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)

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

- \(n_1\)
- \(g^{n_{2}n_{3}}\)
- \(g^{n_{4}n_{5}}\)
- \(n_2\)
- \(n_3\)
- \(n_4\)
- \(n_5\)
- \(n_6\)
Key Tree (n_3’s view)

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

Diagram:

- Root: GROUP KEY
  - Left Branch: $g^{n_1} g^{n_2 n_3}$
    - Left Branch: $g^{n_1}$
    - Right Branch: $g^{n_2}$
  - Right Branch: $g^{n_6 g n_4 n_5}$
    - Left Branch: $g^{n_6}$
    - Right Branch: $g^{n_4}$
Key Tree (n₃’s view)

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

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

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

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

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

Member knows all keys on the key-path and all blinded keys
Key Tree (n3’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)

\[
\text{GROUP KEY} = g^{g^{g_{n_1}g_{n_2}n_3}g_{n_6}g_{n_4}n_5}
\]
Join (n₃’s view)
Join (n₃’s view)
Join (n₃’s view)
Join (n₃’s view)
Join (n₃’s view)
Join ($n_3$’s view)
Join (n_3’s view)
Join (n3’s view)
Join \( (n_3 \text{'s view}) \)
Leave (n₂’s view)
Leave (n2’s view)
Leave (n2’s view)
Leave (n₂’s view)
Leave ($n_2$’s view)
Leave (n₂’s view)

\[ g^{n_3n_4} \]

\[ g^{n_3} \quad g^{n_4} \]
Leave (n₂’s view)
Leave (n₂’s view)
Partition (n₅’s view)
Partition (n₅’s view)

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

\[ g^{g^{n_2}n_3} \]

\[ g^{n_1} \]

\[ g^{n_4} \]

\[ g^{n_6} \]

\[ n_5 \]
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 (n₅’s view)
Partition: Both Sides
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