Multicast Defined

- Multicast is a method of communication between one or many senders and many recipients in a group.
Introduction

- Multicasting uses
  - Internet Radio
  - Stock Tickers
  - Pay Per View Movie
  - Video Conferencing

Benefits of Multicast

- Distributes content from one location to many clients
  - Low network overhead
  - Simultaneous distribution
  - Group collaboration
Enter secure multicast

- Protection of intellectual property
- Ensure payment for services

Problems with Secure Multicast

- Multicast protocols allow anyone to join
- Can’t use SSL
  - Would reduce communication to a unicast
Problems with Secure Multicast

- Multicast protocols allow anyone to join
- Can’t use SSL
  - Would reduce communication to a unicast
- Ensure backward secrecy
- Ensure forward secrecy
Peer-to-peer look up protocols and algorithms: a comparison.

Problems with Secure Multicast

- Multicast protocols allow anyone to join
- Can't use SSL
  - Would reduce communication to a unicast
- Ensure backward secrecy
- Ensure forward secrecy
- Provide a “Group Key”

Proposed Solutions

- Key Management
  - Modified topology of group
  - Key storage schemes
  - Key generation schemes
  - Modified access to group
Physical Key Tree

- Physically breaks the group up into sub groups
- Key management provided by group controller and sub group controllers

[CC, 2003]

Logical Key Tree

- Breaks up group into a tree structure logically
- Tree management handled by multicast server
Peer-to-peer look up protocols and algorithms: a comparison.

Logical Key Tree Diagram

A logical key tree with a depth of 3 [PL, 2000]

Contributory Key Agreement

- Operate via extension of Diffie-Hellman key agreement protocol
- Keys not distributed by server
- Group members play integral role in key determination
Contributory Key Agreement Diagram

M1

M2

M3

M4

...Mx

GK1 + GK2

GK1 + GK2 + GK3

GK1 + GK2 + GK3 + ...GKx

Group Access Control

- Focus of security is not on key management but on who can access the group transmissions at all.
- Forces group members to “authenticate” before receiving any group broadcasts.
Peer-to-peer look up protocols and algorithms: a comparison.

Protocol Comparison

- Physical Tree
  - Iolus
- Logical Tree
  - Standard
  - Park and Lee
- Contributory Key Agreement
  - CLIQUES
- Access Control
  - Gothic

Iolus

- Minimal number of re-keying messages sent due to leave/join
- Group management is distributed
- Long messaging delays due to decryption and Re-encryption by Intermediate level Security Agents
Peer-to-peer look up protocols and algorithms: a comparison.

Logical Tree

- Network traffic scales logarithmically to size of tree for member join/leave
- Greater dependence upon group controller
- Has to maintain all keys in group in memory/on disk

Logical Tree by Park

- Network traffic scales logarithmically to size of tree for member join/leave
- Doesn't have to store all the keys for entire tree
- Requires good deal of processor power at server
CLIQUES

- Only protocol that encrypts with a symmetric key
- Allows for a less powerful server
- Best for small multicast groups
- Can “implode” if final client cannot perform calculations and redistribute the code

Gothic

- Great backwards secrecy
- All members must pass through the “authentication server” before receiving any group communication
  - Potential issue with flash crowd joins
## Metrics Evaluated

- Number/size of keys stored at the Group Controller
- Number/size of keys stored at each client
- Number/size of re-key massages for a member leave
- Strength of processor of the client needed
- Single point of failure in system

## Conclusions

<table>
<thead>
<tr>
<th></th>
<th>Physical Tree (Iolus)</th>
<th>Logical Trees (many protocols)</th>
<th>Contributory key agreement (CLQUIES)</th>
<th>Group access control (Gothic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number/size of keys</td>
<td>O(n/m)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>stored at Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number/size of keys</td>
<td>O(1)</td>
<td>O(log n)</td>
<td>O(n)</td>
<td>O(1)</td>
</tr>
<tr>
<td>stored at each client</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number/size of re-key</td>
<td>O(1)</td>
<td>O(k log n)</td>
<td>O(n)</td>
<td>0</td>
</tr>
<tr>
<td>massages for a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member join</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number/size of re-key</td>
<td>O(n + m)</td>
<td>O(log n)</td>
<td>O(n)</td>
<td>Variable;</td>
</tr>
<tr>
<td>massages for a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>member leave</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strength of processor</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
<td>min = 0</td>
</tr>
<tr>
<td>of the client needed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[CC 2003], [JA, 2002]
Conclusions

- No single protocol has complete superiority
- Most protocols still have single points of failure
  - At Multicast server
  - At client due to “implosion”

Future Research

- Work to eliminate single point of failure
  - Draw inspiration for distribution of load from P2P systems
- Incorporate Gothic’s control scheme with a more scalable key management system
Questions?