Topic 5.3 Group Key Distribution

Outline

- Overview of group key distribution
- A naïve solution
- Iolus: A Framework for Scalable Secure Multicasting
- Logical key hierarchy (LKH)

Group Key Distribution

- Group session keys are determined by the group manager
  - Usually used for large groups.
A Naïve Solution

- Use a separate secure unicast connection from the group manager to EACH group member.
- Requirement
  - Each client shares a unique key with the controller.
- Poor scalability:
  - $n$ secure unicast connections
  - $n$ secret keys

Problems Specific to Group Communication

- “1 affects $n$” problem
  - The actions of one member affects the entire group

![Diagram](https://example.com/diagram1.png)

Problems Specific to Group Communication (Cont’d)

- “1 does not equal $n$” problem
  - Cannot deal with the group as a whole
  - Must consider the conflicting demands of members on an individual basis

![Diagram](https://example.com/diagram2.png)

Example: Cannot use the old group key to distribute the new group key.
Iolus

- Divide a large group into smaller groups
- Introduce entities that manage and connect the subgroups
  - Group security controllers (GSC)
  - Control the entire group
  - Group security intermediaries (GSI)
  - Control the subgroups on behalf of GSC
  - GSC and GSI are both referred to as group security agents (GSA)
  - With GSC as the root, GSAs form a hierarchy of subgroups
    - A lower-level GSA is a member of the group headed by the higher-level GSA

Iolus (Cont’d)

- Joins
  - GSA generates $K_{GSA-MBR}$
  - Store this key along with other information
  - Send $K_{GSA-MBR}$ to the new member in a secure channel
  - Generate a new group key $K’_G$
  - Send $(K’_G)K_G$ to the group
  - Send $K’_G$ to the new member in a secure channel
Iolus (Cont’d)

- Leaves
  - Generate a new group key $K'_G$
  - Send $K'_G$ to each member MBR individually in the secure channel encrypted with $K_{GSA-MBR}$

Iolus (Cont’d)

- Data transmission
  - Data retransmitted within each subgroup

Iolus (Cont’d)

- Iolus for group key management
  - Replace the data with the group key in data transmission
Key Tree Approaches

- Two types of keys
  - SEKs (Session Encryption Key)
  - KEKs (Key Encryption Key)
- A Group Controller constructs a tree based hierarchy of KEKs

Logical Key Hierarchy (LKH)

- Keys are organized in a (logical) hierarchical tree
  - Group key is located at the root
  - Key encryption keys are the non-root, non-leave nodes
  - Each member corresponds to one leave node
- Updates the group key and the key encryption key by means of the encryption of key-nodes
- Rekey with only $O(\log N)$ messages

LKH (Cont’d)

- Initialization
User, Key, or Group Oriented Rekeying

- User-oriented re-keying
  - Grouping re-keying messages by users
  - Less but bigger messages
- Key-oriented re-keying
  - Grouping re-keying messages by keys
  - More but smaller messages
- Group-oriented re-keying
  - Putting all re-keying messages together to generate a big, fat message
  - Only one gigantic message
Example

- User oriented
  \[ x' \rightarrow \{ 1, \ldots, n_a \} \quad : \quad \{ \{ a \Rightarrow a \} \}_{a < n} \]
  \[ x \rightarrow \{ 1, n_b \} \quad : \quad \{ \{ 1 \Rightarrow 2 \} \}_{a < n} \]
  \[ a \rightarrow n_b \quad : \quad \{ \{ y \Rightarrow k_{n_b} \} \}_{a < n} \]

- Key oriented
  \[ x = \{ 1, \ldots, n_a \} \quad : \quad \{ \{ b \Rightarrow b \} \}_{a < n} \]
  \[ x = n_b \quad : \quad \{ \{ a \Rightarrow 2 \} \}_{a < n} \]
  \[ a = n_b \quad : \quad \{ \{ y \Rightarrow k_{n_b} \} \}_{a < n} \]

- Group oriented
  \[ x = \{ 1, \ldots, n_a \} \quad : \quad \{ \{ a \Rightarrow 2 \} \}_{a < n} \]
  \[ x = n_b \quad : \quad \{ \{ y \Rightarrow k_{n_b} \} \}_{a < n} \]