On a Recent Problem of Communication-Storage Tradeoffs for Secure Multicast for Large Networks

by R. Poovendran and John S. Baras

CSHCN T.R. 99-37
(ISR T.R. 99-78)

JB 99-29

Entitled:

On a Recent Problem of Communication-Storage Tradeoffs for Secure Multicast for Large Networks

Authors:

R. Poovendran and J.S. Baras

Conference:

Second Workshop on Security in Communication Networks
September 16-17, 1999
Amalfi, Itlay

On a Recent Problem of Communication-Storage Tradeoffs for Secure Multicast for Large Networks (Extended Abstract)

R. Poovendran, J. S. Baras
{*radha, baras*}*@isr.umd.edu*
Dept. of Electrical and Computer Engineering & Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

Abstract

A variety of rooted-tree based secure multicast networks with different efficiencies and storage requirement that is linear in group size have been proposed [15, 16, 18, 19]. Recently, Canetti et al presented a scheme [21] based on clustering that had sub-linear storage requirements at the group controller. However, they were unable to prove or disprove that the scheme was optimal, and posed it as an open question. In this paper we answer the question with affirmative NO! Additionally we use our results from [25, 26] to show that the optimal clustering in this context is related to maximum entropy of member revocation event, and corresponding optimal strategy is to partition members such that each cluster has the same probability of being revoked.

Key Words: Multicast Key Distribution, Internet, Member Deletion/Revocation, Entropy, Dynamic Groups.

1 Introduction

Providing secure multicast for large, distributed, dynamic groups is becoming an important issue in the Internet commerce and Military communications. Network related problems such as intelligent routing, content replication, caching, and security are primary areas of interest for service providers with large user base. Among the secure communication models such as one-to-many, many-to-many, vast majority of the commercial sector applications tend to be one-to-many models with a single service provider having large number of end users subscribing to a particular service.

Many of the distributed applications like Internet newscast, stock quote updates, and distributed conferencing registration may benefit from secure group communications. Providing an effective key management scheme for these applications is complicated by the nature of a group that is *netgraphically* distributed and exhibits varying degrees of trust, i.e. different parts of the group may have different security strengths that can be assumed in key management. Issues such as the nature of the security parameters, inter-domain and intra-domain interoperability, and authentication across domains need to be addressed in providing a secure multicast key management.

Although the commercial environment may be able to tolerate the compromise or failure of end users up to a threshold number with reasonable financial risk, such is often not the case for military networks.

In the context of military tactical networks, the secure communication "information flow" model is often hierarchical. Moreover, the end users are platoons or clusters consisting of members with the following stringent requirements on the key management:

1. Ability to remove a single member from the secure multicast group with minimal additional messages.
2. Ability to communicate with every member, even if the rest of the cluster members have failed.
3. Ability to merge or split two or more clusters.
4. Ability provide fast authentication.

In order to explain the complexities involved in secure multicast compared to secure unicast, we first present the secure communication model under consideration. Many recent papers provide detailed review of this model and requirements [18, 22, 13, 26, 5, 6, 15, 16]. Among these, the review in [18] is timely, exceptionally well written, and is updated very frequently. We present the basic model that can be found in [18, 25, 26]. In particular, we follow the description in [18, 21, 25] due to the close nature of the problem described in this paper.

1.1 Description of the Communication Model

A centralized Group Controller (GC) is assumed to be responsible for distributing all the required cryptographic keys to the group members. In general, members can use their public keys to communicate with GC. However, if the communication is such that the GC has to send identical message to very large membership, it is convenient for storage, and efficient in terms of encryption, to let members share a common data or traffic encryption key called Session Key (SK). If the SK needs to be updated over a period of time for a variety of reasons including key lifetime expiration, compromise of the key, and/or temporary failure of one of the members, there has to be a mechanism to securely update the SK of all valid members. Although the use of public keys is one approach to achieve this goal, a specific key called the Key Encrypting Key (KEK) can be distributed to the members to reduce the encryptions done by GC. Instead of using a single KEK, each member is given a set of KEKs that are shared with different members for broadcast efficiency. Clearly, minimization of the number of keys to be stored by the GC becomes an issue for very large networks.

One of the main focus of the current research has been to find efficient key distribution schemes under worst case scenario (single member revocation) with storage minimization without introducing vulnerabilities such as user collusion. In passing we note that some of the seemingly efficient schemes [22, 20] have serious security weakness.

In a group communication model, removal or addition of one or more members does not necessarily terminate the session. Since there is more than one member involved in the communications, the group size may vary during the session due to a variety of reasons. These changes in turn prompt the SK update to prevent unauthorized access to group communications. The SK may have to be updated due to any of the following reasons:

- Expiration of the lifetime of the session key.
- Join/Admission of a member.
- Deletion/Revocation of a member.

- Voluntary leave of a group member.

We now illustrate the issues in key distribution by considering two extreme cases.

1.2 Two Extreme Cases of Key Distribution

The secure group communication requires KEKs to securely distribute the updated SK. If every member has an individual public key, for a group consisting of N members, the SK update will involve $\mathcal{O}(N)$ encryption by the GC. The linear increase of the required number of encryptions in group size is not suitable for very large scale applications common in the Internet, due to the amount of computational burden on the GC.

A simple way to reduce the number of encryption by the GC at the time of SK update is to provide a common KEK to all the members of the group as suggested in [13]. If the SK is to be updated due to its lifetime expiration, the GC can perform a single encryption and update all the group members. If the SK is to be updated due to a new member admission, before admitting the new member, GC may choose a new SK and the future KEK, encrypt both using the current KEK and update all the members. The newly admitted member is given the new SK and the KEK separately. However, this approach fails to support the secure communication if a single member is to be deleted/revoked. Since the whole group, including the deleted/revoked one share a single KEK, a revoked member will have access to all future key updates. Hence, this approach doesn't provide an efficient recovery mechanism for the valid members in the presence of a single member failure.

On the other spectrum, the set of keys are partitioned into two groups with respect to each member. One of these sets is called the complement set and contains keys that are not distributed to a particular member. If each member has a unique complementary set, this set can be used for key updates in the event the corresponding member is revoked. The GC associates a KEK and a member in a one-to-one manner. If there are N members in the group, there will be N KEKs each representing a single member. The GC then distributes these N KEKs such that a member is given all the KEKs except the one associated with him/her. Hence, the complementary set contains a single KEK for each member. If the GC wants to delete/revoke a member, it needs to broadcast only the member index to the rest of the group. Since all members except the revoked one has the associated KEK of the revoked member, they can use that KEK for SK updates. This approach requires only one encryption at the GC and allows the GC to update the SK under single member compromise. In fact this approach seem to allow even multiple member deletion/revocation. Considering the complementary sets of any two members reveals that all the KEKs of the group are covered by the KEKs held by any two members. Hence, any two deleted/revoked members can collaborate and have access to all future conversations. Thus, under user collusion, this key scheme does not scale beyond two members. Thus the scheme doesn't have perfect forward secrecy under collusion of revoked members. This approach requires KEK storage that scales as $\mathcal{O}(N)$.

The above mentioned schemes are two extremes of KEK distribution. Depending on the degree of user clustering, a large variety of key management schemes with different amounts of KEKs per member can be generated. Currently available solutions for key management can be grouped into two classes based on the available results. The first class is called the non-tree types, and the second one is called the rooted-tree type. Among these, rooted-tree based schemes have become popular due to their communication-storage efficiency with no functional relationship requirements among the keys. These schemes were the focus of the work reported in [21] where the *open problem* addressed in this paper was posed.

The paper is organized in the following manner. Section 2 presents the review of non-tree schemes. Section 3 presents the review of basic concepts behind the rooted-trees based key distribution. Section 4 presents the work reported in [21] and the open problem posed. Section 5 presents some necessary preliminaries that can also be found in [25]. Section 6 addresses the optimal clustering of a given secure multicast group. Section 7 presents the analysis of the storage-communication minimization problem in the context of the clustering presented in [21] and shows that the storage requirements at GC can be computed explicitly. We then show that the results of this calculation reveal that the issue of optimality does not arise due to the nature of this function. We also prove this point by considering a simple ordinary differential equation that there is no such minima. We however show that there is a minimal point which corresponds to the case of minimal storage scheme, and there can be a host of parameter selections that will yield results better than the sub-linear storage $\mathcal{O}(\frac{N}{\log N})$ given in [21].

2 Review of the Non-Tree Based Key Distribution Schemes

Non-tree based approaches are listed in [5]-[9]. Among these [5] uses group Diffie Hellman technique for generating the keys. This scheme makes use of the difficulty of performing the discrete logarithm in generating the keys. The computations increase linearly with the group size.

In [6], an approach with computations independent of the group size was proposed. However this scheme has problem due to member collusion.

In [14], a cluster based approach was proposed. In this scheme, each cluster has its own session keys and KEKs. This method with suitable modifications has been incorporated into some of the tree based schemes [22, 21].

Schemes based on the assumption of information theoretic security have been recently presented in [7, 8, 9]. Notion of key capacities and information rates also was used by Maurer [10] in secret key agreement problem by two members. Additional results along the lines of information rates were studied by Csiszar et al [11, 12].

3 Review of the Rooted Tree Based Key Distribution Schemes

For clarity, we present the original rooted-tree scheme proposed by [15]. Hierarchical rooted-tree based keys have been used for different applications. The first attempt at using a rooted-tree based key distribution approach for efficient member revocation was independently proposed in [15] and [16]. Modifications to reduce the computational and storage requirements of these two methods were later presented in [18, 19, 22, 20]. The most elegant key distribution approach on the rooted-trees with rigorous proofs on security provided to date is given in [18]. We will briefly review the needed basic concepts behind the rooted-tree based key distribution in this section.

3.1 Distribution of Keys on the Tree

As a concrete illustration, Figure 1 presents a KEK distribution based on a binary rooted tree for 8 members. In this approach, each leaf of the tree represents a unique member of the group; i.e. the leaves are in a one-to-one correspondence with members. Each node of the tree represents a key. The set of keys along the path from the root to a particular leaf node are assigned to the member represented by that leaf node. For example, member M_1 in Figure 1 is assigned KEKs $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$.

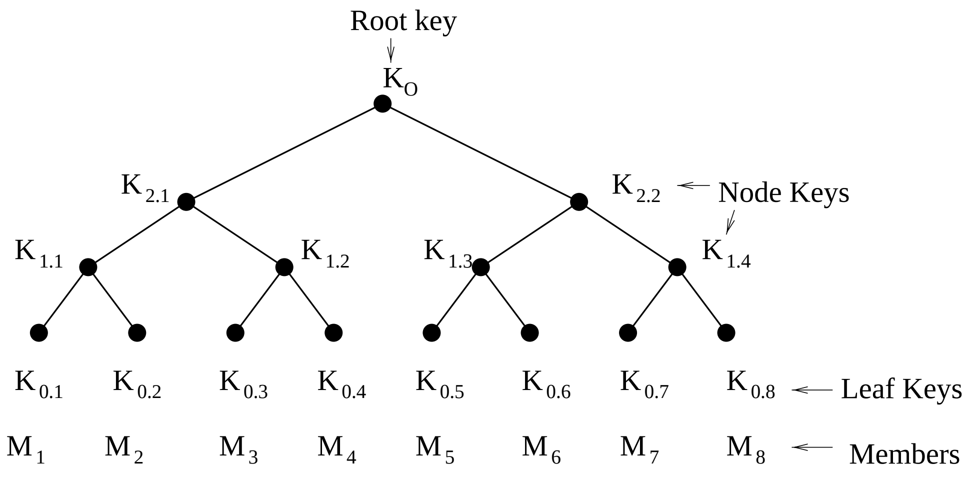
Root key
K_O
$K_{2.1}$
$K_{2.2}$
Node Keys
$K_{1.1}$
$K_{1.2}$
$K_{1.3}$
$K_{1.4}$
$K_{0.1}$ $K_{0.2}$ $K_{0.3}$ $K_{0.4}$ $K_{0.5}$ $K_{0.6}$ $K_{0.7}$ $K_{0.8}$
Leaf Keys
M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8
Members

Figure 1: The Logical Key Tree of [18, 15, 16, 20, 22]

If there is no member deletion/revocation or compromise, the common KEK denoted by K_O can be used to update the SK for all the members. The tree based structure also induces a natural hierarchical grouping among the members. By logically placing the members appropriately, the GC can choose the appropriate keys and hence selectively update, if needed, the keys of the group. For example, in Figure 1, members M_5, M_6, M_7, and M_8 exclusively share the key $K_{2.2}$. The GC can use the key $K_{2.2}$ to selectively communicate with members M_5, M_6, M_7, and M_8. Hence, the local grouping of the members and the keys shared on the tree may be decided by the GC based on application specific needs. In order to be able to selectively disseminate information to a subset of group members, the GC has to ensure that the common key assigned to a subset is not assigned to any member not belonging to that subset. Using the notation $\{m\}_K$ to denote the encryption of m with key K, and the notation A $\longrightarrow$ B : $\{m\}_K$ to denote the secure exchange of message m from A to B, GC can selectively send a message m to members five through eight by the following transmission:
GC $\longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2.2}}$

If, however the key $K_{2.2}$ is invalidated for any reason, GC needs to update the key $K_{2.2}$ before being able to use a common key for members M_5, M_6, M_7, and M_8. It can do so by first generating a new version of $K_{2.2}$, and then performing two encryptions, one with $K_{1.3}$ and the other with $K_{1.4}$. The following two messages are needed to update key $K_{2.2}$ to the relevant members of the group.
GC $\longrightarrow M_5, M_6 : \{K_{2.2}\}_{K_{1.3}}$
GC $\longrightarrow M_7, M_8 : \{K_{2.2}\}_{K_{1.4}}$

3.2 Member Revocation in Rooted Trees

From now on, we will use the term keys to denote SK or KEKs unless there is a need for clarification. Since the SK and the root KEK are common to all the members in the multicast group, they have to be invalidated each time a member is revoked. Apart from these two keys, all the intermediate KEKs of the revoked member need to be invalidated. In the event there is bulk member revocation, the GC has to

- Identify *all* the invalid keys,
- Find the minimal number of valid keys that need to be used to transmit the updated keys.

For an arbitrary tree that may not hold members in all the leaves these two problems need to be solved by exhaustive search. The general principle behind the member revocation is discussed below.

Member M_1 in Figure 1 is indexed by the set of four keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$. Revoking M_1 is equivalent to invalidating these four keys, generating four new keys, and updating these keys of the appropriate valid members. When M_1 is revoked, the following key updates need to be performed: (a) all member need new K_O, (b) members $M_2 - M_4$ need to update $\{K_{2.1}\}$, (c) members $M_3 - M_4$ need to update $\{K_{1.2}\}$, and (d) member M_2 needs to update $\{K_{1.1}\}$.

The following observations can be made towards the rooted tree based key distributions.

- Since each member is assigned $(2 + \log_d N) = \log_d Nd^2$ keys, deletion of a single member requires $(2 + \log_d N)$ keys to be invalidated.
- Since there are $(1 + \log_d N)$ nodes between the root and a leaf and $\log_d N$ nodes are shared with other members, and for each common node one encryption is required, the GC needs to perform a total of $\log_d N$ encryptions.
- For a $d - ary$ tree with depth $h = \log_d N$, the GC has to store $1 + 1 + d + d^2 + \cdots + d^h = \frac{d(N+1)-2}{(d-1)}$ number of keys. Setting $d = 2$ leads to the binary tree for which the required amount of storage works out to be $\frac{2(N+1)-2}{2-1} = 2N$. This result can be independently checked by noting that a binary tree with N leaves has $2N - 1$ nodes. Hence the GC has to store the SK and $(2N - 1)$ KEKs, leading to $2N$ keys that need to be stored.

The above given storage number doesn't make any assumption about the type of key generation. Several modifications to the original scheme have been proposed recently. In [21], an approach for storage-communication minimization was proposed. We summarize the approach and the results of [21].

4 Summary of Results in [21] and the Open Problem

Results in [21] were derived using a hybrid technique. For continuation of the session under member revocation, (a) all the users should have the SK, (b) each user i should share a unique member specific key k_i with the GC, (c) each user may have one or more KEKs it shares with different set of members.

4.1 Reduction of Storage Requirement Using Pseudo-random Functions

Canetti *et al* noted [21] that the GC can minimize the key storage requirements by generating the member specific keys as outputs of a pseudo-random function with indexing. In this scheme [21], the GC holds a single secret key r, an index to a pseudo-random function f_r [21]. The keys are generated for a member i by $k_i = f_r(i)$. When a user is compromised, the GC computes SK, encrypts the new SK with the individual keys of each valid member and distributes. Security of this key generation scheme is based on the security of the pseudo-random function and the encryption scheme used.

4.2 Hybrid Approach

For the KEKs shared by two or more members, we can verify [21] that an attempt to use the technique of pseudo-random function based key generation on the rooted-tree with a single secret key will require the entire tree to be updated and thus increase the encryption requirements from being $\mathcal{O}(\log N)$ to $\mathcal{O}(N)$.

Hence, a mix model of key distribution was proposed in [21] to minimize the key storage requirements at GC. The model can be summarized in the following steps: Given a group of size N,

1. form clusters with fixed size M.
2. build a $a - ary$ rooted-tree with depth
$$b = \lceil \log_a \lceil \frac{N}{M} \rceil \rceil \tag{1}$$
3. assign each cluster to a unique leaf node of the $a - ary$ rooted-tree of depth, denoted by b.
4. form a $M - ary$ tree of one unit deep inside each cluster.

The last interpretation is ours and is not explicit in [21]. Once the $a - ary$ rooted-tree of depth b is constructed, the key distribution is done using the technique in [15]. For the $M - ary$ tree with one unit depth, the key distribution is done using pseudo-random function technique described below [21].

4.2.1 Communication-Storage Parameters

Using the minimal storage scheme in conjunction with the rooted-tree, the user storage, GC storage, and the number of encryptions needed were presented in [21] in a tabular format. We reproduce the results below.

	general M, a	Example 1	Example 2
user storage	$\log_a(\frac{N}{M})$	$\mathcal{O}(\log n)$	2
GC storage	$\frac{N}{M}\frac{a}{(a-1)}$	$\mathcal{O}(\frac{n}{\log n})$	$n^{0.5} + 1$
Encryptions	$(M-1) + (a-1)\log_a(\frac{N}{M})$	$\mathcal{O}(\log n)$	$2n^{0.5} - 2$

Table 1. Parameters of the tradeoff scheme in [21].
Setting $a = 2$, $M = 1$ leads to results in [15]. In example 1, $a = 2$, $m = \mathcal{O}(\log n)$, in example 2, $a = m = n^{0.5}$

4.2.2 Discussion

In [21], authors posed a question whether the sub-linear storage requirement at the center for example 1 was the optimal solution. They left it as an open question. We note that the answer to their question is affirmative NO!.

We also note that if the group membership is not homogeneous in terms of degree of security, there is an analytical proof that the storage requirements on the rooted tree can be further reduced and hence the theoretical model presented for clustering in [21] can be improved upon by studying the basic concepts from information theory. We used these ideas to solve the optimal number of keys per member, and computationally affordable key length in [25, 26]. In this paper we use entropy in yet another useful manner in clustering. Although the idea of using information theory is same, the applications and the results in this paper are completely different from those in [25, 26]. We first review the basic ideas that are useful [25, 26] for probabilistic modeling in the next section.

5 Probabilistic Modeling of Member Revocation

Since the key updates are performed in response to member revocation, statistics of *member revocation event*, is appropriate for system design and performance characterization. We denote p_i as the probability of revocation of member i.

5.1 Defining the Shannon Entropy of Member Revocation Event

In physical processes that involve probabilistic modeling, one can often define the uncertainty of the occurance of an event using a suitably defined entropy of the process. We will use Shannon entropy [1] to express the amount of uncertainty as to which member will be revoked. We first state the definition of the *Shannon entropy* in the context of member revocation event.

Definition: We define the $d-ary$ entropy H_d of the member revocation event by

$$H_d = -\sum_{i=1}^{N} p_i \log_d p_i \tag{2}$$

where p_i is the probability of revocation of member i. As mentioned earlier, the entropy expresses the uncertainty as to which member will be revoked in $d-ary$ digits.

We note the following important points:

1. A word of caution is in place since the Shannon entropy is often used to describe the rates in the source coding literature. We use it in the context of its physical interpretation which is the amount of uncertainty about the occurance of an event.
2. We also note that the definition of the entropy of member revocation event did not require the knowledge about the keys or key distributions. The basic process is independent of the key distribution. The member revocation probability can be related to the keys *iff* we can map the individual members to another event in a unique manner.
3. Mapping of the individual members to unique leafs of the tree for example forms a one-to-one map. Then, revoking a member can be interpreted as revoking the leaf node of the tree. Hence the probabilities of member revocation can be interpreted as probabilities of leaf node (or leaf node key) revocation. Massey discusses the separation and the mapping of a physical process to an algorithm in detail in [3].
4. Since the member revocation event and the leaf node key revocation event are probabilistically identical, entropy of the member revocation event is same as the entropy of the leaf key revocation event. This important observation is summarized as a theorem below.

Theorem 2. *Leaf Key Revocation Entropy* is the entropy or uncertainty as to which of the leaf key will be revoked. Since the leaf key revocation probability is in one-to-one correspondence with the member revocation probabilities, Leaf Key Revocation entropy is identical to the entropy of the member revocation event.

We now show, without proofs that if members of the group have different probabilities of revocation, the strategy of forming clusters with equal number of members should be replaced by the strategy of forming clusters of equal probability to obtain a rooted-tree that is full and has all the leafs at the same depth. Otherwise the strategy is to build a tree such that a cluster having probability of revocation $\hat{p}_i$ be at depth $\log_a \hat{p}_i$ from the root [25, 26].

6 Cluster Size Selection

Since the set of KEKs assigned to a cluster should be unique, and the KEKs are distributed on the nodes of the tree, the unique indexing requires that the *number of keys* assigned to a cluster should satisfy the Kraft inequality [25, 26]. Denoting the number of keys assigned to a cluster with probability of revocation p_i by l_i, we note

$$\sum_{j=1}^{N} a^{-l_i} \leq 1. \tag{3}$$

Minimization of the average number of keys held by a member with the unique indexing leads to the solution that the optimal number of keys assigned to a cluster with revocation probability $\hat{p}_i$ is given by $l_i = -\log_a \hat{p}_i$. The following theorem summarizes the result in [21].

Theorem 1. For a rooted-tree based key assignment that satisfy Kraft inequality, optimal average number of keys, excluding the root key and the SK, held by a member is given by the $a - ary$ entropy $H_a = -\sum_{i=1}^{N} \hat{p}_i \log_d \hat{p}_i$ of the *member revocation* event. For a member i with probability of revocation $\hat{p}_i$, satisfying the optimization criteria, the optimal number of keys l_i, excluding the root key and the SK, is given by

$$l_i = -\log_d \hat{p}_i. \tag{4}$$

From the formula, we note that the depth of *all* the leafs from the root are equal *iff* $l_i = \text{constant}; \forall i$. Hence, the probabilities of revocation should be equal for all the members to form clusters of equal size. Under this case, the entropy of member revocation scheme attains its maximum value of $\log_a N$.

Hence, we have shown that the cluster selection in [21] corresponds to a special type of network which has equal probability of member revocation.

7 Answer to the Open Question

The simplest approach in answering the question posed in [21] is to count the number of keys to be stored at the center. The key distribution structure in [21] has an $a - ary$ rooted-tree with depth b followed by cluster of M members at each leafs. The total number of keys excluding the SK, is given by

$$\begin{aligned} S &= \textstyle\sum_{i=0}^{b} a^i \\ &= \frac{a^{b+1} - 1}{a - 1} \\ &= \frac{aN - M}{M(a-1)}. \end{aligned} \tag{5}$$

Setting $a = 2$, $M = 1$ leads to the familiar result in [15] for binary rooted-tree in case that has (2n -1) keys excluding the SK.

Results of sub-linear case was derived in [21] by setting $M = \log N$. To check if this is an optimum point at all, we just need to check if the storage function

$$\begin{aligned} \frac{aN - M}{M(a-1)} &= \frac{a\frac{N}{M} - 1}{a-1} \\ &= \frac{a\frac{g(M)}{M} - 1}{a-1} \end{aligned} \tag{6}$$

where $g(M)$ is a function of M. We note that the function $\frac{a\frac{N}{M}-1}{a-1}$ is a hyperbola in M for a fixed value of a and N. For a hyperbola, the minimal point is the value of the hyperbola at the highest value of the domain of the function. In this specific case, this point is $M = N$ and this forces $a = N$. In fact any point between $\log_a N < M \leq N$ will yield improved storage minimization since the function $\frac{a\frac{N}{M}-1}{a-1}$ is monotonic in M.

We present numerical examples in a tabular manner for a binary tree ($a = 2$). For this case the storage function reduces to $\frac{N-M}{M}$.

N	$\log N$	Range for improved results
2^{10}	10	$N \geq M > 10$
2^{15}	15	$N \geq M > 15$
2^{20}	20	$N \geq M > 20$

It can also be shown that the following differential equation needs to be satisfied

$$\frac{\frac{d}{dM}g(M)}{g(M)} = \frac{1}{M} \tag{7}$$

This leads to the solution that $N = g(M) = M$. This solution of course is the minimal point discussed earlier.

8 Conclusion

Providing a key management scheme for distributed large scale networks is a difficult problem. Although many elegant schemes have been proposed, an appropriate solution which can handle group dynamics and still be "light weight" seem to pose a major challenge. In commercial applications it may be appropriate to tolerate a degree of session key loss. This may be a better solution than trying to re-key each time a member is compromised. However, the military networks don't have that flexibility. If the military network tends towards wireless and low power devices, that are distributed in a next generation battle field, the affordable computational power becomes an issue. At the same time, for a military network, re-keying under every single member removal/compromise is mandatory. Hence, the problem of re-keying under worst case is a very relevant and one of the most difficult problem.

We have not shown the utilization analysis due to page limit. Depending on the allowed final pages of the journal paper, we will provide the results of utilization analysis, and the removal of logic circuit evaluation as the method of grouping the valid members.

Acknowledgments

We thank professors A. E. Ephremides and Jim Massey for providing references [3, 4] on the information theoretic analysis of rooted-tree based algorithms by Massey.

References

[1] T. Cover, J. Thomas, Elements of Information Theory, John Wiley & Sons, Inc, NY, 1991.

[2] R. Gallager, Information theory and reliable communication, Wiley, NY, 1968.

[3] J. L. Massey, "An Information-Theoretic Approach to Algorithms", Impact of Processing Techniques in Communications, In NATO Advanced Study Institutes Series E91, pp. 3-20, 1985.

[4] J. L. Massey, "Some Applications of Source Coding to Cryptography", In European Trans. on Telecom., Vol. 5, pp. 421-429, July-August 1994.

[5] M. steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication", 3rd *ACM Conf. on Computer and Communications Security"*, 1996.

[6] A. Fiat and M. Naor, "Broadcast Encryption", *Advances in Cryptology- Crypto'92*, Lecture Notes in Computer Science. vol. 773, pp. 481-491, Springer-Verlag, Berlin Germany, 1993.

[7] D. R. Stinson, and T. V. Trung, "Some New Results on Key Distribution Patterns and Broadcast Encryption", to appear in Design, Codes and Cryptography.

[8] D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption", to appear in Design, Codes and Cryptography.

[9] M. Brumester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", *Advances in Cryptology- Eurocrypt'94*, Lecture Notes in Computer Science. vol. 950, pp. 275-286, Springer-Verlag, Berlin Germany, 1994.

[10] U. Maurer, "Secret Key Agreement by Public Key Discussion from Common Information", IEEE Trans. Information Theory, vol 39, No 3, pp. 733 -742, May 1993.

[11] R. Ahlswede, I. Csiszar, " Common Randomness in Information Theory and Cryptography - Part I: Secret Sharing", IEEE Trans. Information Theory, vol 39, No 4, pp. 1121 - 1132, July 1993.

[12] R. Ahlswede, I. Csiszar, " Common Randomness in Information Theory and Cryptography - Part II: CR Capacity", IEEE Trans. Information Theory, vol 44, No 1, pp. 225-240, January 1998.

[13] H. Harney and C. Muckenhirn, "GKMP Architecture", *Request for Comments(RFC)* 2093, July 1997.

[14] S. Mittra, "Iolus: A framework for Scalable Secure Multicasting", In *Proceedings of ACM SIGGCOM'97*, pages 277–288, September 1997.

[15] D. M. Wallner, E. C. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures", Internet Draft, September 1998.

[16] C. K. Wong, M. Gouda, S. S. Lam,"Secure Group Communications Using Key Graphs", In *Proceedings of ACM SIGCOMM'98*, September 2-4, Vancouver, Canada.

[17] R. Canetti, and B. Pinkas, "A taxonomy of multicast security issues", *Internet draft*, April, 1999.

[18] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Efficient Reconstructions",to appear In *Proceedings of IEEE Infocom'99.*

[19] D. A. McGrew and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", *Manuscript, 1998.*

[20] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, "Efficient Security for Large and Dynamic Groups", In *Proc. of the Seventh Workshop on Enabling Technologies*, IEEE Computer Society Press, 1998.

[21] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption", In Eurocrypt 99, pp. 456 - 470.

[22] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques", To appear in Proceedings of IEEE Infocom'99.

[23] R. Canetti, P-C. Cheng, D. Pendarakis, J. R. Rao, P. Rohatgi, D. Saha, "An Architecture for Secure Internet Multicast", *Internet Draft*, November 1998.

[24] B. Quinn, "IP Multicast Applications: Challenges and Solutions", *Internet draft*, November 1998.

[25] R. Poovendran, and J. S. Baras, "An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes", Springer Verlag Lecture Notes in Computer Sciences, *Advances in Cryptology*- CRYPTO'99, August 1999, Santa Barbara, USA.

[26] R. Poovendran, and J. S. Baras, "An Information Theoretic Approach to Multicast Key Management", in Proceedings of IEEE Information theory and Networking Workshop, Metsovo, Greece, June, 1999.