Experimental Performance of Five Notable Group Key Agreement*

IRTF GSEC WG

February 7, 2002

Yongdae Kim, Gene Tsudik

*Originally from Y. Amir, Y. Kim, C. Nita-Rotaru, G. Tsudik, "The Performance of Group Key Agreement", in submission

Outline

- Definitions and concepts
- Short Introduction to Protocols
  - GDH
  - TGDH
  - STR
  - BD
  - CKD
- Performance Comparison
  - Theoretical Analysis
  - Experimental Results (Computational)
  - Experimental Results (WAN)
- Related work (Time permits)

Group Communication Settings

- Few-to-Many
  - Single-source broadcast: Cable/sat. TV, radio
  - Multi-source broadcast: Televised debates, GPS
- Any-to-Any
  - Collaborative applications need inherently underlying peer groups.
  - Video/Audio conferencing, collaborative workspaces, interactive chat, network games and gambling
  - Rich communication semantics, tighter control, more emphasis on reliability and **security**

Dynamic Peer Groups (DPG)

- No hierarchy
- Any member can be sender and receiver
- Frequent membership changes

- Relatively small (<100 of members)

My focus: key management in DPGs

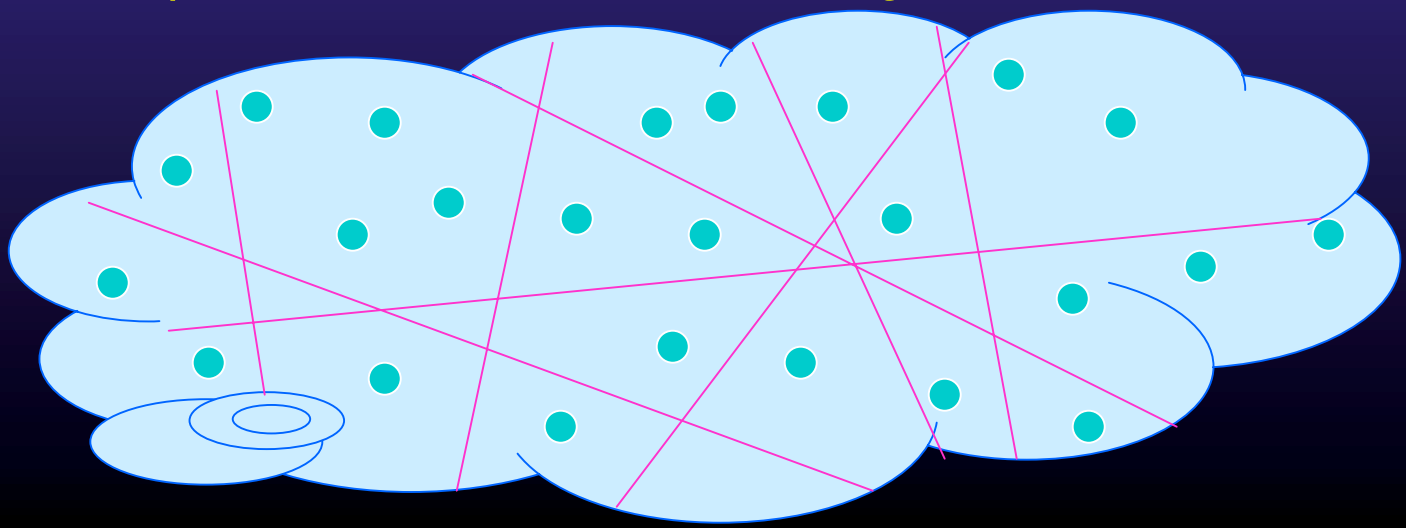
Group Key Management

- Group key: a secret quantity known only to current group members
- Group Key Distribution
  - One party generates a secret key and distributes to others
  - Centralized vs. decentralized (floating key server)
- Group Key Agreement
  - Secret key is derived jointly by two or more parties.
  - Key is a function of information contributed by each member.
  - No party can pre-determine the result.

Can we use Key Distribution in DPG?

- Centralized key server
  - Single point of failure
  - Attractive attack target
- Can key server be sufficiently replicated? ⇒ Very costly
  - Availability of a key server in any and all possible partitions
    - » Network can have arbitrary faults!

We have to use decentralized group key distribution
⇒ still expensive secure channel management



Group Communication System

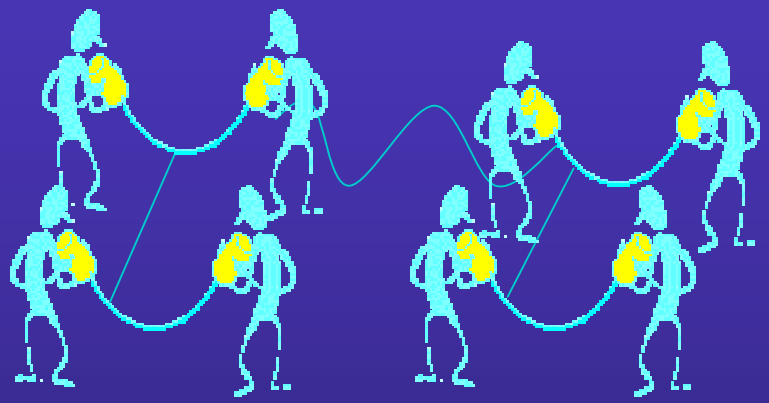
- 2-party key management is easy
- Group key management is hard: multi-party, multi-round

- GCS Offers
  - Efficient messaging : any-to-any
  - Dynamic membership
  - Message / event ordering
  - Fault-detection service
  - Fault-tolerant : resistant against cascaded failures

  to peer group

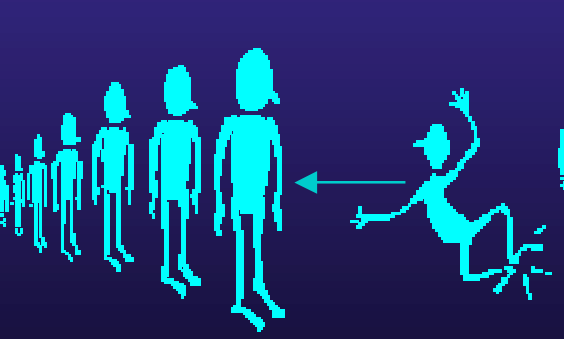
- Different from single source multicast (e.g. IP)
- Examples: Horus, Transis, Totem, Spread

Membership Operations

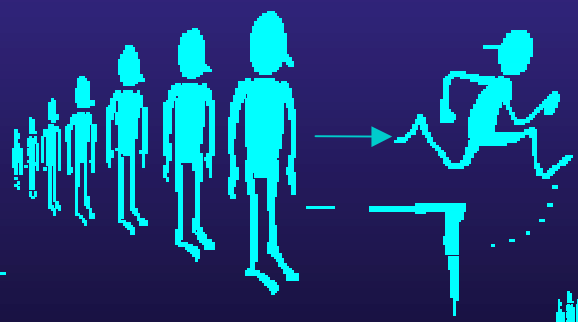


Formation

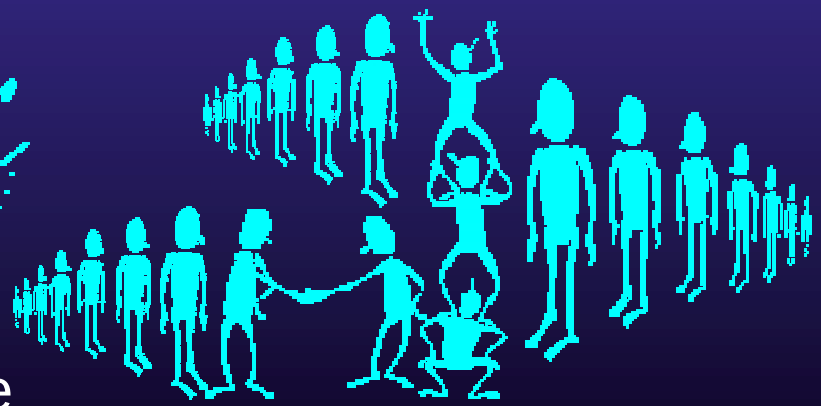
Group partition



Member join



Member leave



Group merge

Secure Group Communication

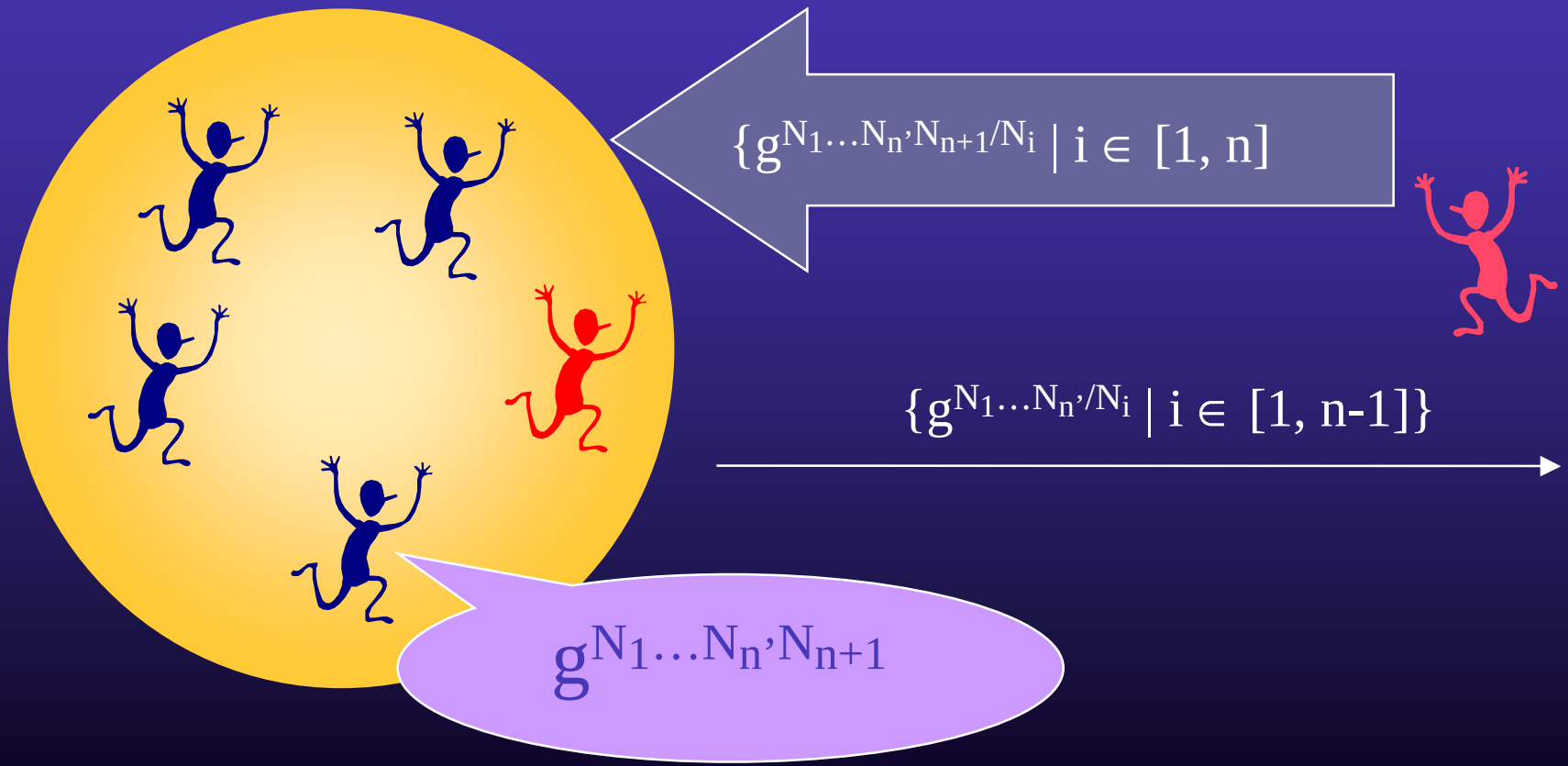
- Group key agreement protocols rely on GCS for:
  - Protocol message transport
  - Strong membership semantics (Notification of a group membership)
  - Used for safety or correctness, not for security
- GCS needs specialized security mechanisms.

Mutual benefit and interdependency

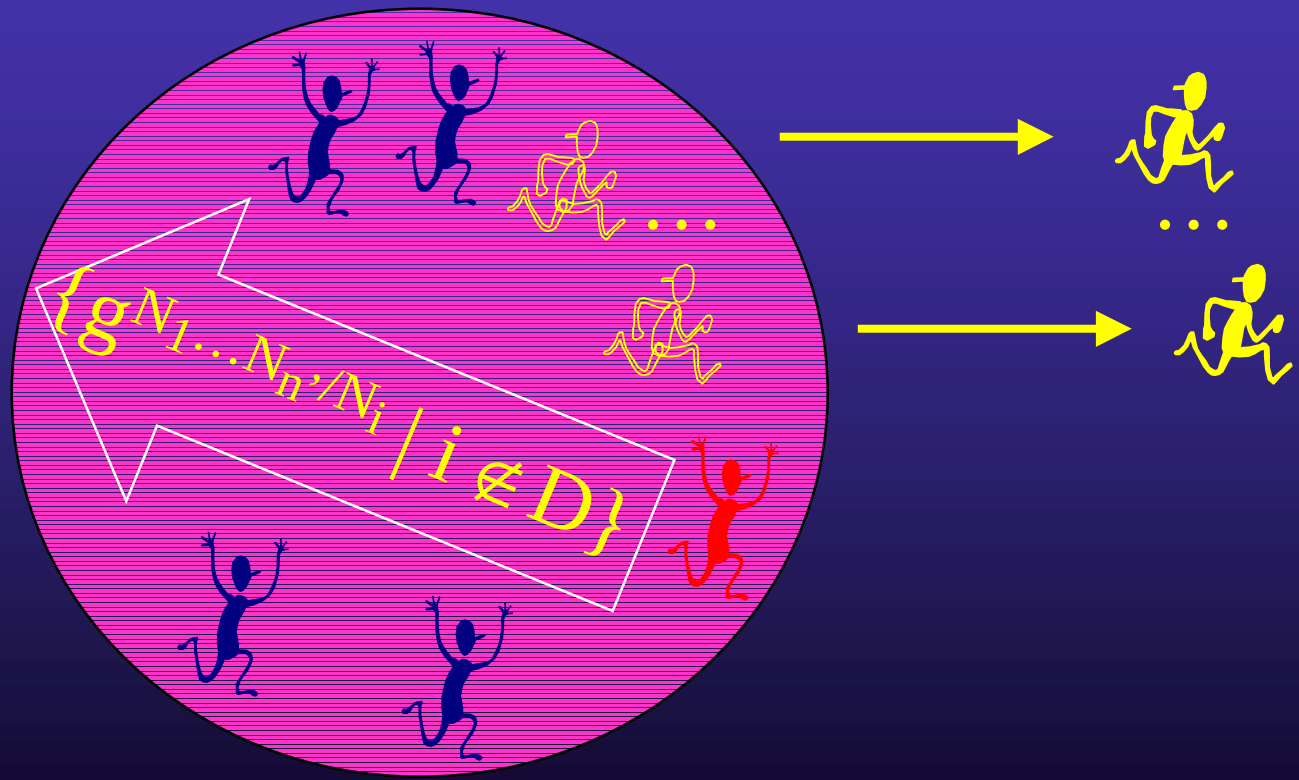
Cliques GDH

- Steiner, Tsudik, Waidner, in IEEE TPDS 2000
- Contributory group key agreement protocols
- Security
  - Formal proof of security (includes Key Independence)
- Efficiency: few rounds except for merge
- Introduces dynamic group operation
- natural extension of Diffie-Hellman protocol to n members
  - Group key: $g^{N_1 N_2 \ldots N_n}$ where N_i is member i's secret share
  - Partial key for each member i: $g^{N_1 N_2 \ldots N_n / N_i}$

Joins


$\{g^{N_1 \ldots N_n, N_{n+1}/N_i} \mid i \in [1, n]$
$\{g^{N_1 \ldots N_n/N_i} \mid i \in [1, n-1]\}$
$g^{N_1 \ldots N_n, N_{n+1}}$

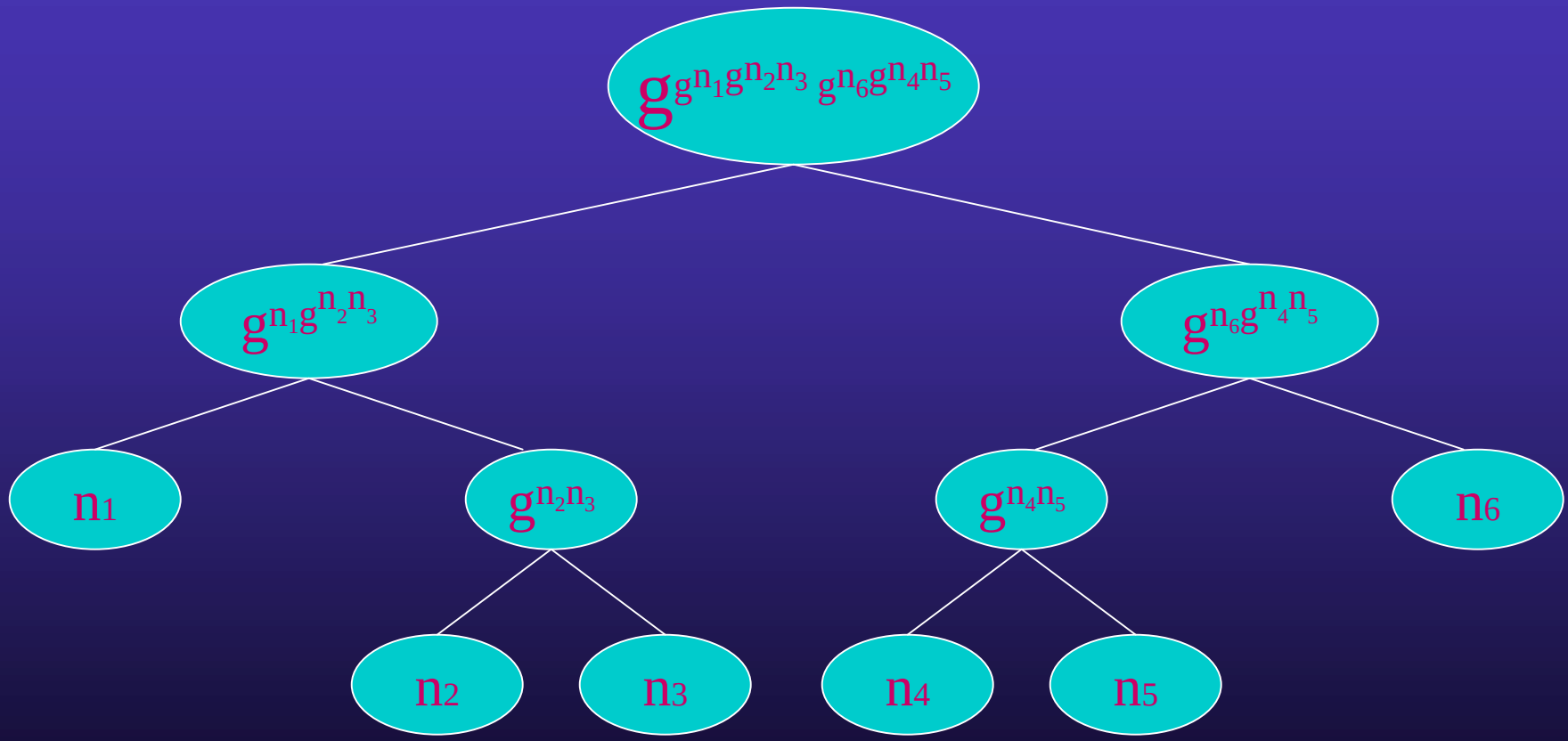
Leave or Partition



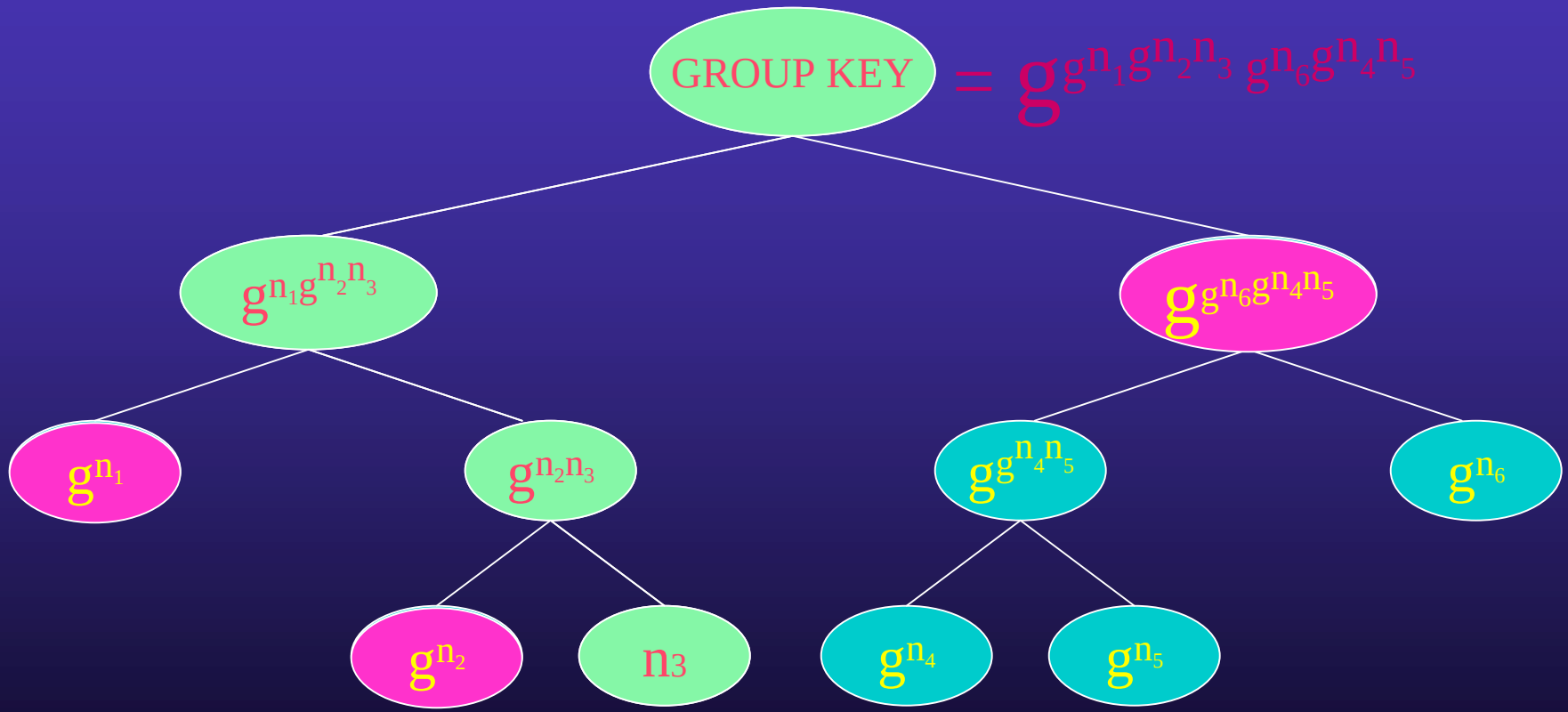
TGDH

- Kim, Perrig, Tsudik, ACM CCS 2000
- Simple: Two functions enough
- Fault-tolerant: Robust against cascaded faults
- Secure
  - Provable security (including key independence)
- Efficient
  - O(log n) computation, communication cost

Key Tree (General)



Key Tree (n_3's view)



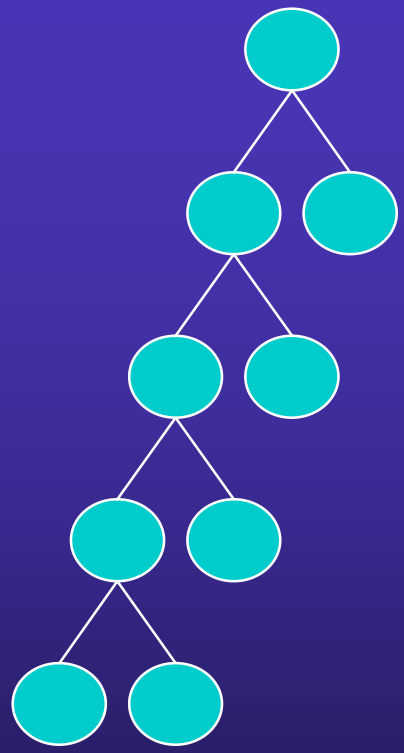
Member knows all keys on the key-path and all blinded keys

Key-path Co-path Siblings of nodes on the key-path Out nodes

Discussion

- Efficiency (h: height of the key tree)
  - Maximum number of mod exp: 4 h
  - Maximum number of round: h
- Robustness: easily provided due to self-stabilization property
- Self-clustering
  - Logical Key Tree: Not depending on the physical location of the group members
  - After a partition, members on the same partition will form a cluster
  - After merge, next partition on the same link is much easier

STR



- Kim, Perrig, Tsudik, IFIP SEC 2001
- Using completely unbalanced tree
- Communication efficient
  - Max 2 rounds
  - Max 2 b-casts
- Simple: two functions enough
- Fault-tolerance: easier than TGDH
- Security:
  - Contributory
  - Provable security (including key independence)
- Computation is bit more expensive than TGDH
  - Max # exps = 4(N-1)
  - N is # users

BD

- Burmester, Desmedt, Eurocrypt'94
- Computation efficient
  - 3 exps plus $O(n^2)$ mults
  - 2 rounds
  - Each round requires n broadcasts
- No special join, leave, merge, and partition protocols
  - When membership changes, build new key
- Fault-tolerant
  - When cascaded event happens, start from scratch
- Secure
  - Contributory
  - Provable security
  - Key independence

Protocol

1. Each U_i selects random integer r_i and computes and broadcasts $z_i = g^{ri} \bmod p$
2. Each U_i computes and broadcasts
$$X_i = (z_{i+1}/z_{i-1})^{ri} \bmod p$$
3. Each U_i computes the conference key
$$K_i = (z_{i-1})^{n\ ri}\ X_i^{n-1}\ X_{i+1}^{n-2}\ \ldots\ X_{i-2} \bmod p$$

$$\begin{aligned} Ki &= (z_{i-1})^{n\ ri}\ X_i^{n-1}\ X_{i+1}^{n-2}\ \ldots\ X_{i-2} \\ &= (z_{i-1})^{n\ ri}\ (z_{i+1}/z_{i-1})^{\ r_i\ n-1}\ (z_{i+2}/z_i)^{\ r_{i+1}\ n-2}\ \ldots\ (z_{i-1}/z_{i-3})^{\ r_{i-2}} \\ &= (g^{r_{i-1}})^{n\ ri}\ (g^{(r_{i+1} - r_{i-1})r_i})^{n-1}\ (g^{(r_{i+2} - r_i)r_{i+1}})^{\ n-2}\ \ldots\ (g^{(r_{i-1} - r_{i-3})r_{i-2}}) \\ &= g^{\ r_1 r_2 + r_2 r_3 + r_3 r_4 + \ldots + r_n r_1} \end{aligned}$$

Discussion

- Efficiency
  - Constant 2 modular exponentiation
  - Constant 2 communication round
  - Each round requires n broadcasts
- Robustness is easy since protocol starts from scratch

CKD

- “Naïve” Decentralized Group Key Distribution
- Maintaining secure channel (DH) between the current group controller and all other members
- Key transport using the secure channel
- Secure channels re-established when controller leaves

Theoretical Analysis

		Comm				Comp	Robust
		Round	Msg	Uni	Broad	Exp	
CLQ	Join	4	n+3	n+1	2	n+3	Hard
	Leave, Partition	1	1	0	1	n-1	
	Merge	k+3	n+2k+1	n+2k-1	2	n+2k+1	
TGDH	Join, Merge	2	3	0	3	2log n	Easy
	Leave	1	1	0	1	log n	
	Partition	log n/2	log n	0	log n	log n	
STR	Join	2	3	1	3	7	Easy
	Leave, Partition	1	1	0	1	3n+6	
	Merge	2	3	0	3	4k+4	
BD		2	2n	0	2n	3 (4?)	Easy
CKD	Join, Merge	3	k+2	k	2	k+2	Easy
	Leave, Partition	1	1	0	1	1	

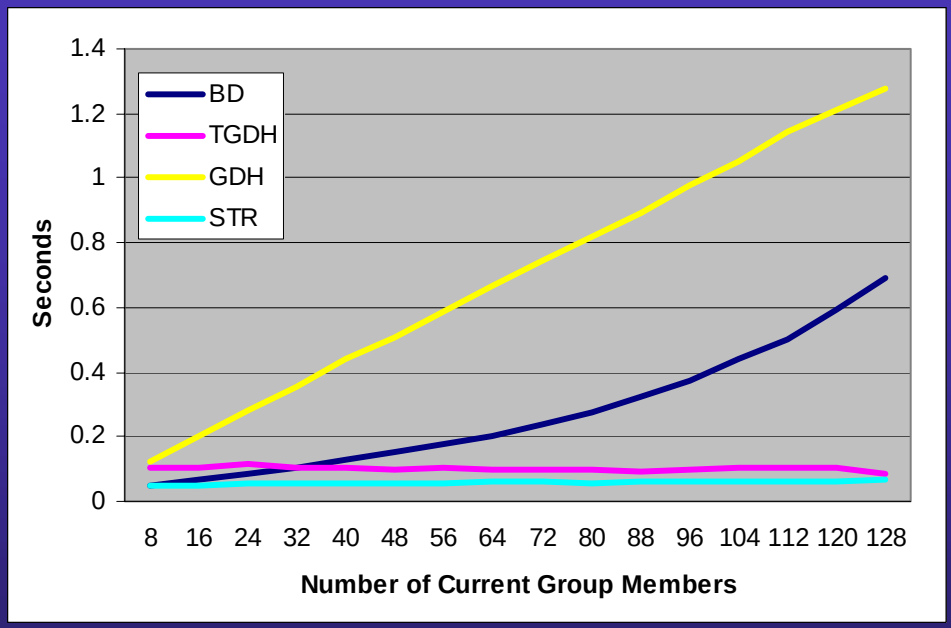
expensive when key server leaves

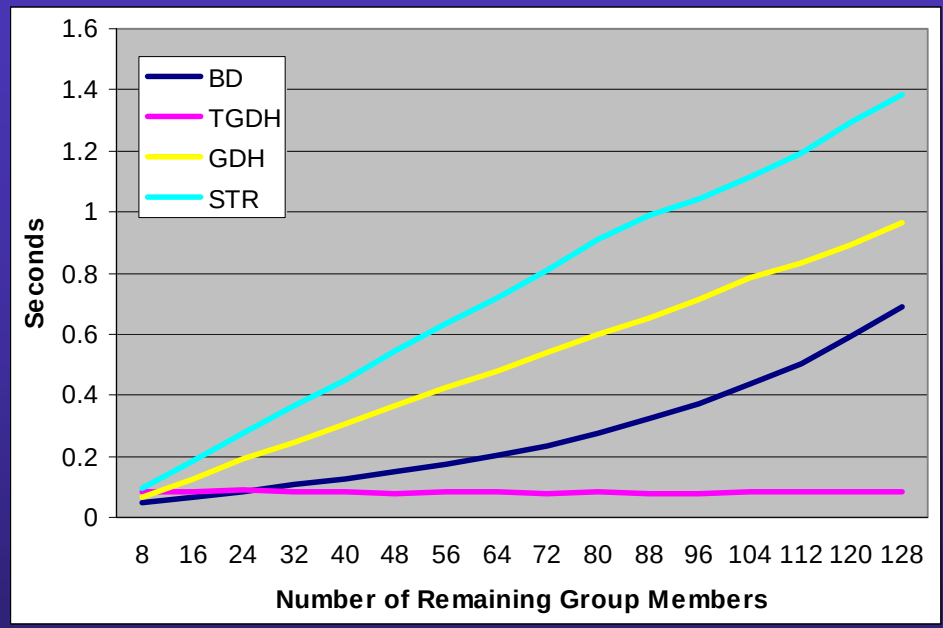
Experimental Results (Computation)

- Simulation Results without communication
- Meaningful results for LAN
- Average time for each membership event

- Considerations
  - 1024 Bit RSA signature with public exponent 3 for all messages
  - Signing: 0.007 sec, Verifying: 0.0001 sec
  - TGDH: Random Tree
  - STR: picking random member for subtractive event

Computational Cost (Join and Leave)

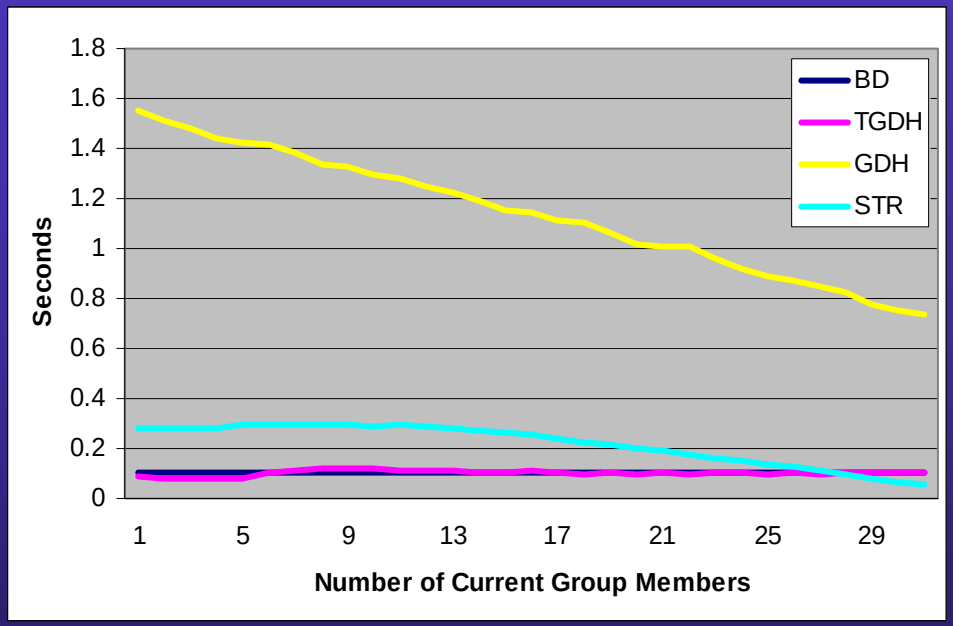


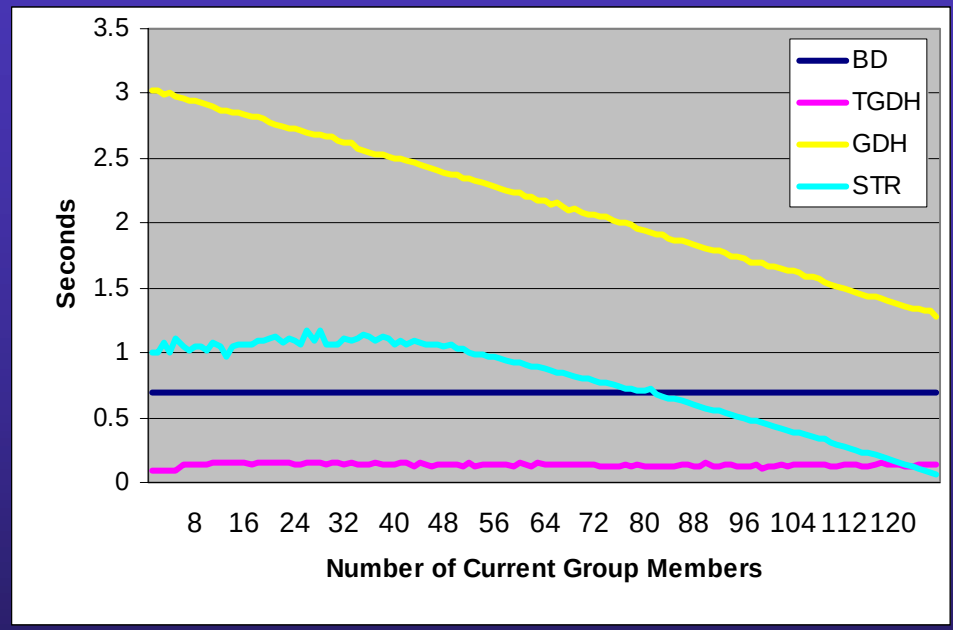


- x-axis: # members before join
- TGDH, STR: almost 0.1 sec
- GDH worst
- TGDH: Joining node is near to root due to random tree
- BD: hidden cost => signature verification, modular multiplication

- x-axis: # members after leave
- TGDH best
- STR worst

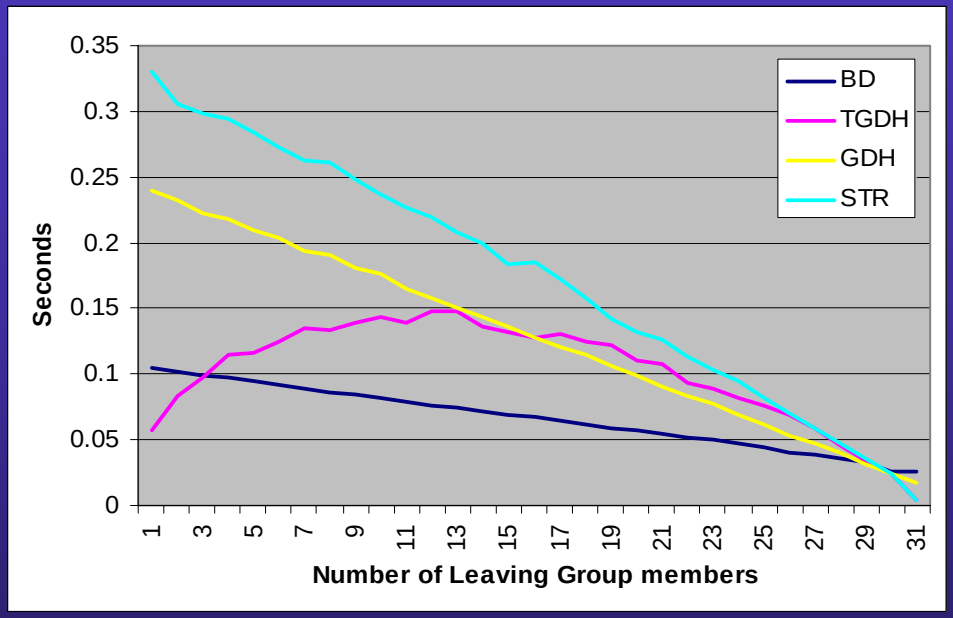
Computational Cost (Merge)

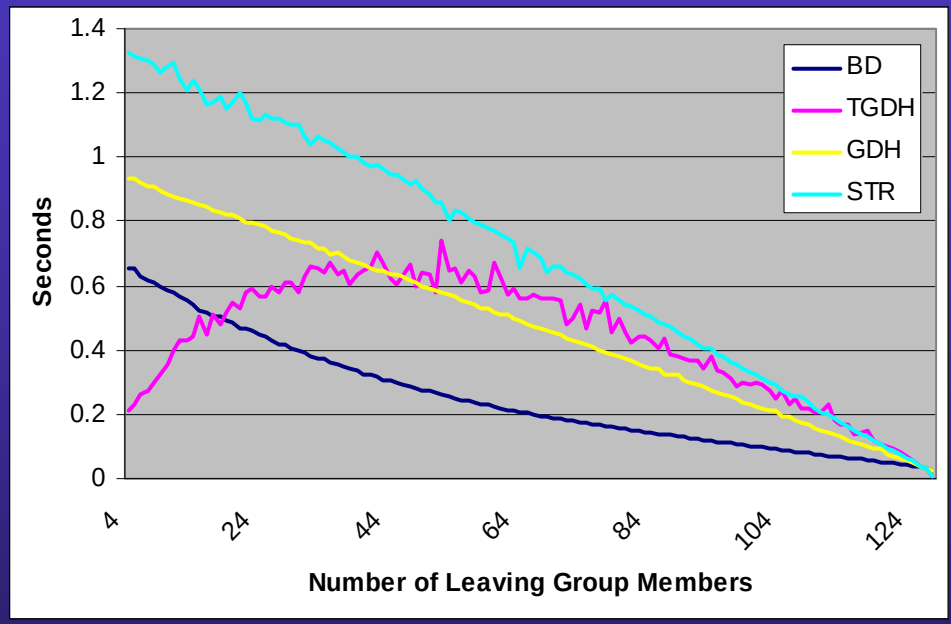




- Delay for member in current group when merging 2 ~ 5 groups to result in 32 (left) or 128 (Right) members
- x-axis: Number of current group members
- For small groups (< 32), BD performs best
- For larger groups, TGDH is best (usually merge to root)
- If # of merging groups increases, TGDH results worsen

Computational Cost (Partition)



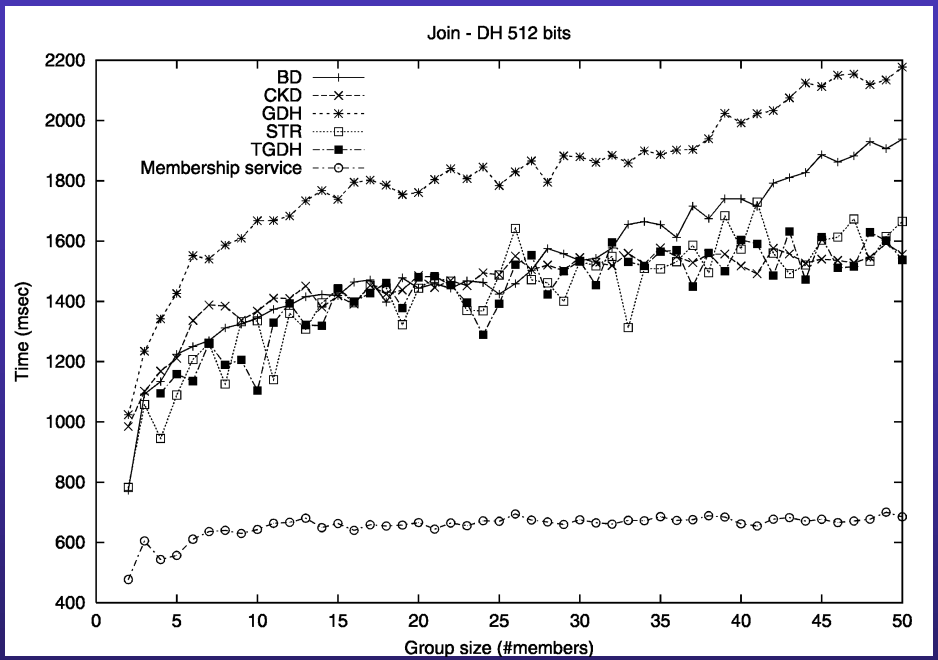


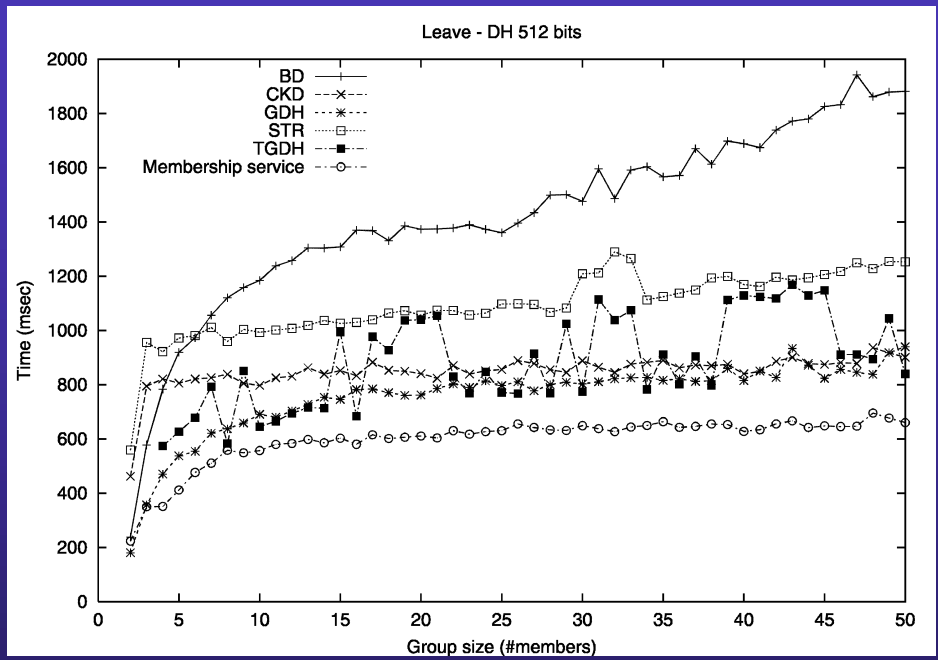
- Delay for member in current group of starting size 32 (Left) / 128 (Right) when x members leave
- Usually BD is best
- NOTE: clustering effect in TGDH with repeated partitions/merges!

Experimental Result (WAN)

- Using Spread GCS (by JHU) over high delay WAN
  - JHU: 11 machines
  - UCI: 1 machine
  - ICU (Korea): 1 machine
- Delay (msec)
  - Ping: JHU – UCI = 70, UCI – ICU = 300, ICU – JHU = 270
  - Actual Spread delay from Sender (different due to ack)
    - » at JHU: 392
    - » at UCI: 328
    - » at ICU: 334
- DH parameter: |p| = 512, |q| = 160 bit
- 1024 RSA with public exponent 3
- Membership cost is pretty high: around 600 msec

Experimental Result on WAN





- Computational cost does not matter much
- Communication cost is most important
- On high delay network, hard to use any group key agreement
  - Imagine merge or partition cost
- Join: implemented with merge
  - For smaller delay WAN, TGDH will be best performer overall

Related Work (Group Key Agreement)

- Only provide formation of a group key
  - Steer et. al (1988): fast join, slow leave
  - Burmester and Desmedt (BD, 1993): fast but too many broadcasts
  - Becker and Wille (1998): always *log n* communication rounds and computation overhead
  - Tzeng and Tzeng (1999, 2000): Fast but does not provide forward and backward secrecy

Related Work (GKA, Continue)

- Cliques: Key Agreement in Dynamic Peer Groups (STW96)
  - Group Diffie-Hellman key agreement protocols
  - Dynamic membership operations
  - Slow computation: O(n) computation for each membership event
  - Communication overhead: k rounds for merge (k: # of new members)
- TGDH (KPT00)
  - Computation overhead reduced from O(n) to O(log n)
  - Providing robustness against cascaded failure inherently
  - Self-clustering
- STR (KPT01)
  - Communication overhead is lower than any other methods
  - Inherent robustness against cascaded faults

Related Work (DGKD)

- Dynamical selection of key distribution center
  - If center leaves, another member becomes the center
- CKD (AAHKNSSST00)
  - DH between the center and other members
  - Flat structure
- DISEC (DMS00)
  - Decentralized version of OFT
  - Tree Structure
- RBD00
  - Decentralized LKH with AVL tree
  - Tree structure

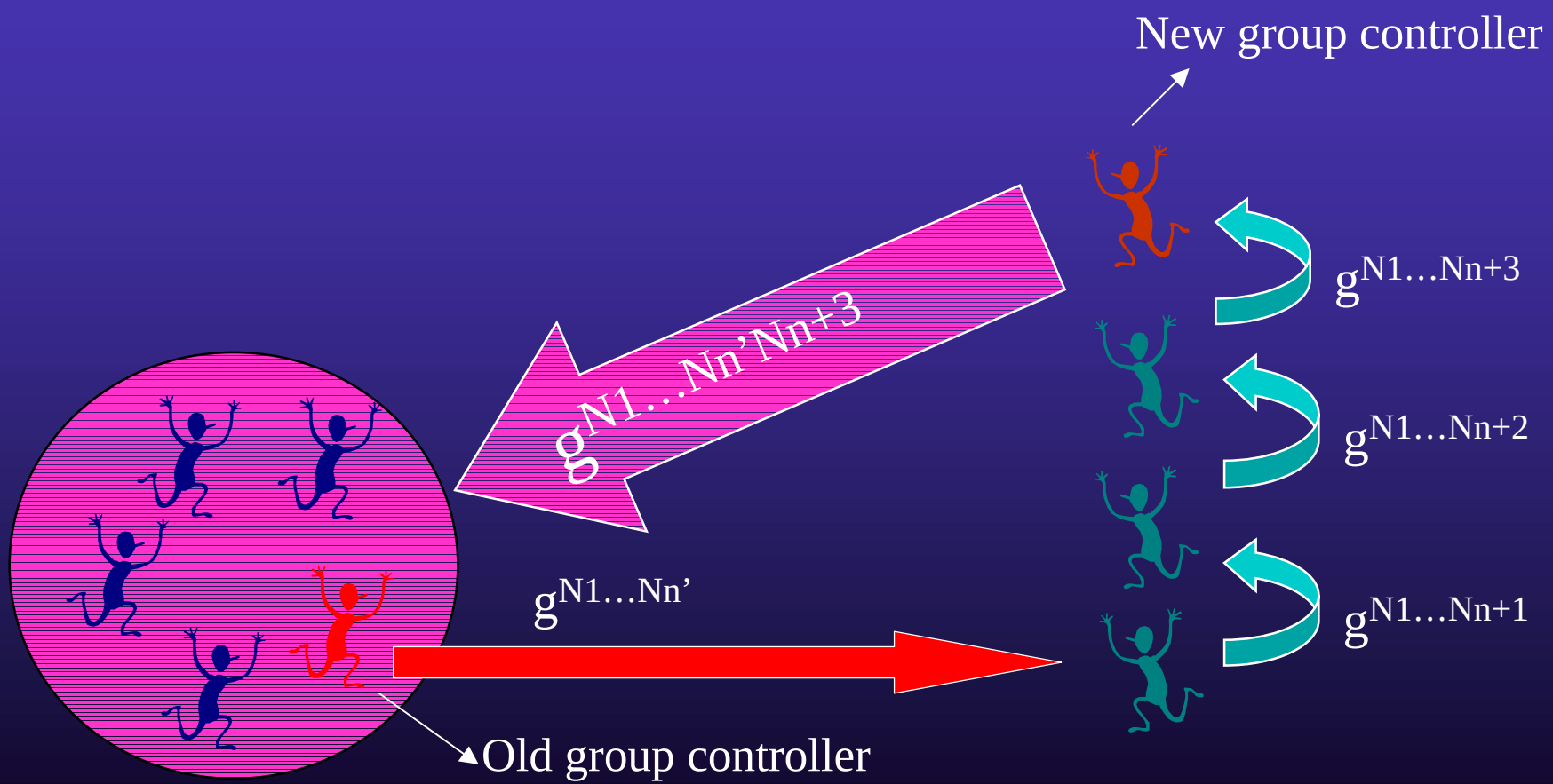
Thank you!!!

Questions?

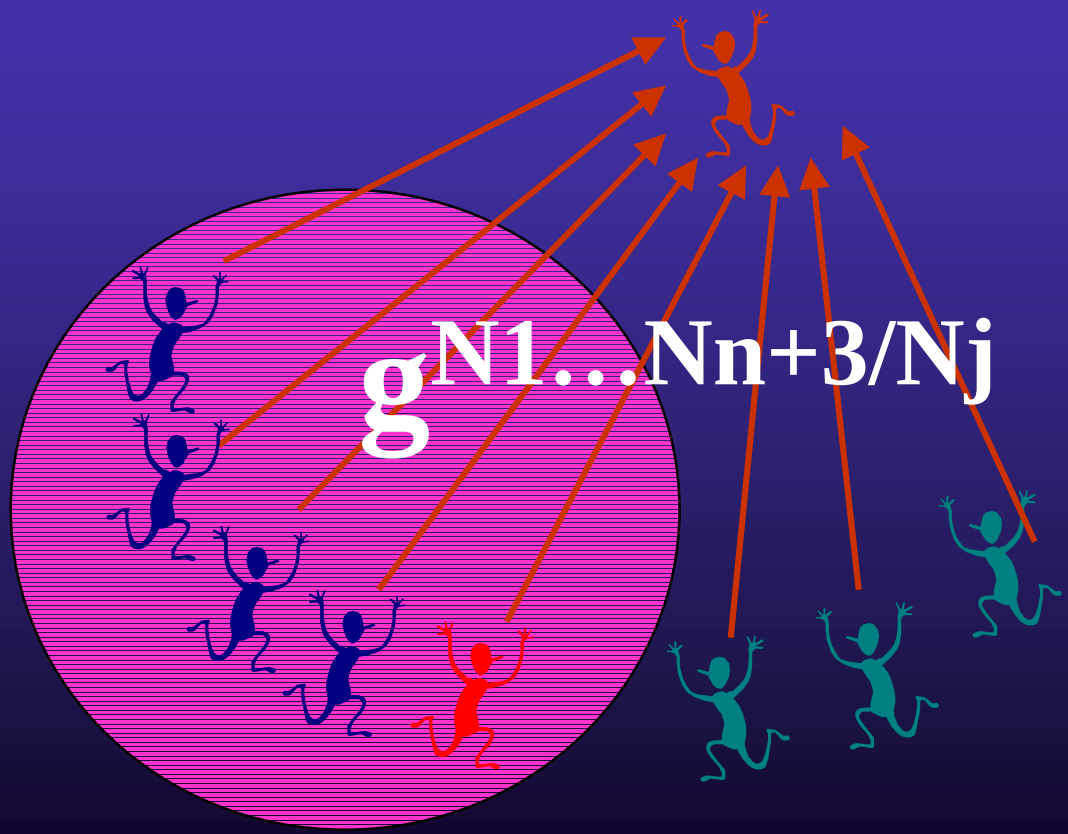
{kyongdae, gts}@ics.uci.edu

http://sconce.ics.uci.edu/cliques

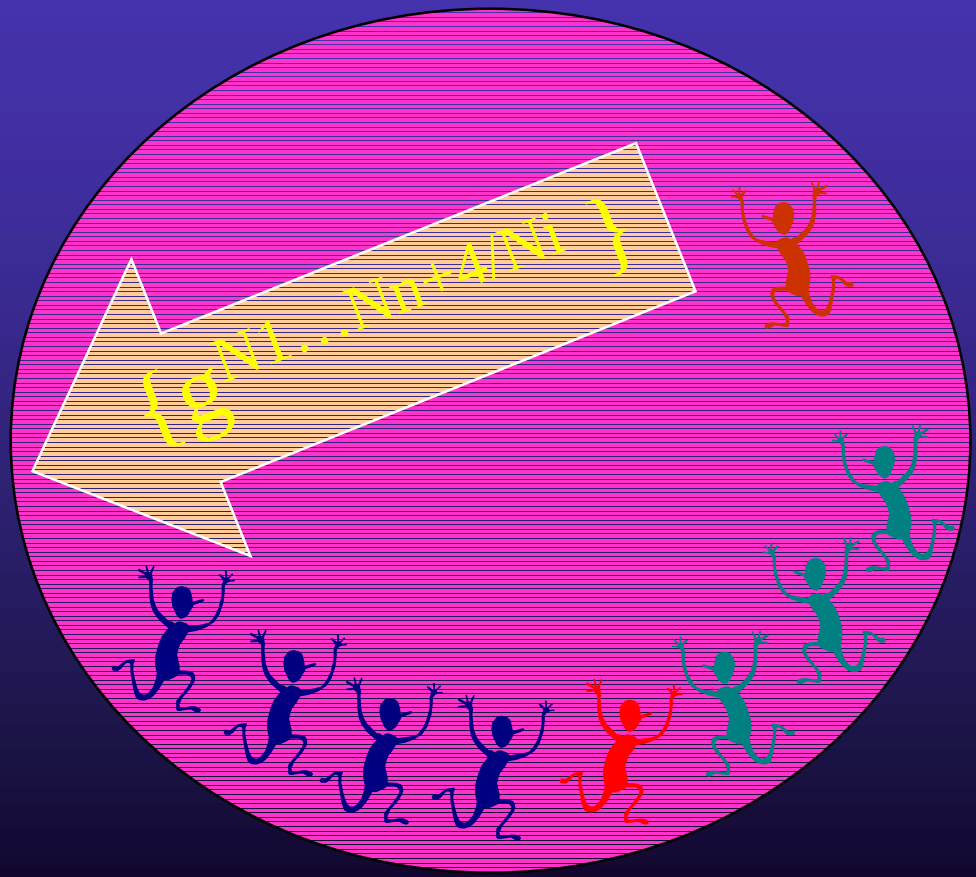
Merge I


New group controller
$g^{N1\ldots Nn'Nn+3}$
$g^{N1\ldots Nn+3}$
$g^{N1\ldots Nn+2}$
$g^{N1\ldots Nn+1}$
$g^{N1\ldots Nn'}$
Old group controller

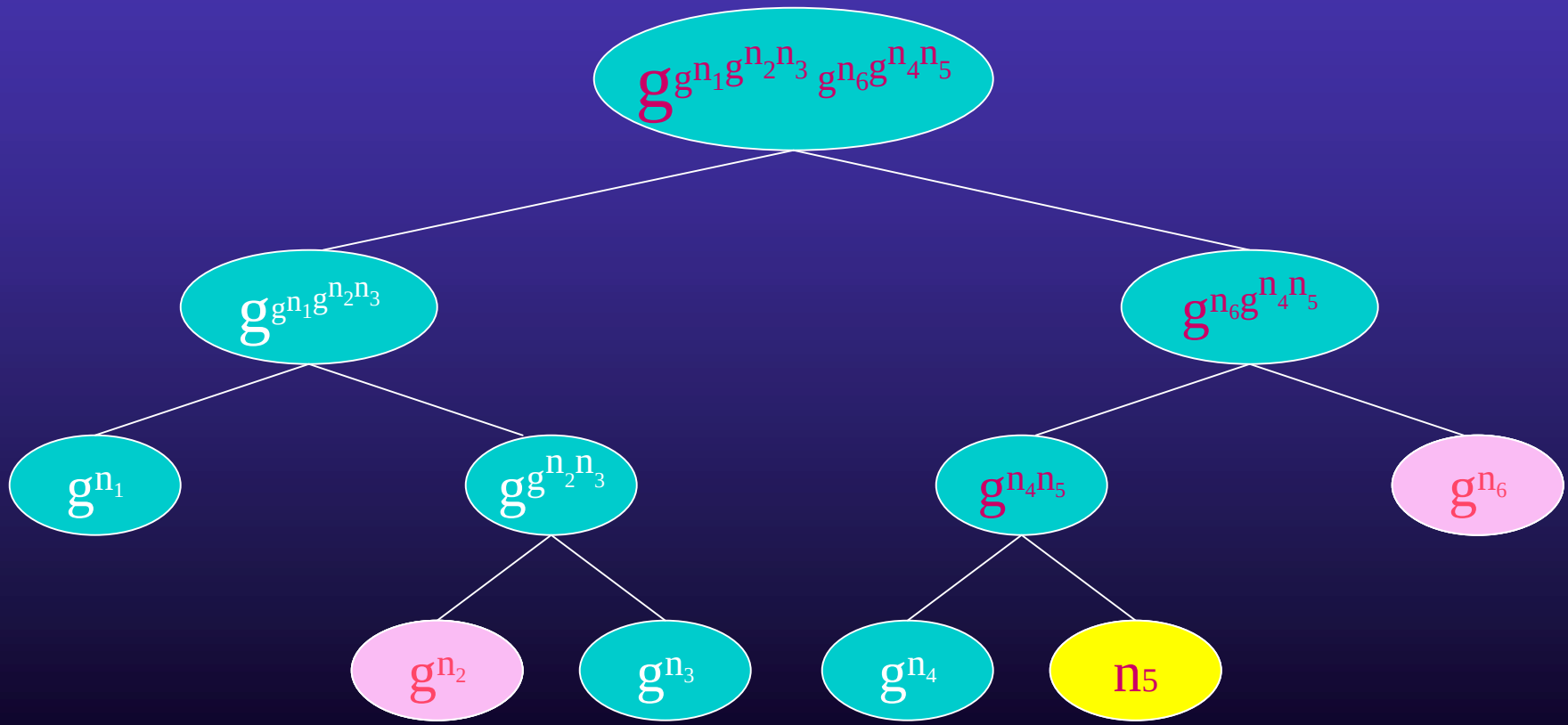
Merge II



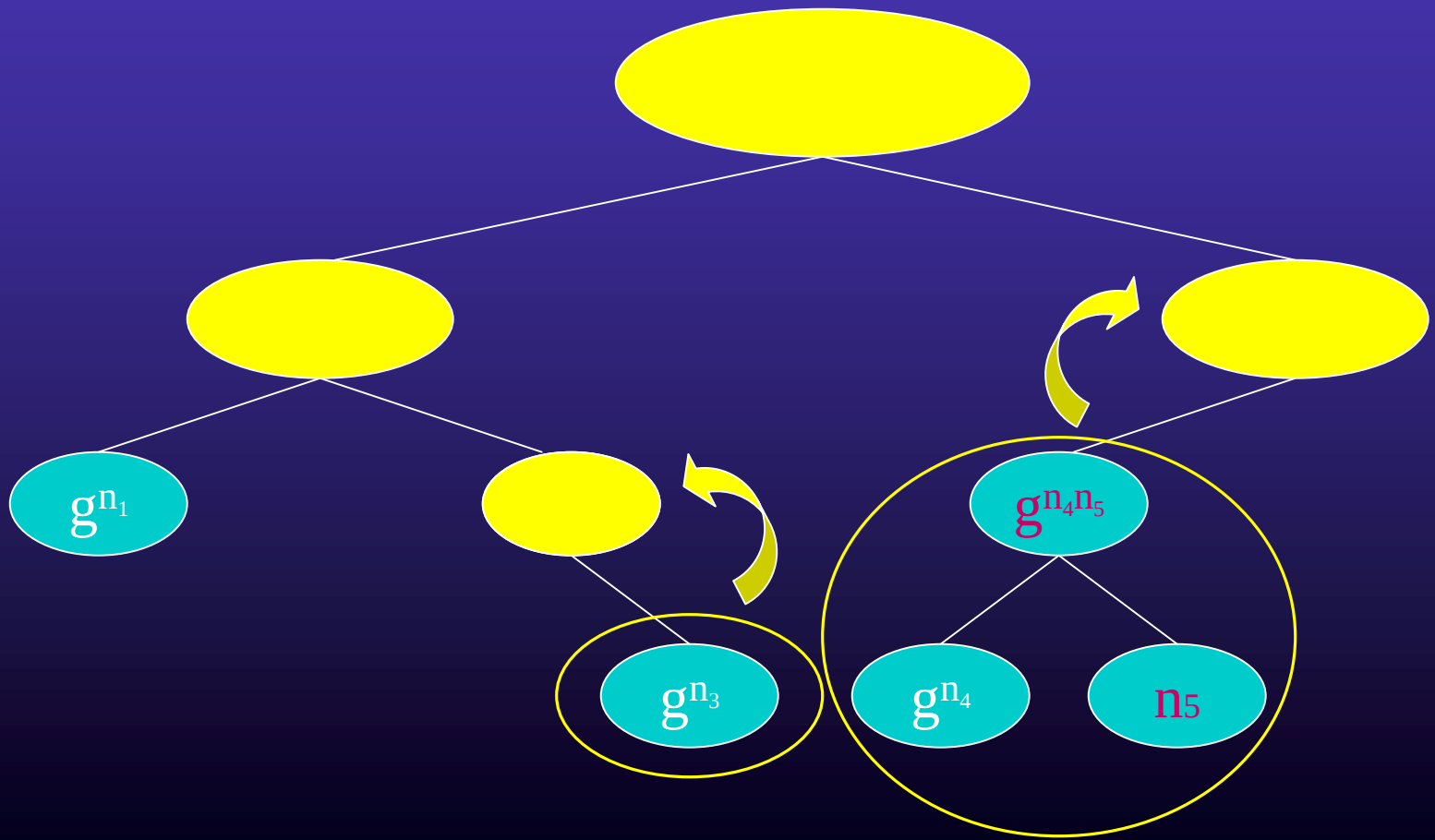
Merge III



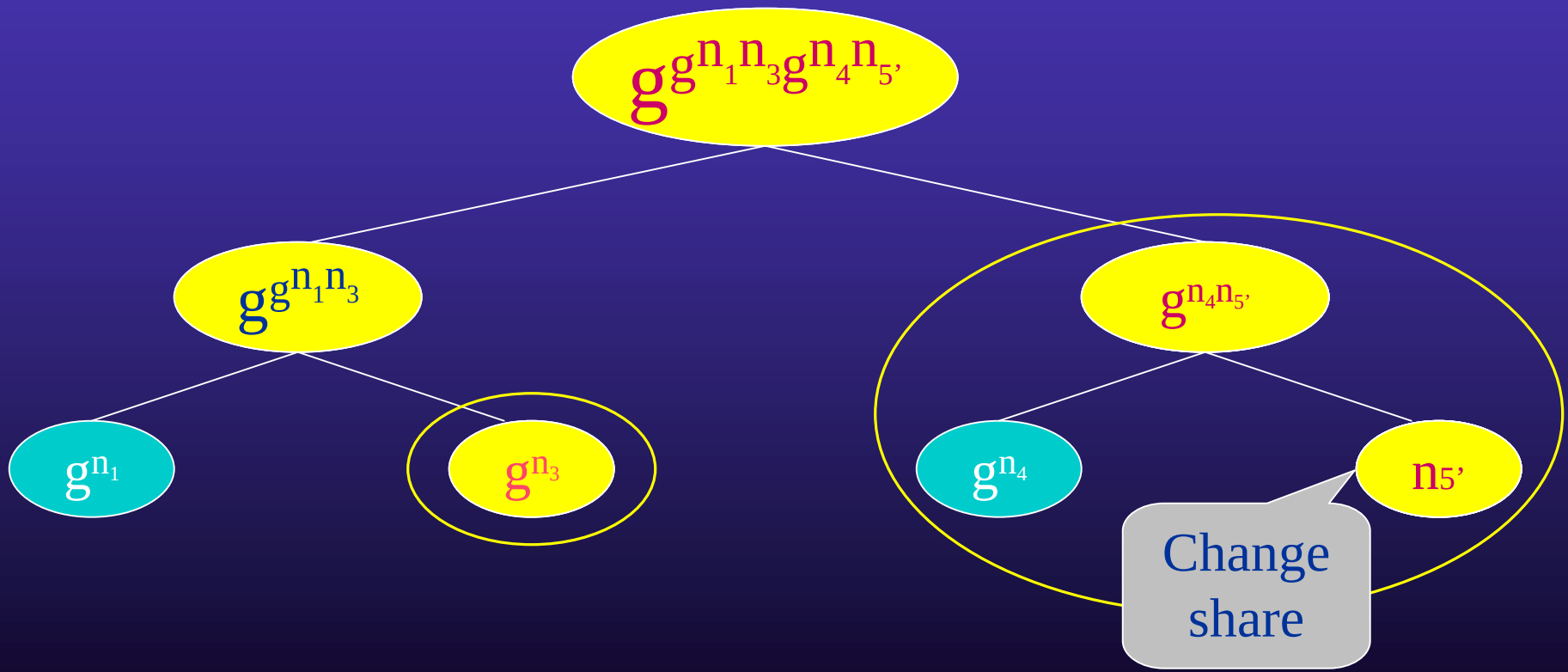
Partition (n_5's view)



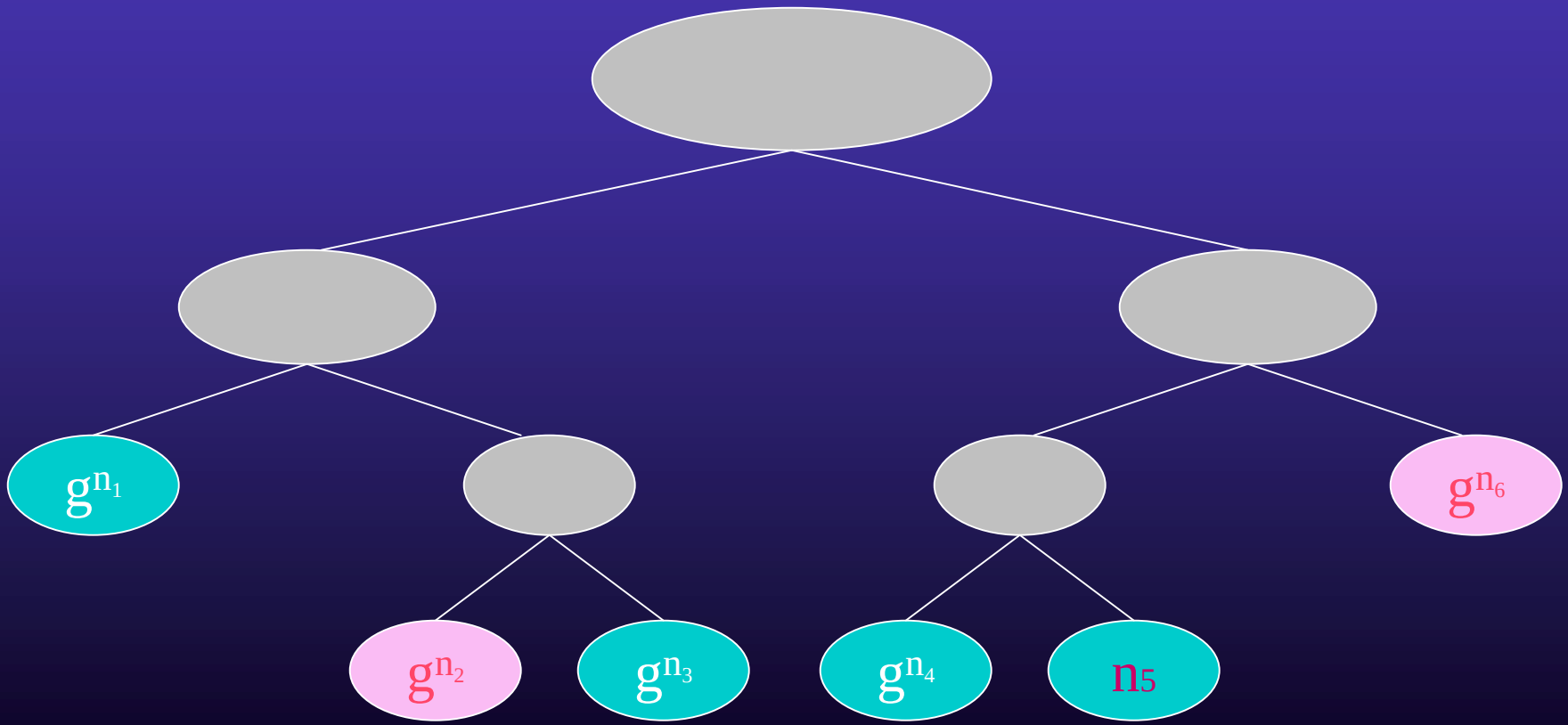
Partition (n_5's view)



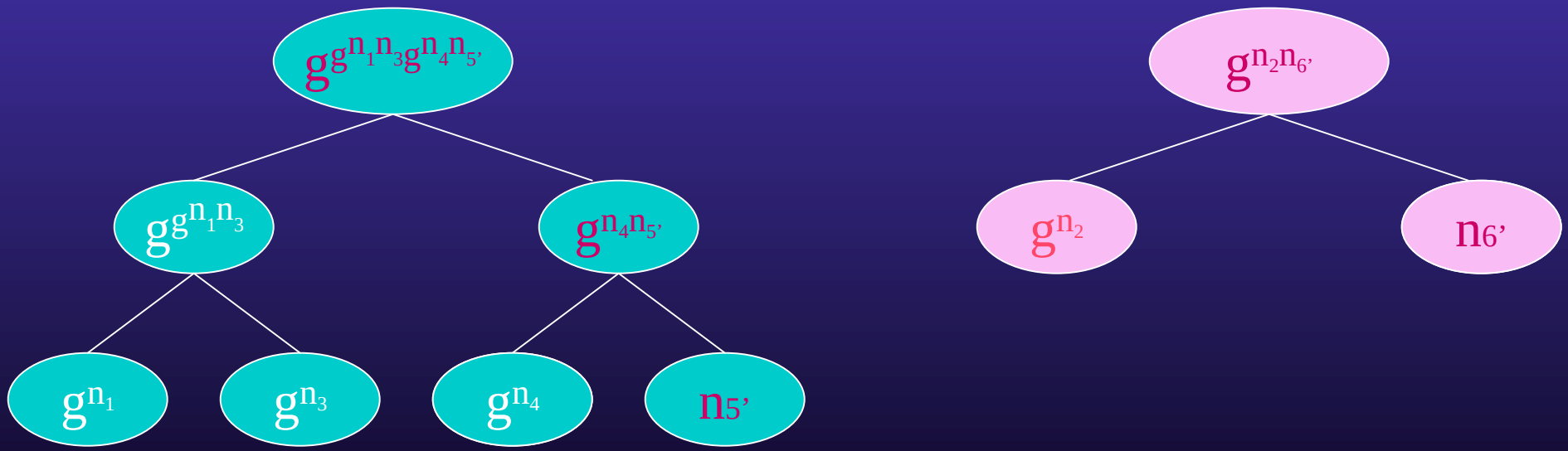
Partition (n_5's view)



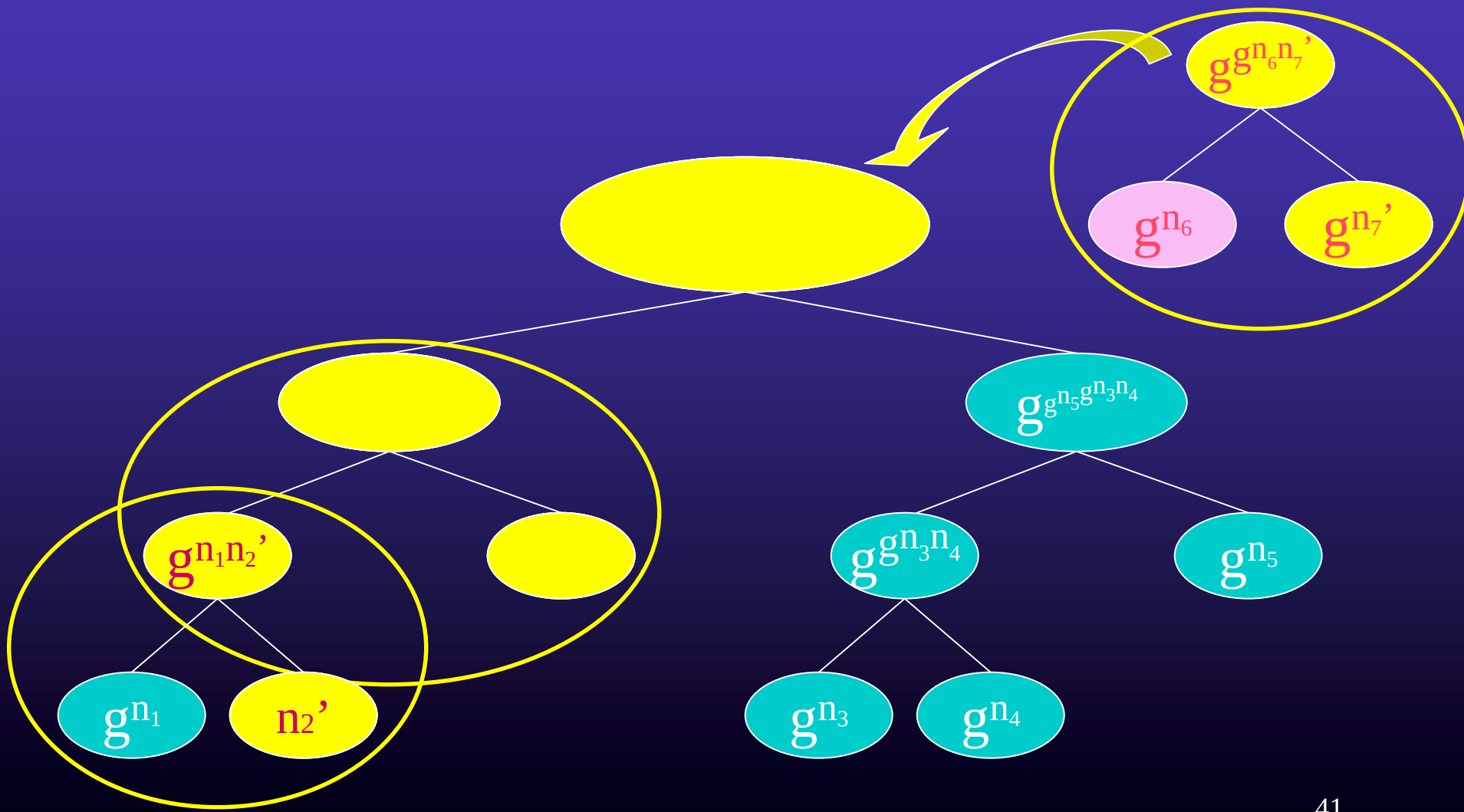
Partition: Both Sides



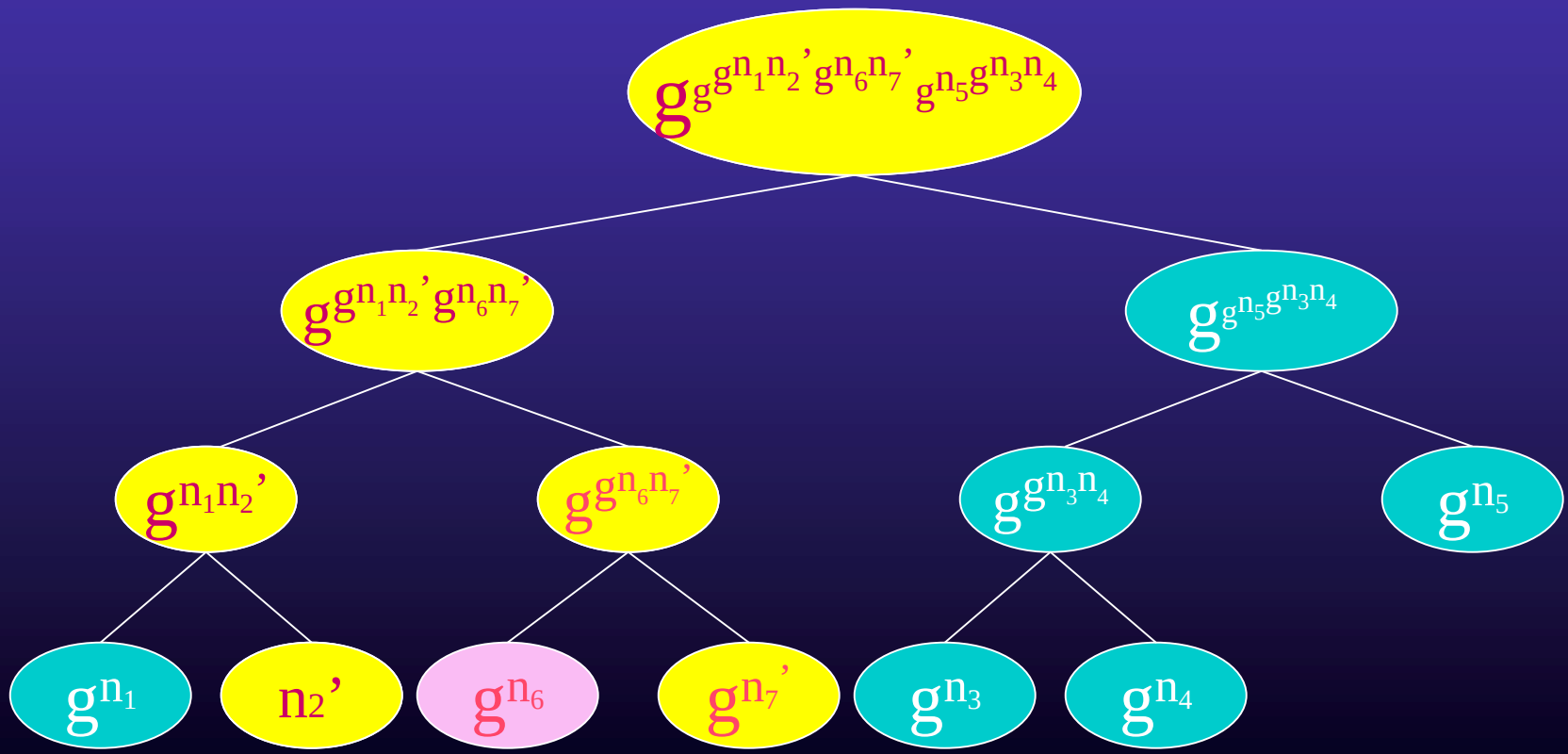
Partition: Both sides (N_5 and N_6's view)



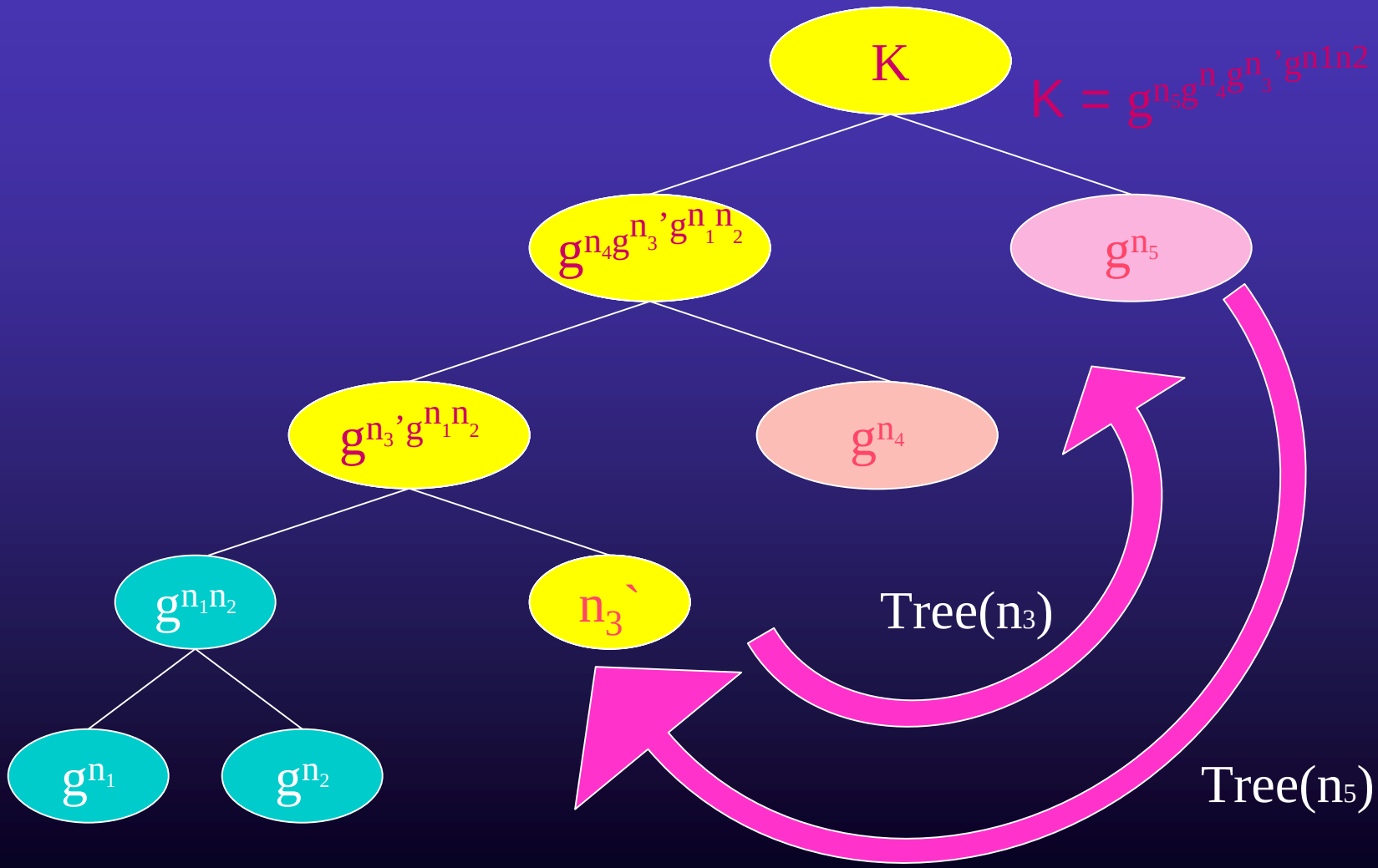
Merge (to intermediate node, N_2's view)



Merge (to intermediate node)



Merge


K
$K = g^{n_5} g^{n_4} g^{n_3} {}' g^{n_1 n_2}$
$g^{n_4} g^{n_3} {}' g^{n_1 n_2}$
g^{n_5}
$g^{n_3} {}' g^{n_1 n_2}$
g^{n_4}
$g^{n_1 n_2}$
n_3`
g^{n_1}
g^{n_2}
Tree(n_3)
Tree(n_5)