Group Key Agreement

Ph.D. Dissertation Proposal

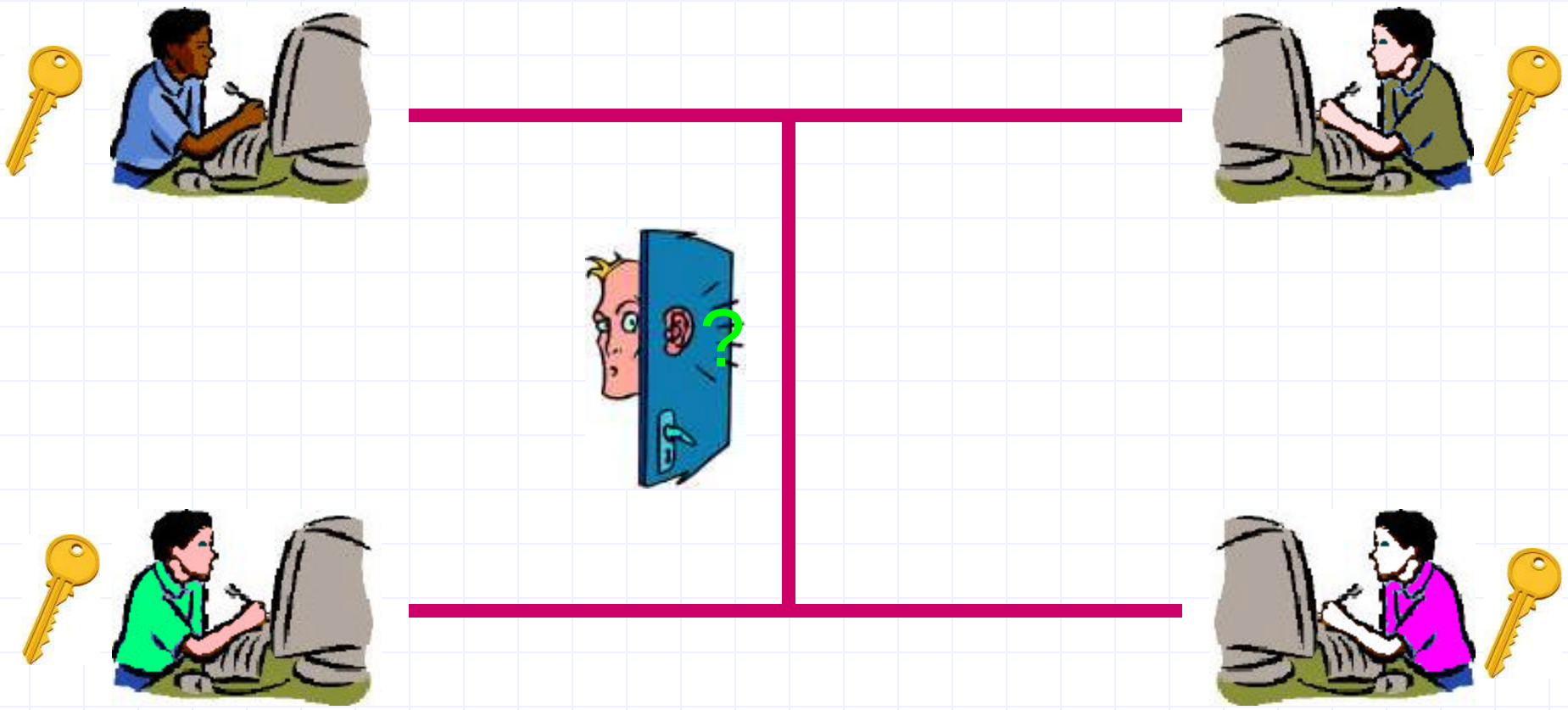
June 20, 2001

Yongdae Kim

Outline

- Definitions and concepts
- Motivations and goals
- Related work
- Work done
  - Protocols
  - Implementation and Integration
- Research plan and expected contribution

Background



Group Communication Settings

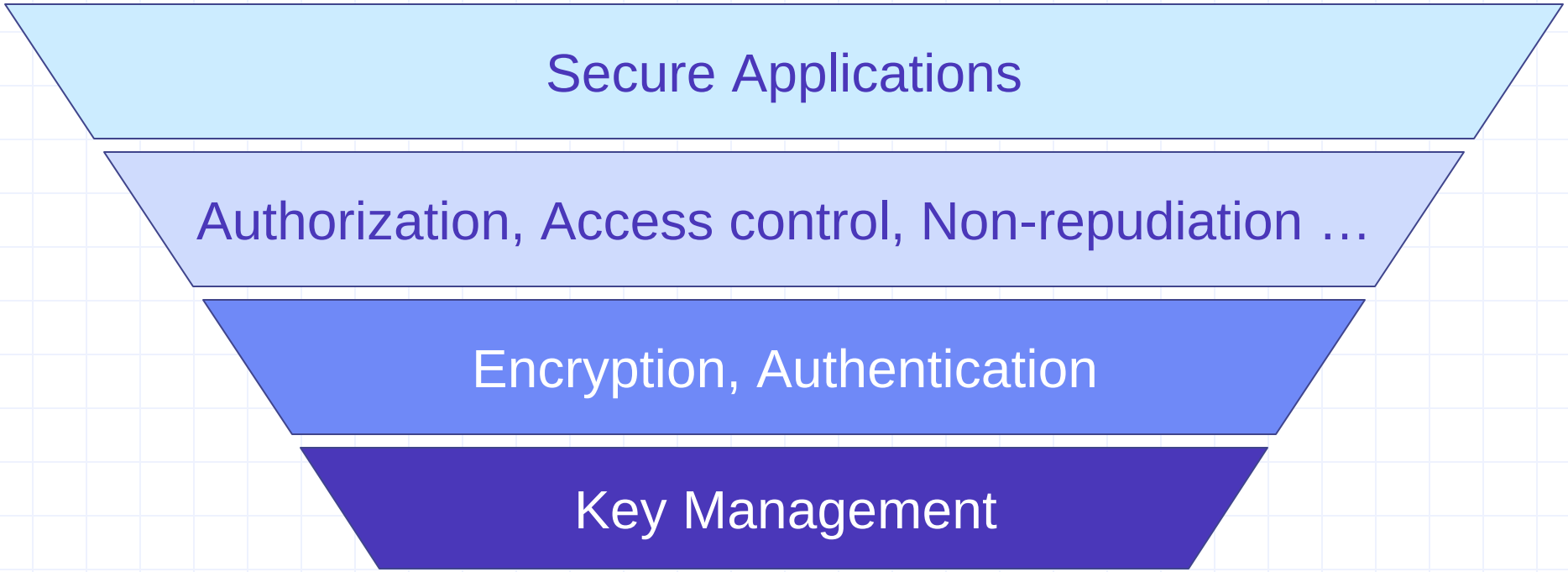
- 1-to-Many
  - Single-source broadcast: Cable/sat., TV, radio
- Few-to-Many
  - Multi-source broadcast: Televised debates, GPS
- Any-to-Any
  - Collaborative applications need inherently underlying peer groups.
  - Video/Audio conferencing, collaborative workspaces, interactive chat, network games and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

Dynamic Peer Groups (DPG)

- Relatively small (<100 of members)
- No hierarchy
- Frequent membership changes
- Any member can be sender and receiver

My focus: key management in DPGs

Key Management is a building block

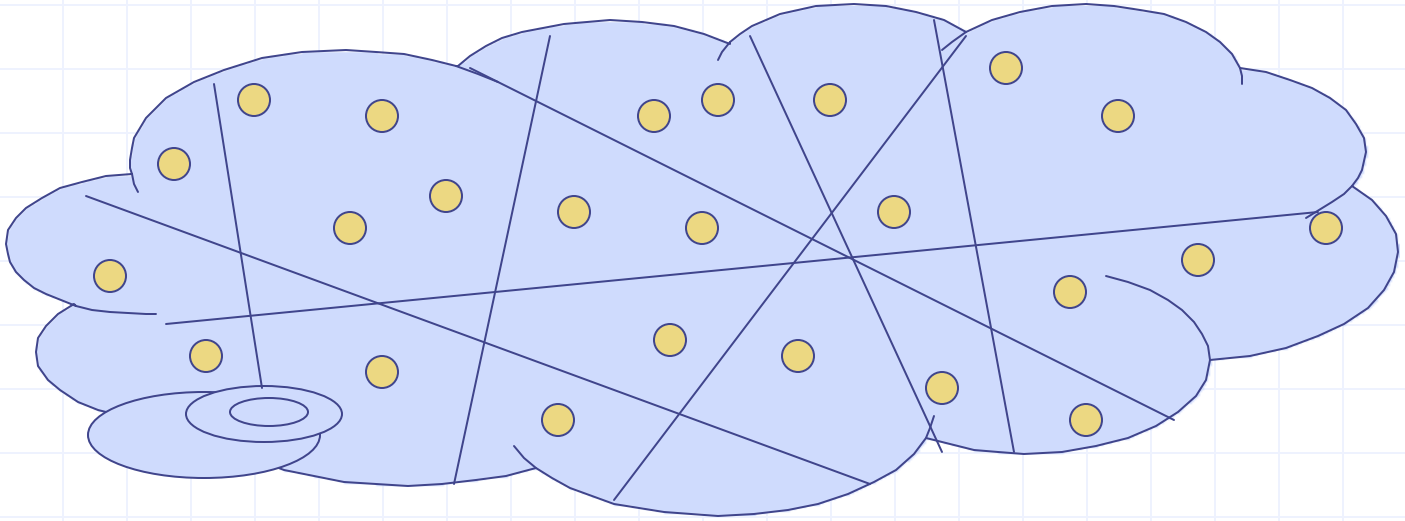


Group Key Management

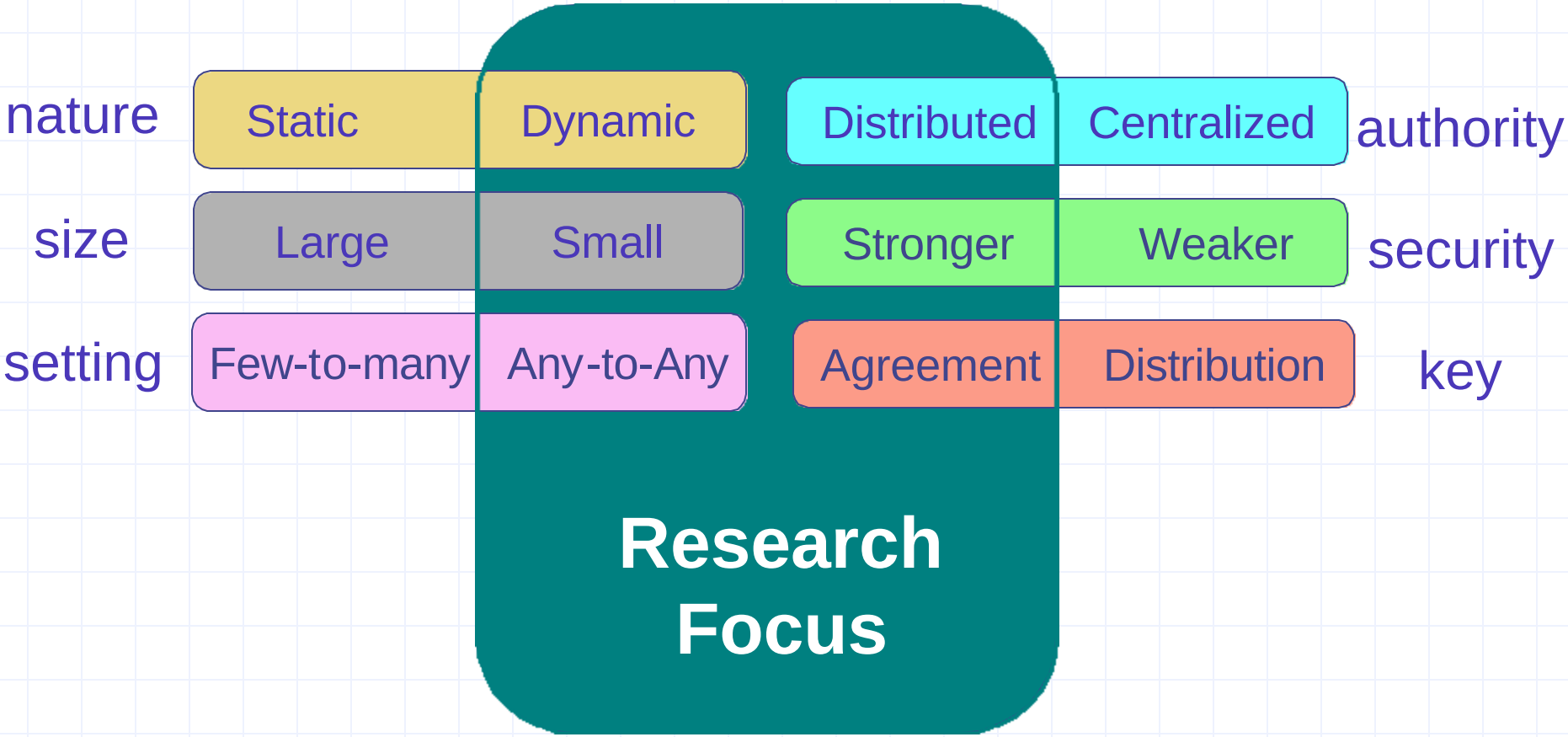
- Group key: a secret quantity known only to current group members
- Group Key Distribution
  - One party generates a secret key and distributes to others.
- Group Key Agreement
  - Secret key is derived jointly by two or more parties.
  - Key is a function of information contributed by each member.
  - 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!



Settings for Group Key Management

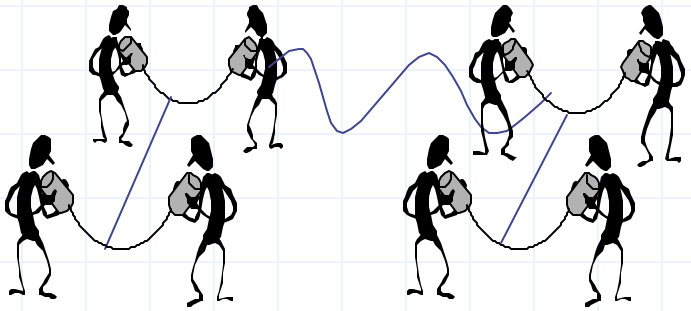


Secure Group Communication

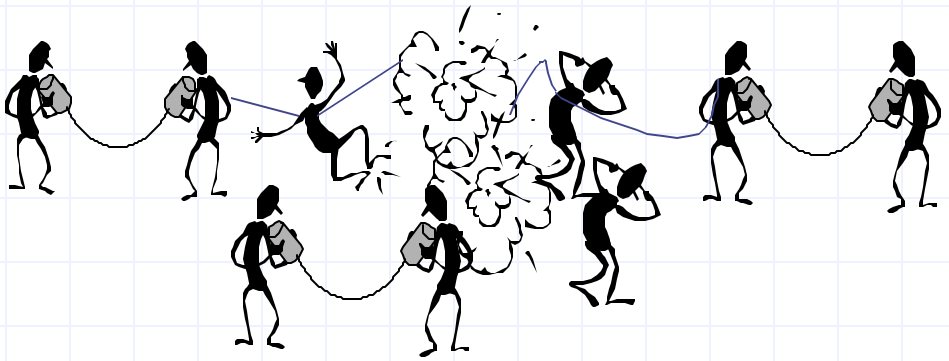
- Group key agreement protocols rely on the underlying group communication systems.
  - Protocol message transport
  - Strong membership semantics (Notification of a group membership)
  - Not for security reasons
- Group communication system needs specialized security mechanisms.

Mutual benefit and interdependency

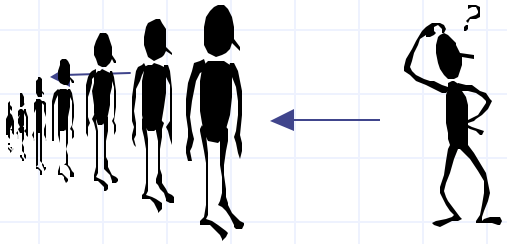
Membership Operations



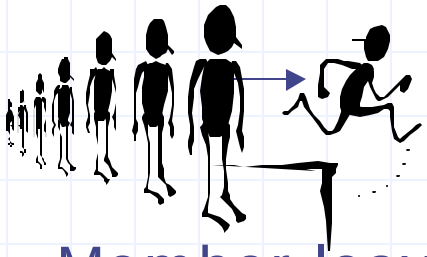
Formation



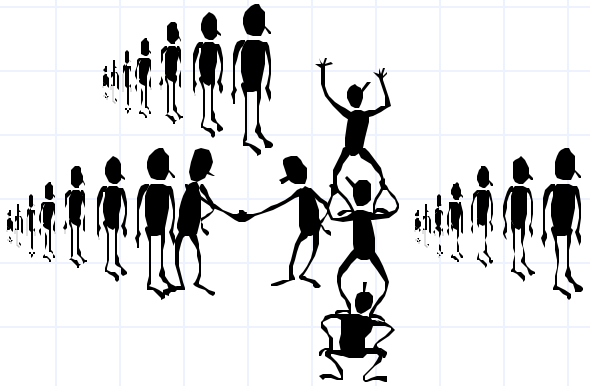
Group partition



Member add



Member leave



Group merge

Motivation

- We need group key agreement methods satisfying the following:
  - Strong security
  - Dynamic operation
  - Robustness
  - Efficiency in communication and computation
  - Implementation, integration, and measurement

Why care about computation overhead?

- Most group key agreement methods rely on modular exponentiation.
  - 512 bit modular exponentiation on Pentium 400 Mhz = 2 msec
  - 1024 bit modular exponentiation = 8 msec
- Most methods require a lot of modular exponentiations for each membership operation.
  - Cliques: When current group size is *n* , join of a member to this group requires *2 n + 1* modular exponentiation.

Goals

- To design efficient group key agreement protocols
  - Low communication and computation overhead
  - Suitable for various network environments
- Rigorous proof of security
- Development of group key management software
- Integration with group communication systems
- Evaluation of the group key agreement methods

Security Requirements

- Group key secrecy
  - computationally infeasible for a passive adversary to discover any group key
- Backward secrecy
  - Any subset of group keys cannot be used to discover previous group keys.
- Forward secrecy
  - Any subset of group keys cannot be used to discover subsequent group keys.
- Key Independence
  - Any subset of group keys cannot be used to discover any other group keys.
  - Forward + Backward secrecy

Functional Requirements

- Group key agreement
- Dynamic membership operation
- Robustness against cascaded failures

Cascade faults: when a membership event occurs while handling prior one.

Outline

- Definitions and notions
- Motivation and goals
- Related work
- Work done
  - Protocols
  - Implementation and Integration
- Research and evaluation plan

Related Work

- 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

- Cliques project
  - DARPA-sponsored project (1997 ~ 2000)
  - Follow-on project from 2000 co-work with JHU
- *Cliques protocol: Foundation of the proposed work*
  - *Key Agreement in Dynamic Peer Groups (1996, 1997, 2000)*

    Steiner, Tsudik and Waidner

    Group Diffie-Hellman key agreement protocols

    Dynamic membership operations
  - *New Multi-party Authentication Services and Key Agreement Protocols (1998, 2000)*

    Ateniese, Steiner and Tsudik

    A notion of group key authentication is considered
  - Drawbacks

    Slow computation: O(n) computation for each membership event

    Communication overhead: k rounds for merge (k: # of new members)

Outline

- Definitions and notions
- Motivation and goals
- Related work
- Work done
  - Protocols
  - Implementation and Integration
- Research and evaluation plan

Work done: Protocols

- *Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups*
  - TGDH (Tree-based Group Diffie-Hellman)
  - Y. Kim, A. Perrig, G. Tsudik
  - ACM CCS 2000, Nov. 2000
  - Computation overhead reduced from O(n) to O(log n)
  - Providing robustness against cascaded failure inherently

- *Communication-Efficient Group Key Agreement*
  - STR
  - Y. Kim, A. Perrig, G. Tsudik
  - In submission
  - Communication overhead is lower than any other methods

Work done: Implementations

- *The Design of a Group Key Agreement API*
  - CLQ_API (Cliques Application Program Interface)
  - G. Ateniese, O. Chevassut, D. Hasse, Y. Kim, and G. Tsudik
  - DARPA DISCEX 2000, Jan. 2000
- Related APIs
  - TREE_API: Implementation of TGDH, May 2000
  - STR_API: Implementation of STR, June 2000
  - BD_API: Implementation of BD, Aug. 2000

Work done: Integration

- *Secure Group Communication in Asynchronous Networks with Failures: Integration and Experiments*
  - Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton and G. Tsudik
  - IEEE ICDCS 2000, April 2000
  - Integrating Cliques with Spread
  - Have some measurement ⇒ Will be used in our evaluation

- *Exploring Robustness in Group Key Agreement*
  - Y. Amir, C. Nita-Rotaru, Y. Kim, J. Schultz, J. Stanton and G. Tsudik
  - Accepted to IEEE ICDCS 2001
  - First paper which provides robustness in secure group communication

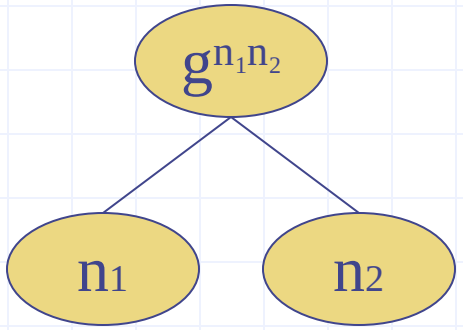
Outline

- Definitions and notions
- Motivation and goals
- Related work
- Work done
  - Protocols
  - Implementation and Integration
- Research and evaluation plan

Diffie-Hellman

- Setting
  - p – large prime (e.g. 512 or 1024 bits)
  - Zp* = {1, 2, … , p – 1}
  - g – base generator

- $A \rightarrow B : N_A = g^{n1} \bmod p$
- $B \rightarrow A : N_B = g^{n2} \bmod p$
- $A : N_B{}^{n1} = g^{n1n2} \bmod p$
- $B : N_A{}^{n2} = g^{n1n2} \bmod p$



- Diffie-Hellman Key : $g^{n1\ n2}$
- Blinded Key of n1 : $N_A = g^{n1} \bmod p$

Diffie-Hellman Problem

- Computational Diffie-Hellman Assumption (CDH)
  - Loose Definition: Having known g^a, g^b, computing g^{ab} is hard.
  - CDH is not sufficient to prove that Diffie-Hellman Key can be used as secret key.
    - Eve may recover part of information with some confidence
    - One cannot simply use bits of g^{ab} as a shared key
- Decision Diffie-Hellman Assumption (DDH)
  - Loose Definition

    Knowing g^a and g^b, and guessing g^c, can you check $g^c = g^{ab}$?
  - Stronger than CDH

Proof in Cryptography

- Common Assumption
  - Factorization is hard $\Rightarrow$ RSA
  - Computing discrete logarithm is hard $\Rightarrow$ ElGamal
  - DDH problem is hard $\Rightarrow$ Diffie-Hellman, Group key agreement methods
- We usually prove that the given problem can be formally reduced to a known common assumption.
  - If our system is broken, then the common assumption will be broken.

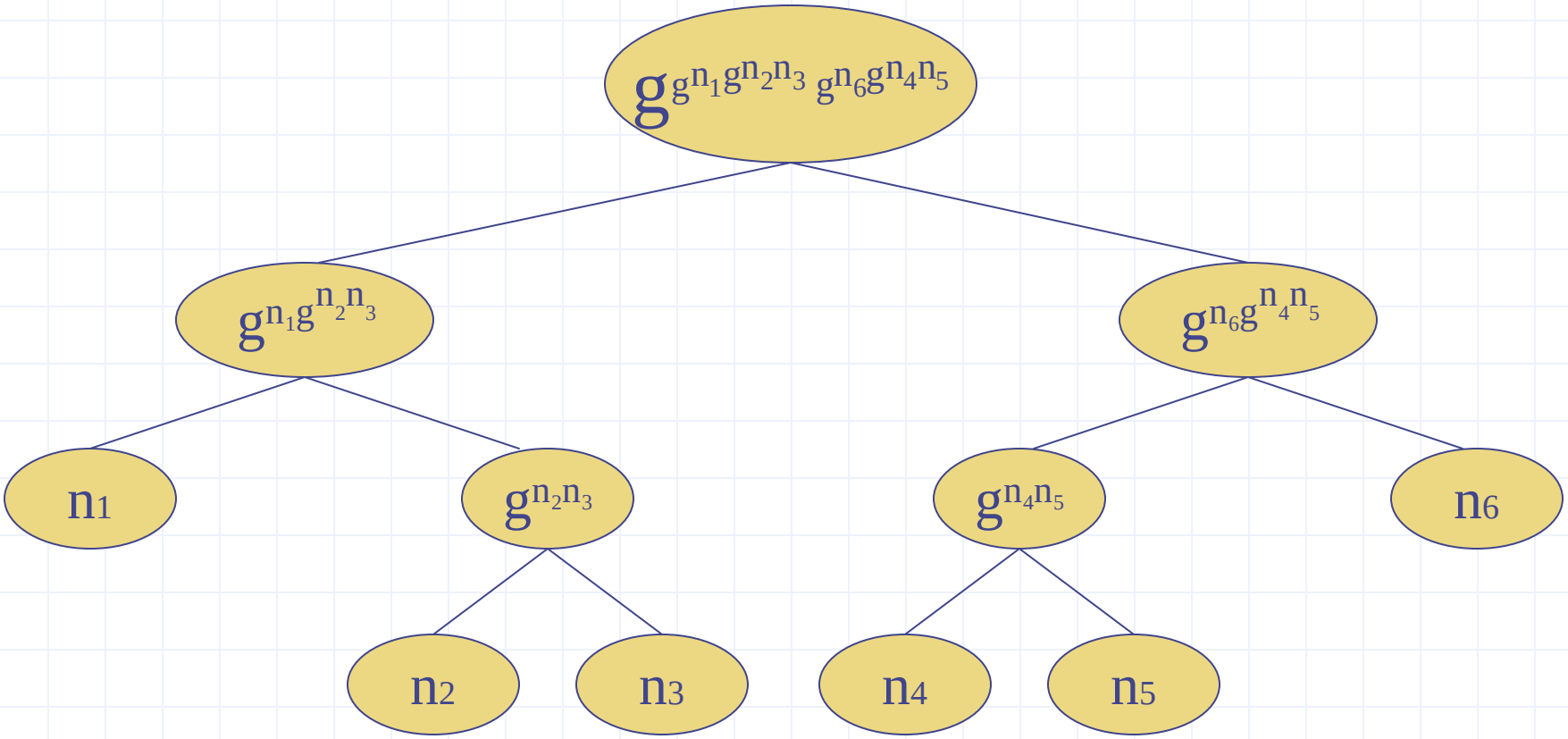
Cliques

- Steiner, Tsudik, and Waidner in ACM CCS ’96
- Contributory group key agreement protocol
- Security
  - Formal proof of security
  - Authentication
  - Key Independence
- Efficiency
  - Small communication round except merge
- Introduce dynamic group operation

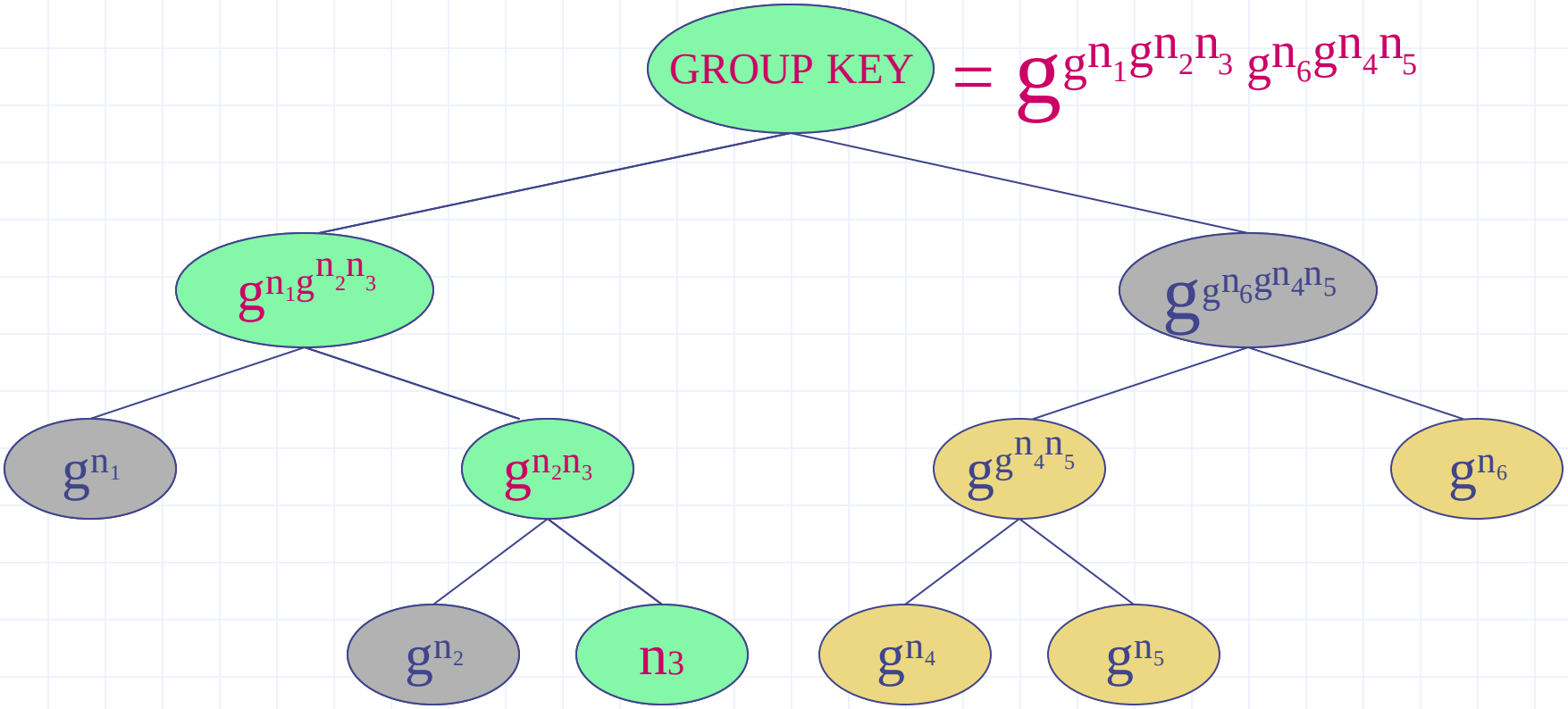
TGDH

- Simple: One function is enough to implement it
- Fault-tolerant: Robust against cascaded faults
- Secure
  - Contributory
  - Provable security
  - Key independence
- Efficient
  - d is the height of key tree (< $O(\log_2 N)$), N is the number of users
  - Maximum number of exponentiation = 4(d-1)
  - # of exp. in Cliques = 2N+1

Key Tree (General)

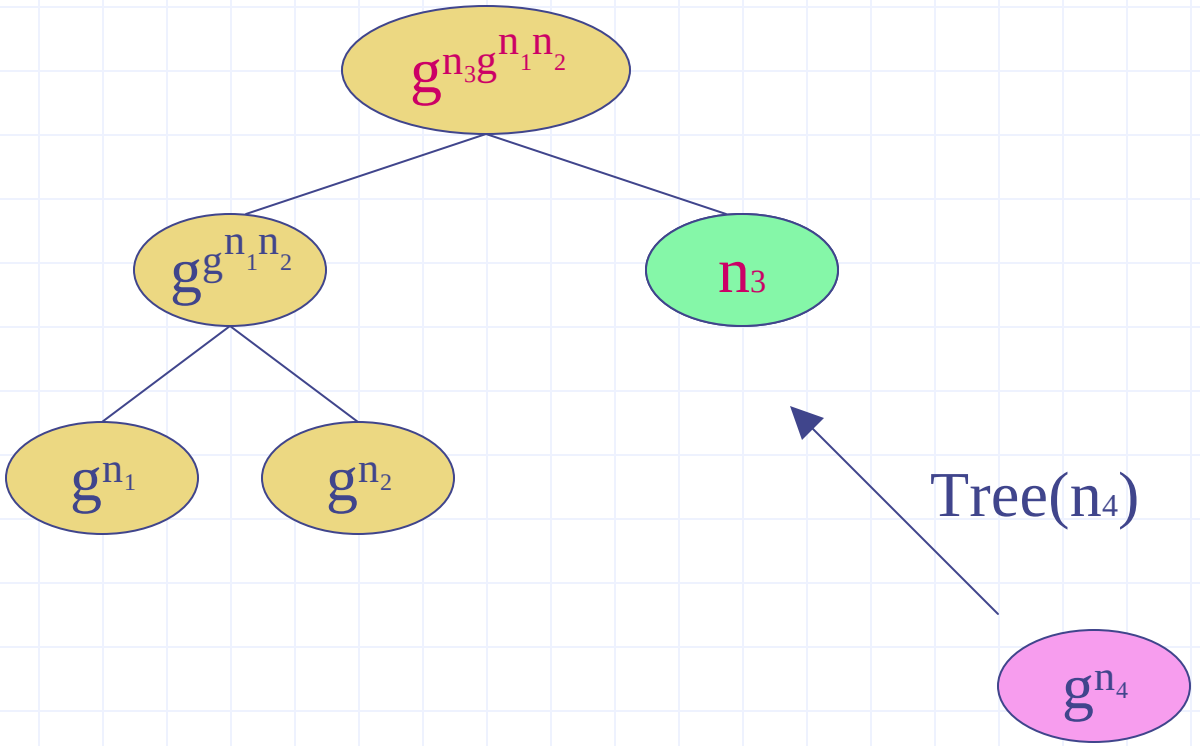


Key Tree (n_3's view)

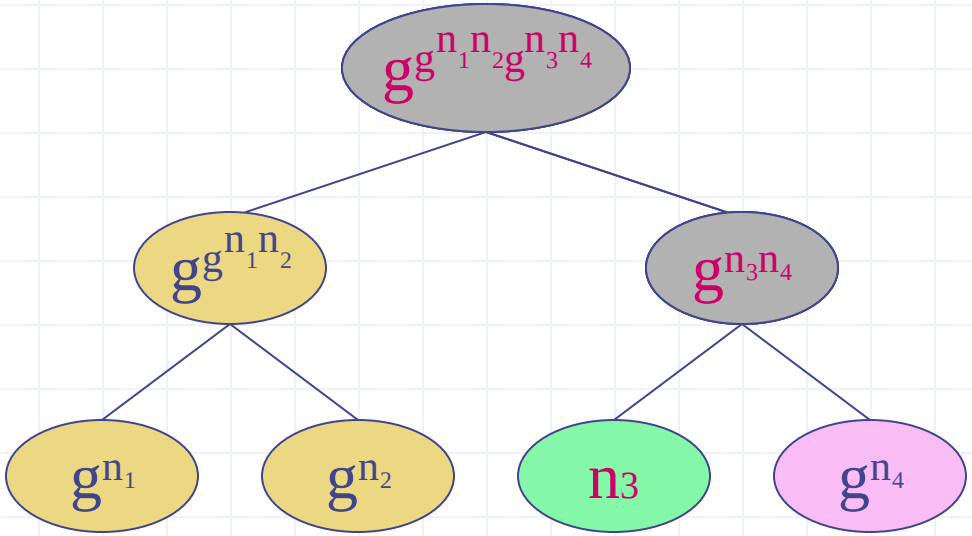


Any member who knows blinded keys on every nodes and its session random can compute the group key.

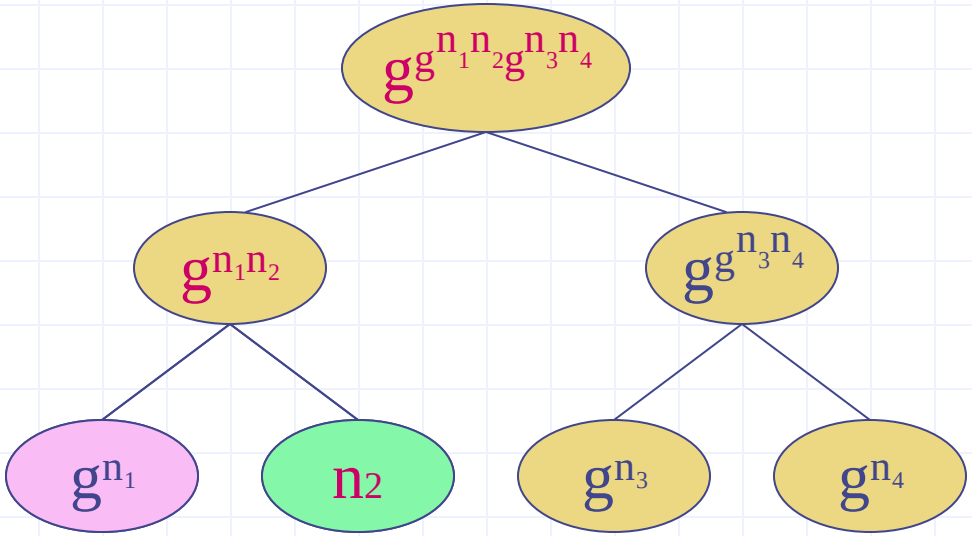
Join (n_3's view)



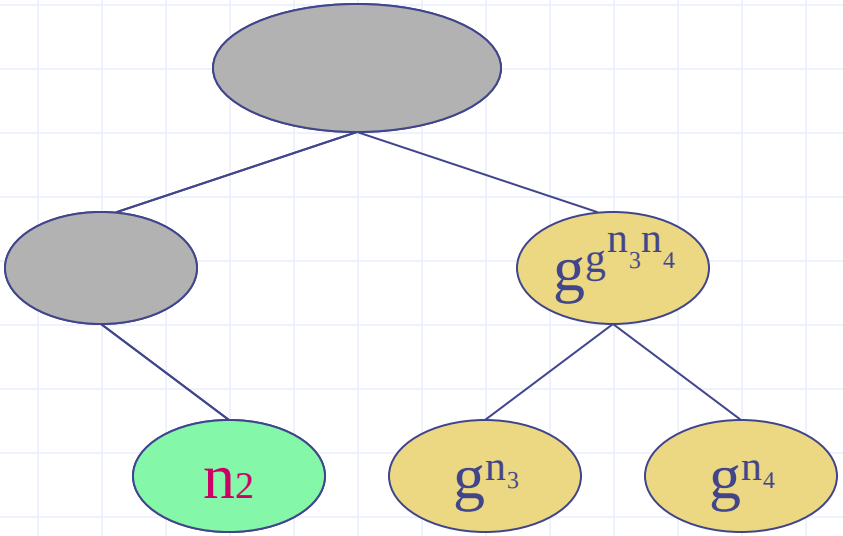
Join (n_3's view)



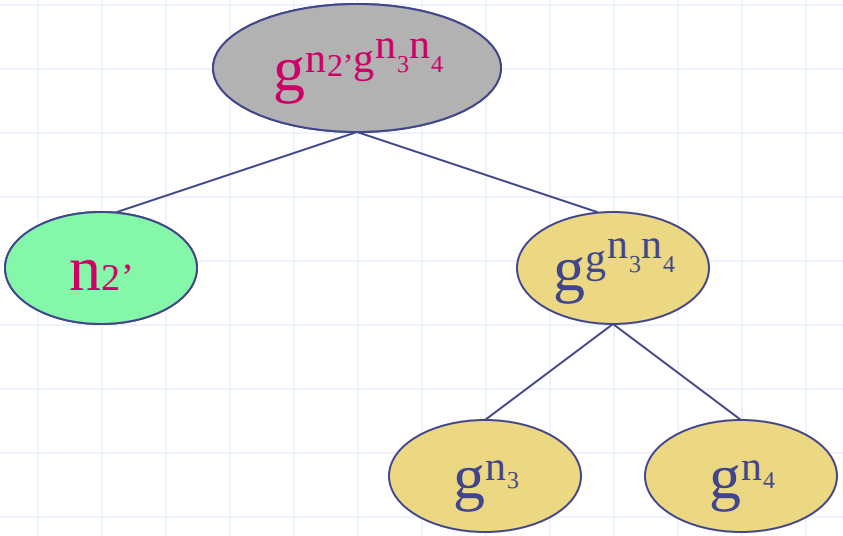
Leave (n_2's view)



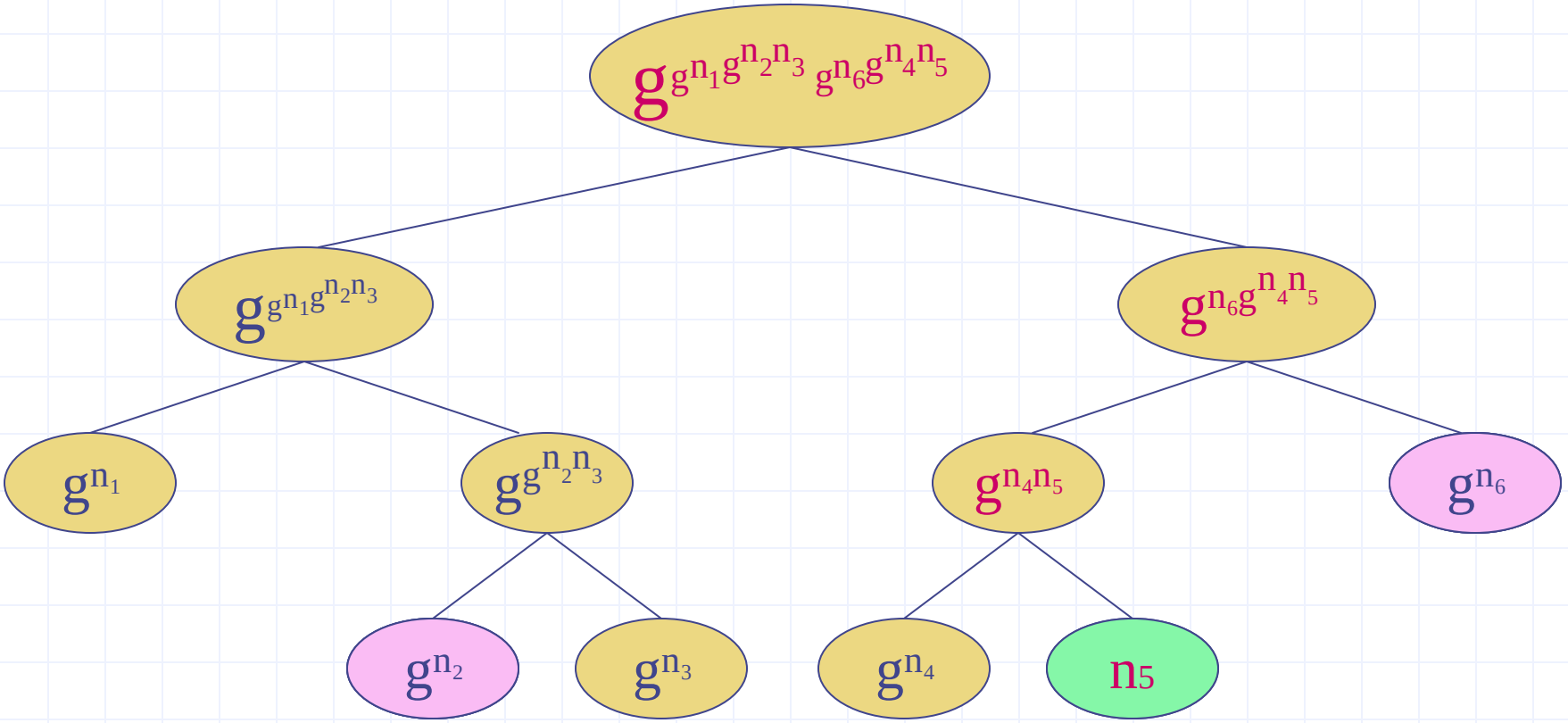
Leave (n_2's view)



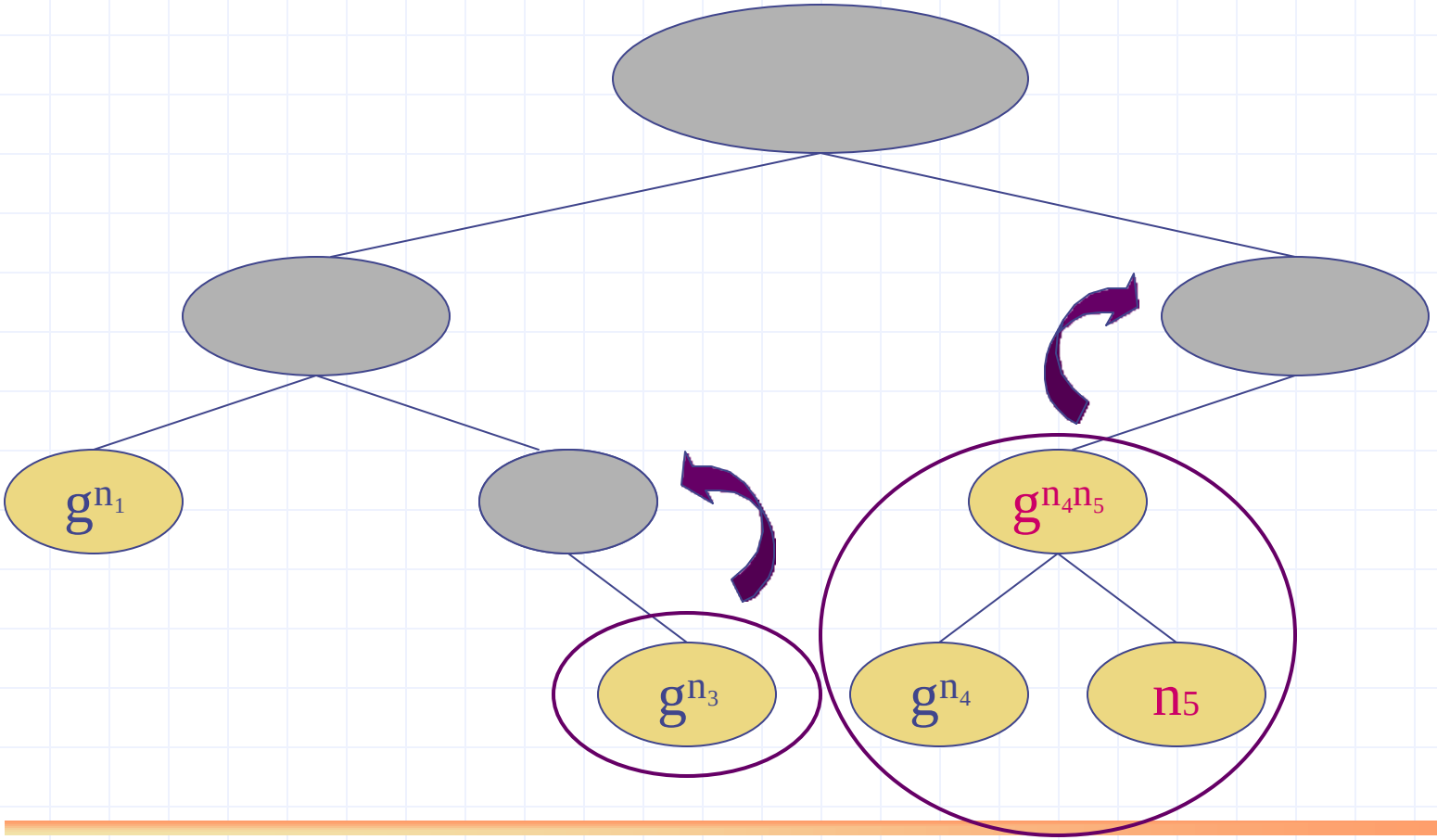
Leave (n_2's view)



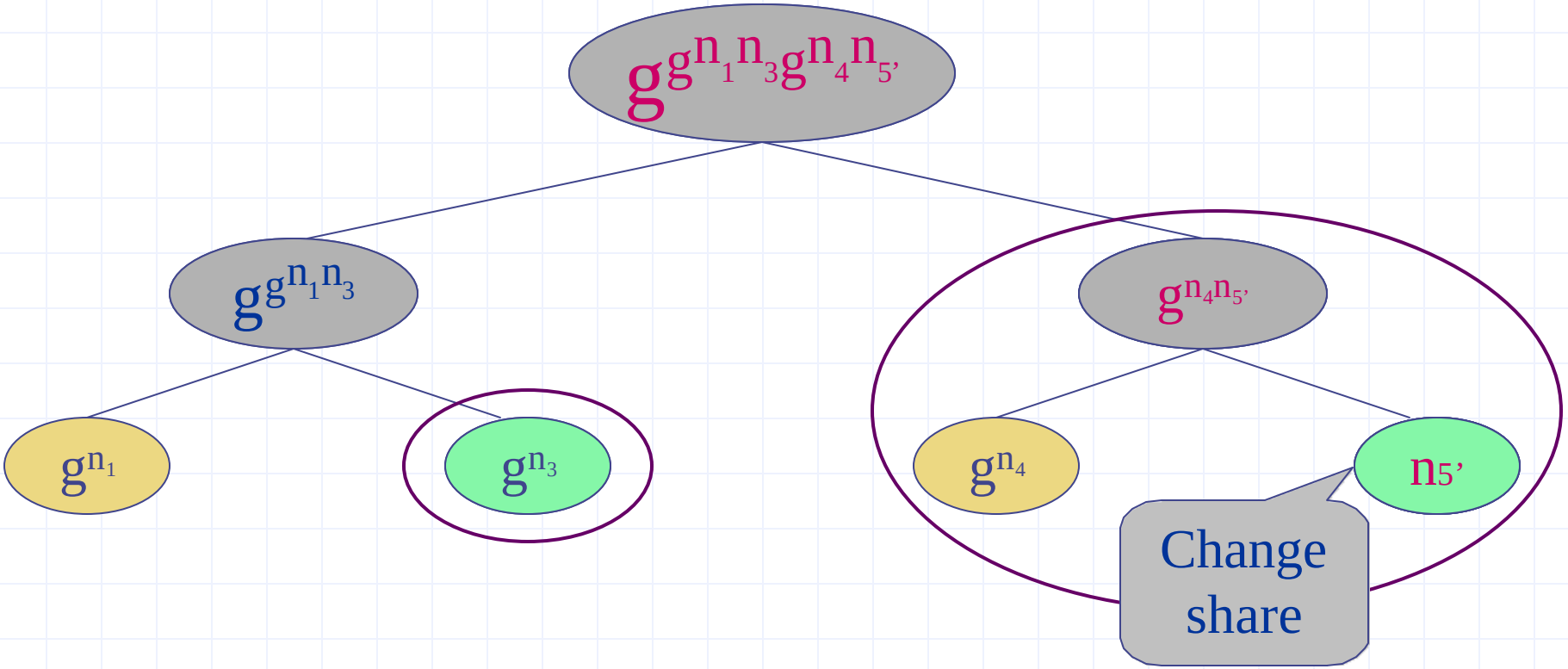
Partition (n_5's view)



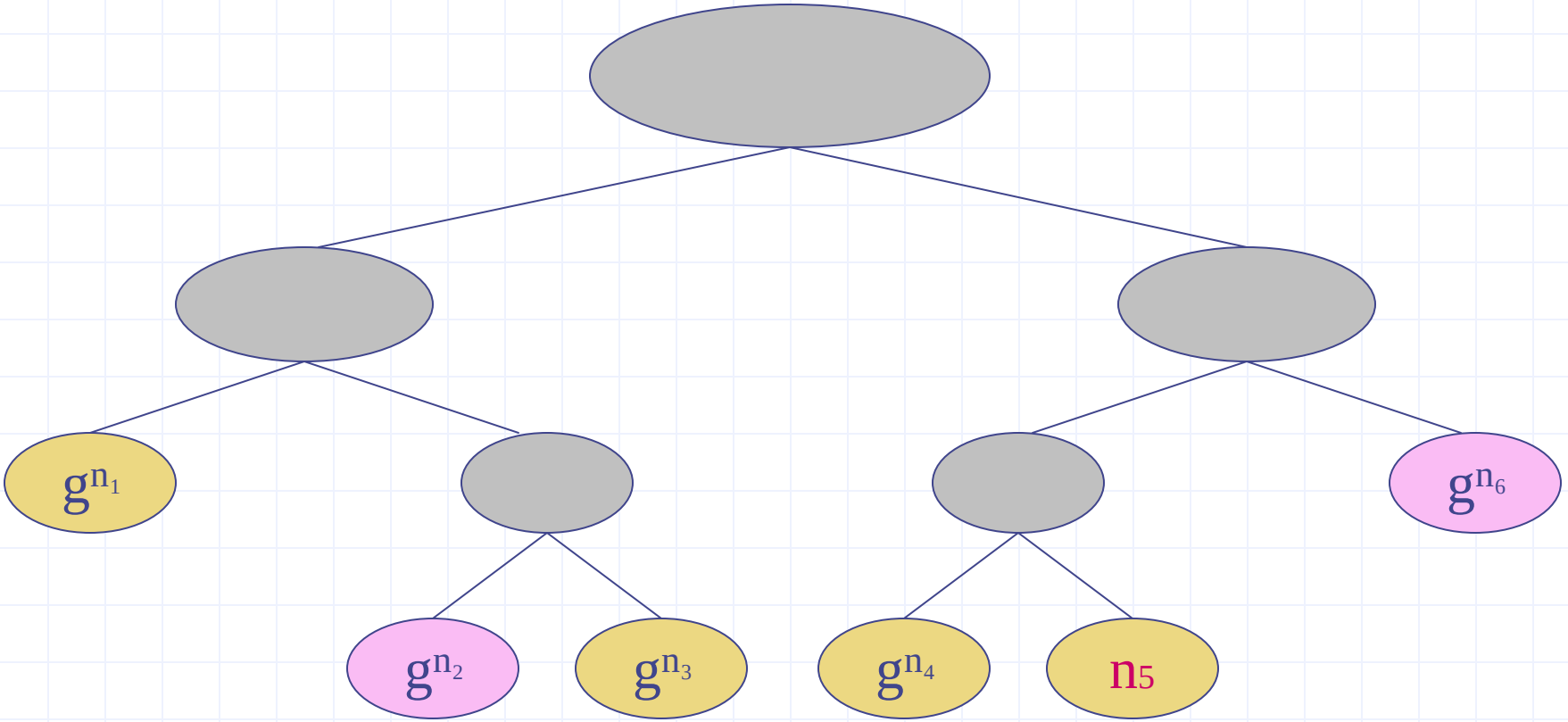
Partition (n_5's view)



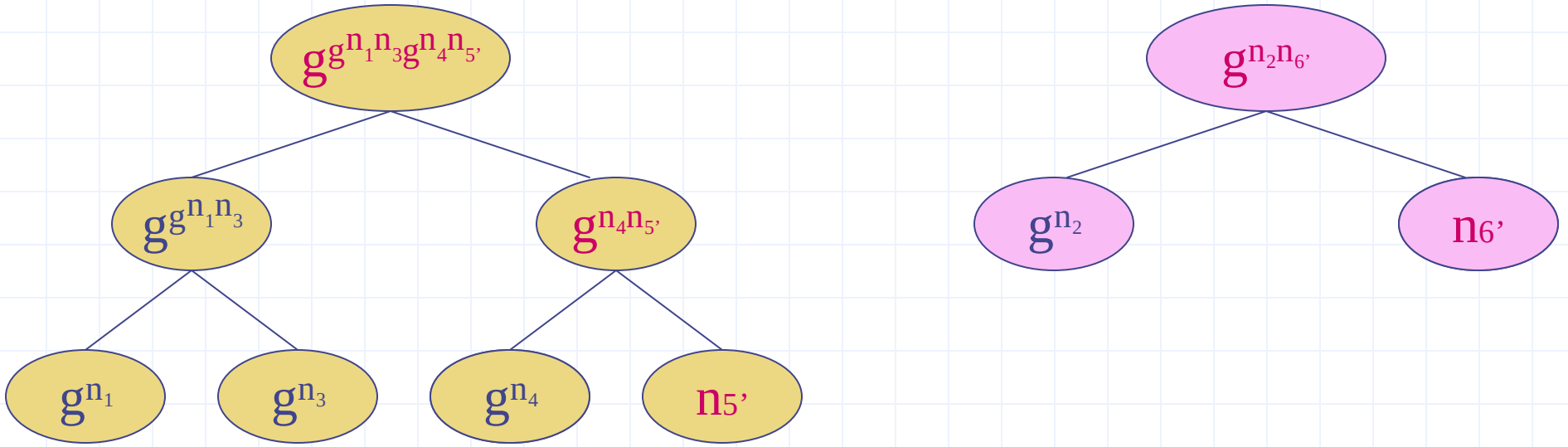
Partition (n_5's view)



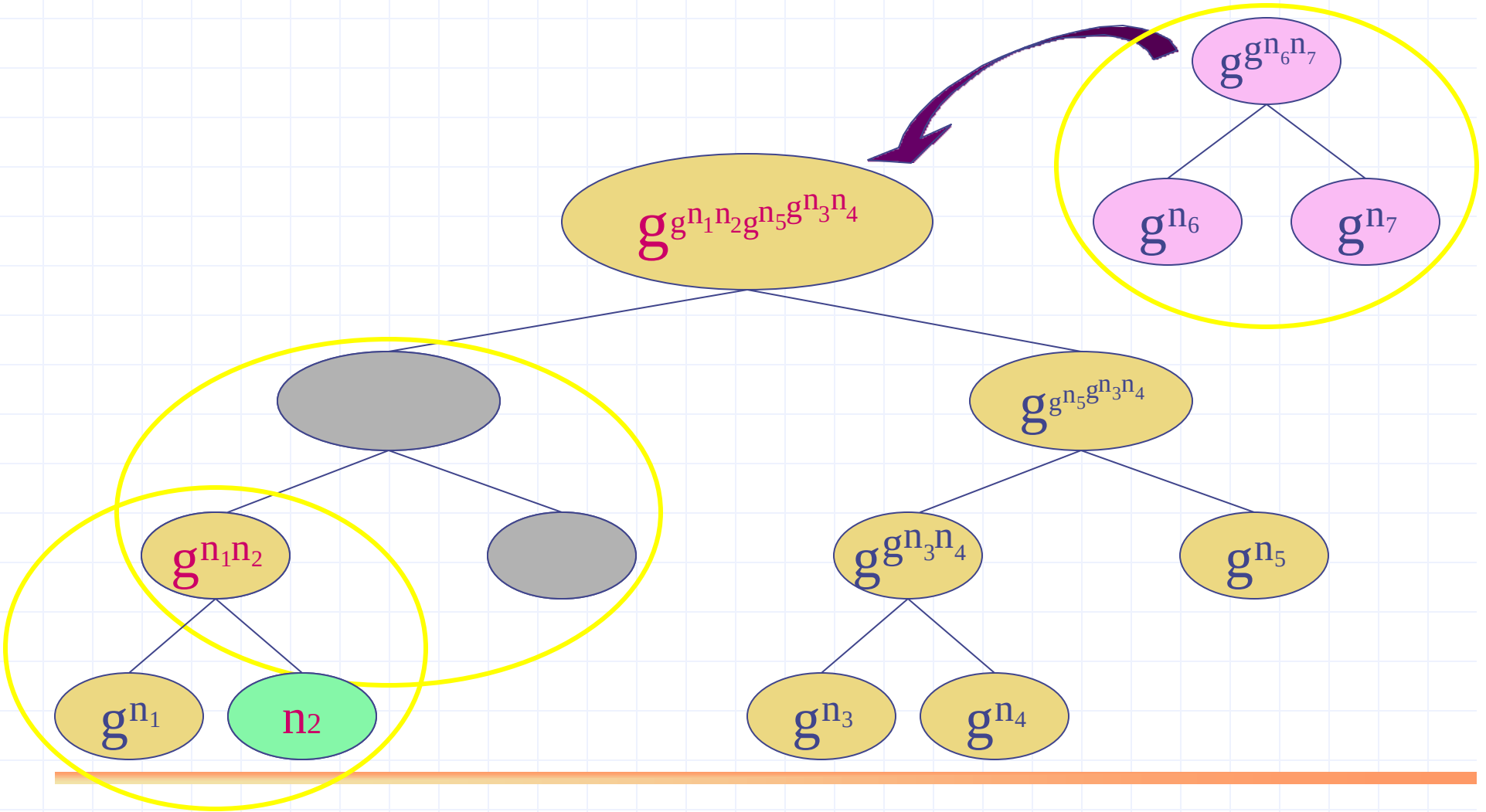
Partition: Both Sides



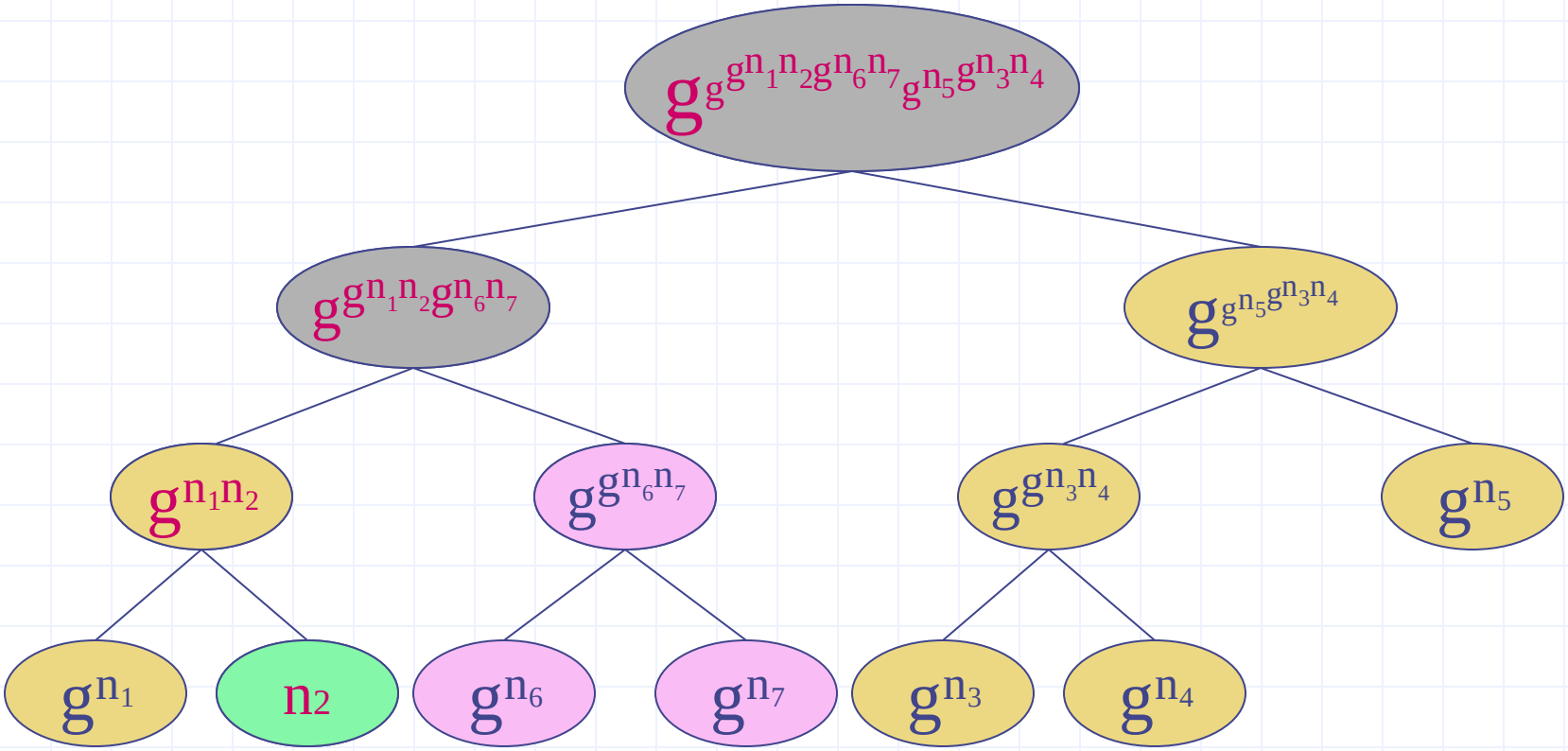
Partition: Both sides (N_5 and N_6's view)



Merge (to intermediate node, N_2's view)



Merge (to intermediate node)



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 < $2 \log_2 N$)
- Future Work
  - Other tree management technique
  - Rebalancing

Self-stabilization and Fault-tolerance

- Each protocol represents different strands of a single protocol.
- Every protocol consists of
  - Tree management (delete or add) for each membership event
  - Compute missing keys if I can
  - If I computed, broadcast my tree
- This leads us to design self-stabilizing protocol.
- If cascaded event finishes, then our protocol will also finish.

Security

- Group key secrecy
  - We have proofs using
    - Random Oracle Model
    - Becker and Wille's proof
  - Rigorous DDH proof $\Rightarrow$ next page
- We can provide key independence.
  - By changing session random of a member on every additive event

Tree DDH Problem

- Intuitive Definition

  Given all blinded keys of a random key tree, can we distinguish the group key with the random number?

- Proof goal

  If we can distinguish, we can distinguish 2-party DDH.

- We proved three-party case.

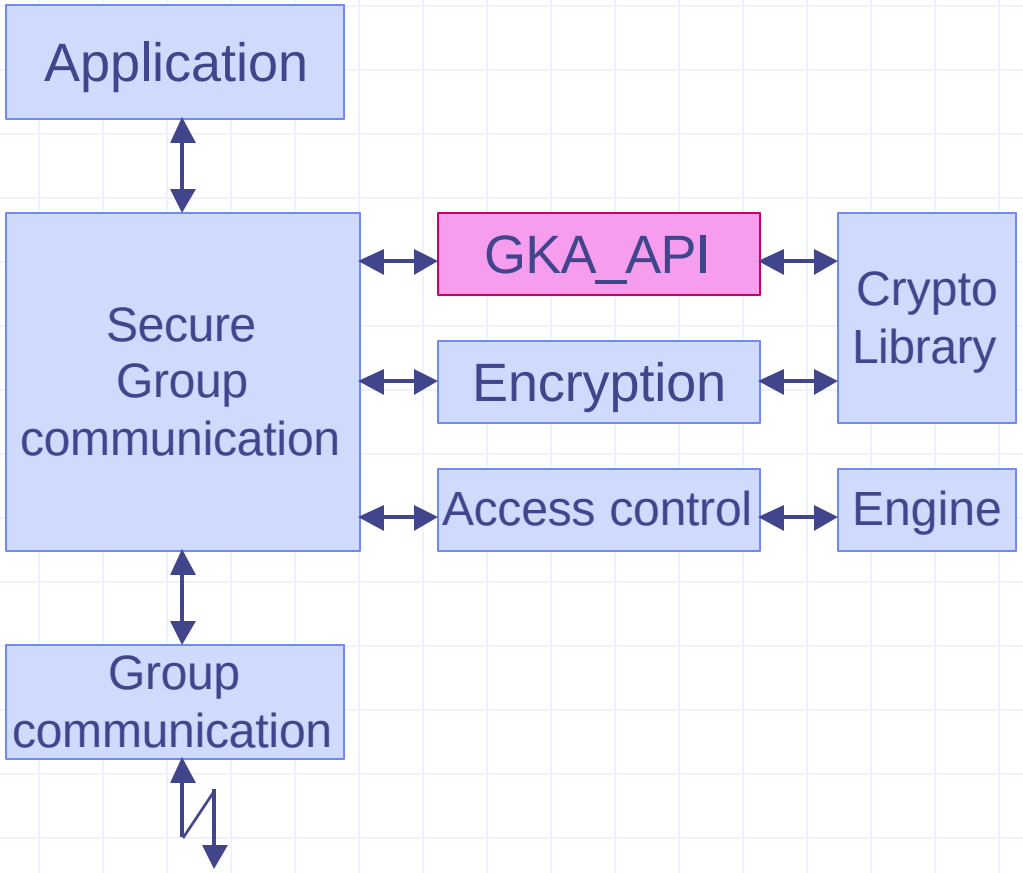
- Simple induction does not work.

STR

- Communication efficient
  - Maximum 2 communication round
  - Maximum 2 broadcast messages
- Simple: One function is enough to implement it.
- Fault-tolerant: Easier than TGDH
- Secure
  - Contributory
  - Backward and forward secrecy
  - Provable security
  - Key independence
- Computation is bit more expensive.
  - Maximum number of exponentiation = 4(N-1)
  - N is the number of users.

Implementation

- OpenSSL
- Group communication system
  - Message transport
  - Membership control
- Access control and encryption are out of scope



Research Plan:

- Evaluation:
  - Depending on application, different GKA may be used.
  - Based on complexity analysis and measurement
    - High/low delay network (LAN,WAN,Satellite,sensor,etc.)
    - CPU speed
    - Key size
  - We will measure the efficiency of our protocol after the integration.
    - Not whole APIs, but at least we will compare TGDH and Cliques.
  - We will collect some group dynamics from DoE and Spread.
    - Life time of a group
    - Dynamicity
    - Cascaded events

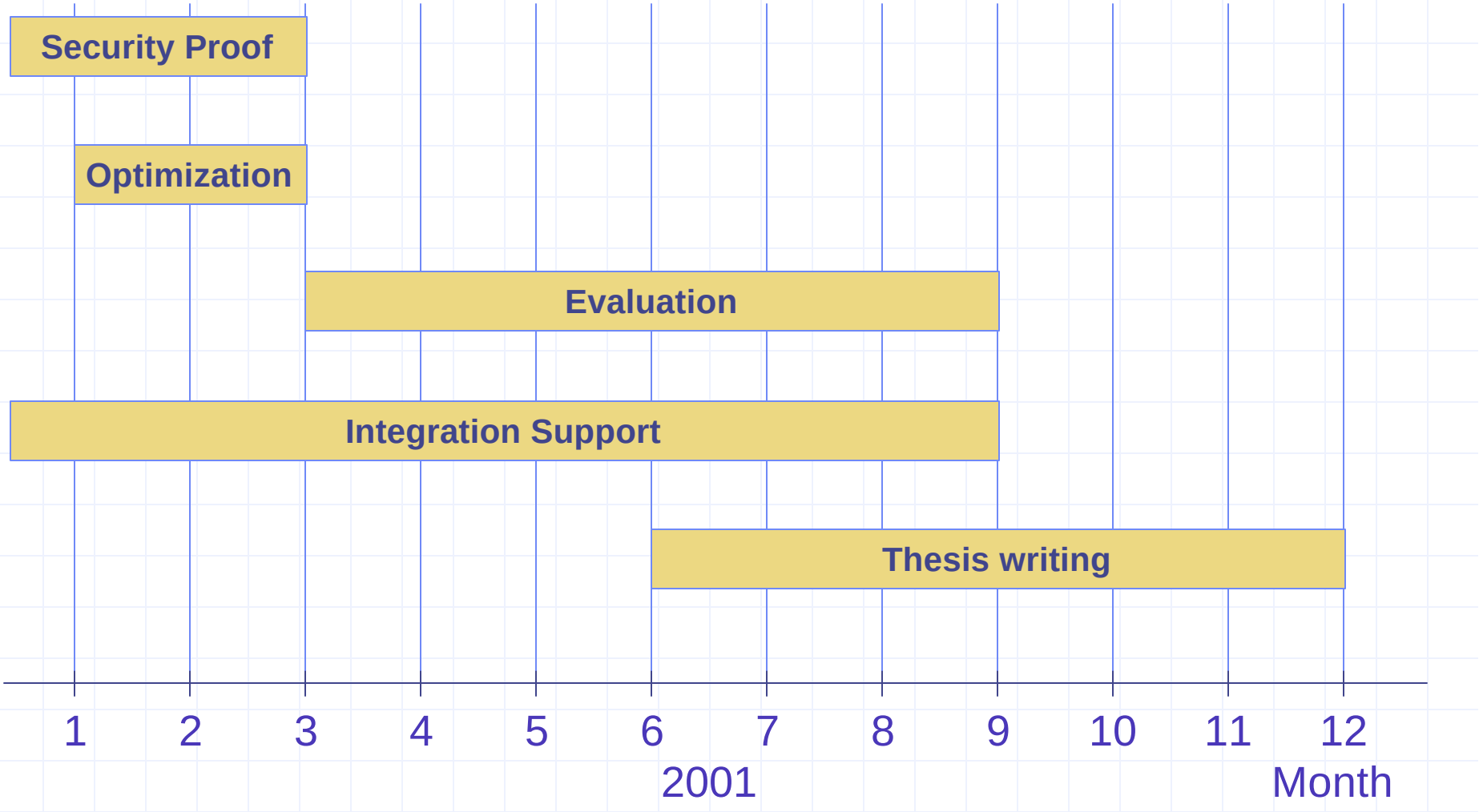
Research Plan:

- Theoretical to-do:
  - Complete security proofs (Tree-DDH problem)
- More Integration activities:
  - Depends on other parties (JHU, LBL)
  - Hybrid group key agreement (STR + TGDH?)

Summary

	Work Done	Future Work
Protocols	TGDH, STR*	?
Theoretical		Tree-DDH, Tree balancing
Implementation	CLQ_API, TREE_API, STR_API, BD_API	?
Integration	Cliques+Spread (Integration, Robust)	TGDH+Spread (STR+Spread?, Hybrid?)
Evaluation		Evaluation, Recommendation, Measurement

Time Line (optimistic)



Expected Contribution

- Practical and Provably Secure Group Key Agreements
- Proof of Security: Tree DDH equivalent to 2-party DDH
- Evaluation of Group Key Agreement Schemes
- Group Key Agreement API
- Integration with Group Communication System

End

- Thank you very much!

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 a single group
  - Network fault heal: previously disconnected partitions reconnect
  - Explicit merge: application decides to merge multiple pre-existing groups into a single group

Key Distribution vs. Agreement

	Key Distribution	Key Agreement
Key Generation	Center	Each member's contribution
Crypto Primitive	Secret key Encryption Hash and MAC	Diffie-Hellman variants
Communication	IP Multicast	Group communication
Computation Overhead	Low (very high for center)	High (Similar complexity)
Group Size	> 10,000	< 100
Contributory	No	Yes
Number of rounds	Single (no acks)	Multiple
Fault-tolerance	Supported	Required
Single point of attack?	Yes	No

Comparison

		Comm				Comp	Robust
		Round	Msg	Uni	Broad	Exp	
CLQ	Join	2	2	1	1	2n	Hard
	Leave, Partition	1	1	0	1	n	
	Merge	k+3	n+2k+1	n+2k-1	2	n+2k	
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	1	2	1	1	2	Easy
	Leave, Partition	1	1	0	1	2n	
	Merge	2	3	2	1	2	
BD		2	2n	2	2n	3	Easy

Motivation

- Why group communication system is important?
- Why it needs to be secured?
- Application of secure group communication system
- Is this a hard problem?
- Why focusing on DPG?

Evaluation Plan

- How can you say this work is successful?
- If you say “your group key agreement” is practical, why? If you have an

Self-stabilization

- Four protocols actually represent different strands of a single protocol

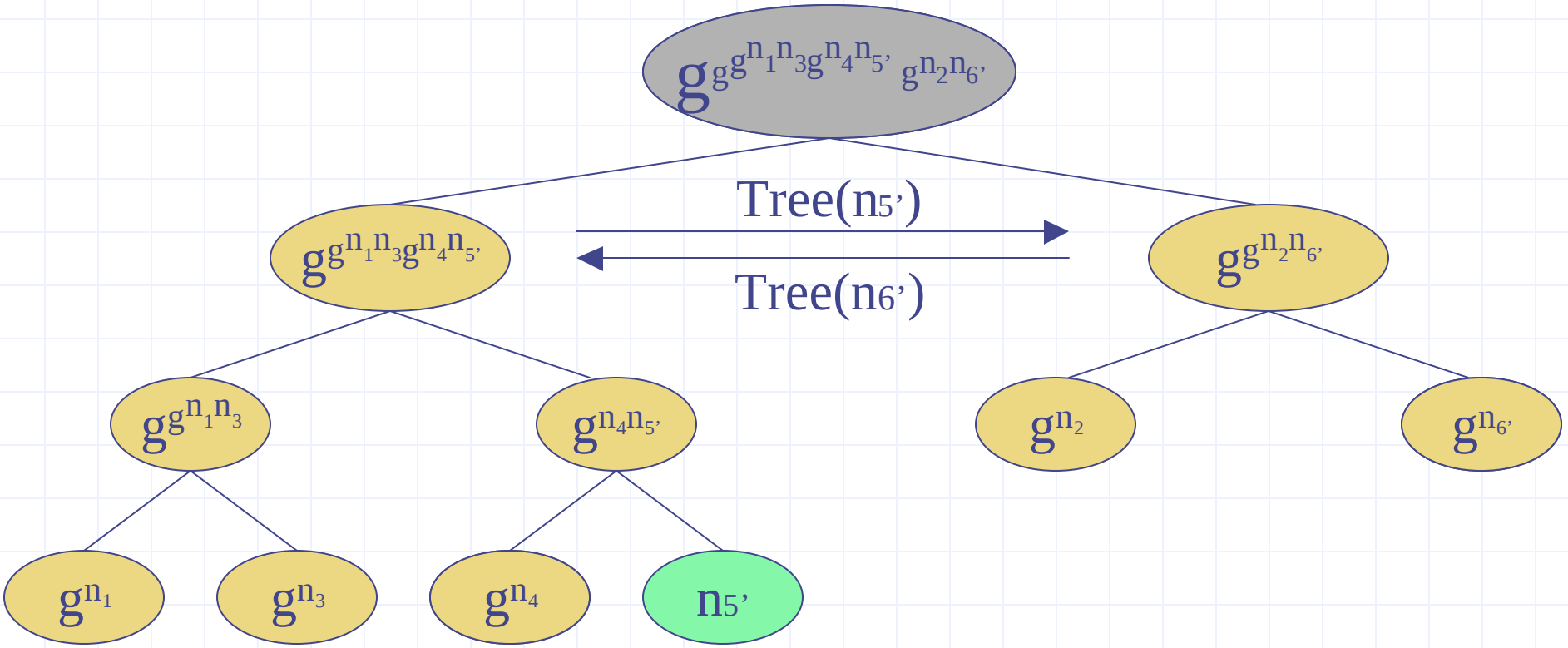
```
receive msg (msg type = membership event)
construct new tree
while there are missing blinded keys
  if (I can compute any missing keys)
    compute missing blinded keys
    broadcast new blinded keys
  endif
  receive msg (msg type = broadcast)
  update current tree
endwhile
```

Cascaded Events

- A join, leave, merge, or partition takes place while a prior event is being handled

```
receive msg (msg type = membership event)
construct new tree
while there are missing blinded keys
  if (I can compute any missing keys)
    compute missing blinded keys
    broadcast new blinded keys
  endif
  receive msg
  if (msg type = broadcast)
    update current tree
  else (msg type = membership event)
    construct new tree
endwhile
```

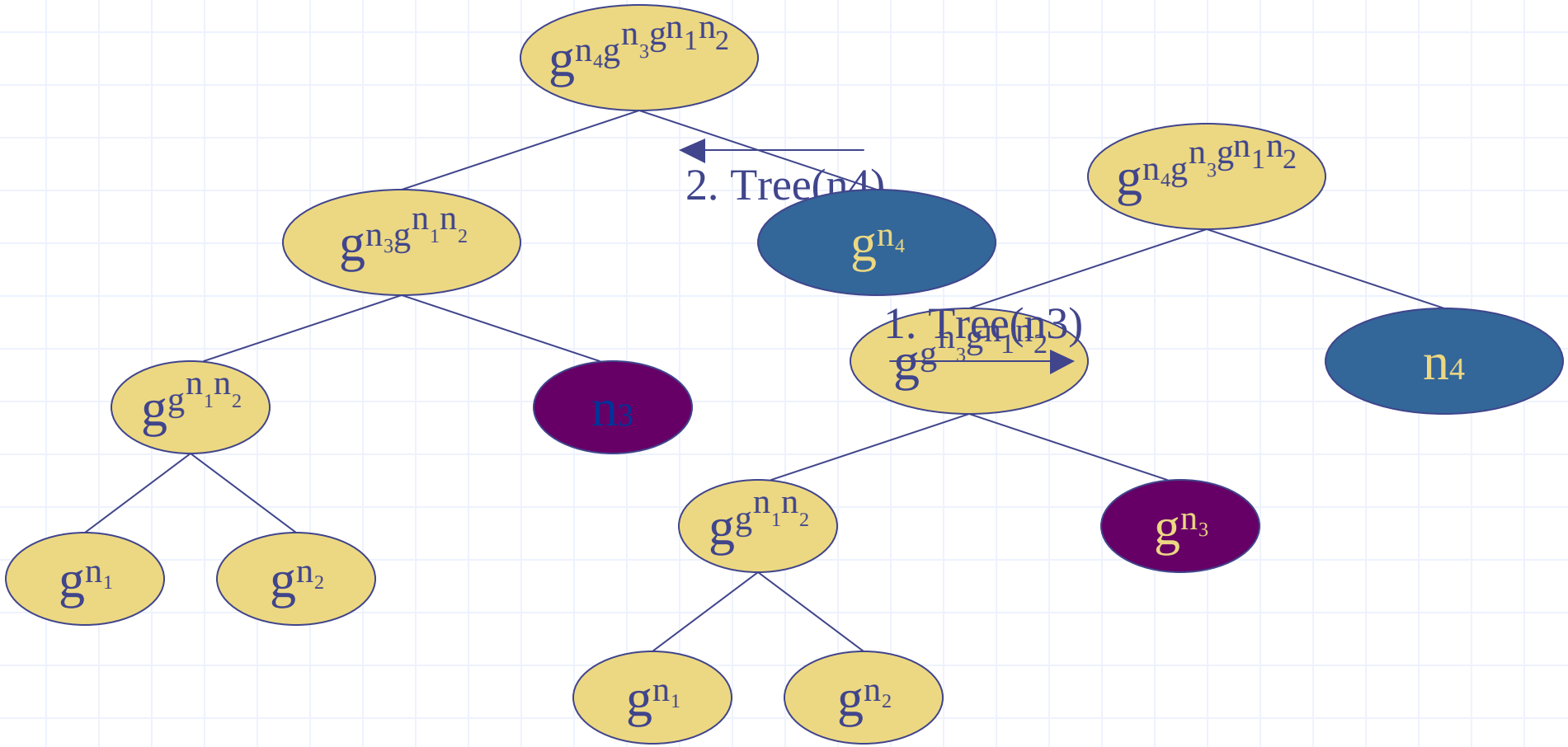
Merge (N_5's view)



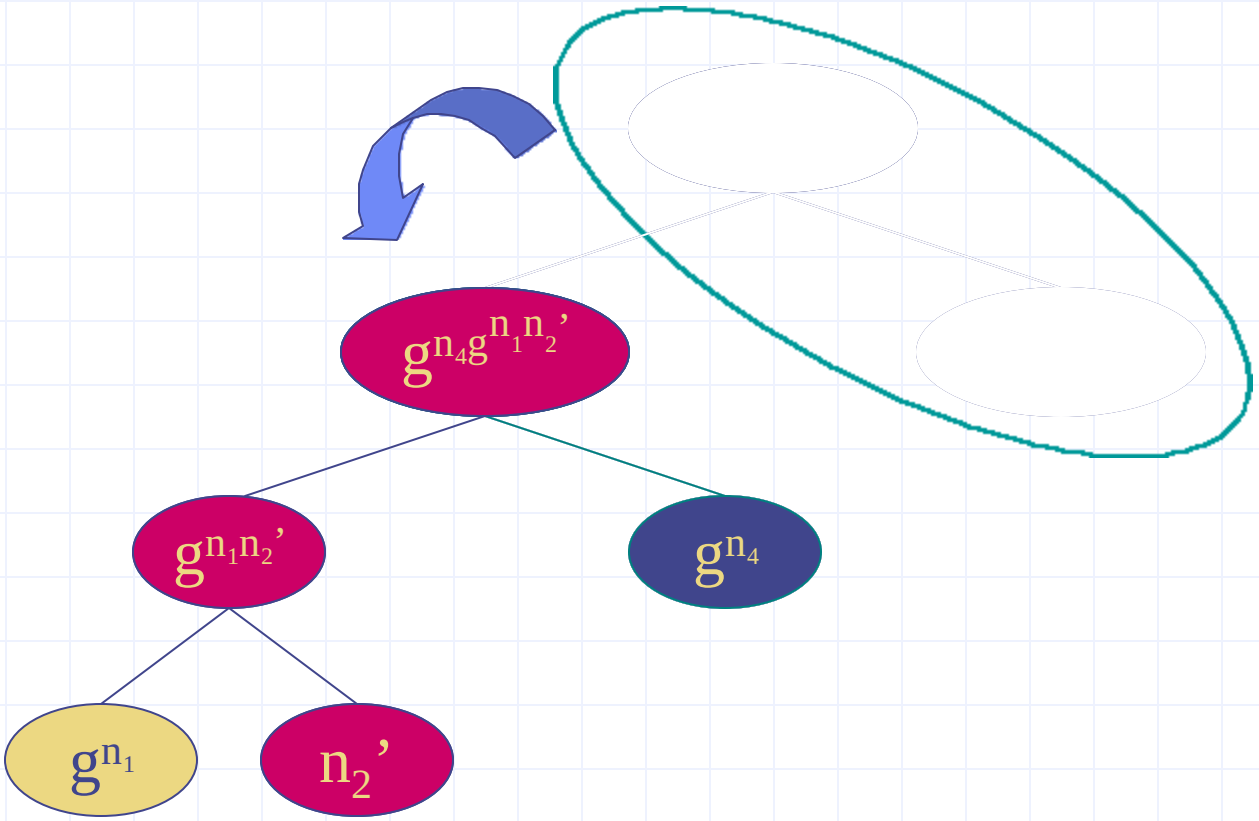
No nodes to merge without increasing height of the tree

Join to root

Join


$g^{n_4}g^{n_3}g^{n_1}n_2$
2. Tree(n4)
$g^{n_3}g^{n_1}n_2$
g^{n_4}
$g^{n_4}g^{n_3}g^{n_1}n_2$
1. Tree(n3)
$gg^{n_1}n_2$
n_3
$gg^{n_3}g^{n_1}n_2$
n_4
g^{n_1}
g^{n_2}
$gg^{n_1}n_2$
g^{n_3}
g^{n_1}
g^{n_2}

Leave



Tree Management: Etc

- Why balanced tree
  - # of exponentiation depends on height of tree
- How good is our policy
  - With a fully balanced tree (height 8) with 256 users, generate random partition and merge for 10000 times
  - Maximum height of tree: 15
  - Average height of tree: 12
  - Still logarithmic to number of users

Self-stabilization and Fault-tolerance

- Four protocols actually represent different strands of a single protocol
- Every protocol consists of
  - Tree management (delete or add nodes) for each membership event
  - Compute missing keys if I can
  - If I computed, broadcast my tree
- This leads us to design self-stabilizing protocol
- Cascaded event: Join, leave, merge, or partition occurs while a prior event is being handled