Simple and fault-tolerant key agreement for dynamic collaborative groups

Y. Kim and A. Perrig and G. Tsudik

Friday, April 07, 2003

Anubhav Dhoot
CSC 774 Network Security
North Carolina State University

INTRODUCTION

- Group key ***Agreement***
  - Not Group key ***Distribution***
- Group key ***Distribution*** suffers from
  - ***Single point of failure***
  - ***Attractive target for attacks***
  - ***Network Partitions***
  - ***Replication is too costly***
- Suitable for Dynamic peer groups (sizes < 20)
  - Frequent membership changes
  - Any member can be sender and receiver

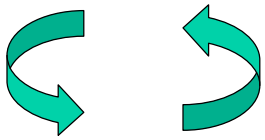
Main Contribution

- Self Stabilizing : One protocol (partition) subsumes all operations required (e.g. join, merge)
- Cascaded events
  - Takes care of network failures such that even partitioned groups are secure and independent
  - E.g. While one partition is taking place, another partition may occur
- Fault-tolerant: Robust against cascaded faults
- Efficient
  - Experimental evaluation shown it to be most efficient for ***dynamic peer*** groups
  - d is the height of key tree ($< O(\log_2 N)$), N is the number of users
  - Maximum number of exponentiation = 4(d-1)
- Provably secure

Assumptions

Group Communication Systems

Group key management



Message Transport

Strong Membership Semantics

Group Key Agreement

Assumptions

- Reliable Group Communication through ***View Synchrony***
  - ***Group members see the same set of messages between 2 sequential group membership events***
  - ***Senders message order (e.g.. FIFO) maintained***
  - ***Messages delivered to all "recipients" as when the sender sent the message.***
- Don't want to reinvent features for it

RELATED WORK

- Steer et. al (1988): fast join, slow leave
- Burmester and Desmedt (BD, 1993): fast but too many broadcast
- Becker and Wille (1998): Minimal communicational overhead of all, but large computation overhead for leaves
- Tzeng and Tzeng (1999, 2000): Fast but does not provide forward and backward secrecy
- Steiner et. al. GDH : leave and partitions efficient, but merge and join is inefficient

  Also the protocol suite Clique, based on GDH, blocks in case of cascaded events.
- Comparisons with related works later

Notations

l → *level*
v → *position of node*
(0 <= v <=2 $^{l-1}$)

- Each node <*l,v*> associated with a
  - Key $K_{<l,v>}$ (known to a few nodes)
    - Session Random for members (represented as leaf nodes)
    - calculated for interior nodes.
  - Blinded key $BK_{<l,v>}$ (known to all)
- $BK_{<l,v>} = f(K_{<l,v>}) = \_^{K<l,v>}$

  where f is modular exponentiation in Diffie-Hellman group.

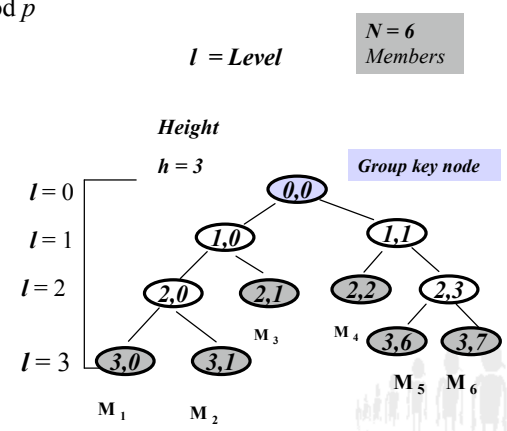
Notations

- $K_{<l,v>} = (BK_{<l+1,2v+1>})^{K<l+1,2v>} \bmod p$

  $= (BK_{<l+1,2v>})^{K<l+1,2v+1>} \bmod p$

  $= \_^{K<l+1,2v>\,K<l+1,2v+1>} \bmod p$

  $= f(K_{<l+1,2v>} K_{<l+1,2v>})$



How M_1 calculates K<2.0>

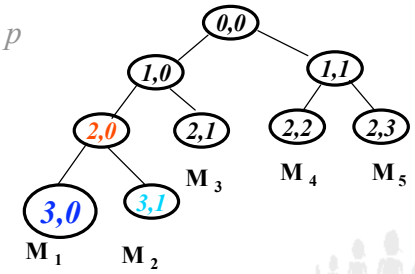
l =2 , v = 0

- $K_{<2,0>} = (BK_{<3,1>})^{K<3,0>} \bmod p$

  $= (BK_{<l+1,2v>})^{K<l+1,2v+1>} \bmod p$

  $= \_^{K<l+1,2v>} {}^{K<l+1,2v+1>} \bmod p$

  $= f\,(K_{<l+1,2v>} K_{<l+1,2v>})$



How M_2 calculates K<2.0>

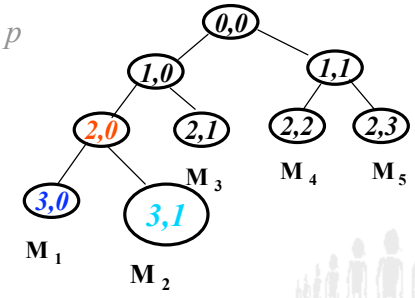
l =2 , v = 0

- $K_{<2,0>} = (BK_{<3,1>})^{K<3,0>} \bmod p$

  $= (BK_{<3,0>})^{K<3,1>} \bmod p$

  $= \_^{K<l+1,2v>} {}^{K<l+1,2v+1>} \bmod p$

  $= f\,(K_{<l+1,2v>} K_{<l+1,2v>})$



Completed Equations

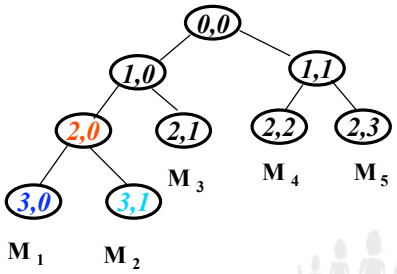
l =2 , v = 0

- $K_{<2,0>} = (BK_{<3,1>})^{K<3,0>} \bmod p$

  $= (BK_{<3,0>})^{K<3,1>} \bmod p$

  $= \_^{K<3,0>\,K<3,1>} \bmod p$

  $= f\,(K_{<3,0>}, K_{<3,1>})$



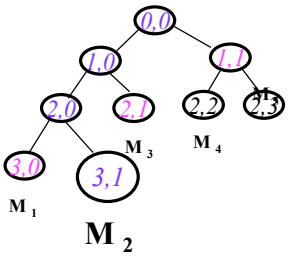
How M_2 calculates K<0.0>

- M_2 knows all keys on the "key path"

  $\{K_{<3,0>}\, K_{<2,0>}\, K_{<1,0>}\, K_{<0,0>}\}$

- and every blinded key

  $\{BK_{<0,0>} BK_{<1,0>,\ldots,} BK_{<3,7>}\}$



- Because M_2 can compute $K_{<2,0>}\, K_{<1,0>}\, K_{<0,0>}$

using K<3,1> (its own Secret Random)

and the known blinded keys $\{BK_{<3,0>} BK_{<2,1>} BK_{<1,1>}\}$

- These form the co-path for M_2

MAIN TECHNIQUES

- Each member contributes its share, equally to group key
- Share is secret
- As group grows, new members share are factored in, but old members shares unchanged
- As group shrinks departing members keys are factored out, *at least one remaining member changes its share*
- All messages signed by the sender. (e.g. RSA)

"Any member can compute the group key from its secret share and all blinded keys from its co path"

– For efficiency all Blinded keys are known to everyone.

– Group key obtained by using a cryptographic hash of the group secret

K group = h (K <0,0>),

Membership Events

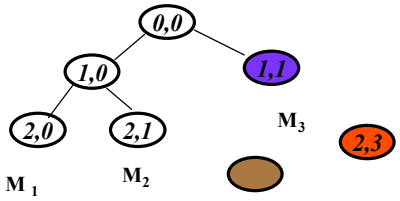
- *Join* - Single member add
- *Leave* - Single member leaves
- *Partition* - Group of members add
- *Merge* - Group of members leave
- *Key refresh* – Key updated
  - special case of "*Leave*", without any member actually leaving

Join Protocol

- M_{n+1} requests to join group $\{M_1,\ldots,M_n\}$ by sending its Blinded key $BK_{<0,0>}$
- Insertion node
  - Shallowest rightmost leaf node such that it does not increase height of tree,
  - Else new node joins to the root
- "Sponsor" node
  - Sponsor is the rightmost leaf node in subtree rooted at insertion tree

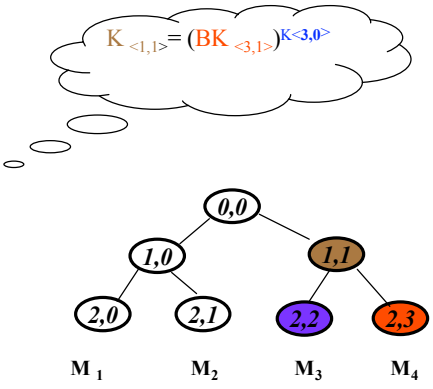
Join Protocol

- Insertion Node
- Sponsor (M_5)
  - Renames the insertion node <1,1> to <2.2>
  - creates a new intermediate node <1,1> and a new member node,<2,3>
  - Makes the new intermediate the parent of the insertion and the new node



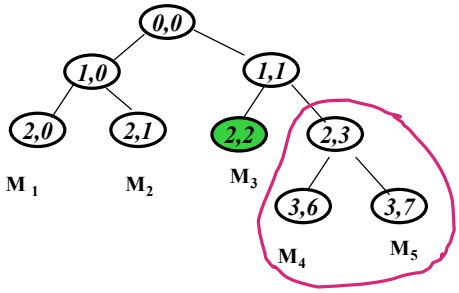
Join Protocol

- New node <2,3> used its session random K<2.3> to generate its BK<2.3> when it requested to join
- Sponsor calculates K<1,1> using BK<2.3> it and its K<2,2>
- Now using BK<1,0> and K<1,1>it can calculate the group key K<0,0>
- Similarly everyone can calculate the group key once the blinded keys are known



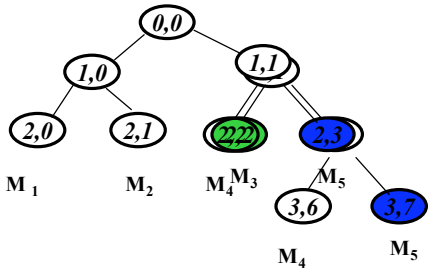
Leave Protocol

- Leaving node M_d
- Sponsor
  - Rightmost leaf node of the **subtree** rooted at the leaving member's sibling node.



Leave Protocol

- Delete leaving node M_d
- Former sibling of M_d → replaces parent of M_d
- Sponsor
  - picks a new share
  - calculates all the keys on its key path
  - Broadcasts new set of blinded keys



Partition Protocol

General Idea

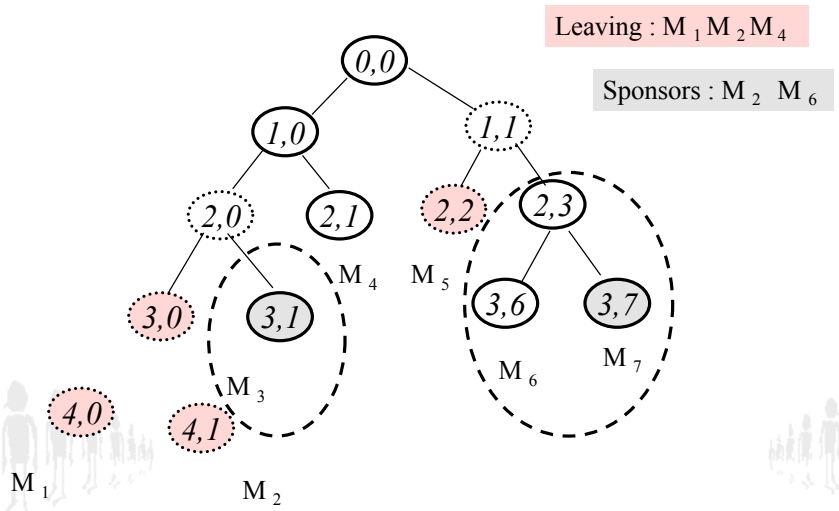
- Multi-round protocol
- Viewed by other members as a concurrent leave of multiple members
- Delete all members and their parent nodes

Steps Followed in Partition Protocol

- Deletion method optimized
  - All leaving nodes sorted on depth
  - Each pair of leaving siblings collapsed into their parent which is termed as leaving
  - Nodes is reinserted into the leaving nodes list
  - Resulting tree has a set of leaving nodes, each having a sibling which remains
- Each sponsor *computes & broadcasts* **keys** and **blinded keys** on its **key-path** as far up .

To prevent reuse of old group key, sponsor which was the shallowest rightmost initially, changes its share.

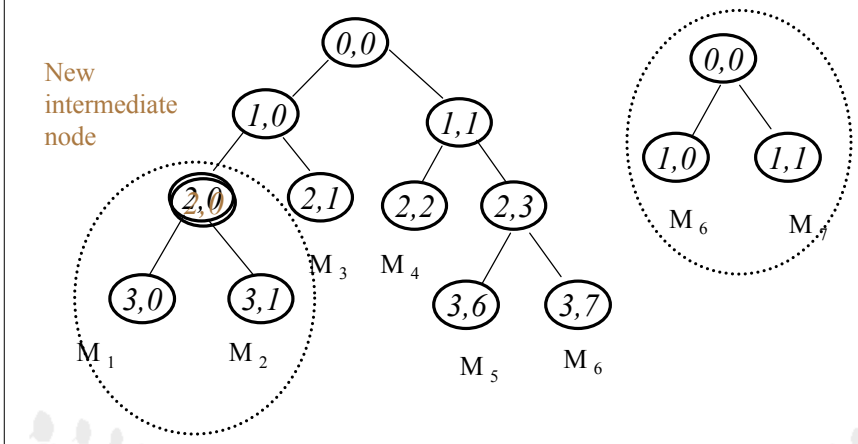
Partition Protocol



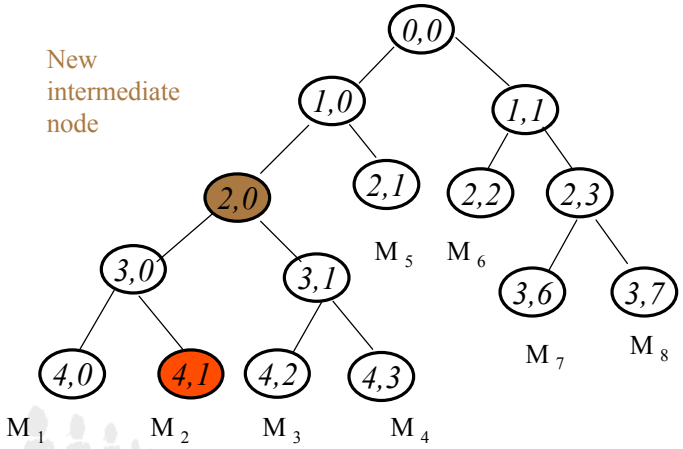
Merge Protocol

- Each sponsor (rightmost member of each group)
  - broadcasts tree information (including blinded keys) to other group
- The *insertion node* is
  - Root of the other tree for same heighted trees or if any other node increases height of the tree.
  - Rightmost shallowest node
- Rightmost member of subtree rooted at joining location is the *sponsor* for key update.
- Each sponsor computes keys and blinded keys and broadcasts set of blinded keys.

Merge Protocol



Merge Protocol



Unified Protocol

Join and *Leave* are special cases of *Merge* and *Partition*

Merge special case of *Partition*

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

Protocol for handling Cascaded Events

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

Cascaded events

- Each protocol represents 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 to a self-stabilizing protocol.
- If cascaded event finishes, then our protocol will also finish.
- ***(Please refer to example on paper to understand the issue, which is the question on the message board)***

Security Features provided

- Group key secrecy
- Forward Secrecy
- Backward Secrecy
- Key Independence

Q : How is PFS achieved ?

Security Proof

- Group secrecy
  - Group key secrecy reducible to Decision Hellman problem
- Weak backward secrecy
  - Group key secrecy, implies simply blinded keys are not enough to construct group key
  - As one needs at least one secret key K, a new node cannot calculate earlier keys.
- Weak forward secrecy
  - When a node M leaves the group,
    - at least one node changes its group shares
    - M's node is deleted and M's sibling replaces M's parent
  - all contributions from M are removed from the m's former key path

Complexity Analysis

- Height of tree determines exponentiation cost
  - All additions to root would need minimum exponentiations during addition
  - Makes tree unbalanced and very costly for other operations
- Heuristic used to reduce height
  - balance tree by adding nodes to rightmost shallowest node so that it doesn't increase height of tree

Comparison with Related Work

- ***On the Performance of Group Key Agreement Protocols***

Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, Gene Tsudik
http://sconce.ics.uci.edu/cliques/paper/aknt01.pdf

Comparisons of *Tree-Based Group Diffie-Hellman(TGDH) with*

- *Centralized Group Key Distribution (CKD),*
- *Burmester-Desmedt (BD),*
- *Steer et al. (STR),*
- *Group Diffie-Hellman (GDH)*

Conclusion - TGDH is the best choice of a key agreement protocol for dynamic peer groups in both local and wide area networks.

POSSIBLE FUTURE WORK

- Able to implement higher level policies
- Extend it for large groups
  - Make it applicable to LANs, WANs universally.
- Adapt to ad-hoc networks –
  - remove reliance on RSA authentication
- Another protocol STR similar to TGDH
  - lesser communication cost
    - Maximum 2 communication round
    - Maximum 2 broadcast messages
  - More computational cost
    - Maximum number of exponentiation = 4(N-1)
    - N is the number of users.