Collusion-Resistant Group Key Management Using Attribute-based Encryption

Presented by:
Anurodh Joshi
Overview of the Paper

- Presents a ciphertext-policy attribute-based encryption (CP-ABE) scheme to solve the collusion issue of Group Key Management.
- CP-ABE scheme proposed by Bethencourt, Sahai, and Waters is (BSW) used to implement collusion resistant flat-key group key management.
- Mechanism for refreshing the keys (for added security) are discussed.
Setting & Motivation:
- IP Multicast: an efficient way to distribute information to a large group of users
- However, any host can join (access control issue)

Solution: Encrypt Data using Data Encryption Key (DEK)

DEK can be generated and distributed securely to the Group Members (GMs). However, the main challenge is efficiency of selective key distribution; this problem is called Group Key Management

Related issues:
- Group Backward Secrecy (new member joins)
- Group Forward Secrecy (old member leaves)
Existing Group Key Management approach:

- Most schemes use key encryption keys (KEKs) on flat table key management
- Distribute unique set of KEK to each GM
- Group controller (GC) encrypts DEK with a combination of KEKs

Problem?

- Vulnerable to Collusion
Use CP-ABE instead of KEK

- \( S = \) set of Attributes; \( SK = \) secret Key
- Associate GM with \( S, SK \) rather than KEK
- GC computes access structure that is satisfied by \( SK \) of current members only

Advantage

- SKs are computed based on \( S \) using a randomization factor
- collusion resistant (follows from CP-ABE)
Outline of the paper

- Background
  - Flat table key management
  - CP-ABE
- Proposed scheme for Group Key Management
- Results & Performance
- Concluding Remarks
Flat table key management

- A way to manage group key

- Each GM has a n bit unique ID: \( X_n, X_{n-1}, \ldots, X_0 \)
- \( X_i \in [0,1] \)
- So maximum size of the group is \( 2^n \)
- Two KEKs corresponding to each n bit
  - KEKs \( \{k_{i,b} \mid i \in \mathbb{Z}_n, b \in \mathbb{Z}_2\} \). So total \( 2n \) KEKs
The GC maintains the DEK (K) + 2n KEKs
Each GM has n KEKs + 1 DEK

Join:

- Group backward secrecy:
  - Suppose id: X_n, X_{n-1}, .. X_0 is joining
  - N KEKs + K is refreshed
  - Example: (3 bit group ids)
  - |1| 1|0 | (New)
  - KEK => k_{2,1} | k_{1,1} | k_{1,0}
Join:

- GC encrypts the new keys with corresponding old keys and multicasts
- new member is given $k'_{i,b} \mid i \in \mathbb{Z}_n, b \in \mathbb{Z}_2$ and $K'$ via secure unicast

Leave:

- Suppose id $X_n, X_{n-1}, .. X_0$ is leaving
- Forward Secrecy:
  - The $n$ KEKs + 1 ($K$) held by the leaving member is refreshed
  - The GC multicasts the new DEK encrypted once with each of the $n$ KEKs not held by the leaving GM \{K\}'_{k_0,x_0}^{k_{n-1},x_{n-1}}, ..., \{K\}'_{k_0,x_0}
    - leaving GM can’t decrypt any of these messages
    - rest of the GMs should be able to decrypt at least one of these messages hence obtaining $K'$
  - The GC multicasts the new KEKs encrypted with both the new DEK and the KEKs. Again existing GMs can decrypt and update appropriate KEK but the leaving GM can not.
Multiple Leaves:

- Leave procedure can be repeated to remove multiple members from the group.
- More efficient approaches such as using Boolean function minimization ($m$) has been proposed:
  - $m(X_n, X_{n-1}, .. X_0) = 0$ if it is leaving id
  - $m(X_n, X_{n-1}, .. X_0) = 1$ if it is existing id
- The GC runs the Quine-McCluskey algorithm which returns sum of products expression (SOPE).
- Example:
  - 011 and 101 are to be removed from the group. The membership function `m` can be reduced to
    - $m(X_2, X_1, X_0) = X_2X_1 + X'_2X'_1 + X'_0$
    - $m(0, 1, 1) = 0(1) + (1)(0) + 0 = 0$
    - $m(1, 0, 1) = 1(0) + 0(1) + 0 = 0$
- To Update the DEK three messages are multicast by GC:
  - $\{K'_k\}_{k,1}, \ {K'_k}_{1,1}$, and $\{K'_k\}_{k,0}$.
Collusion

- Leaving group members can collude to possibly figure out the new DEK and KEKs

Example:
- If leaving ids are \( |0|1|1| \) and \( |1|0|1| \)
- Can start to collectively figure out rekey messages
- \( \{K\}'_{k2,1, k1,1} \) and \( \{K\}'_{k2,0, k1,0} \)
- Use a simplified version of BSW scheme (CP-ABE scheme)
- Use SKs instead of KEKs
- GM with id $X_n$, $X_{n-1}$, .. $X_0$ has SK associated with $S := \{ A_{ix_i} \mid i \in Z_n \}$
- Can decrypt a message only if $S / SK$ satisfies the access structure
- Advantage:
  - Leaving group members cannot collude to breach the system
Algorithms in CP-ABE scheme:
- Setup
- Encrypt
- KeyGen
- Decrypt
- Delegate (optional)

Setup:
- Generates the public key
  \[ PK := < G, g, g^\beta, g^{(1/\beta)}, e(g,g)^\alpha, H > \]
CP-ABE (BSW)

- **Encrypt**
  - Input: PK, M, T (access tree)
  - Output: CT (cipher text)

- **KeyGen**
  - Input: set S of attributes
  - Output: SK associated with S

- **Decrypt**
  - Input: CT, SK
  - Output: Message if S satisfies T (access structure)

- **Delegate**
  - Used by Sub group controller (SGC) for decentralized management
Concept:
- Use flat table key management
- Use CP-ABE for rekey operations in order to achieve collusion resistance instead of KEK

Group Initialization
- GC plays the role of central authority (CP-ABE)
- To initialize GC runs the setup algorithm and gets the PK (public key) and MK (master key)
- Selects random DEK (K) ∈ G
- No KEKs needed
Group Members

- n bit id: $X_n, X_{n-1}, \ldots, X_0$ with attribute set:
  \[ S := \{ A_{i,b} \mid i \in Z_n, b \in Z_2 \} \]

Join

- Joining GM establishes a secure unicast with the GC
- GC selects new DEK ($K'$) $\in G$ at random
- Multicasts $\{K'\}_K$. Current members can decrypt and update the DEK
- GC runs KeyGen with attribute set $S := \{ A_{i,b} \mid i \in Z_n, b \in Z_2 \}$ to get SK
- The joining member receives the secret key SK and $K'$ via secure unicast
Leave

- Single leave and multiple leaves work exactly the same way
- Let $C'$ be the set of active members in the group after some GMs leave
- GC runs the Quine-McCluskey algorithm to obtain the SOPE, $E = E_0 + E_1 + \ldots + E_L$
- The GC selects a random $K' \in G$.
- For each $l$, GC runs Encrypt on $K'$ and $T_l$ to obtain the $CT_l$ and multicasts $CT_0, CT_1, \ldots, CT_L$.
- Each active GM should be able to decrypt at least one message to recover $K'$
- Inherits collusion resistance from CP-ABE. This ensures leaving GMs can not collude to decrypt $K'$
GC chooses random $K' \in G$ and $\alpha \in Z_p$

PK and MK are updated

$PK' = \langle G, g, g^\beta, g^{(1/\beta)}, e(g,g)^\alpha, H \rangle$

$MK' = \langle \beta, g^\alpha \rangle$

GC broadcasts two messages

- $\{K'\}_K$ – GMs know $K$ so they can decrypt and update $K'$
- $\{g^{(\alpha - \beta / \beta)}\}_K$ – this is called the conversion factor and used to update the SKs by the GMs

$SK' = \langle D \cdot g^{(\alpha - \beta / \beta)}, \{D^0_j \mid j \in S\}, \{D^1_j \mid j \in S\} \rangle$

$K'$ and $\alpha$ are random so do not reveal anything about older versions of the same
Current Scheme:
- GC manages the whole group
- Responsible for storage, KeyGen and overall communication
- Workload can be distributed by assigning responsibilities to subgroup of trusted members who act as Subgroup Controllers (SGCs)
- Number of SGCs is small so communication between SGC and GC is done via secure unicast (multicast for other direction)
SGCs maintain membership of their own subgroups using the delegate algorithm.

GC generates DEK and $\alpha$ parameters when they are requested or periodic refresh.

BFM computations are done by SGC and leave operations are also handled within a subgroup.

Notations:
- Given a subgroup identified by $\text{gid}$, every member of the subgroup possesses the attribute $\text{B}_{\text{gid}}$ and only SGC possesses attribute $\text{C}_{\text{gid}}$.
- GC encrypts messages to SGC only and SGC encrypts messages to subgroup members.
GC assigns a subgroup id gid and gives the SGC a secret key SK with the attribute set 
\{A_{i,b} \mid i \in \mathbb{Z}_n, b \in \mathbb{Z}_2 \} \cup \{B_{gid}, C_{gid}\}

- gid is added to the list of active subgroups
- SGC also receives instruction on who may join the group and how IDs for subgroup members are obtained (may be assigned from some ID space)
GM who wants to join contacts the SGC
SGC verifies the request and sends a secure unicast message to GC to signal a join
GC multicasts \( \{K'\}_K \) to the whole group
SGC uses delegate algorithm with attribute set \( \{A_{i,b} \mid i \in \mathbb{Z}_n, b \in \mathbb{Z}_2 \} \cup \{B_{gid}\} \) to get SK
SGC uses secure unicast to give the SK and \( K' \) to the joining GM
- The leaving GM contacts the SGC from whom he got his SK.
- SGC sends a leave signal to the GC.
- GC multicasts $K'$ to SGCs only by encrypting the new DEK with the C attributes.
- SGC performs the BFM computation within the subgroup and multicasts rekey messages with the additional attribute $B_{gid}$. 
Global Revoke

- Occurs when the GC makes a revocation decision based on malicious behavior from a GM
- GC multicasts $K'$ to SGC
- Works same way as a leave (Only difference is it is instigated by GC)
Periodic Refresh

- Same as before. Still done by GC

Remove

- Suppose GC decides to remove SGC with subgroup ID $\text{gid}_0$ or SGC decides to leave
- GC multicasts $K'$ encrypted with CP-ABE with attributes $C$ other than the SGC who is leaving
- Each SGC does BFM computation and multicasts rekey messages within subgroup

Note:

- The GMs that were in the $\text{gid}_0$ are effectively removed from the group because they can’t decrypt any rekey messages
- They must contact another SGC to join the group again (downside)
The number of messages do not go up a lot even for large number of members leaving

### Fig. 2. Average Number ofMsgs/Bytes Per Leave Event in 8-Bit ID Space.

<table>
<thead>
<tr>
<th>Group Size</th>
<th>Percentage of Members Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td>32</td>
<td>6 / 5510</td>
</tr>
<tr>
<td>64</td>
<td>9 / 9662</td>
</tr>
<tr>
<td>128</td>
<td>14 / 18588</td>
</tr>
</tbody>
</table>
Simple Concept:

- KEK is vulnerable to collusion attack
- CP-ABE is collusion resistant
- So replace KEK with CP-ABE in Flat Table Group Key Management Scheme
- Use Decentralized System to distribute load of GC
- Disadvantage:
  - If SGC is removed, all GMs in that group lose membership and have to rejoin a different group