Membership Operations

- **Join**: a prospective member wants to join
- **Leave**: a member wants to (or is forced to) leave
- **Partition**: a group is split into smaller groups
  - Network failure: network event causes disconnectivity
  - Explicit partition: application decides to split the group
- **Merge**: two or more groups merge to form one group
  - Network fault heal: previously disconnected partitions reconnect
  - Explicit merge: application decides to merge multiple pre-existing groups into a single group
Tree-Based Group Diffie Hellman

- Simple: One function is enough to implement it
- Fault-tolerant: Robust against cascade faults
- Secure
  - Contributory
  - Provable security
  - Key independence
- Efficient
  - $d$ is the height of key tree ($< O(\log_2 N)$), and $N$ is the number of users
  - Maximum number of exponentiations per node $3d$

Key Tree (General)

Key Tree (n’s view)

Key-path: Set of nodes on the path from member node to root node

Co-path: Set of siblings of nodes on the key-path

Member knows all keys on the key-path and all blinded keys

Any member who knows blinded keys on every node and its session random can compute the group key.
Join (n’s view)

Leave (n’s view)
Partition (ns’s view)

Partition (ns’s view)

Partition: Both Sides
### Tree Management: do one’s best

- **Join or Merge Policy**
  - Join to leaf or intermediate node, if height of the tree will not increase.
  - Join to root, if height of the tree increases.

- **Leave or Partition policy**
  - No one can expect who will leave or be partitioned out.
  - No policy for leave or partition event

- **Successful**
  - Still maintaining logarithmic (height < $2 \log_2 N$)

### Discussion

- **Efficiency**
  - Average number of mod exp: $2 \log_2 n$
  - Maximum number of round: $\log_2 n$

- **Robustness** is easily provided due to self-stabilization property