A Distribute Key Assignment Protocol for Secure Multicast Based on Proxy Cryptography

Chun-Ying Huang, Yun-Peng Chiu, Kuan-Ta Chen, and Chin-Laung Lei

Distributed Computing and Network Security Lab
Department of Electrical Engineering
National Taiwan University
Outline

- Secure Multicast
- Problem Statements and Assumptions
- Our Scheme
Secure Multicast

- **The Goal**
  - Share common secrets between group members.

- **Secure Multicast: Classifications and Problems**

  - **Centralized**
    - Need a group controller.
    - Bad scalability.

  - **Decentralized**
    - Scale better, however ...
    - May still need a group controller.

  - **Distributed (Contributory)**
    - Need to know group members.
    - Bad for a large group.
The Problem Statement and Assumptions

- For a large communication group …
  - It would be better to adopt “decentralized” mechanisms.
  - However, we don’t like the group controller.
  - Besides, we should only put limited trust on intermediate branch nodes.

- Our scheme is based on ElGamal proxy encryption
  - It can be easily applied on a source-based multicast tree.
  - It reduces the trust-level on intermediate nodes.

- Assumptions
  - The sender and the receivers are trusted.
  - Proxies are semi-trusted.
Our Scheme – Adopt Proxy Encryption

- Proxies are only semi-trusted
  - Transform the cipher-text without revealing the message.

```
KEY = K0
MESSAGE = m
Send: [c1 = g', c2 = mg^{K0}]

KEY = K1-K0
Transform: c2' = c2 \times g^{(K1-K0)}
Send: [c1' = g', c2' = mg^{K1}]

KEY = K1
Decrypt: m = c2'/(c1')^{K1}
```
Our Scheme – Remove the Group Controller

- **Basic Ideas**
  - The sender and the proxies generate their own secret keys.
  - The receiver obtains the decryption key using the proposed key-composition protocol.

```
Sender                  Proxy-1                   Proxy-2                  Receiver

KEY = K0
CoKEY = R+K0
MESSAGE = m
Send: \{g', mg^{K0}\}

KEY = K1
CoKEY = R+K0+K1
Transform with g^{K1}
Send: \{g', mg^{(K0+K1)}\}

KEY = K2
CoKEY = R+K0+K1+K2
Transform with g^{K2}
Send: \{g', mg^{(K0+K1+K2)}\}

KEY = ?
Obtain R, compute CoKEY-R = K0+K1+K2
Decrypt using the composed KEY.
```
Thank You!

Comments or Questions?