Group key management algorithms (GKMA)

Lakshminath R. Dondeti
Brian Weis
IETF-56 MSEC WG meeting
San Francisco, Mar 17 2003

Introduction

- **Group key management architecture**
- **Group key management protocols**
  - GDOI, GSAKMP, MIKEY
- **Group key management algorithms**
  - LKH, OFT, OFC, MARKS, Subset diff etc.

Group key management algorithms

- **LKH, OFT, OFC (stateful): expired I-Ds**
- **LKH is an informational RFC**
- **Subset difference (stateless): expired I-Ds**
- **MARKS: expired I-D**
- **GSAKMP and GDOI include support for LKH**
  - Allow extensions to support other GKMAs

GKMA standardization

- **LKH, OFT and OFC use similar logical trees**
  - One RFC may cover this
  - Consider immediate as well as batch rekeying
  - Brian and Lakshminath working on an I-D ☺
- **Previous efforts by David McGrew and Lakshminath**
  - Group key transport protocol (GKTP)
- **Subset difference? MARKS?**
  - Any known efforts to standardize these?

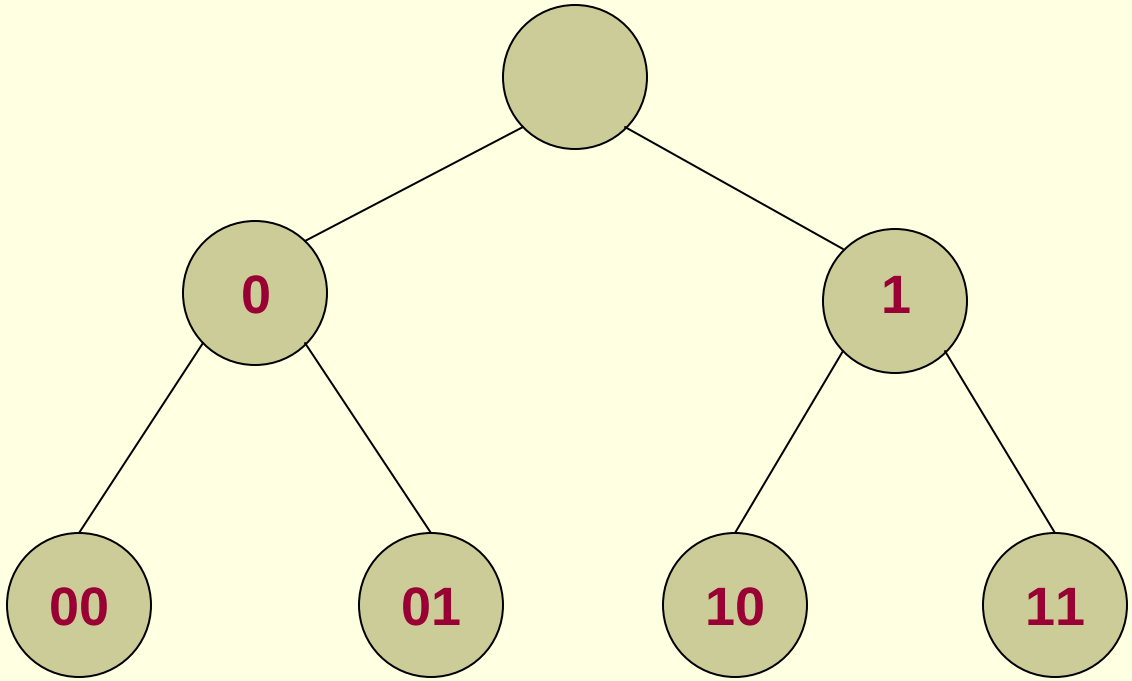
Group key transport protocol

- **GKTP**
  - Rekey and feedback messages
    - Rekey messages are part of GKM arch I-D
    - Feedback messages are covered in a separate I-D
      - New I-D submitted by Lakshminath and Thomas
  - Key tree management
    - Key tree encoding

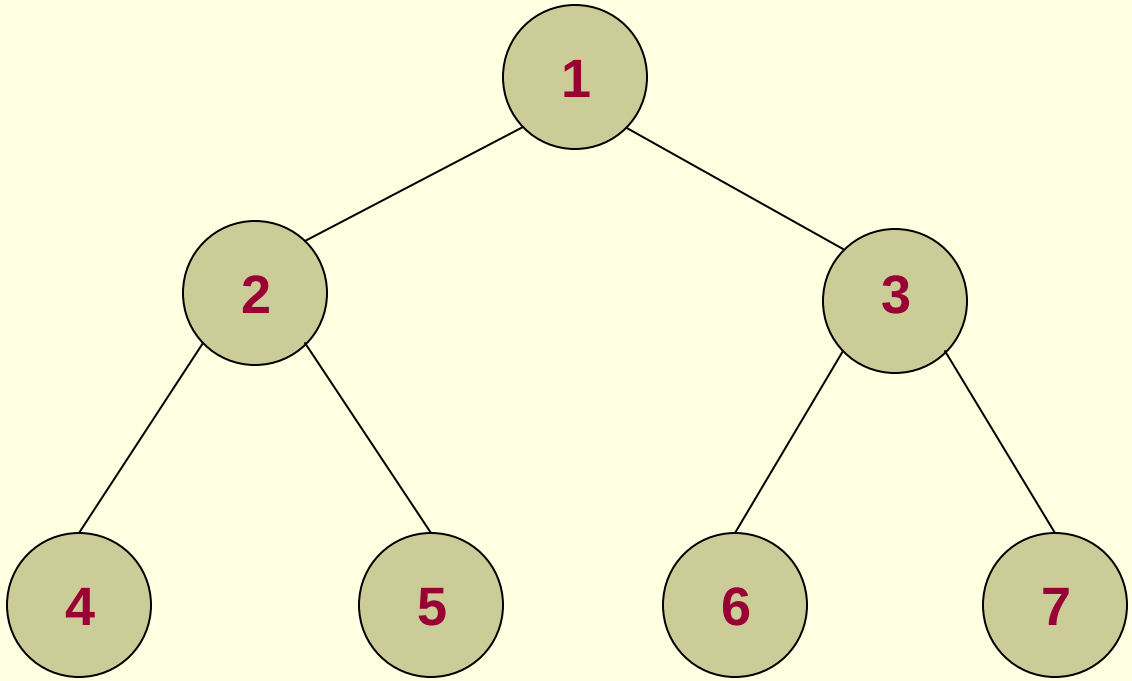
Key tree management

- **Key trees**
  - may grow and shrink with membership changes
  - Fixed-size trees
- **Do we need to include node ID changes?**
  - Yes
  - No (implicit in key changes)
- **Several known key tree encoding schemes**
  - Binary number encoding
  - Natural number encoding (works for d > 2)

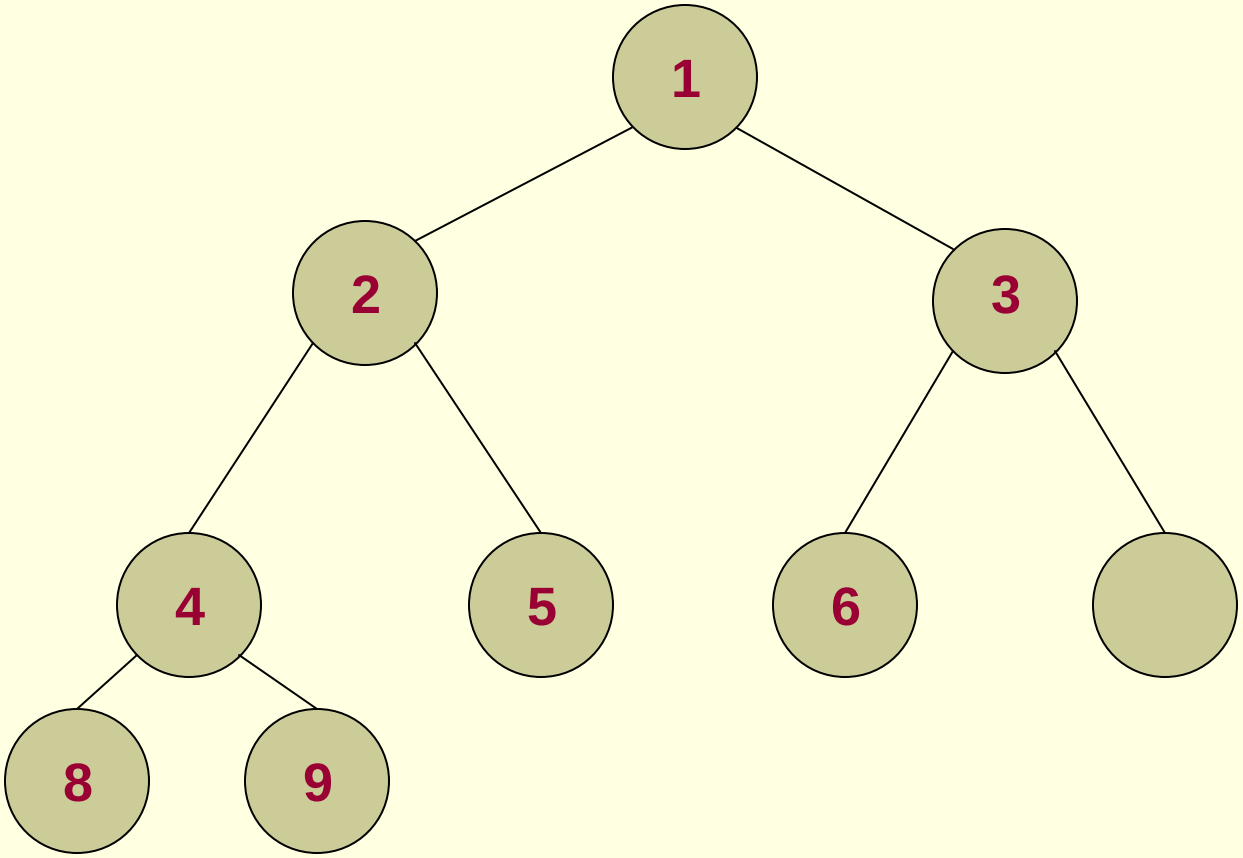
Binary encoding



Natural number encoding of key trees



After two joins and member 7's departure



Key tree management

- **In the previous schemes, tree grows**
  - by splitting a leaf node
  - split an internal node or the root node itself
- **Tree shrinking occurs**
  - Only if the highest numbered node has departed
- **Next member to join receives position “7”**
- **Node number update messages may be sent in or along with rekey messages**

Encoding proposed by Zhang et. al.

- **Natural number encoding**
- **$\{k_j\}k_i$ is identified with the ID of k_i**
  - Assuming k_j is k_i's parent
    - **Too simplistic for OFT**
- **The key tree is always full and balanced**
  - Null nodes are introduced to make it so
- **Joins and leaves may change the leaf-node IDs**
- **Members can determine their new ID using old ID and the max internal key node ID.**

Rekey messages

- Send the max. internal key node ID in rekey messages
- Works for immediate and batch rekeying
- Might not work for OFT?

Next steps: Option 1

- **Standardized key tree encoding**
- **Try to design a scheme that works for LKH / OFT / OFC etc.**
- **Don't send key node ID changes in rekey messages**
- **Keep the trees as efficient as possible**
  - No static trees!
  - Static trees inefficient if instantaneous membership size is typically smaller than subscription base

Next steps: Option 2

- **GCKS may advertise a standardized scheme and use it**
  - Tradeoffs in footprint, communication cost etc.,
- **This stems from different people having different ideas in key tree management**
  - But we will make them publish a standards RFC before they can use a “scheme”
- **What do other people think?**