# Trees for Group Key Management with Batch Update

Nan Zang

Ph. D. Defense

May 30th, 2008

# Outline

- Secure group key management overview
- Jumping sequence problem
  - Properties of optimal jumping sequences
  - Properties of optimal GKM trees
- Approximation algorithms to build GKM trees
  - The GLR Algorithm
  - The LR Algorithm (q → 1)
- Summary

# Introduction

- Secure group communication
  - A group of members (n)
  - Members in this group broadcast messages
  - The group is secure
  - Network applications
    - Pay-per-view video streaming, teleconferencing
- Group key (symmetric key)
  - All the messages broadcast by the members are encrypted by $K_g$
    - $K_g$ is shared and only shared by the members in the group.
  - Trusted central group key server (GC)
    - verify member
    - give each verified member a private key (a unique key for each member)
    - In charge of producing and maintaining the group key $K_g$.

# Introduction

- Security
  - Rekeying Process after each member joins or leaves

- Efficiency
  - Batch rekeying process ( ’01 Li, Yang, Gouda, Lam)
    - Reassign the keys after a period of time (several joins/leaves)
    - Less secure (set the length of the period based on application)
    - More efficient

# Introduction

- Group Key Management (GKM) Problem
  - Benefit from IP Multicast (Big Problem: Control the access)
  - Two Tasks
    - Identification and authorization
    - Maintain the group keys
  - A popular topic, but less theoretical analysis
- One specific model( ’03 Zhu, Chan, Noubir )
  - Network service has limited resources
  - Only can serve n members
  - Each member in a batch period has probability p of leaving
  - There are always members waiting to join

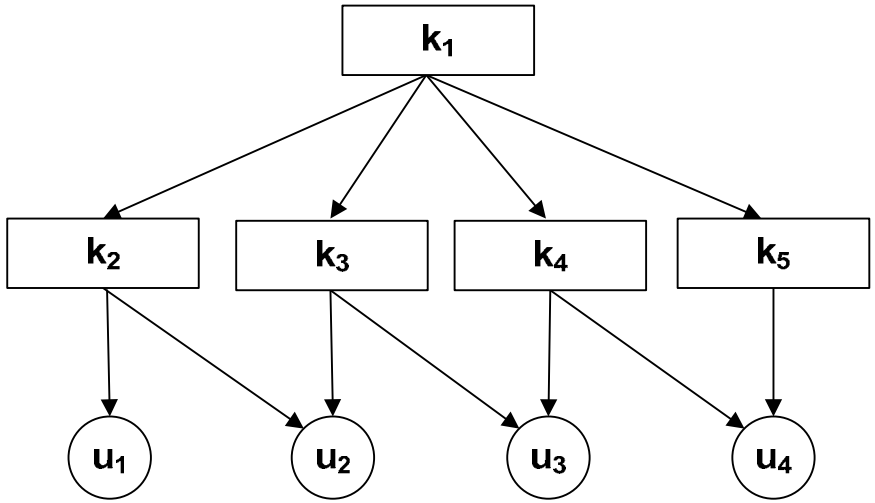
# Introduction

- Model definition
  - n members in the group
  - Each member has the same probability p of leaving.
  - The update cost = the expected number of rekeying-messages that have to be sent by the key server.
  - Task:

    Minimize the update cost in each batch period

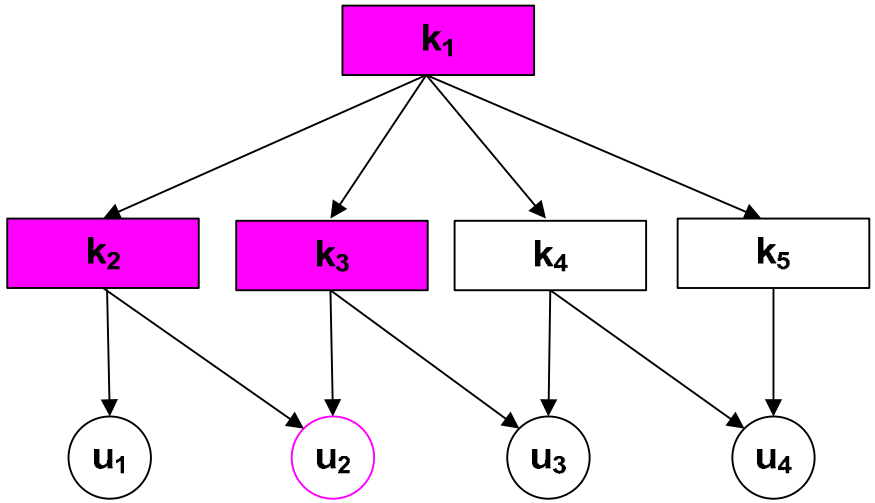
# Key graph

- Key graph ('98 Wong, Gouda, Lam)
  - DAG -- Directed Acyclic Graph
  - Key - user relation
    - Users (u-node): All the nodes with only incoming edges.
    - Keys (k-node): All the other nodes.
    - User $U_i$ contains Key $K_j$ ⇔ There is a directed path from $K_j$ to $U_i$.

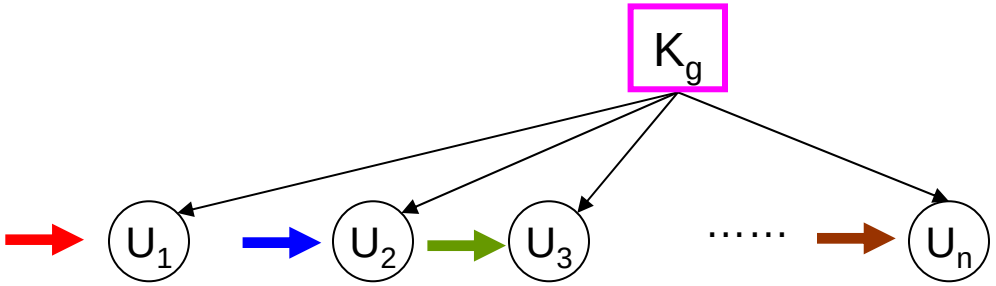


# Key graph

- Key graph ('98 Wong, Gouda, Lam)
  - DAG -- Directed Acyclic Graph
  - Key - user relation
    - Users (u-node): All the nodes with only incoming edges.
    - Keys (k-node): All the other nodes.
    - User $U_i$ contains Key $K_j$ ⇔ There is a directed path from $K_j$ to $U_i$.



# n-star tree example




$E_{Ki}\{K_g'\}$

$K_g'$ is the plain message.

$K_i$ is the encryption key.

- If the member $U_1$ changes, the group key $K_g$ has to be changed.
- $K_g$ is not secure. To tell every user the new $K_g$, n different messages have to be broadcast, each is encrypted new $K_g$ by user's private key $K_i$.
- Each member has the probability p of leaving.

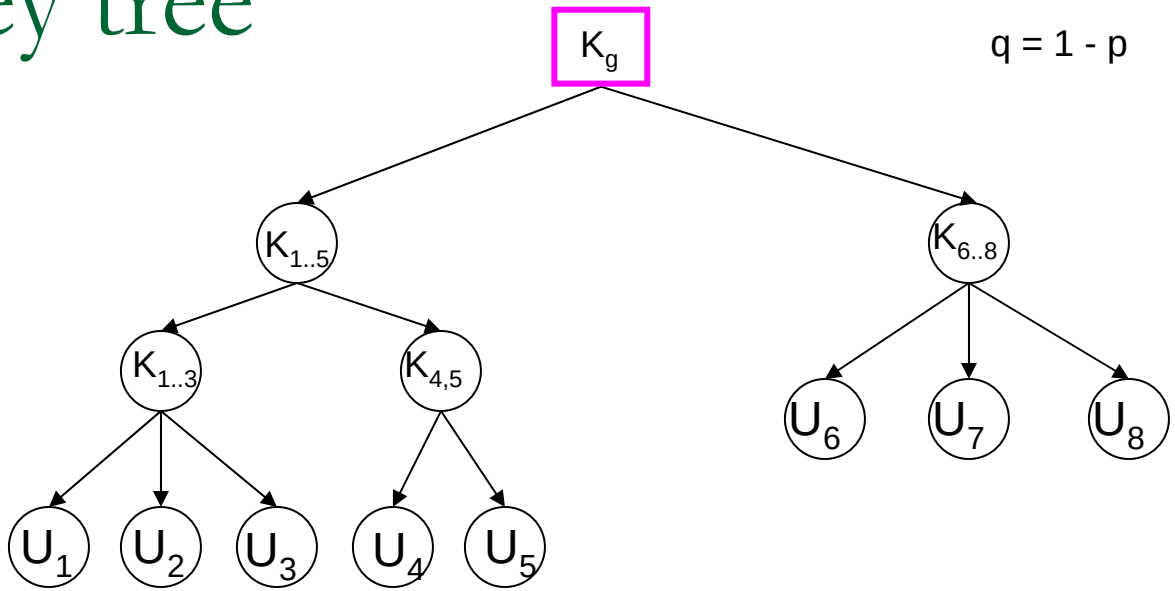
Probability ($K_g$ changes) = $1-(1-p)^n$ The cost of the n-star = $n(1-(1-p)^n)$

- Problem: No common keys other than $K_g$.
- Is there any better way to assign the keys to the users?

# Key tree

p: the prob. of each member changing

q = 1 - p




## Hierarchical key structure

users - keys

Manage users in subgroups

One key for each subgroup

## Key tree

leaves (users); internal nodes (keys)

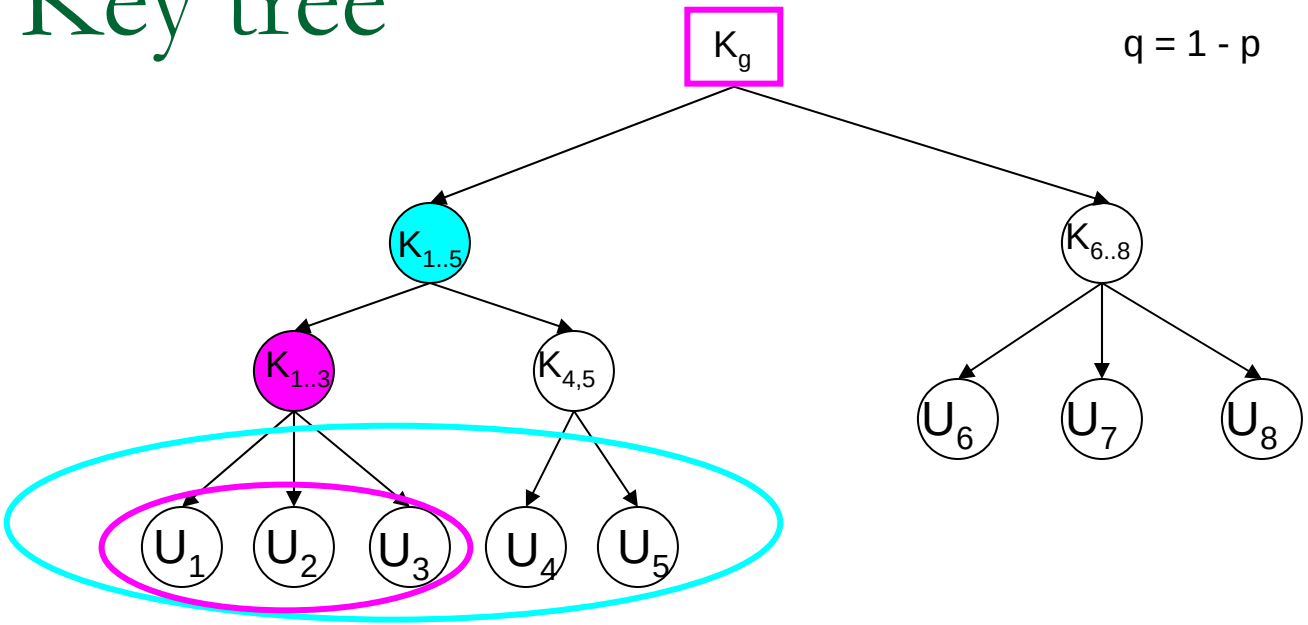
Subtrees

The root of each subtree

# Key tree

p: the prob. of each member changing

q = 1 - p




| Subgroup | Subgroup Key |
|---|---|
| $\{U_1, U_2, U_3\}$ | $K_{1..3}$ |
| $\{U_1, U_2, U_3, U_4, U_5\}$ | $K_{1..5}$ |

$K_g$: Group Key; Traffic Encryption Key (TEK)

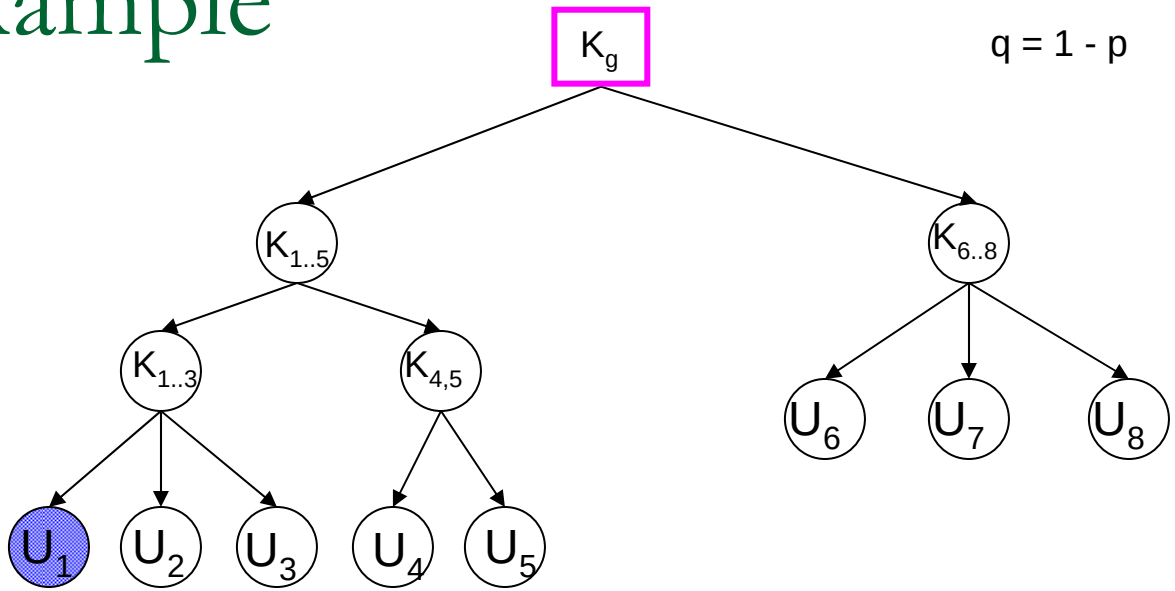
$K_{1..3}$, $K_{4..5}$, $K_{1..5}$, $K_{6..8}$ : Subgoup Keys; Key Encryption Key (KEK)

Subgroup key changes if and only if any member in this subgroup changes.

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

**If member $U_1$ changes, how to update the keys?**

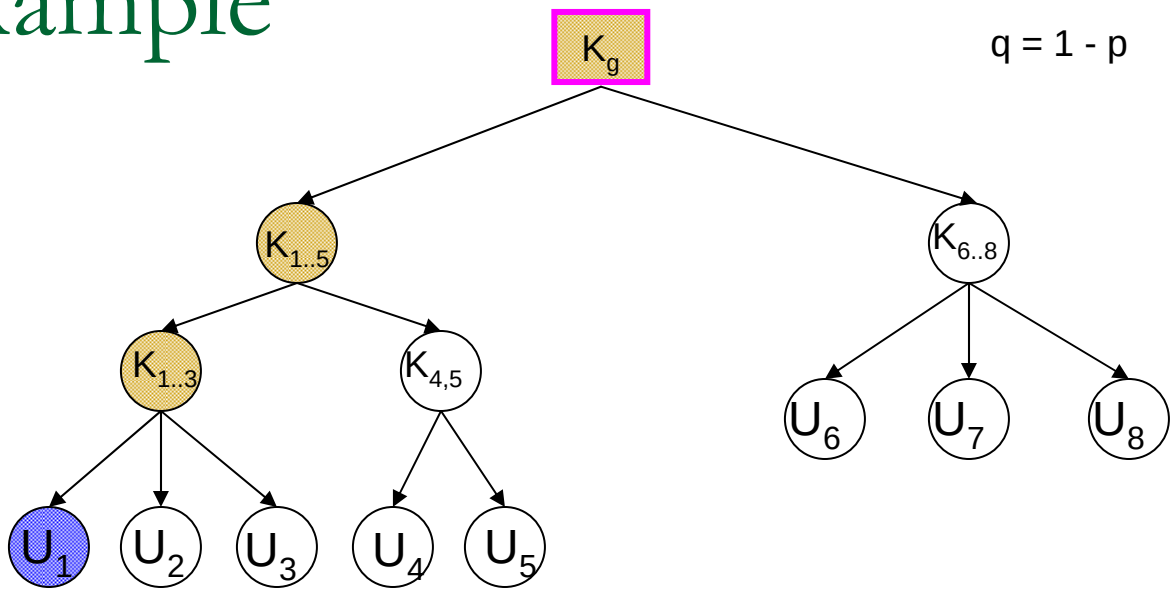
**1. Update all the keys $U_1$ holds.**

**2. Update the keys from a bottom-to-top order.**

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

**If member $U_1$ changes, how to update the keys?**

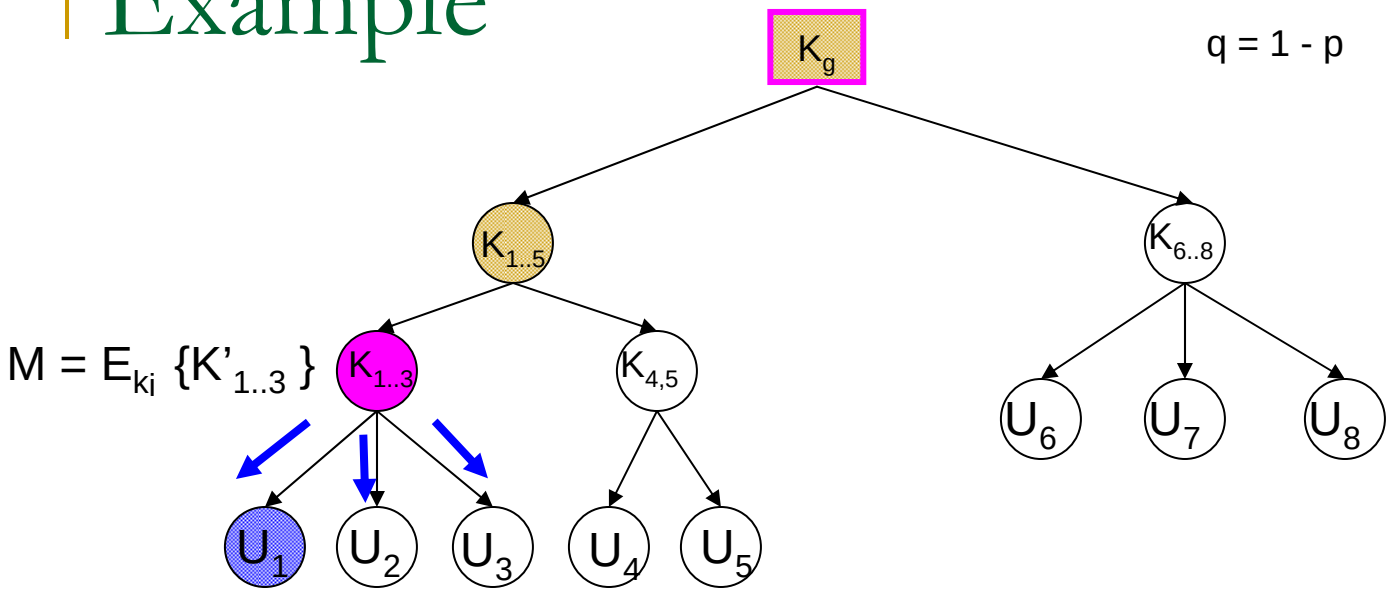
**1. Update all the keys $U_1$ holds: $\{K_{1..3}, K_{1..5}, K_g\}$**

**2. Update the keys from a bottom-to-top order.**

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

**If member $U_1$ changes, how to update the keys?**

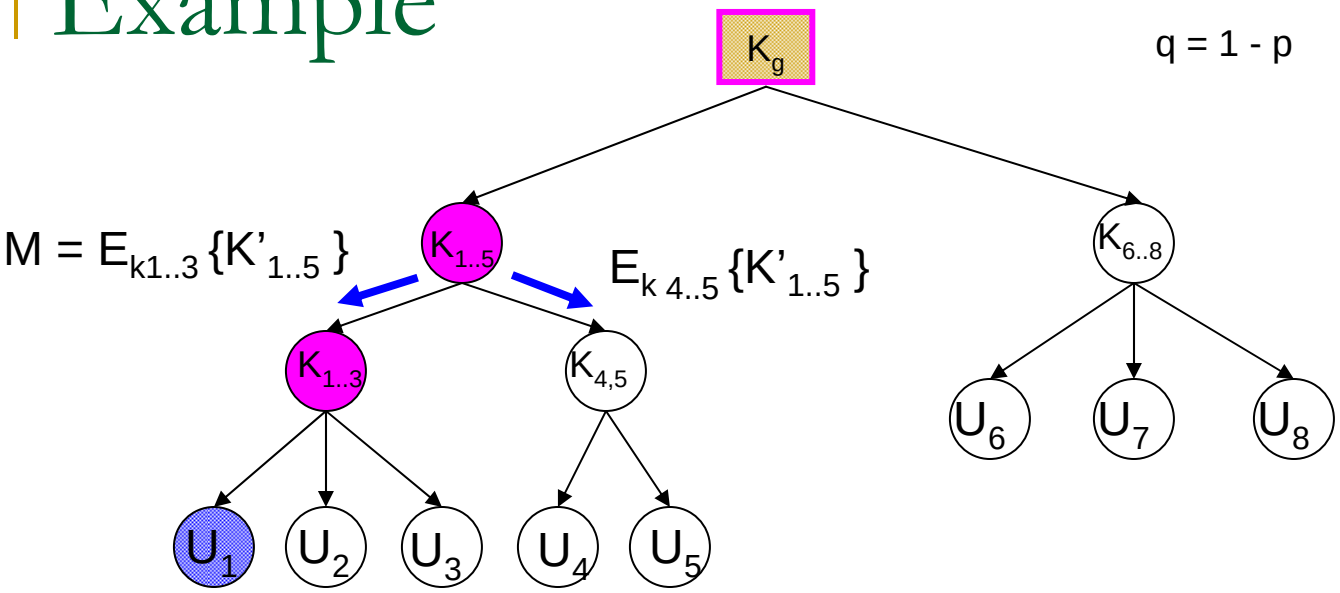
**1. Update all the keys $U_1$ holds: $\{K_{1..3}, K_{1..5}, K_g\}$**

**2. Update the keys from bottom to top.**

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

**If member $U_1$ changes, how to update the keys?**

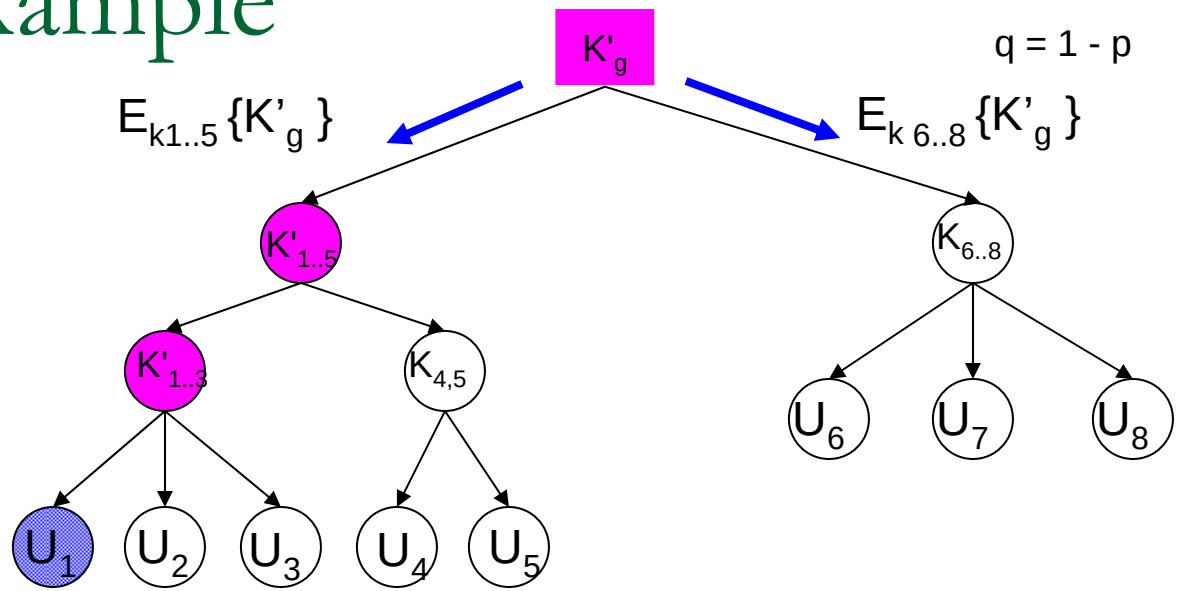
**1. Update all the keys $U_1$ holds: $\{K_{1..3}, K_{1..5}, K_g\}$**

**2. Update the keys from bottom to top.**

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

**If member $U_1$ changes, how to update the keys?**

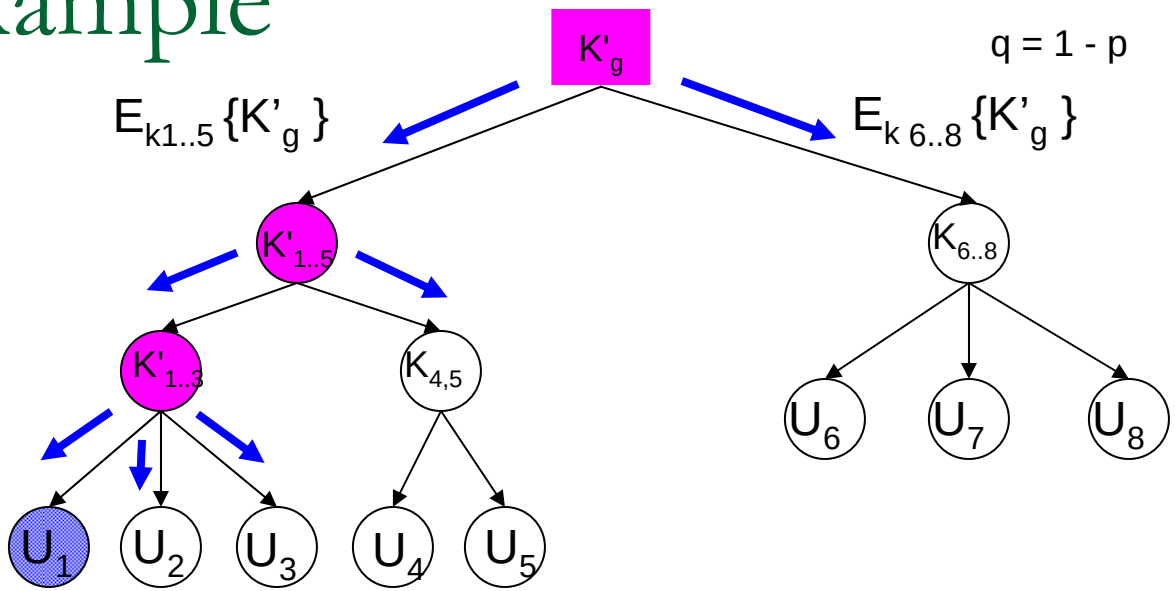
**1. Update all the keys $U_1$ holds: $\{K_{1..3}, K_{1..5}, K_g\}$**

**2. Update the keys from bottom to top.**

# Example

p: the prob. of each member changing

q = 1 - p




**Example:**

7 Messages have to be sent!

**If member $U_1$ changes, how to update the keys?**

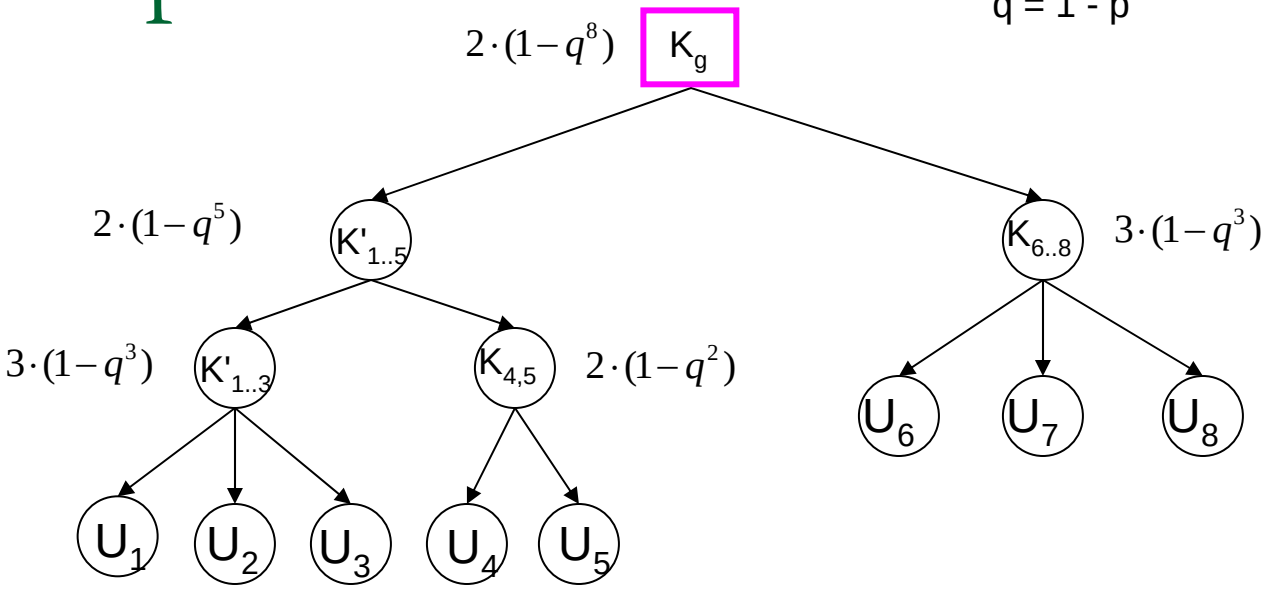
**1. Update all the keys $U_1$ holds: $\{K_{1..3}, K_{1..5}, K_g\}$**

**2. Update the keys from bottom to top.**

# Example

p: the prob. of each member changing

q = 1 - p




| Subgroup Key | Probability of Changing | No. of Messages |
|---|---|---|
| $K_{1..3}$ | $1 - (1 - p)^3 = 1 - q^3$ | 3 |
| $K_v$ | $1 - (1 - p)^{N(v)} = 1 - q^{N(v)}$ | d(v) |

$N_v$ is the number of leaves under the key node $K_v$.

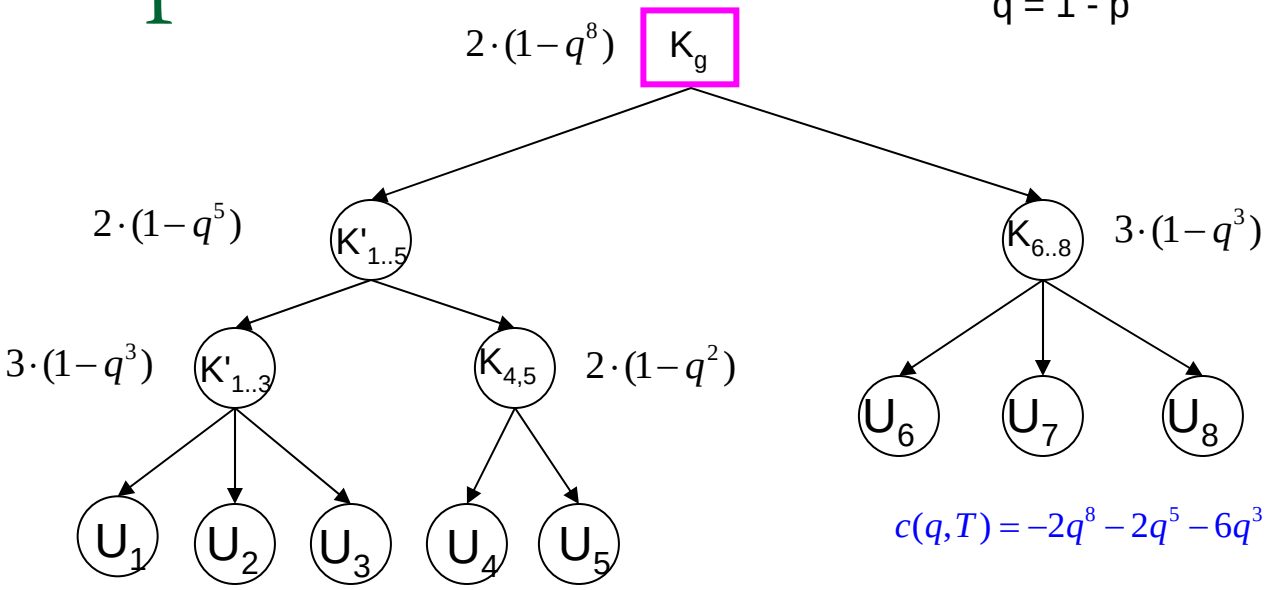
$d_v$ is the outgoing degree of the key node $K_v$.

For each $K_v$, Exp No. of Messages to update $K_v = d(v)\cdot(1 - q^{N(v)})$

# Example

p: the prob. of each member changing

q = 1 - p

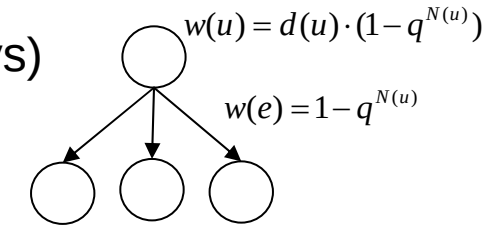



$$c(q,T) = -2q^8 - 2q^5 - 6q^3 - 2q^2 + 10$$

For any tree structure, assuming each leaf has probability **p** of changing.

Cost of the tree = Expected number of encrypted rekeying-messages.

(Expected number of encryptions needed to reassign the keys)




Cost of Tree: $C(q,T) = \sum_{u\in V} d(u)\cdot(1-q^{N(u)}) = \sum_{e\in E}(1-q^{L(e)})$

d(u) = degree of u; N(u) = number of leaves under u; for e = (u, v), L(e) =N(u)

# Mathematical Model

## Math Model

- Given n, and //n is the number of leaves
- q, where q = 1-p,
- cost of tree structure T

$$C(q,T) = \sum_{u \in V} d(u) \cdot (1 - q^{N(u)})$$

- Objective:

  Find a tree T with minimum cost among all the trees with n leaves.

## GKM Problem

- n Members in a group
- Each member changes with probability p.
- Expected number of rekeying messages to be sent.
- Objective:

  How to arrange the members to minimize the expected number of rekying messages.

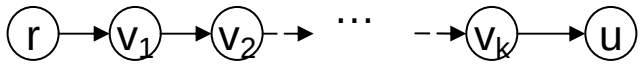
# Contributions

- Jumping sequence problem
  - Properties of optimal jumping sequences
  - Properties of optimal GKM trees
- Approximation algorithms to build GKM trees
  - Properties of optimal trees as $n \to \infty$
  - The GLR Algorithm
- GKM problem as q → 1
  - Properties of optimal jumping sequences as q → 1
  - The LR Algorithm

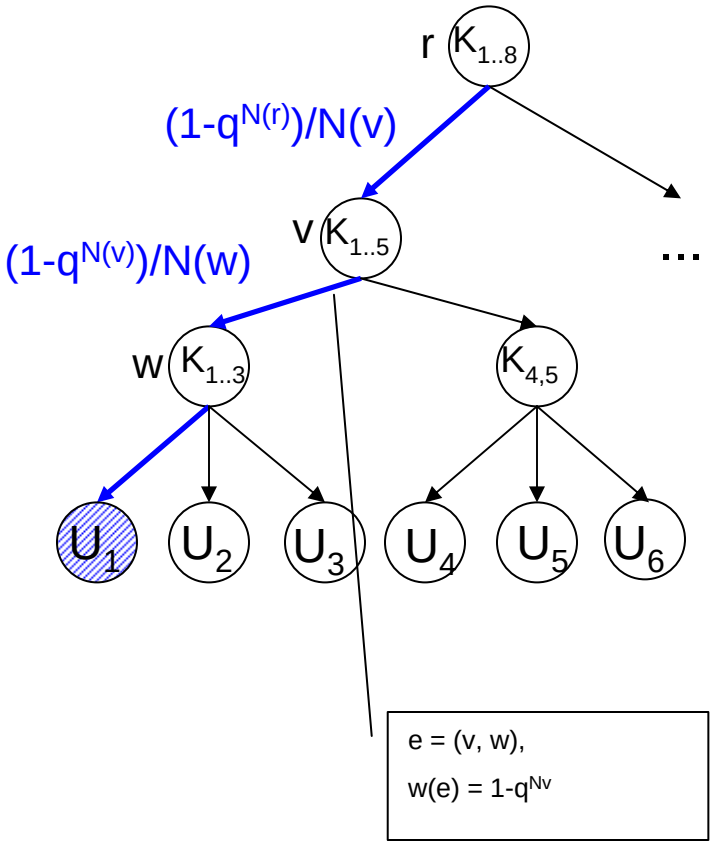
# Definitions of Jumping Sequences

- Definitions
  - Cost Per Leaf (CPL)
    - Cost of the tree $C(q,T) = \sum_{e \in E} (1 - q^{L(e)})$
    - Allocate the cost of the tree to each leaf
    - CPL= The total cost of the path from the leaf to the root.

Path(r, u): from r to u in the tree.



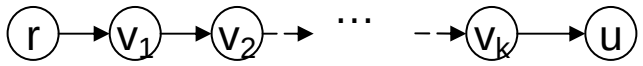

$$CPL(u) = \sum_{e(v,w) \in Path(r,u)} (1 - q^{N(v)}) / N(w)$$



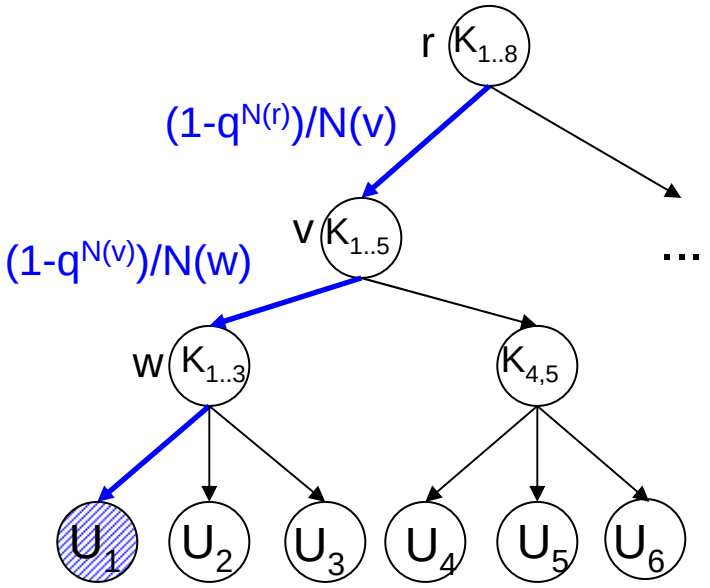
# Definitions of Jumping Sequences

- Definitions
  - Cost Per Leaf (CPL)
    - Cost of the tree $C(q,T)=\sum_{e\in E}(1-q^{L(e)})$
    - Allocate the cost of the tree to each leaf
    - CPL= The total cost of the path from the leaf to the root.

Path(r, u): from r to u in the tree.




$$CPL(u)=\sum_{e(v,w)\in Path(r,u)}(1-q^{N(v)})/N(w)$$

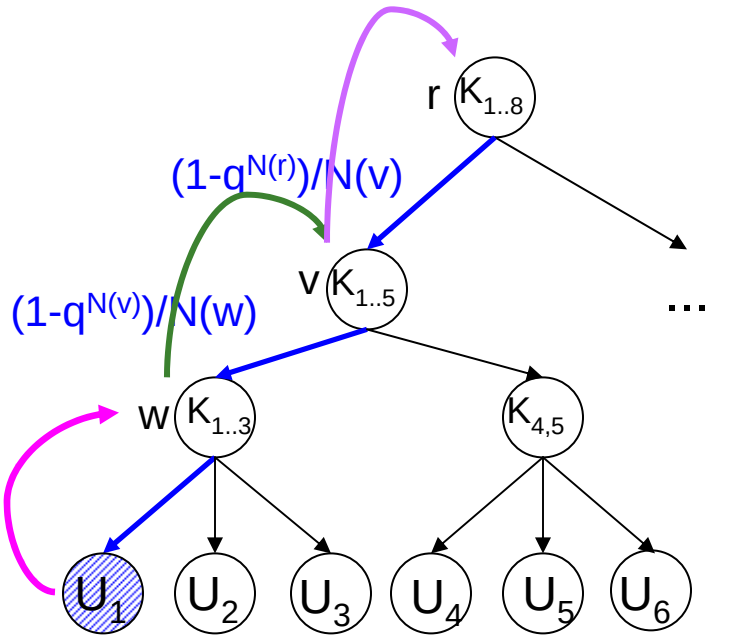



$$CPL(u_1)=1-q^3+\frac{1}{3}(1-q^6)+\frac{1}{6}(1-q^n)$$

# Definitions of Jumping Sequence

- The jump from a to b costs $(1-q^b)/a$.
- The jump starts with 1 (leaf).
- The jump ends with n (root).
- The jumps are along integers.
- In this Figure, the jumping sequence is

  $(1, N_w, N_v, N_r) = (1, 3, 6, n)$
- We can rewrite each path from the leaf to the root as a jumping sequence from 1 to n.




$$CPL(u_1) = 1-q^3 + \frac{1}{3}(1-q^6) + \frac{1}{6}(1-q^n)$$

# Definitions of Jumping Sequences

- Definition (Jumping Sequence Problem)
  - $S_n = (1 = a_0, a_1, a_2, \ldots, a_{k-1}, a_k = n)$, $1 = a_0 < a_1 < \ldots < a_k = n$, $a_i$ is a real number.
  - The jump from $a_{i-1}$ to $a_i$ costs $\frac{1 - q^{a_i}}{a_{i-1}}$
  - Cost of $S_n$ is: $c(q, S_n) = \sum_{i=1}^{k} \frac{1 - q^{a_i}}{a_{i-1}}$
  - Find the best sequences from 1 to n with the minimum cost.

Note: Jumping Sequence Problem is the real number version of GKM problem, and $opt(q, n) \geqslant n\, f(q, n)$.

f(q, n ) denotes the minimum cost from 1 to n and F(q, n) denotes the corresponding optimal real number sequences. .

OPT(q, n ) denotes the optimal GKM tree and opt(q, n) denotes the minimum cost.

# Properties of Jumping Sequences

- Theorem

  An optimal jumping sequence from 1 to n, $S^*_n = (1 = a_0, a_1, a_2, \ldots, a_{k-1}, a_k = n)$ has the following properties:

  - All the ratios $a_{i+1}/a_i$, except the last one , are bounded
  - $\sqrt{2} \le \frac{a_{i+1}}{a_i} < 4.75$ (i = 0, 1, …, k-2)
  - If $n \leqslant 2$ or $q < e^{-1/e,}$ the optimal jump sequence is only from 1 to n.
  - If $n \geqslant 5$ and q > 0.9, there is at least one intermediate jump from 1 to n

# Properties of Jumping Sequences

- Proof Outline
  - Recursive Property:
    - If $1 < a_1 < a_2 < \ldots < a_{k-1} < n$ is an optimal jump sequence for q and n, then for any $1 \leqslant i \leqslant k$, the sequence $1 < a_{i+1}/a_i < a_{i+2}/a_i < \ldots < n/a_i$ is an optimal jump sequence for $q' = q^{a_i}$ and $n' = n/a_i$.
  - If an optimal jump sequences starts $1 < b \ldots < n$, then $b < 4.75$.
  - If an optimal jump sequence starts $1 < a < b \ldots < n$, then $a \geq \sqrt{2}$, and $b \geqslant 2$.

# Properties of Jumping Sequences

- If n $\geqslant$ 2, then

$$f(q,n) \geq 1 - q^{\sqrt{2}}, \text{ and } opt(q,\ n)\ \geq n \cdot (1 - q^{\sqrt{2}}).$$

f(q, n ) denotes the minimum cost from 1 to n.

opt(q, n ) denotes the cost of the optimal GKM tree.

- For any n, k is the number of jumps from 1 to n, then

$$k \geq \min(0.64\ln(n) - 0.04, -0.64\ln(\ln\frac{1}{q}) - 0.45),$$

$$k \leq \min(2.89\ln(n), -2.89\ln(\ln(\frac{1}{q})) - 0.89).$$

# Properties of optimal GKM trees

- For any n, k is the number of jumps from 1 to n, then

$$k \geq \min(0.64\ln(n) - 0.04, -0.64\ln(\ln\frac{1}{q}) - 0.45),$$
$$k \leq \min(2.89\ln(n), -2.89\ln(\ln(\frac{1}{q})) - 0.89).$$

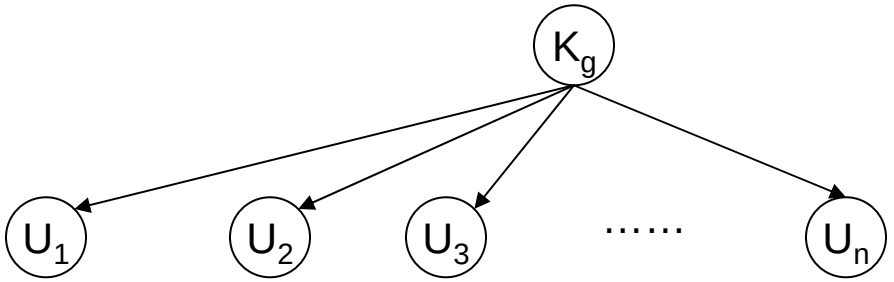
  - As n → $\infty$, the number of jumps (the number of levels of an optimal GKM tree) is bounded by O(ln(1/ln(1/q))).
  - As q → $1$, the number of jumps (the number of levels of an optimal GKM tree) is bounded by O(ln n).

# Outline

- Secure group key management overview
- Jumping sequence problem
  - Properties of optimal jumping sequences
  - Properties of optimal GKM trees
- Approximation algorithms to build GKM trees
  - The GLR Algorithm
  - The LR Algorithm (q → 1)
- Summary

# Related work ('07 Graham, Li, Yao)

- Optimal GKM trees
  - When $0 \leqslant q < 3^{-1/3}$, the n-star is strictly better than any other tree structure. ( $3^{-1/3} \approx 0.69336$, q = 1-p)




The cost of an n-star:

$$n \cdot (1 - q^n)$$

- We only consider $3^{-1/3} \leqslant q \leqslant 1$.

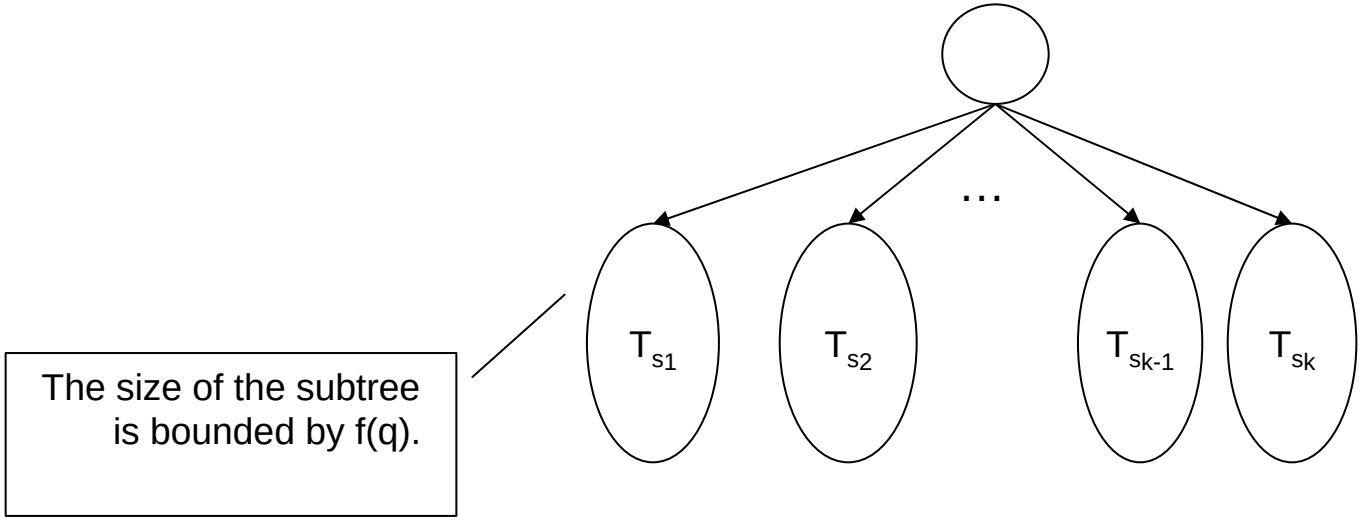
# Idea of GLR Algorithm

- First, we show there is always a “good” and “symmetric” subtree $T_{DS}$ connected to the root, when n is large.
  - The size of the $T_{DS}$ is bounded;
  - The structure of $T_{DS}$ only depends on q.
- Second, for a fixed n and q, to build an approximate GKM tree structure, we use $T_{DS}$ as often as possible.
- Computer simulation shows GLR is efficient. The approximation ratio of GLR is still open.

# Properties of optimal GKM trees

## Property (’07 Graham, Li, Yao)

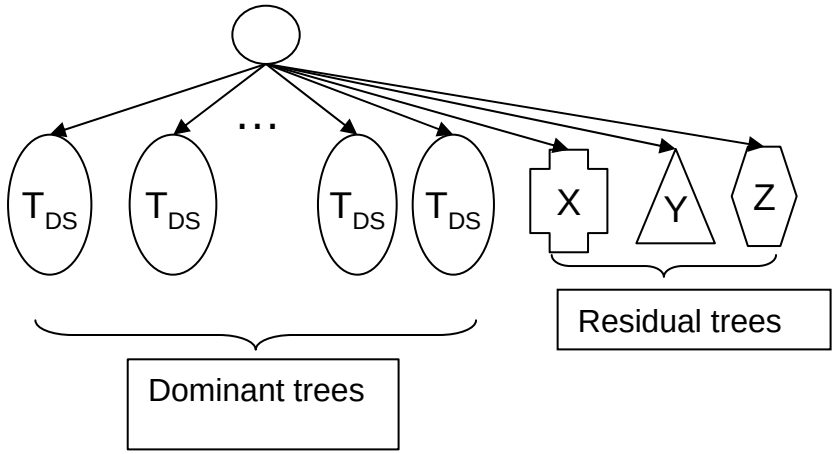
- As n → ∞, the size of each subtree connected to the root is bounded by max{ 4 ( log (1/q)) -1,1}
- Consistent with my results: As n → ∞, the number of jumps is bounded by O(ln(1/ln(1/q))), and each jump is bounded by 9/4.
- As n → ∞, the degree of the root is unbounded.



# Properties of optimal GKM trees

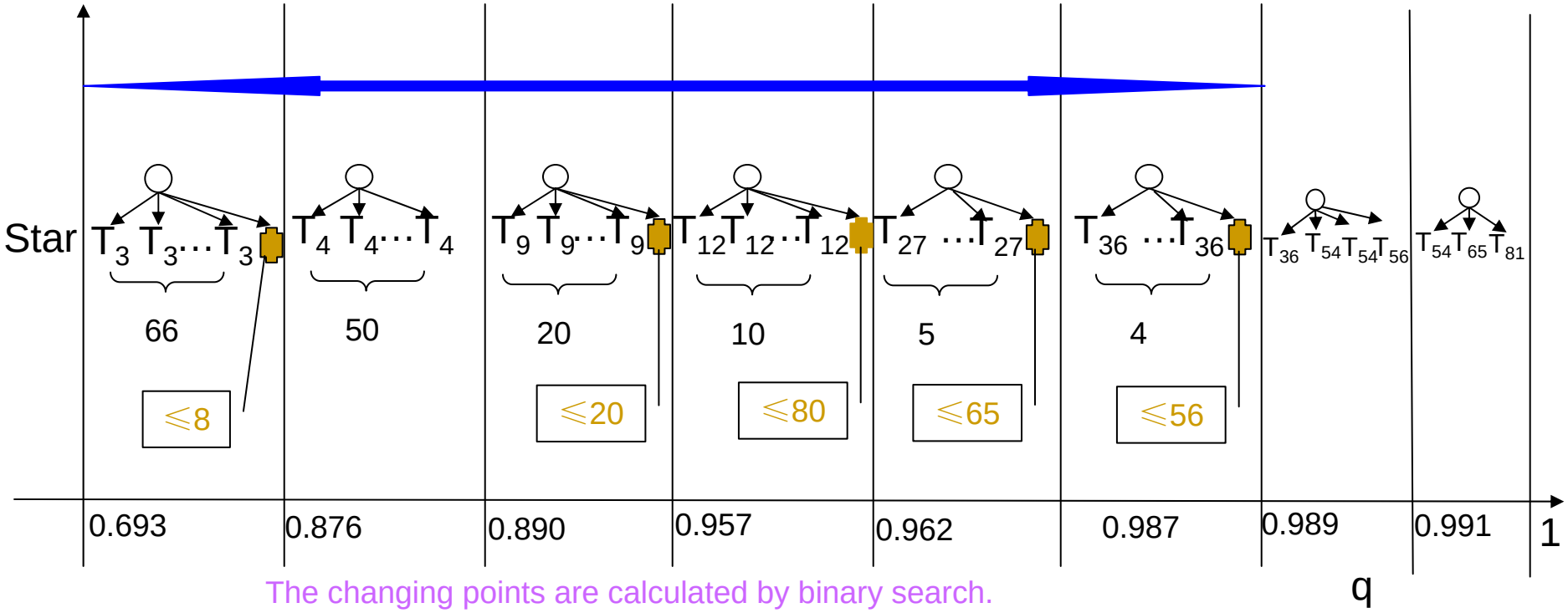
## Lemma

- If q is constant, as $n \to \infty$ , in the optimal tree opt(q,n),
  - The root has an unbounded number of subtrees;
  - But these subtrees only have a constant number of structures.
  - There must be at least one subtree T which gives the minimum average CPL (cost per leaf) value.
  - We call this subtree structure the **dominant subtree**.
  - We call other subtree structures the **residual subtrees**.

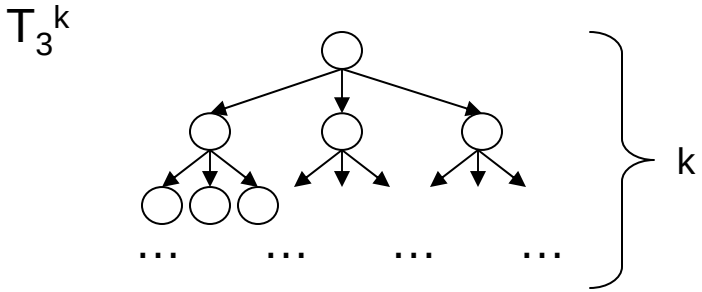


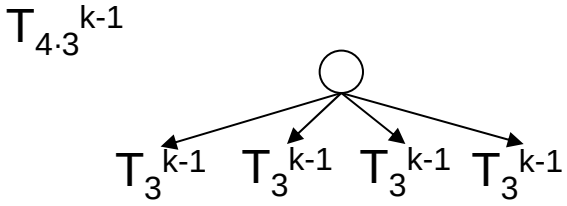
# Experimental results

Opt tree structures (n=200)



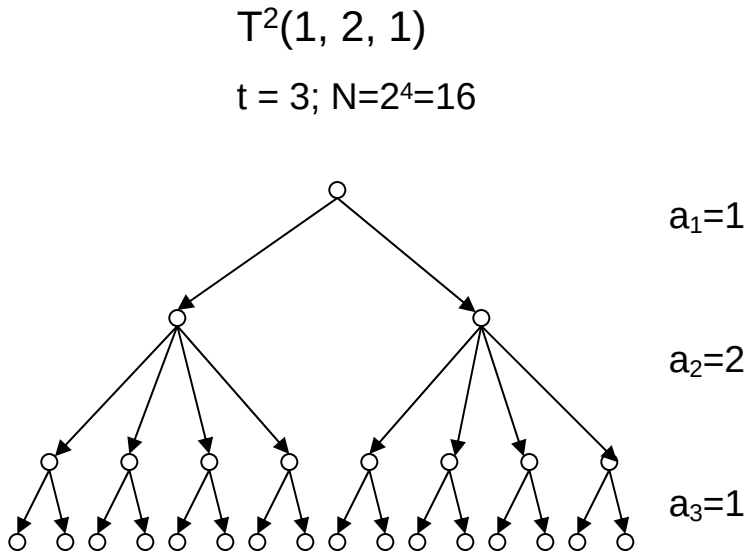

The changing points are calculated by binary search.



# Related work ('03 Zhu, Chan, Noubir)

- A special set of key trees
  - The number of members $N = 2^k$
  - $T^2(a_1, a_2, \ldots, a_t)$
    - A tree has t levels
    - The node at the i-th level has $2^{ai}$ branches

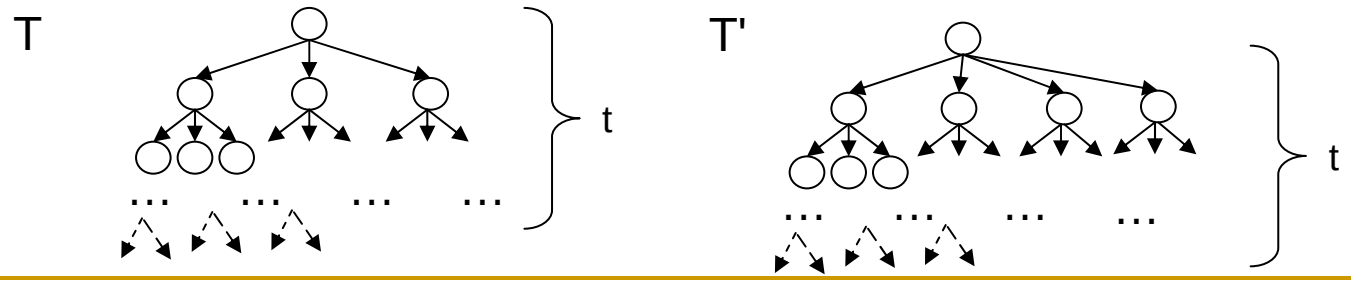



- Main results
  - The optimal tree must be a star; or, can be written as $T^2(a_1, 2, \ldots, 2, a_t)$, where $a_1 \geqslant 2$, and $a_t = 1$ or 2.
    - Almost a 4-ary tree
    - The bottom level internal nodes can have outdegree 2 or 4.

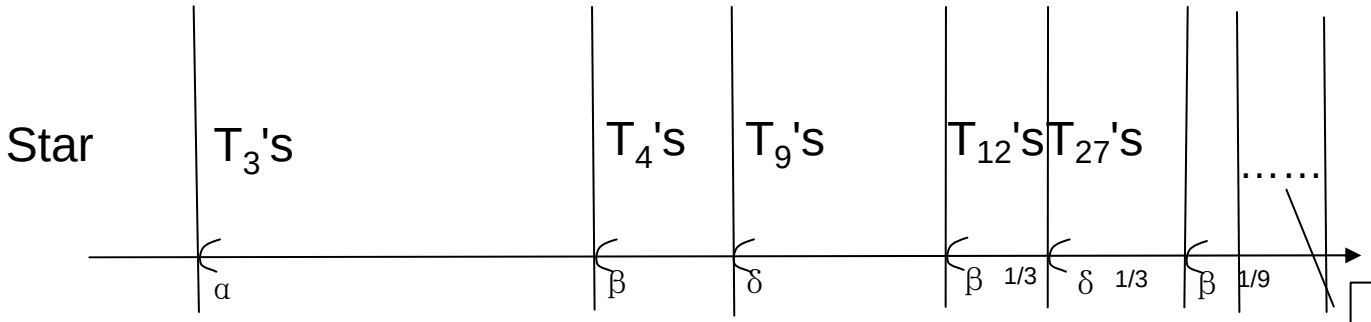
# Definition of Uniform Property

- Uniform Property
  - All nodes at the same level have the same degree.
  - The tree can be written of the form T($a_1$, $a_2$, …, $a_t$)
    - The tree has t levels.
    - The node at the i-th level has outdegree $a_i$.
- Theorem

As $n > n_0(q)$, if the dominant subtree has the Uniform Property, then the dominant subtree can be written as
T((4), 3, …, 3, (2)). [(x) means the level might be missing]



# How to find the best dominant tree




$T_3^k$, $T_{4\cdot3}^{k-1}$ will come out alternately for k=1,2…..

Jump points: $\alpha = 1/3^{(1/3)} \approx 0.6934$; $\beta \approx 0.8760$ ; $\delta \approx 0.8902$.

$\beta$ is the root of $x^4-x^3+1/12=0$; $\delta$ is the root of $1/3x^9+x^4-x^3+7/36=0$

$\beta$ and $\delta$ are both in $[1/3^{1/3}, 1]$.

$T_4$ is a subtree with 4 branches, $T_9$ is a complete ternary tree.

$T_3^k$ is a complete ternary tree.

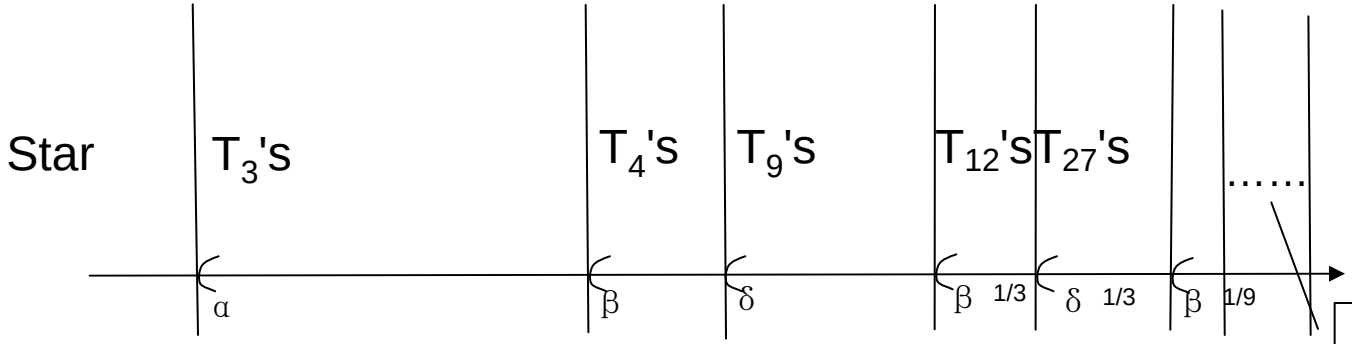
$T_{4\cdot3}^{k-1}$ is a subtree with 4 branches on the first level, the degree of the nodes on the other level is 3.

## Theorem

Among the trees of the form T((4), 3, 3,…, 3):

1. If q is in ( $\alpha$ , $\beta$ ], $T_3$ is the best.
2. If q is in ( $\beta^{1/3^k}$, $\delta^{1/3^k}$], $T_{4\cdot3k}$ is the best dominant subtree. (k = 0, 1, 2, …)
3. If q is in ( $\delta^{1/3^k}$, $\beta^{1/3^{k+1}}$], $T_{3k+2}$ is the best dominant subtree. (k = 0, 1, 2, …)

# How to find the best dominant tree




$T_3^k$, $T_{4\cdot3}^{k-1}$ will come out alternately for k=1,2…..

How can we find the dominant subtree for given q?

1. Given q, find the first candidate dominant tree $T_1 = T(a_{11}, 3, ..., 3)$.
2. Given $q^2$, find the corresponding dominant tree $T_2 = T(a_{12}, 3, ..., 3)$,

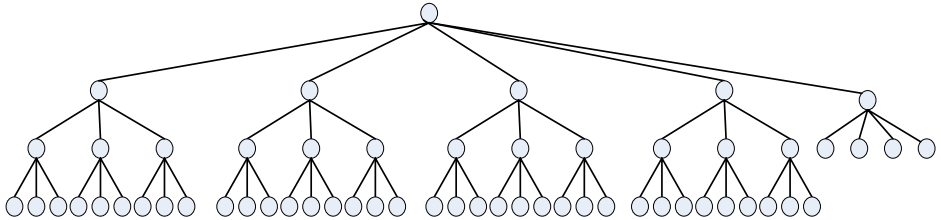
   $T_2' = T(a_{12}, 3, 3, ..., 3, 2)$;
3. Compare the cost of $T_1$ and $T_2'$, and find the dominant subtree.

# GLR Algorithm

- Given: q, n
- Output: An approximate GKM tree GLR(q, n)
- Algorithm
  - Find the best dominant tree $T_{DS}$ among the trees with the Uniform Property.
    - If the number of leaves $T_{DS}$ $|L(T_{DS})| > n$, then choose the $T_{DS}$ in the form of T((4), 3, …, 3) with the biggest number of leaves with $|T_{DS}| \leqslant n$.
  - GLR(q, n) contains $n_1+1$ subtrees:
    - $n_1$ of which are $T_{DS}$'s
    - one of which is GLR(q, L1) // $n_1 = \text{floor}(n / |T_{DS}|)$, $L_1 = n \bmod |T_{DS}|$.

# Example (GLR)

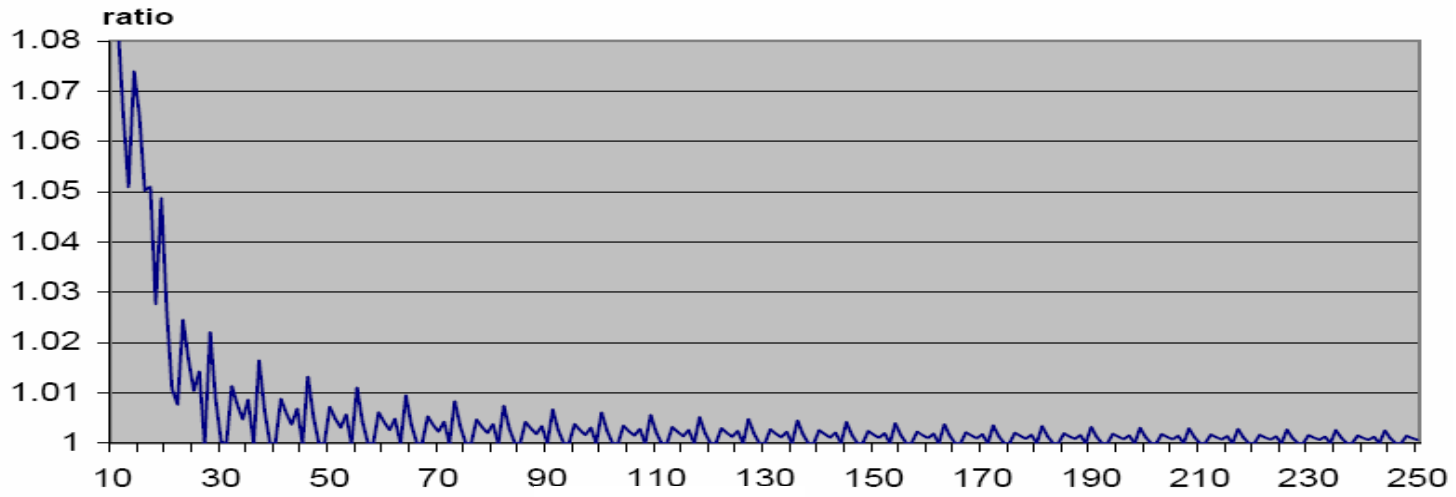
- q = 0.9; n = 40.
- Determine the structure of $T_{DS}$ by the value of q
  - The best dominant tree $T_{DS}$ is $T_9$, when q = 0.9.
  - $n_1$ = floor(40/9) = 4; $L_1$ = 40 mod 9 = 4.



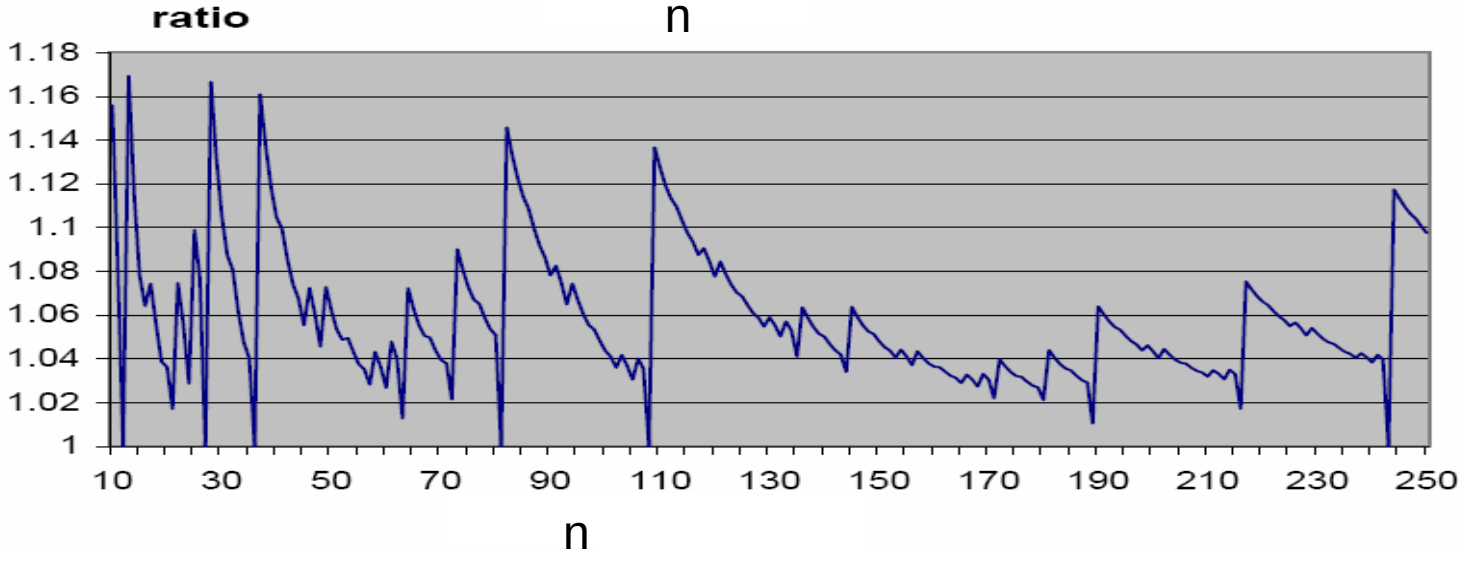
GLR(.9, 40)

# Simulation Results of Ratio glr(q, n)/opt(q, n)

q = .9




q = .999

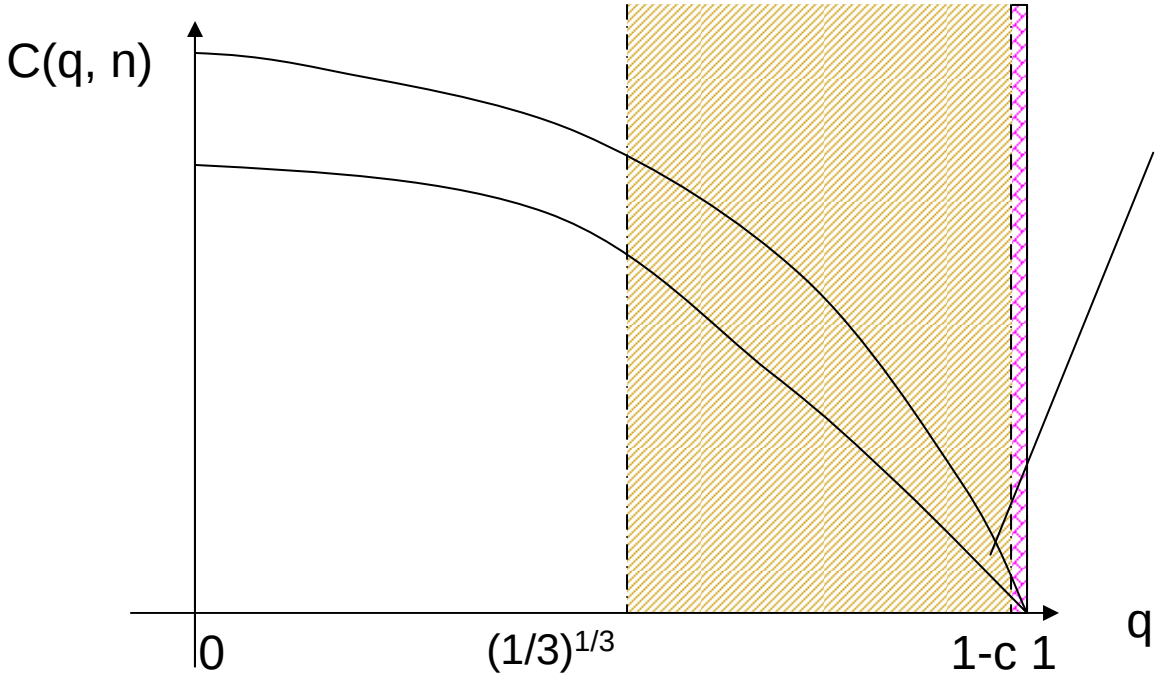


# Outline

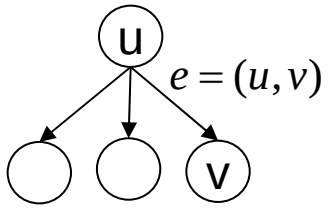
- Secure group key management overview
- Jumping sequence problem
  - Properties of optimal jumping sequences
  - Properties of optimal GKM trees
- Approximation algorithms to build GKM trees
  - The GLR Algorithm
  - The LR Algorithm (q → 1)
- Summary

# GKM tree problem as q → 1

- Cost of a GKM Tree T: $C(q,T)=\sum_{e\in E}(1-q^{L(e)})$




$$C'(q,T)\big|_{q=1}=-\sum_{e\in E}L(e)$$




$e=(u,v)$

$w(e)=1-q^{L(e)}$

$L(e)=N(u)$

The opt GKM tree is the tree with min $\sum_{e\in E}L(e)$

# Related work ('07 Graham, Li, Yao)

- Limiting case: $q \to 1$;
- The optimal tree structure T with n leaves, which minimize the cost function

$$C'(q,T)|_{q=1} = -\sum_{e \in E} L(e)$$

- Theorem

As $q \to 1$, if T*(n) is the optimal tree structure with n leaves, then:

- if n = 2, T*(2) has degree 2;
- if $n = 3^t$, T*(n) is a complete ternary tree;
- if $n = 4 \cdot 3^t$, T*(n) is of the form $T(4, \underbrace{3, 3, \ldots, 3}_{t})$.
- Otherwise, T*(n) has root degree 3.

# Idea of LR Algorithm

- As q → 1, the level of an optimal GKM tree is bounded by O(ln n).
- $T_{DS}$ is almost a complete ternary tree, except possibly the top and the bottom levels.
- As q → 1, if $n = 4 \cdot 3^t$, the optimal GKM tree T*(n) has root degree 4; otherwise, T*(n) has root degree 3.
- We construct an almost balanced ternary by adding new leaves from a top-down and left-to-right order.

# LR Algorithm

- LR (n) //Building approximate GKM trees as q→ 1
  - An Almost ternary tree
  - Fill the bottom level of the tree from left to right order
  - Approximation Ratio of LR(n) is $2.01(1+\frac{1}{\lfloor \log_3 n \rfloor})$

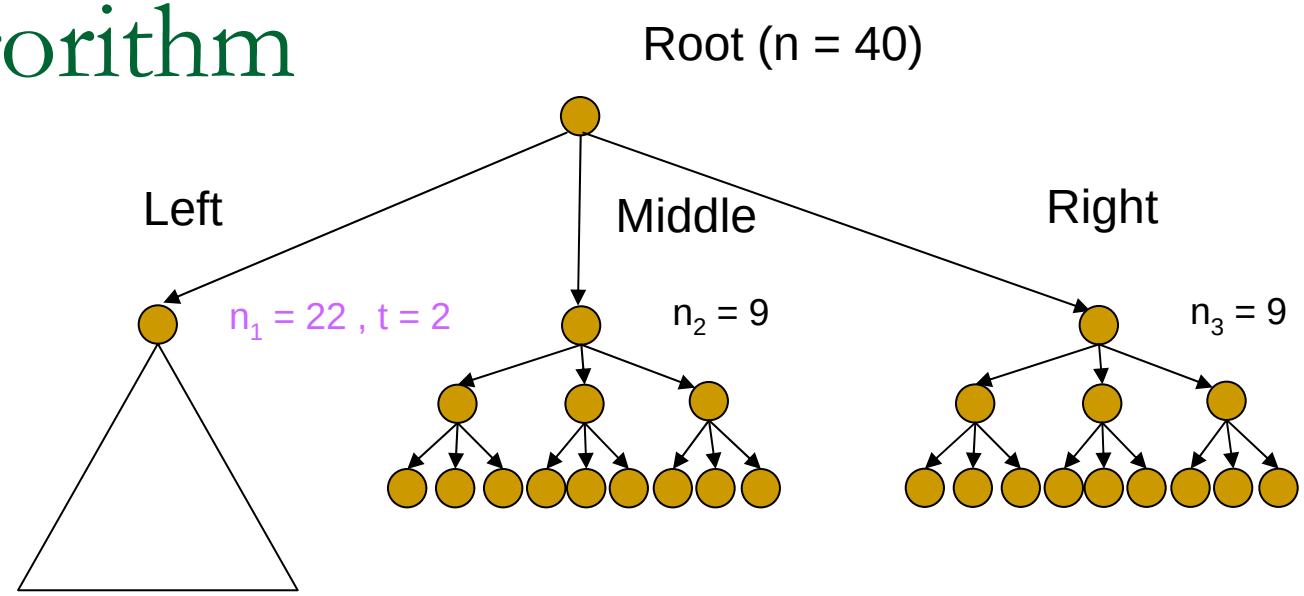
| | Leaves (Left) | Leaves (Middle) | Leaves (Right) |
|---|---|---|---|
| $3^t \leqslant n < 5 \cdot 3^{t-1}$ | $n - 2 \cdot 3^{t-1}$ | $3^{t-1}$ | $3^{t-1}$ |
| $5 \cdot 3^{t-1} \leqslant n < 7 \cdot 3^{t-1}$ | $3^t$ | $n - 4 \cdot 3^{t-1}$ | $3^{t-1}$ |
| $7 \cdot 3^{t-1} \leqslant n < 3^{t+1}$ | $3^t$ | $3^t$ | $n - 2 \cdot 3^t$ |

# LR Algorithm

Root (n = 40, t = 3)

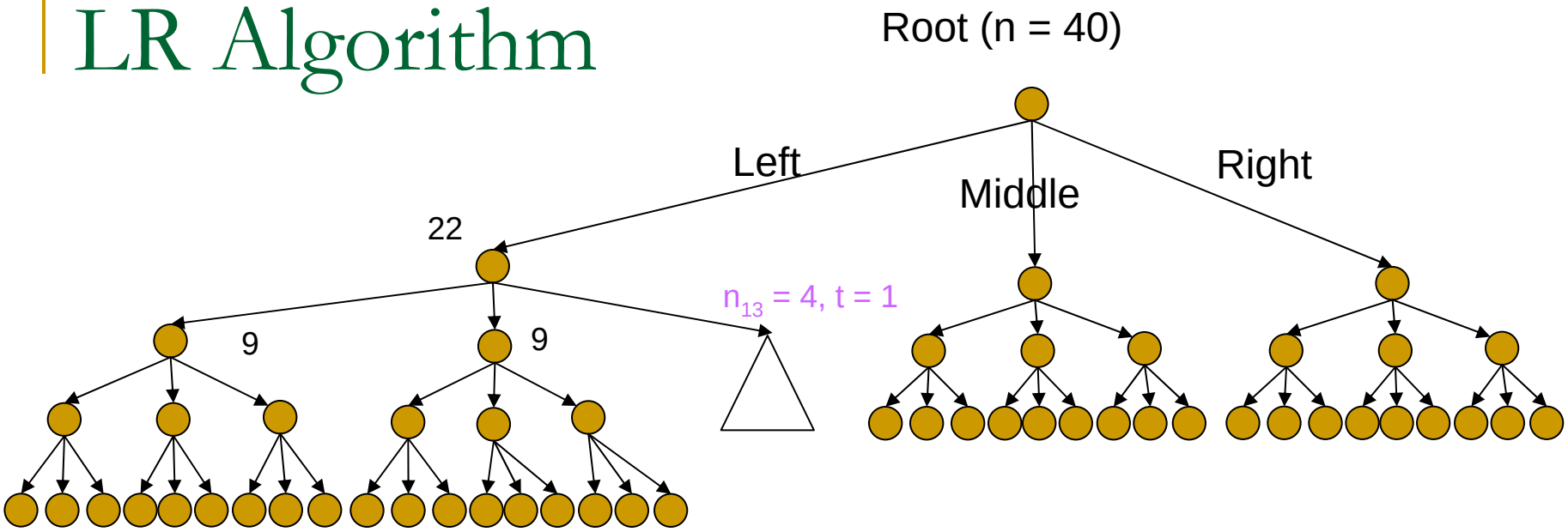
| | Leaves (Left) | Leaves (Middle) | Leaves (Right) |
|---|---|---|---|
| $3^t \leqslant n < 5 \cdot 3^{t-1}$ | $n - 2 \cdot 3^{t-1}$ | $3^{t-1}$ | $3^{t-1}$ |
| $5 \cdot 3^{t-1} \leqslant n < 7 \cdot 3^{t-1}$ | $3^t$ | $n - 4 \cdot 3^{t-1}$ | $3^{t-1}$ |
| $7 \cdot 3^{t-1} \leqslant n < 3^{t+1}$ | $3^t$ | $3^t$ | $n - 2 \cdot 3^t$ |

# LR Algorithm



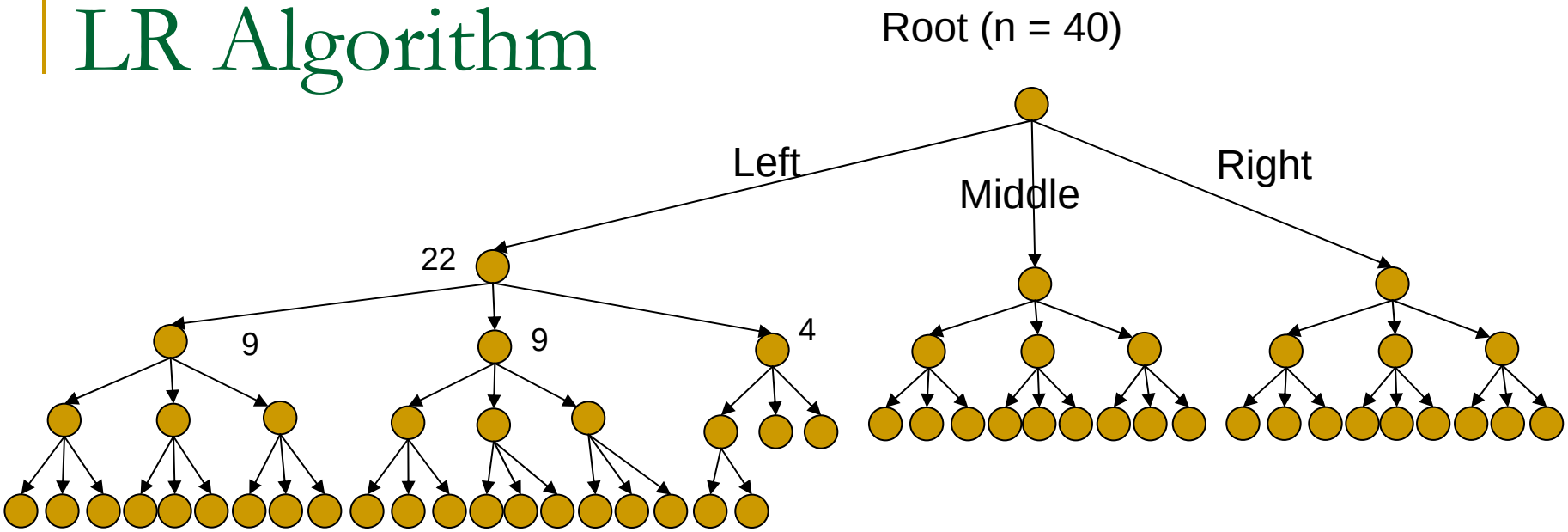

| | Leaves (Left) | Leaves (Middle) | Leaves (Right) |
|---|---|---|---|
| $3^t \leqslant n < 5 \cdot 3^{t-1}$ | $n - 2 \cdot 3^{t-1}$ | $3^{t-1}$ | $3^{t-1}$ |
| $5 \cdot 3^{t-1} \leqslant n < 7 \cdot 3^{t-1}$ | $3^t$ | $n - 4 \cdot 3^{t-1}$ | $3^{t-1}$ |
| $7 \cdot 3^t \leqslant n < 3^{t+1}$ | $3^t$ | $3^t$ | $n - 2 \cdot 3^t$ |

# LR Algorithm




| | Leaves (Left) | Leaves (Middle) | Leaves (Right) |
|---|---|---|---|
| $3^t \leqslant n < 5 \cdot 3^{t-1}$ | $n - 2 \cdot 3^{t-1}$ | $3^{t-1}$ | $3^{t-1}$ |
| $5 \cdot 3^{t-1} \leqslant n < 7 \cdot 3^{t-1}$ | $3^t$ | $n - 4 \cdot 3^{t-1}$ | $3^{t-1}$ |
| $7 \cdot 3^{t-1} \leqslant n < 3^{t+1}$ | $3^t$ | $3^t$ | $n - 2 \cdot 3^t$ |

# LR Algorithm




| | Leaves (Left) | Leaves (Middle) | Leaves (Right) |
|---|---|---|---|
| $3^t \leqslant n < 5 \cdot 3^{t-1}$ | $n - 2 \cdot 3^{t-1}$ | $3^{t-1}$ | $3^{t-1}$ |
| $5 \cdot 3^{t-1} \leqslant n < 7 \cdot 3^{t-1}$ | $3^t$ | $n - 4 \cdot 3^{t-1}$ | $3^{t-1}$ |
| $7 \cdot 3^{t-1} \leqslant n < 3^{t+1}$ | $3^t$ | $3^t$ | $n - 2 \cdot 3^t$ |

# Definition of Jumping Sequences q → 1

- Definition (Jumping Sequence Problem)
  - $S_n = (1 = a_0, a_1, a_2, \ldots, a_{k-1}, a_k = n)$.
  - The jump from $a_i$ to $a_{i+1}$ costs $a_{i+1} / a_i$
  - Cost of $S_n$ is: $c(S_n) = \sum_{i=1}^{k} \frac{a_i}{a_{i-1}}$

$$\frac{1 - q^{a_i}}{a_{i-1}}$$

$$c(q, S_n) = \sum_{i=1}^{k} \frac{1 - q^{a_i}}{a_{i-1}}$$

  - Find the best sequences from 1 to n with minimum cost. g(n) is defined as such minimum cost.

# Properties of Jumping Sequences q → 1

- Jumps along the integers

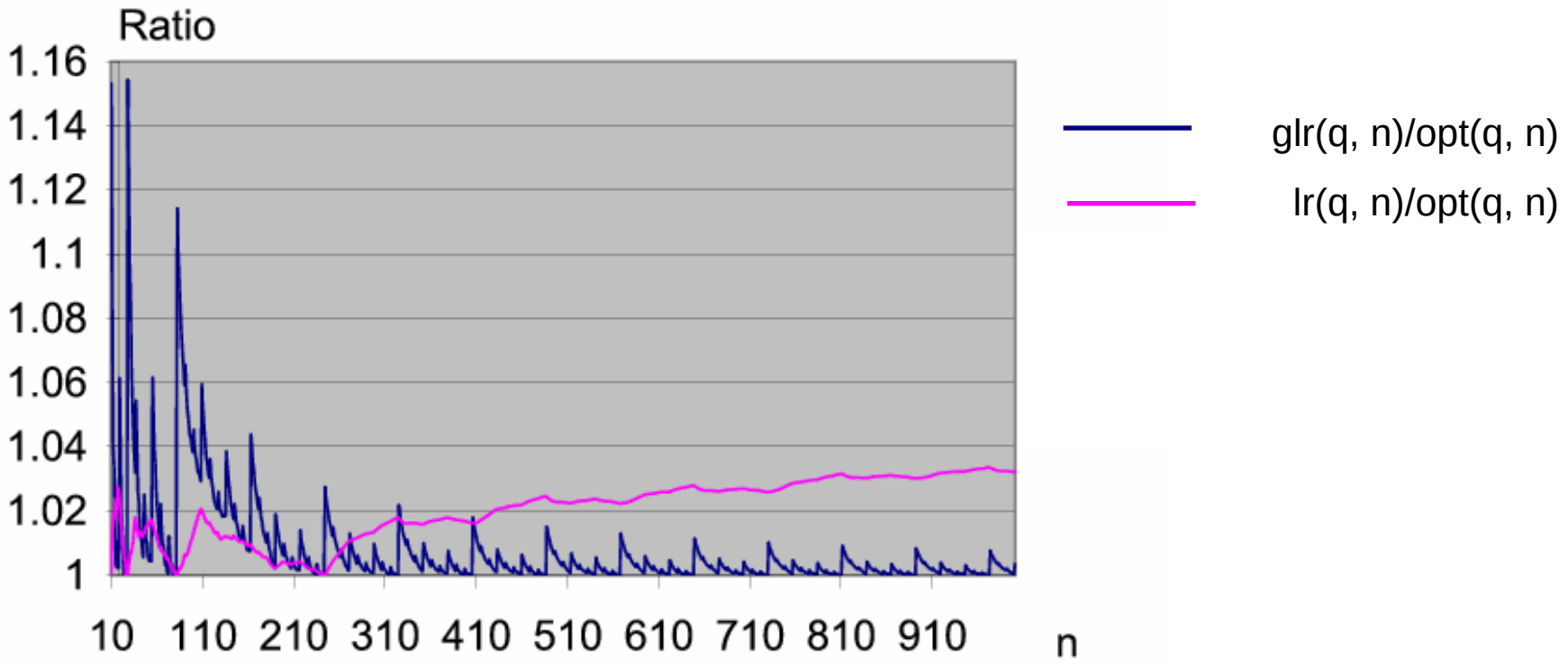
  $g_Z(n)$ is defined as the minimum cost of jumping along integers from 1 to n, and $g_Z(n)$ satisfies:

  - If $n_1 > n_2$, then $g_Z(n_1) > g_Z(n_2)$.
  - If $n = 3^t$ and $S_{3t} = (1, 3, 9, \ldots, 3^t)$
    - $c(S_{3t}) = 3t$;
    - $g_Z(3^t) \geqslant 0.996 \cdot 3t$.

# Analysis of LR Algorithm

- Approximation ratio: $2.01(1+\frac{1}{\lfloor \log_3 n \rfloor})$
- Proof Ideas: ($3^t \leqslant n < 3^{t+1}$)
  - For any LR tree with n leaves lr(n) < lr ($3^{t+1}$)
  - LR($3^{t+1}$) is a complete ternary tree, the cost of LR($3^{t+1}$) is $3\cdot(3^{t+1})\cdot(t+1)$.
  - LR($3^t$) is a complete ternary tree, the cost of LR($3^t$) is $3\cdot(3^t)\cdot t$.
  - $g_Z(3^t) \geqslant 0.996 \cdot 3\cdot t$
  - $lr(n) < lr(3^{t+1}) = 3\cdot(3^{t+1})\cdot(t+1) = 3\cdot(1+1/t)\cdot lr(3^t)$
    $\leqslant 3\cdot(1+1/t)\cdot(1/0.996)\cdot 3^t\cdot g_Z(3^t)$
    $\leqslant 3.015\cdot(1+1/t)\cdot n\cdot g_Z(n)$
    $\leqslant 3.015\cdot(1+1/t)\cdot opt(n)$
  - More careful analysis will give us an approximation ratio $2.01\cdot(1+1/t)$.
  - Note: lr(n) denotes the cost of LR(n).

# Comparison of LR and GLR (q = .99)



# Summary

- Analyze the properties of optimal GKM trees
- Approximation algorithms to build GKM trees
  - The GLR Algorithm
  - The LR Algorithm (q → 1)
- Jumping Sequence Problems
  - Properties of the optimal Jumping Sequences
    - Case 1: 0< q <1
    - Case 2: q →1

# References


- ['98 Wong, Gouda, Lam]  C. K. Wong, M. Gouda, and S. S. Lam Secure group communications using key graphs, Proceedings of the ACM SIGCOMM '98 conference on ATAPCC.

- ['01 Li, Yang, Gouda, Lam] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, batch re-keying for secure group communications, WWW10, 2001, Hong Kong.

- ['03 Zhu, Chan, Noubir] F. Zhu, A. Chan, and G. Noubir, Optimal tree structure for key management of simultaneous join/leave in secure multicast, Proceedings of MILCOM, 2003.

- ['07 Graham, Li, Yao] R. Graham, M. Li, and F. Yao Optimal tree structures for group key management with batch updates, to appear in SIAM Journal on Discrete Mathematics.

# Related Publications

- "Approximately optimal trees for group key management with batch updates" (with Minming Li, Ze Feng, R. Graham and Frances F. Yao). Accepted for publication by Special Issue of Theoretical Computer Science.

- "Optimal jumping patterns" (with Steve Butler and R. Graham). Submitted to Journal of Combinatorics and Number Theory.

Thank you!