

A Detailed Mathematical Analysis and Design of Rooted- Tree-Based Multicast Key Management Schemes

April 13, 2004

Jing Zhang – jzhang@tamu.edu

Navendu Misra – navendu@tamu.edu

Part I Introduction

Tree based multicast key distribution

Unicast and Multicast communication

- unicast

a point-to-point data delivery system

- multicast

a group based network communication

- A group is made up of members interested in sharing some information. While ensuring that this information is not available to those outside of the group.

Security

The goal is to provide essential features of security to a group based communication: confidentiality, authenticity, integrity of message between members of the group. Multicast is more susceptible to attacks [2, 3]. Such as an attacker posing as a possible member or an attacker going after the well publicized server.

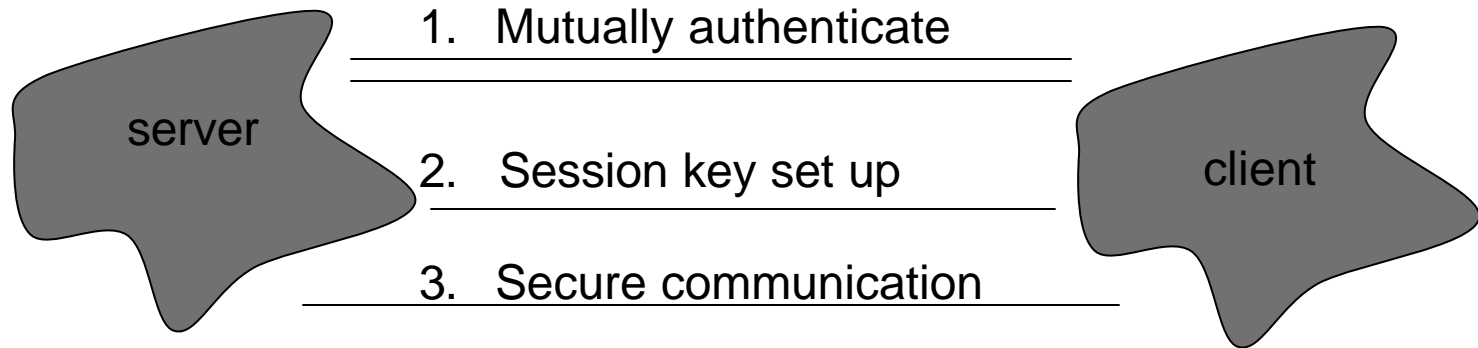
Security in unicast

Security in unicast is well known and has been thoroughly discussed in the class:

- integrity and confidentiality were covered in the lectures on cryptography
- authentication was covered in the lectures on MAC algorithms

Unicast security illustration

Bellow is an illustration of secure communication in unicast



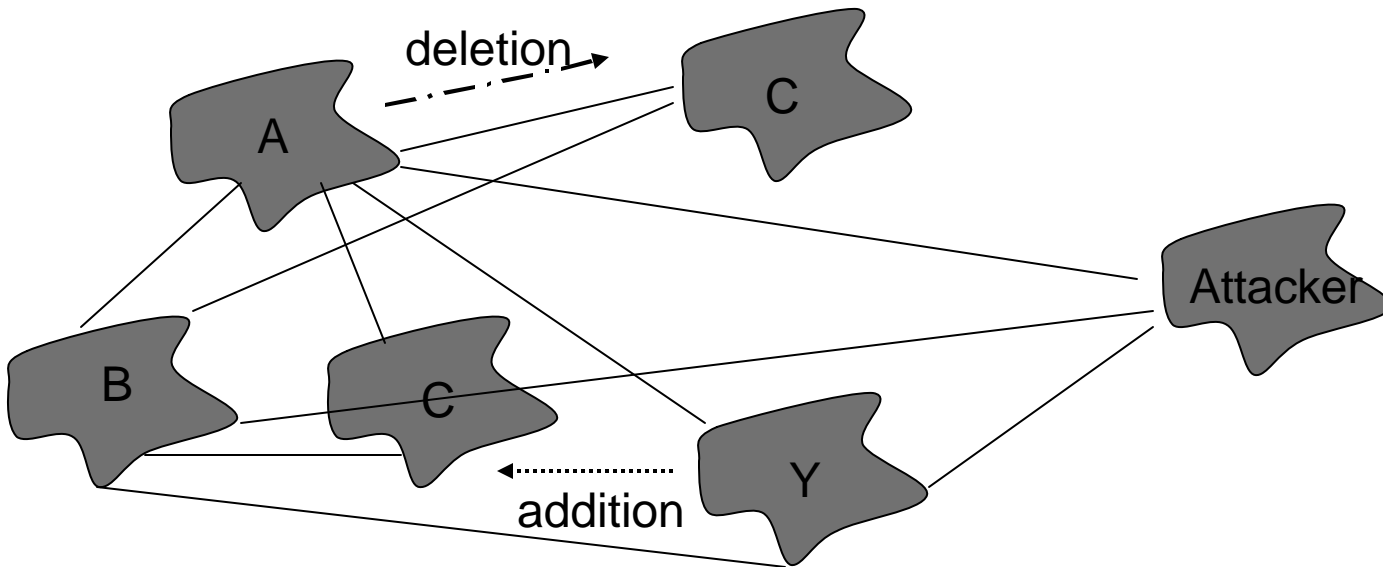
Security in multicast illustration

C leaves the group and should no longer have access to private group data exchange.

Y is added should gain access to future group communications but not past exchanges.

Secure communication

Inaccessible communication



Desire and Problem?

“The desire to create small groups of users all interconnected and capable of communicating with each other, but who are securely isolated from all other users ... “ [2]

The problem is to maintain keys after a member joins or leaves.

Join is easy leave is hard. We will see this throughout the presentation.

Scaling to multicast

It is going to be very hard to scale to multicast. Multicast protocols have to face two opposing features [2] :

- 1 affects n failure, action of one member affects all others.
- 1 does not equal n failure, protocol can not deal with group as whole must meet demands of each individual user.

We will discuss throughout, how the approaches measure up to each of these two.

Group based (hierarchical) Key Trees

Discussed by two papers that came out at the same time [1, 2]. It improves scalability over the unicast approach to key distribution to group members.

The hierarchical tree approach provides three desirable properties:

1. Secure removal of a node
2. Transmission efficiency
3. Storage efficiency (discussed later in presentation)

Differs from hierarchical agent

Another approach is *lolus* [3] (core-based tree key) which introduces the hierarchical agents. Here each user belongs to a sub-group and each sub-group behaves as a group on to itself. But, the primary difference is that there is no global group key. This makes direct control by a central server much harder.

Tree key graphs are different in that they do not treat each group as a group on it self. In fact as we will see shortly *lolus* gives a physical tree while tree key graphs have virtual trees.

Tree based key distribution

A secure group is defined as follows a triple (U, K, R) [1].

U is the set of users,

K is the set of keys and

R is the relation between keys and users.

Key graphs

A key graph is a graph with two types of nodes:

- u-node representing users
- k-node representing keys

Each u-node has only out going edges.

Each k-node can have out going and incoming edges.

A root has only incoming edges.

* However, not all key graphs use the same representation.

The approaches for key graphs

- Star key graphs
- Tree key graphs

The difference between the two is the number of keys that they each user holds.

The star key graph is a special case of tree key graph.

Star key graph

Here each user has only two keys; the individual key and group key. This presents the problem of scalability.

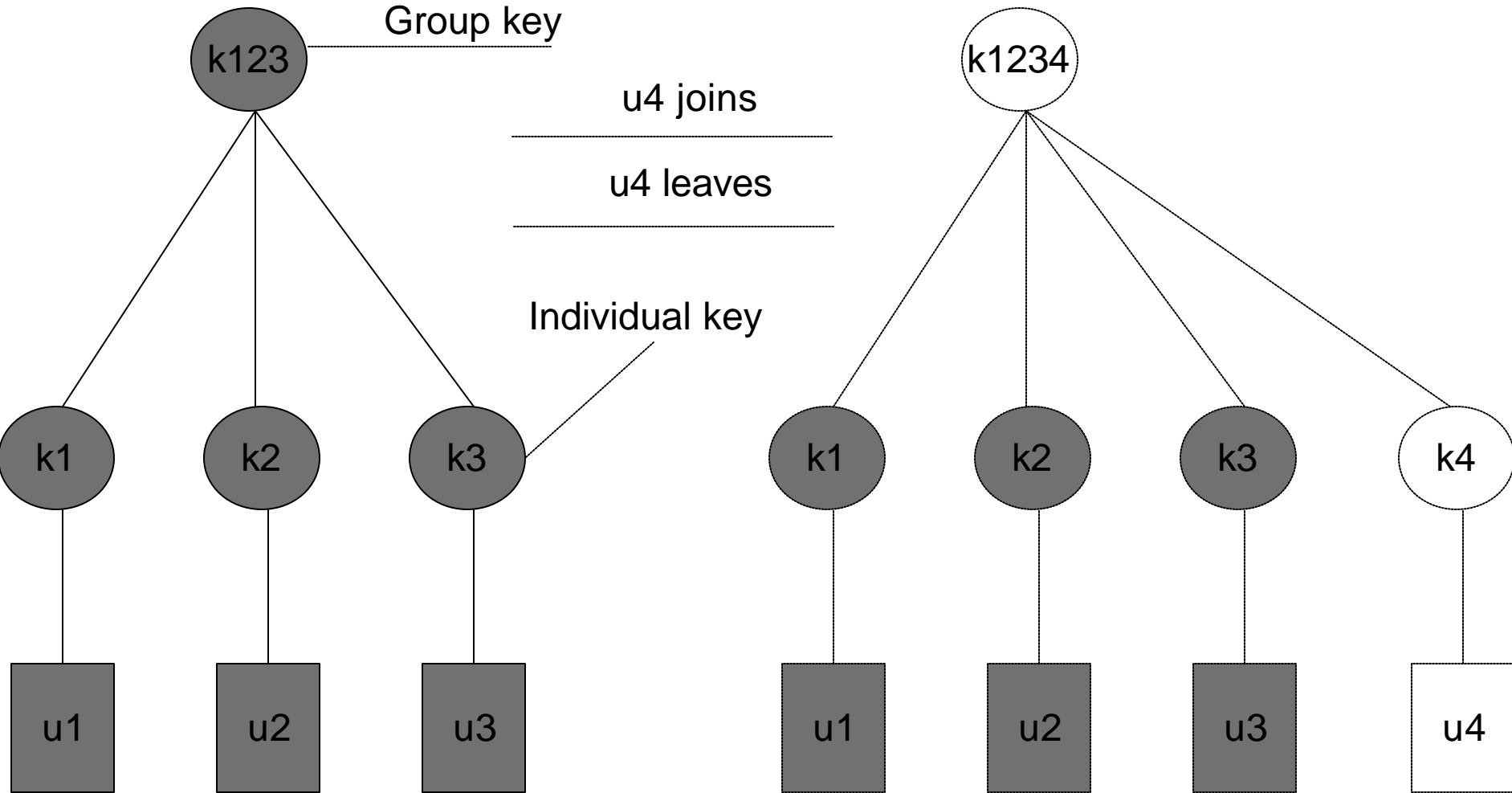
As we will see shortly the cost of updating keys is generally higher.

Joins and leaves in Star Key Graph

Joins: After a user joins the group a new group key is generated. This new key is encrypted by the old group key and multicast to the old members. Then this group key is sent to the new member by its individual key.

Leaves: After a user is acknowledged to be removed by the server. The server generates a new group key. This new key is sent to each member by unicast by encrypting it with each member's individual key.

Joins and leaves in Star Key Graph [1]



Group based Tree key graphs

There is a marked improvement in scalability by the usage of group based key management. A trusted server responsible for group access control and key management. Each member will be given three keys:

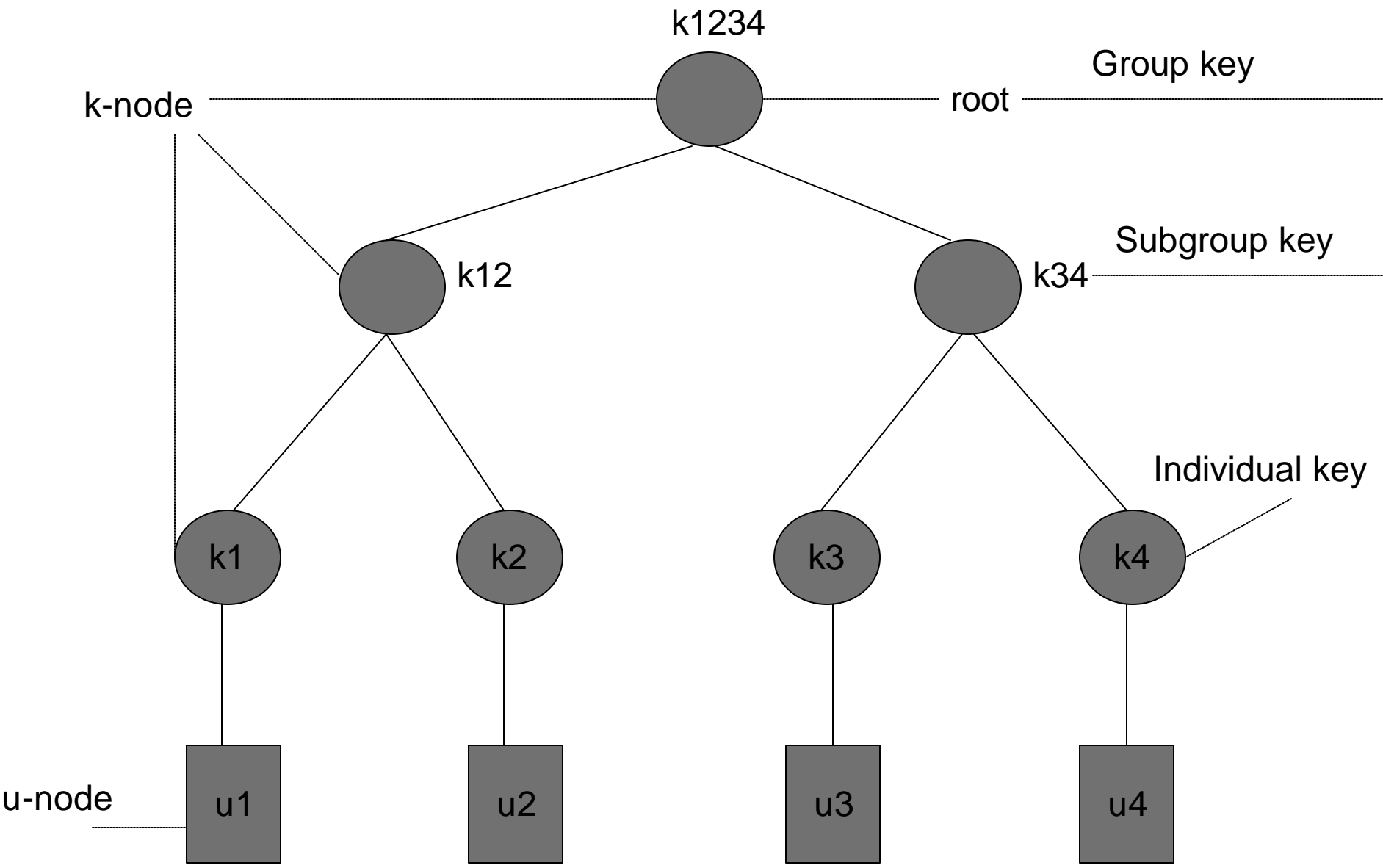
- Individual key I
- Group key G
- Sub-group key S

Group based key graph example [1]

$$U = \{u_1, u_2, u_3, u_4\}$$

$$K = \{k_1, k_2, k_3, k_{12}, k_{34}, k_{1234}\}$$

$$R = \{(u_1, k_1), (u_1, k_{12}), (u_1, k_{1234}), \\ (u_2, k_2), (u_2, k_{12}), (u_2, k_{1234}), \\ (u_3, k_3), (u_3, k_{34}), (u_3, k_{1234}), \\ (u_4, k_4), (u_4, k_{34}), (u_4, k_{1234})\}$$



Group based Key graph [1]

Joining a Tree Key Graph

Unlike star key graphs, tree key graphs have more than one method of re-keying:

1. User oriented re-keying

For each user the server constructs a re-key message that contains precisely the new keys needed by the user and encrypts using the individual key. Costliest.

Joining a Tree Key Graph cont'd

2. Key-Oriented re-keying

For each k-node whose key has been changed the server encrypts the new key with the old key and sends it to the users that share that key. Less costly.

3. Group-Oriented re-keying

The last approach is group oriented re-keying the main focus of the paper. Least costly.

Leaving a Tree Key Graph

There are three ways for the server to distribute keys upon a leave.

1. User oriented re-keying

Each user gets a re-key message in which all the new keys it needs (group and subgroup) are encrypted by the individual key. Costliest.

Leaving a Tree Key Graph cont'd

2. Key-oriented re-keying

Each new key is encrypted individually.
Stores encrypted new keys for use in
different re-key messages.
Less costly.

3. Group-oriented re-keying

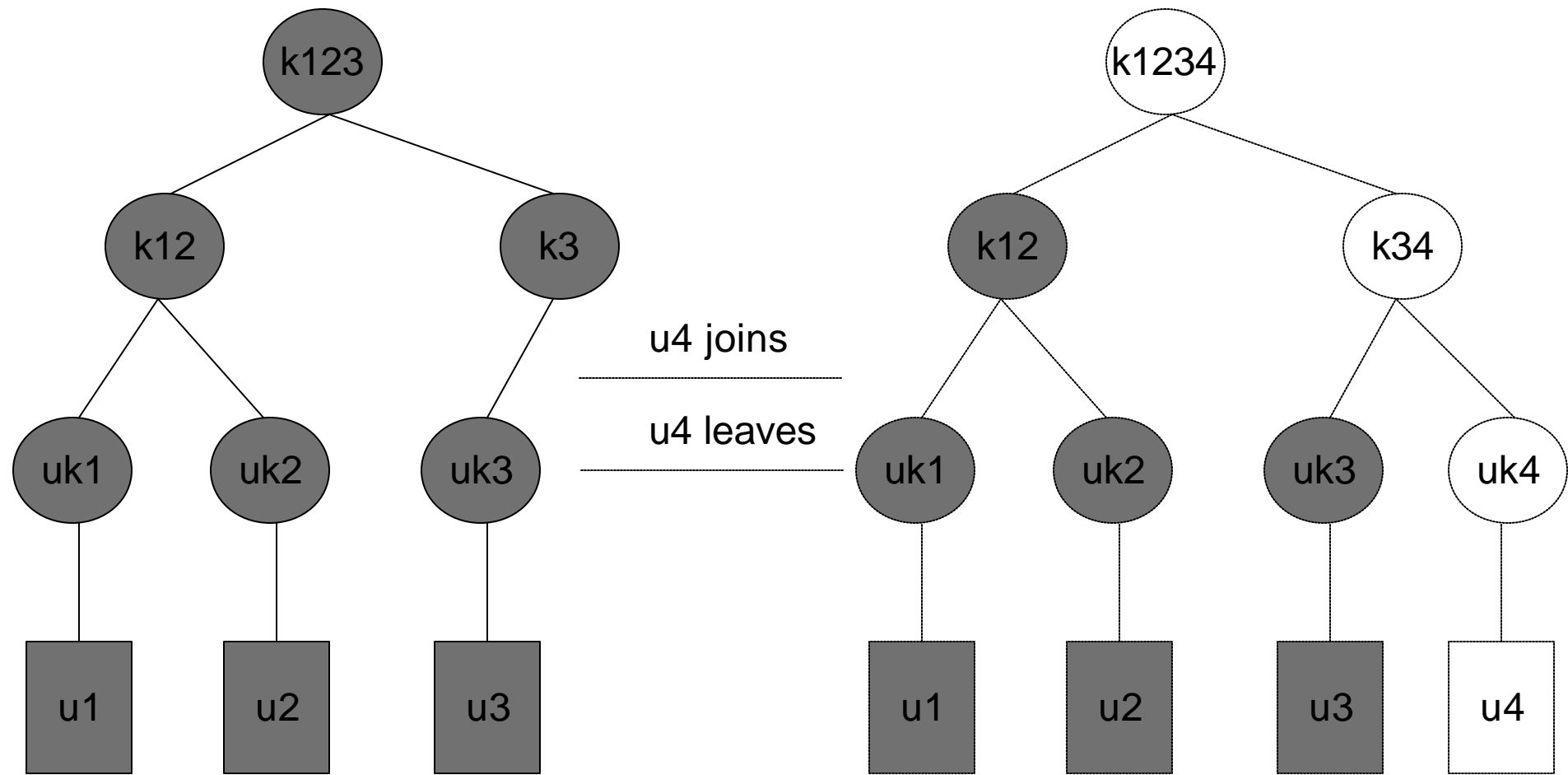
A single re-key message is created
containing all new keys. Least costly.

Joins and leaves using group oriented re-keying

$$U = \{u_1, u_2, u_3, u_4\}$$

$$K = \{k_1, k_2, k_3, k_{12}, k_{234}, k_{1234}\}$$

$$R = \{(u_1, uk_1), (u_1, k_{12}), (u_1, k_{1234}), \\ (u_2, uk_2), (u_2, k_{12}), (u_2, k_{1234}), \\ (u_3, uk_3), (u_3, k_{34}), (u_3, k_{1234}), \\ (u_4, uk_4), (u_4, k_{34}), (u_4, k_{1234})\}$$



Joins and leaves in a key tree graphs[1]

Group-oriented re-keying leaves

Let us now consider that u4 leaves the group.

The server needs to send a new group key (k_{123}) and sub-group keys (k_{12}) and (k_3). This can be done using the key-encryption-keys (KEK).

What new keys need to be sent out?:

- u1 needs a new group key
- u2 needs a new group key
- u3 needs a new group key as well as a new sub-group key

Why is this the case?

Since u1 and u2 were in a different sub-group u4 did not know their sub group key. The only keys that were shared by them was the group key.

However, u3 was in the same sub group as u4 hence they both shared the group and sub-group keys. Thus u3 needs a new set of these.

Example cont'd

The server sends to u1 and u2 a new group key (k_{123}) by encrypting the key with their sub-group (k_{12}) key.

This is safe because u4 did not KNOW this sub-group key.

The server sends to u3 a new group key (k_{123}) and sub-group key (k_3) by encrypting these keys with the individual key of u3 (uk_3).

This is safe because u4 did not KNOW the individual key of u3.

Part II

Mathematical analysis and design of tree-based multicast key management schemes based on information theory

1. Mathematical analysis
2. Design technique
3. Conclusions
4. contribution

Assumptions:

- A group session encryption key (SEK) is used to encrypt the data;
- A group key encryption key (KEK) is used for rekeying (SEK).
- All the node keys are different.

? We will base all our analysis on the member deletion process for this presentation.

Mathematical analysis

- Tree shaping issue
- Cover-free issue (Exclusivity)

Tree shaping

- Example 1.

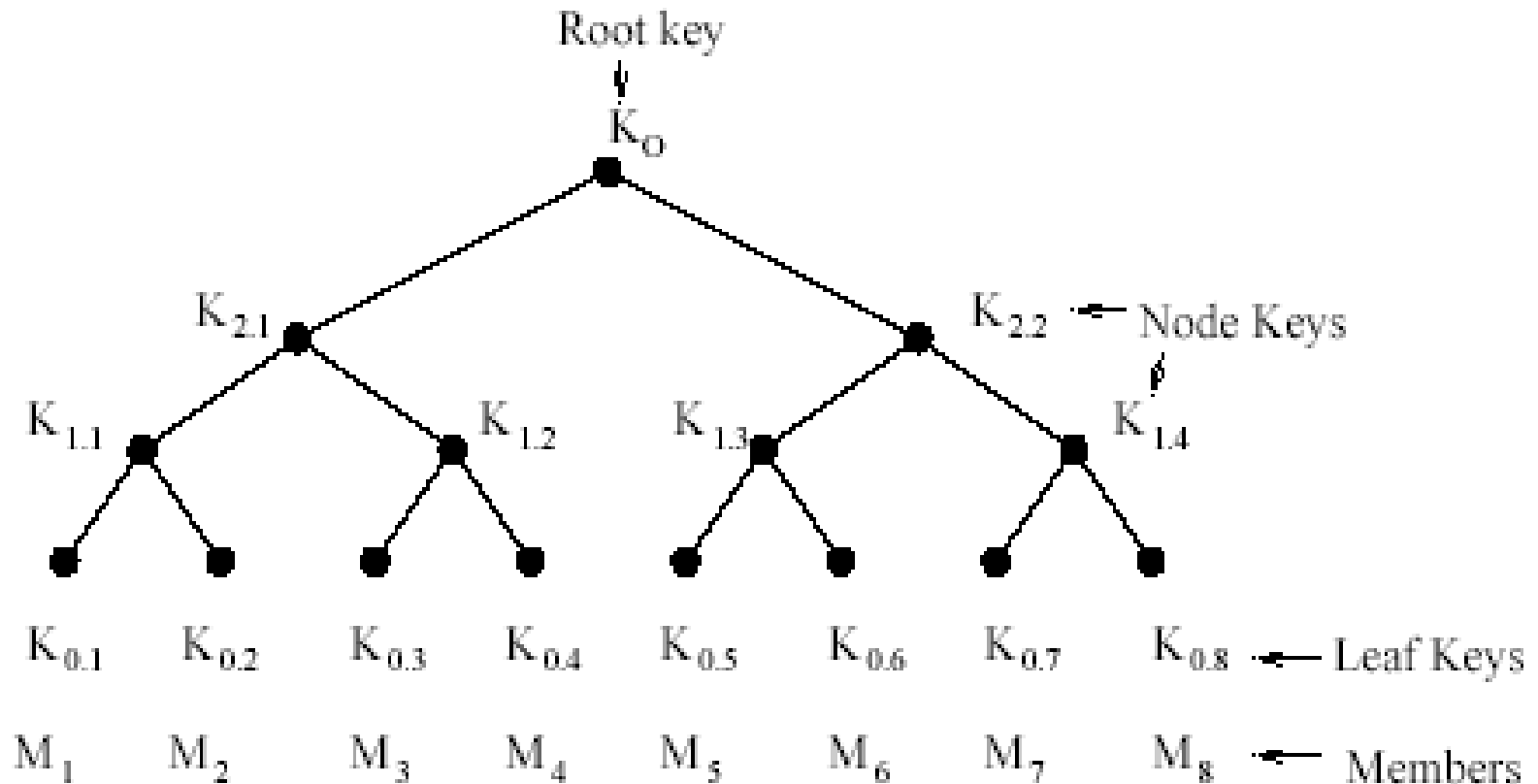


Fig. 1. A logic key tree in [4] (uniform distribution)

● Average # of keys to updated:

$$1 + \log N = O(\log N).$$

● Average per-user storage:

$$2 + \log N = O(\log N).$$

● The storage at GC (Group controller):

$$1 + d^0 + d^1 + d^2 + \dots + d^h = \frac{d(N+1) - 2}{d-1} \Rightarrow 2N = O(N) \text{ with } d = 2.$$

Where

$$h = \log_d N \Rightarrow \log N \text{ with } d = 2.$$

Example 2:

- Average # of keys to be updated:

$$1 + \frac{1 + 2 + 3 + \dots + N}{N} = 1 + \frac{N(N+1)}{2N} = O(N)$$

- Average per-user storage:

$$\begin{aligned} 1 + \frac{2 + 3 + 4 + \dots + (N+1)}{N} \\ = 2 + \frac{N(N+1)}{2N} = O(N). \end{aligned}$$

- The total storage at GC:

$$N + (N-1) + 1 = 2N$$

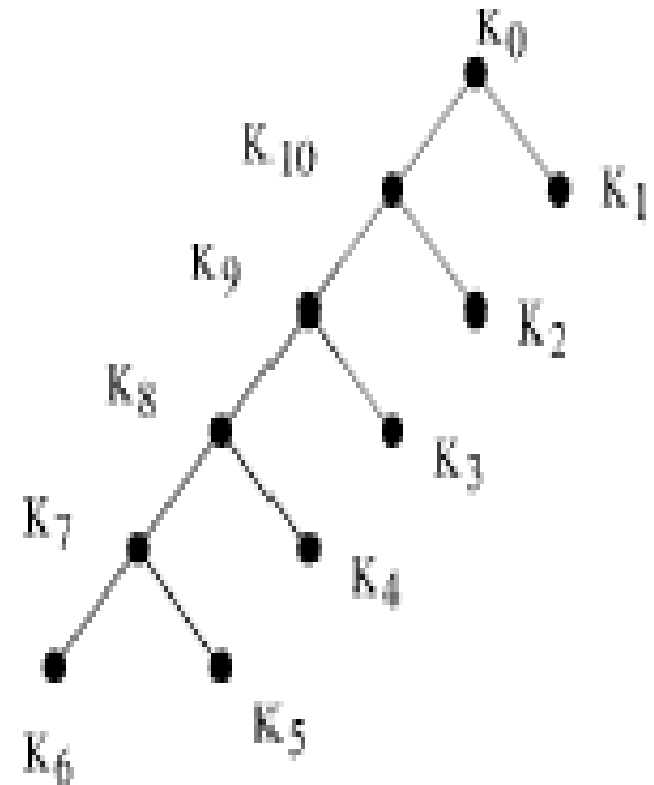


Fig. 2. A logic key tree in [4].
(uniform distribution)

Claim 1: The average number of keys to be updated can be considered as an efficient parameter to evaluate the performance.

since it reflects:

- Transmission efficiency
- Storage efficiency

Systematical way to solve this problem?

? use codeword length optimization
technique

Two important facts

Assume there are N users in the system. The i^{th} user's deletion probability is p_i and the total number of keys to be updated on deletion is l_i , where $i=1, 2, \dots, N$. Then

- The average number of keys to be updated (L) per member deletion can be expressed as:

$$L = \sum_{i=1}^N p_i l_i. \quad (1)$$

- Kraft Inequality needs to be satisfied:

$$\sum_{i=1}^N d^{-l_i} \leq 1.$$

Claim 2:

Tree shaping optimization problem is equivalent to the following optimization problem:

$$\min_{l_i} \sum_{i=1}^N p_i l_i \quad (2)$$

subject to the condition:

$$\sum_{i=1}^N d^{-l_i} \leq 1. \quad (3)$$

? The denoted number of keys l_i doesn't include the SEK because of the way the Kraft inequality is used.

By applying Lagrangian multiplier to this problem, we get the minimized i^{th} user's number of keys to be updated l_i^* and the minimized number of keys to be updated L^* per member deletion. They are:

$$l_i^* = \text{ceil}\{-\log_d p_i\}; \quad (4)$$

$$L^* = \text{ceil}\{H_d\}. \quad (5)$$

If we consider the SEK, Then

- $l_i^* = \text{ceil}\{-\log_d p_i\} + 1; \quad L^* = \text{ceil}\{H_d\} + 1.$
- The minimum average per member storage of total number of keys is $\text{ceil}\{H_d\} + 2$ because of the leaf key.

Discussions about the tree shaping:

- The relationship between the member deletion probability and the cardinality of the member's key set.

$$p_i < p_j \Rightarrow l_i > l_j;$$

p_i is the smallest $\Rightarrow l_i$ is the longest.

- The two longest key sequences have the same length.
- The two longest key sequences differ only in the leaf keys.
- The corresponding two members with the two longest key sequences have the least deletion probabilities.

Claim3: Let P_i and q_i denote the true and assigned deletion probabilities of member M_i , respectively. The average number of redundant keys to be assigned on the rooted-tree-based key distribution is $D(p||q)$, where

$$D_d(p \parallel q) = \sum_{i=1}^N p_i \log_d \frac{p_i}{q_i} \quad (6)$$

Therefore, the actual average number of keys to be updated can be expressed as:

$$\begin{aligned} L' &= \text{ceil}\left\{\sum_{i=1}^N p_i (-\log_d q_i)\right\} \\ &= \text{ceil}\{H_d + D_d(p \parallel q)\} \end{aligned} \quad (7)$$

Cover-free (Exclusivity)

Example 3:

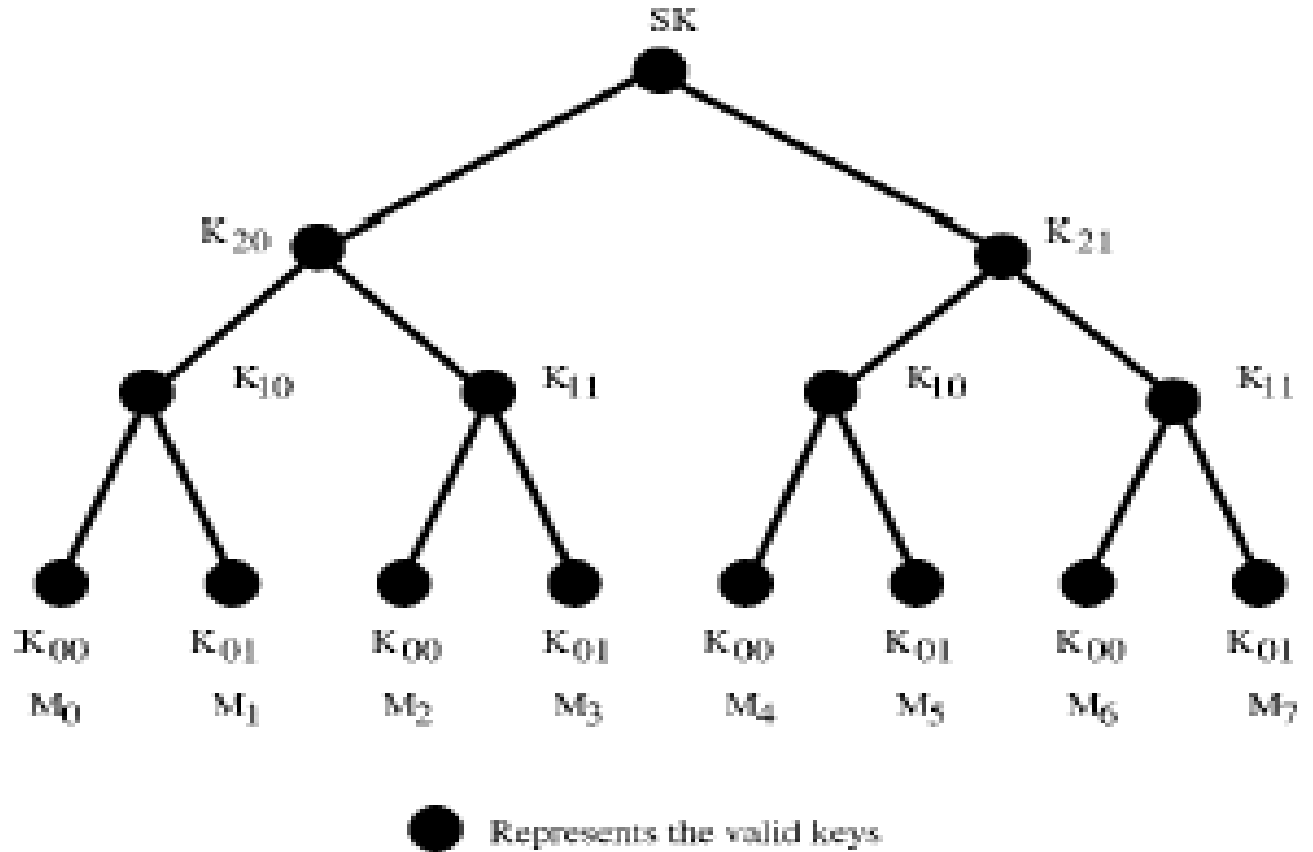


Fig.3. Key distribution in [9].
(uniform distribution)

Problem: If both user M_0 and user M_7 are deleted, the collusion of these two users can compromise the whole system
? the compromised users can't be securely removed.

Solution: To distribute the keys in a way to make the system cover-free.

- Definition: Given a collection of sets $\{S_1, \dots, S_N\}$, a nonempty set S_i is said to be $\{k, a\}$ cover-free if

$$\frac{\left| S_i \setminus \bigcup_{j=1; j \neq i}^k S_j \right|}{|S_i|} \geq a_i$$

where $0 \leq a_i \leq 1$, $1 \leq k \leq N$. When $a_i = 0$, there is no cover-free key distribution for key set S_i . For cover-free condition, $a > 0$.

- Sufficient condition to make the scheme cover free ? choose different keys for all the leaf nodes.

Design technique

--- using Huffman encoding algorithm

- Choose the optimal tree shape.
- Distribute the keys exclusively.

- Example 4 ---- We have 5 members with member deletion probabilities 0.15, 0.15, 0.2, 0.25, 0.25.

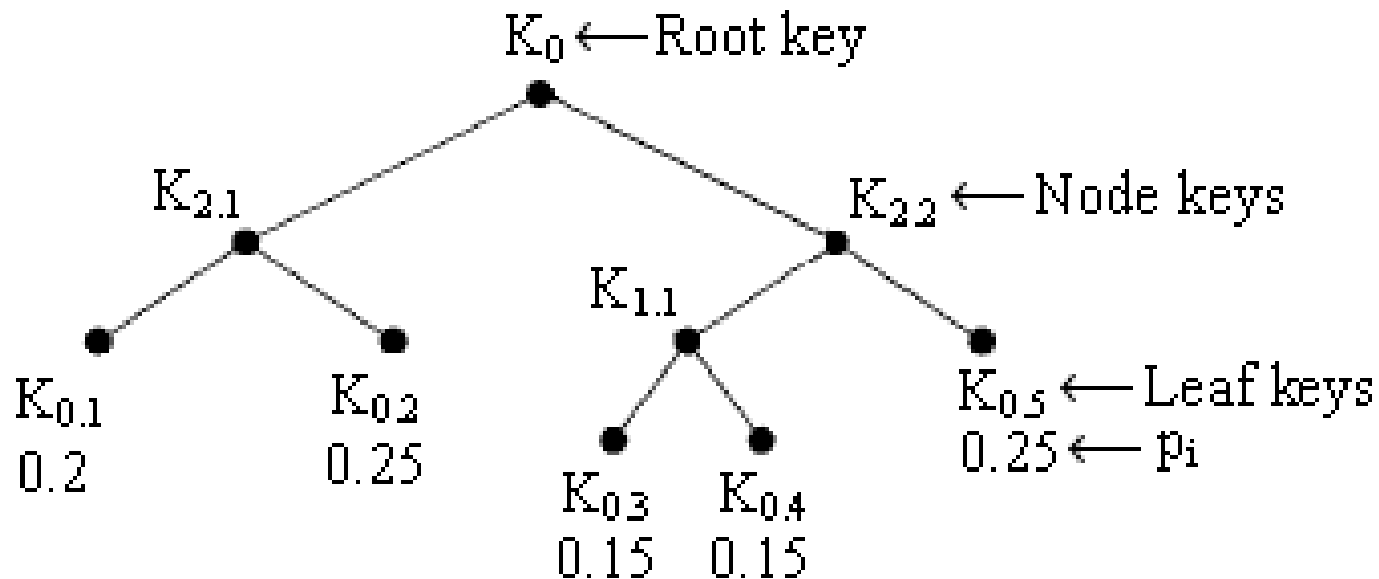


Fig. 4. Optimal key distribution scheme with 5 members.

- Ques: Can we choose any of the internal node keys to be the same?
- Ans: No, for the purpose of transmission efficiency.

- Example 5: --- We have eight members without knowing the member deletion probability distribution. In order to solve this problem, we assume all the members are equally likely to be deleted. (solution: see Fig. 4.)

Notes:

1. If the true member deletion probability is uniform distribution, the average number of keys to be updated achieves its maximum value among all the other distributions.
2. If the true member deletion probability for member M_i is p_i , the number of redundant keys on average is

$$D = \sum_{i=1}^N p_i \log \frac{p_i}{q_i} = \sum_{i=1}^N p_i \log N p_i.$$

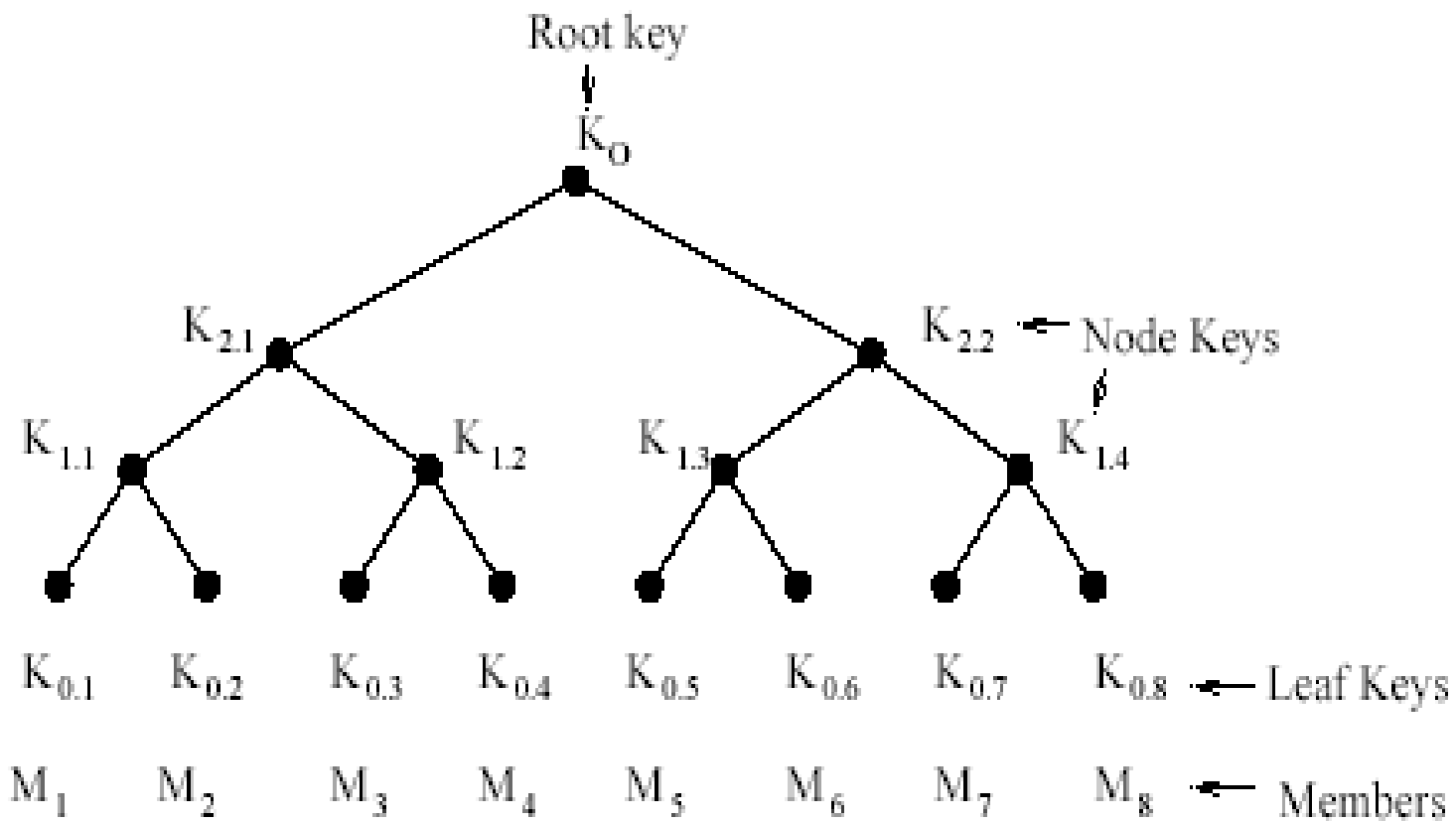


Fig. 5. Key distribution with eight members in [4].

Part III Conclusions

- Tree-based key management schemes can be mathematically analyzed using information theory.
- Tree-based key management schemes can be designed using Huffman coding algorithm to achieve secure member deletion, transmission efficiency, and storage efficiency.
- Wrong member deletion probability distribution introduces redundant keys, making the scheme suboptimal.

Contribution

As far as we know, we are the first to apply Huffman encoding technique to tree-based multicast key management scheme design.

References:

- [1] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Networking*, vol. 8, pp. 16–31.
- [2] D. Wallner, E. Harder, and R. Agee June Request for Comments: 2627 – Key Management for Multicast: Issues and Architectures 1999.
<http://www.faqs.org/rfcs/rfc2627.html>
- [3] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGGCOM'97*, Sept. 1997, pp. 277–288.
- [4] R. Poovendran, An Information Theoretic Approach for Design and Analysis of Rooted-Tree-Based Multicast Key Management Schemes, *IEEE Trans. Information Theory*, Vol. 47, pp.2824-2834, November 2001
- [5] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
- [6] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," in *Proc. Eurocrypt 99*, pp. 456–470. Feb. 2000.
- [7] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using Boolean function minimization techniques," in *Proc. IEEE INFOCOM'99*, pp. 689–698.
- [8] <http://www.cs.bu.edu/groups/aces/gkm/>
- [9] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: A Taxonomy and Efficient Authentication, *IEEE INFOCOM-99*.