CSC 774 Advanced Network Security

Topic 5.2 Tree-Based Group Diffie Hellman Protocol

Acknowledgment: Slides were originally provided by Dr. Yongdae Kim at University of Minnesota.
Membership Operations

Formation

Group partition

Member add

Member leave

Group merge
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)

\[ g^{n_1n_2n_3} g^{n_6n_4n_5} \]

\[ g^{n_1n_2n_3} \]

\[ g^{n_2n_3} \]

\[ n_1 \]

\[ n_2 \]

\[ n_3 \]

\[ n_4 \]

\[ n_5 \]

\[ n_6 \]
Key Tree (n₃’s view)

GROUP KEY = \( g^{g^{n_1}{g^{n_2}{n_3}}} \cdot g^{n_6}{g^{n_4}{n_5}} \)
Key Tree (n₃’s view)

GROUP KEY = g^{g^{n₁}g^{n₂}n₃}g^{n₆g^{n₄}n₅}

Key-path: Set of nodes on the path from member node to root node
Key Tree (n₃’s view)

GROUP KEY = \( g^{n_1}g^{n_2}n_3 \quad g^{n_6}g^{n_4}n_5 \)

Co-path: Set of siblings of nodes on the key-path
Key Tree (n₃’s view)

GROUP KEY = g^{g_{n_1}g_{n_2}g_{n_3}}g_{n_6}g_{n_4}g_{n_5}

Member knows all keys on the key-path and all blinded keys
Key Tree (n_3’s view)

\[ \text{GROUP KEY} = g^{g_{n_1} g_{n_2} n_3} g_{n_6} g_{n_4} n_5 \]

Any member who knows blinded keys on every nodes and its session random can compute the group key.
Key Tree (n₃’s view)

GROUP KEY = g^{g n₁ g n₂ n₃} g^{n₆ g n₄ n₅}

Diagram:
- g^{n₁}
- g^{n₂}
- n₃
- g^{n₄}
- g^{n₅}
- g^{g n₆ g n₄ n₅}
- g^{n₆ g n₃ n₅}
- g^{n₄ n₅}

NC STATE UNIVERSITY  Computer Science  Dr. Peng Ning  CSC 774 Adv. Net. Security 6
Join (n₃’s view)
Join (n₃’s view)
Join (n₃’s view)
Join (n₃’s view)

Tree(n₄)
Join (n₃'s view)
Join (n₃’s view)
Join ($n_3$’s view)
Join (n3’s view)
Join (n₃’s view)
Leave (n₂’s view)
Leave (n2’s view)
Leave (n_2’s view)
Leave (n₂’s view)

Given:

\[ g_{g^{n_1}, n_2} g^{n_3} n_4 \]

Diagram:

- Root: \( g_{g^{n_1}, n_2} g^{n_3} n_4 \)
- Left child: \( g^{n_2} \)
- Right child: \( g_{g^{n_3}, n_4} \)
  - Left child: \( n_2 \)
  - Right child: \( g^{n_3} \)
  - Right child: \( g^{n_4} \)
Leave (n2’s view)
Leave (n₂’s view)
Leave (n₂’s view)
Leave (n₂’s view)
Partition (n₅’s view)
Partition (n5’s view)
Partition (n_5’s view)
Partition (n₅’s view)
Partition (n₅’s view)
Partition (n5’s view)
Partition (n₅’s view)
Partition (n5’s view)
Partition (n5’s view)
Partition (n₅’s view)
Partition (n₅’s view)
Partition (n_5’s view)
Partition (n_5’s view)
Partition: Both Sides

\[ g_{n_1} \]
\[ g_{n_2} \quad g^{n_3} \quad g^{n_4} \quad n_5 \]

\[ g_{n_6} \]
Partition: Both sides (N₅ and N₆)
Merge (N2’s view)
Merge (N2’s view)
Merge (N2’s view)
Merge (N2’s view)
Merge (N2’s view)
Merge (N₂’s view)
Merge (to intermediate node)
Merge (to intermediate node)
Merge (to intermediate node)
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 \( \log_2 N \))
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**