

Topic 7. Broadcast and Multicast Key Distribution and Authentication (chapter 8 in the text)

- 1 **Basic** Models for Multicast Key Distribution
- 2 **Logic Key Tree** Based Multicast Key Distribution
- 3 **Hash Chain** Based Authentication
- 4 Merkle Trees for **Authentication**

1. Basic Models for Multicast Key Distribution

System Model

- We consider **a system** with n communication entities $\{u_0, u_1, \dots, u_{n-1}\}$ and one key management and distribution (KMD) server.
- **KMD server** responds to distribute a common key to each entity. This key is referred to as a **multicast key or group key**, which will be used for protection of multicast communication in the group.

Trust Model

a) **Public-key based approach:**

- ▶ **KMD server's public-key** is bound with a certain group relation or membership relation for carrying out the group communication, which is known to each entity in the multicast group.
- ▶ **Individual keys:** **KMD server** is employed as a certification authority to issue a certificate to a pair of public-key and private key for each entity. Those private keys are referred to as **individual keys**.

b) **Symmetric-key approach:** There exists a secure link between KMD server and an entity for which they can pre-share a key securely prior to its joining the group.

Pre-conditions

KMD server and entities employ their own security associations:

- 1 a **pseudorandom** number generator (PRNG),
- 2 **an encryption** and decryption operators $E_k(m)$ and $D_k(m)$ (e.g., AES Rijndael),
- 3 a **key deviation function (KDF)**,
- 4 and the other requested crypto schemes.

Security Requirement

- **Preceding/succeeding secrecy:** A multicast system is said to have *preceding secrecy* if any newly joining member cannot decrypt the previous established protected communication sessions using his keys, and
- it said to have *succeeding secrecy* if a member who left the system or his individual key is revoked cannot decrypt the protected communication sessions after his leaving or the key revoked.

Key Sharing Scenarios

Case 1. Single common key shared in the whole multicast network, which is distributed by a KMD server.

Two Main Disadvantages:

- (i) **Rekeying Process:** Whenever an entity joining or leaving, the KMD server will revoke the group key, and run a rekeying process, which is essentially the same process as the initial phase for the key distribution, in order to provide the preceding/succeeding secrecy.
- (ii) **Robustness:** An adversary who compromises any entity and extracts his multicast key will compromise the security of the whole system.

Countermeasure: Tamper-proof memory for the shared key is a possible solution for this attack.

Key Sharing Scenarios (cont.)

- 1 **Case 2.** Each entity pre-shares an individual key with the KMD server.
 - ▶ *Disadvantages:* No save for communication cost.
- 2 **Case 3.** Pairwise key shared in each pair of two entities. In this case, each entity could be in the role of the KMD server ← Case 2.

Table: **Comparisons of Three Scenarios of Key Sharing Status**

	$ K(u_j) $	$ K(KMD) $	$ M(x) $	Rekeying needed
Case 1	2	$n + 1$	1	Yes
Case 2	1	n	n for KMD	No
Case 3	n	n	$n - 1$ for u_j	No

- $K(x)$ denotes the set of keys of a party x .
- $M(x)$ is the set consisting of **ciphertexts or authentication tags** that a party x which could be KMD server or any entity in the group sends to the multicast group in a protected manner for delivering one **multicast message**.
- Cases 2 and 3 are of unicast scenario.

PROTOCOL Θ : STAR TOPOLOGY BASE MULTICAST KEY DISTRIBUTION

Initial Phase

- **KMD server:**
 - Run with each entity** the mutual authentication and key establishment protocol (MAKE) to share a key, say k_i with entity i , which serves as the individual key between the entity i and KMD server.
 - Randomly pick** r , and send $c_i = E_{k_i}(r)$ to entity i by unicast.
- **Entity:** Decrypt c_i using the individual key k_i to obtain r , which serves as a multicast key.

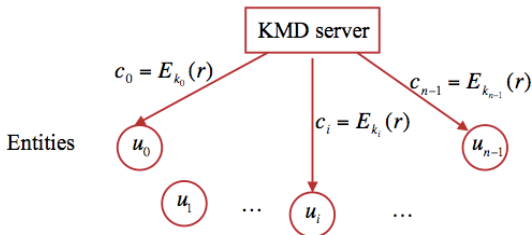


Figure: Multicast key distribution: a naive protocol

Rekeying Phase

KMD server

- 1 **Entity leave**: the KMD server runs the step (b) in the initial procedure except for the leaving entity.
- 2 **Entity join**: the KMD server runs the step (a) with the new entity, and execute the step (b) within the new group.

Parameter Set

- In **protocol** θ , each entity holds two keys where one is the individual key and the other the group key, i.e., $R = 2$.
- **Once an entity** leaves or joins the network, the group key is revoked, and a new group key is delivered to a new group.
- **The numbers of messages** that KMD server sends in the initial key distribution phase and the re-keying phase for a leave or a join, respectively, are equal to

$$N_{init} = n, N_{leave} = n - 1, \text{ and } N_{join} = n + 1.$$

- This is referred to as a parameter set $(n, R, N_{init}, N_{leave}, N_{join})$ of the system.
- The parameters of the protocol θ are $(n, 2, n, n - 1, n + 1)$.

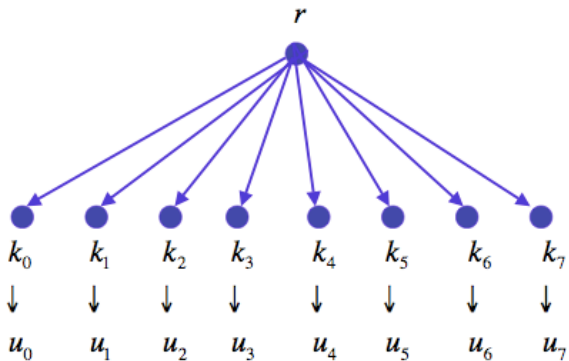


Figure: Key Graph of a Group with 8 Entities

Question:

How to design a scheme which yields a trade-off among the following factors?

- **The number** of the keys held by each entity.
- The **communication** overhead in the rekeying process.
- Tolerant to key **compromising** attack.

2. Logic Key Tree Based Multicast Key Distribution

Basic concepts of graph theory

- **A graph G consists** of two types of elements, namely vertices and edges.
- **Every edge** has two endpoints in the set of vertices, and is said to **connect or join the two endpoints** and we also say that these two vertices are *adjacent*.
- **An edge** can thus be defined as a set of two vertices (or an ordered pair, in the case of a directed graph).
- **The degree** of a vertex is the number of the edges that the vertex being connected.
- A **binary** tree is a connected acyclic graph such that the degree of each vertex is no more than 3.
- **Node**, leaf, root, path,
- **parent (vertex)**, **child (vertex)**, **siblings**.

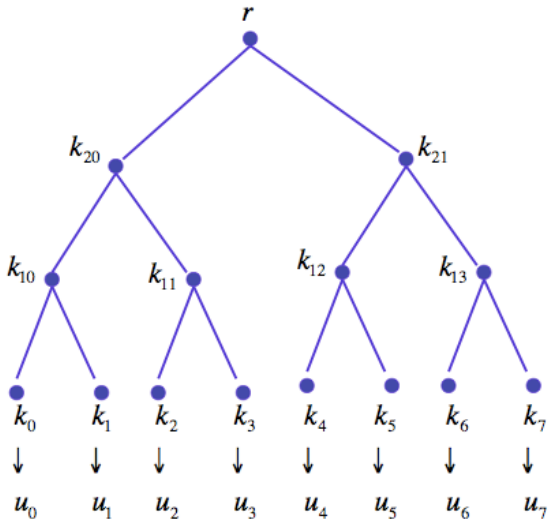


Figure: A Logic Key Tree with 8 Entities

Tree topology based multicasted key distribution protocol

PROTOCOL 1 (INITIAL PHASE)

- KMD server creates **Logic Key Tree**:
 - ▶ **Generation** of a complete tree with height $h = \lceil \log n \rceil$
 - ▶ r , root key or group key,
 - ▶ k_{ij} , attach to vertex (i, j) , called the *parent keys or vertex keys or subgroup keys* depending on the context,
 - ▶ k_i , individual keys of leaves, established later, and each entity is attached to the leaf with its individual key.

- **Establish individual keys** and distribution of group key and subgroup keys:

- ▶ **If the individual** keys are not preshared, then KMD server and the entity u_i run the MAKE protocol to establish individual key k_i .
- ▶ **Finds a path** from vertex u_i to the root. Let

$$K(u_i) = \{k_i, k_{1,j_1}, k_{2,j_2}, \dots, k_{h-1,j_{h-1}}, r\}.$$

The KMD server assigns $K(u_i)$ to u_i by sending the following ciphertexts to u_i :

$$C_i = \{E_{k_i}(k_{1,j_1}), E_{k_{1,j_1}}(k_{2,j_2}), \dots, E_{k_{h-1,j_{h-1}}}(r)\}. \quad (1)$$

Each entity

- 1 **Entity** u_i uses its individual key k_i , shared with the KMD server, to decrypt the first ciphertext to obtain k_{1,j_1} , then to decrypt the second cipher text to obtain k_{2,j_2} using the key k_{1,j_1} , and so on, until r is obtained.
- 2 The entity u_i **stores** his key set $K(u_i)$.

Example

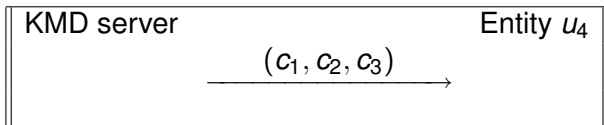
- $n = 8$.
- Find u_4 's key set.
- **A path from** the vertex k_4 to the root r : $k_1, k_{1,2}, k_{2,1}$, and r .
- Key set of entity u_4 is given by

$$K(u_4) = \{k_4, k_{1,2}, k_{2,1}, r\},$$

- **KMD server** prepares the following ciphertexts:

$$\begin{array}{l} c_1 = E_{k_4}(k_{1,2}) \\ c_2 = E_{k_{1,2}}(k_{2,1}) \\ c_3 = E_{k_{2,1}}(r) \end{array}$$

and sends (c_1, c_2, c_3) to the entity u_4 , i.e.,



- **Upon receiving** (c_1, c_2, c_3) , the entity u_4 performs the following successive decryption using his individual key k_4 .

$$\begin{array}{l} k_{1,2} = D_{k_4}(c_1) \\ k_{2,1} = D_{k_{1,2}}(c_2) \\ r = D_{k_{2,1}}(c_3) \end{array}$$

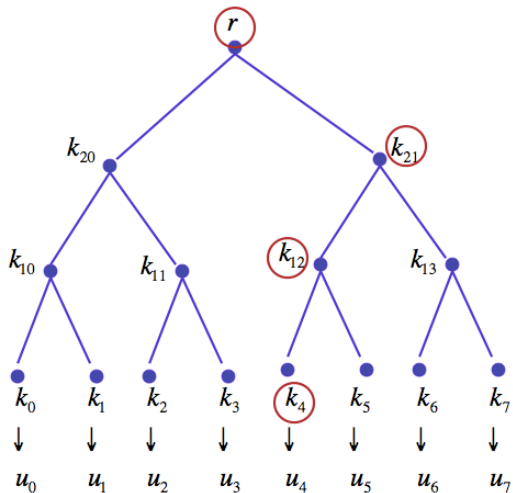


Figure: The key set of entity u_4 in LKT

Rekeying Process for Leave

PROTOCOL 2 (REKEYING PROCESS FOR LEAVE)

KMD server:

- **Find a path** from vertex u_i to the root and retrieve the key set of u_i , $K(u_i)$, as shown below

$$K(u_i) = \{k_i, k_{1,j_1}, k_{2,j_2}, \dots, k_{h-1,j_{h-1}}, r\}.$$

All the keys in $K(u_i)$ need to be revoked including the individual key.

- **Locate the sibling** of u_i , say u_t (note that $t = i$ or $t = i + 1$), which holds the same set of the parent keys as that of u_i .

$$K(u_t) = \{k_t, k_{1,j_1}, k_{2,j_2}, \dots, k_{h-1,j_{h-1}}, r\}.$$

- When u_i leaves, KDM server attaches u_t to its parent vertex $(1, j_1)$, assigns the individual key k_t of u_t to the vertex, and removes the parent key k_{1,j_1} .

- KMD server generates a set of new keys for updating the parent keys in $K(u_i)$ except for k_{1,j_1} since this vertex becomes a leaf vertex. We may assume that the updated $K(u_t)$ is given by

$$K'(u_t) = \{k_t, k'_{2,j_2}, \dots, k'_{h-1,j_{h-1}}, r'\}.$$

- **Paired encryption and multicast transmission:** KMD server updates keys $k_{i,j}$ to $k'_{i,j}$ to the entities in the subtrees which possess the original keys $k_{i,j}$ using the keys of the siblings of the vertexes in the path of u_i to the root in the following fashion.
- **KMD server encrypts** the new parent key using the vertex's newly updated key and its sibling's key respectively, and multicasts this pair of the ciphertexts to the entities attached in their respective subtrees.
- **Entities:** Decrypt to obtain these vertex keys, and update their key sets.