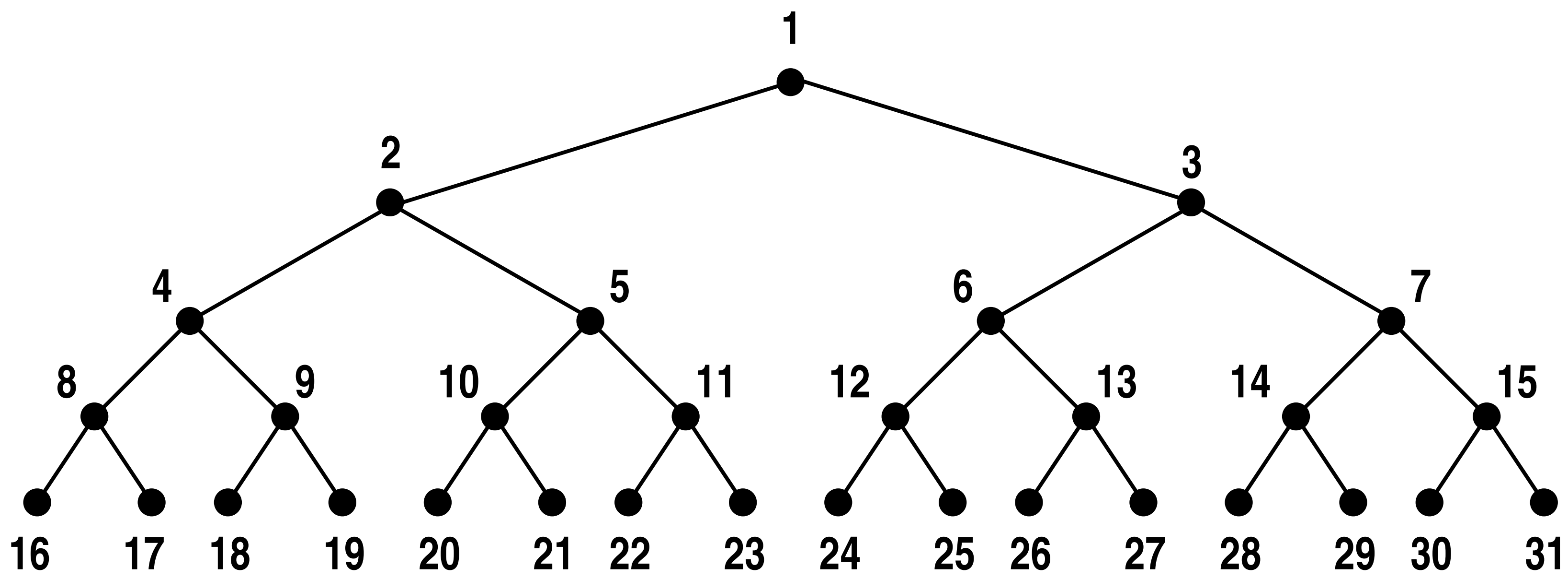
Logical Key Hierarchy (LKH)

- suppose the number of users, n satisfies $2^{d-1} < n \leq 2^d$
- construct a binary tree, say $\mathcal{T}$, of depth d, having exactly n leaf nodes
- the n leaf nodes of $\mathcal{T}$ correspond to the n users: for every user U, let U also denote the (leaf) node corresponding to the user U
- there is a key associated with *every* node in the tree
- for every node X, let $k(X)$ denote the key for node X
- $k(R)$ is the group key, where R is the root node of $\mathcal{T}$
- every user U is given the $d+1$ keys belonging to the nodes of $\mathcal{T}$ that lie on the unique path from U to R in $\mathcal{T}$
- therefore every user has $O(\log n)$ keys

Example

A binary tree with $d = 4$ and $n = 16$, having nodes labelled $1, 2 \ldots, 2^{d+1} - 1 = 31$:



A Simple Data Structure For Binary Trees

- the leaf nodes are labelled $2^d, 2^d+1, \ldots, 2^{d+1}-1$
- the *parent* of node j $(j \neq 1)$ is node $\lfloor \frac{j}{2} \rfloor$
- the *children* of node j $(j \leq 2^d - 1)$ are nodes $2j$ and $2j+1$
- the *sibling* of node j $(j \neq 1)$ is node

$$\begin{array}{ll} j+1 & \text{if } j \text{ is even} \\ j-1 & \text{if } j \text{ is odd} \end{array}$$

- therefore the binary tree can be stored in the form of an array, $T[1], \ldots, T[2^{d+1}-1]$, and it is a simple matter to find parents, children and siblings of nodes in this data structure

User Revocation: Removing User U

- let $\mathcal{P}(U)$ denote the set of nodes in the unique path from a leaf node U to the root node R (recall that R has the label 1)
- it is necessary to change the keys corresponding to the d nodes in $\mathcal{P}(U)\backslash\{U\}$
- for each node $X \in \mathcal{P}(U)\backslash\{U\}$, let $k'(X)$ denote the new key for node X
- let $\mathsf{sib}(\cdot)$ denote the sibling of a given node, and let $\mathsf{par}(\cdot)$ denote the parent of a given node
- the following $2d-1$ items are broadcast by the TA:
  1. $e_{k(\mathsf{sib}(U))}(k'(\mathsf{par}(U)))$
  2. $e_{k(\mathsf{sib}(X))}(k'(\mathsf{par}(X)))$ and $e_{k'(X)}(k'(\mathsf{par}(X)))$, for all nodes $X \in \mathcal{P}\backslash\{U, R\}$

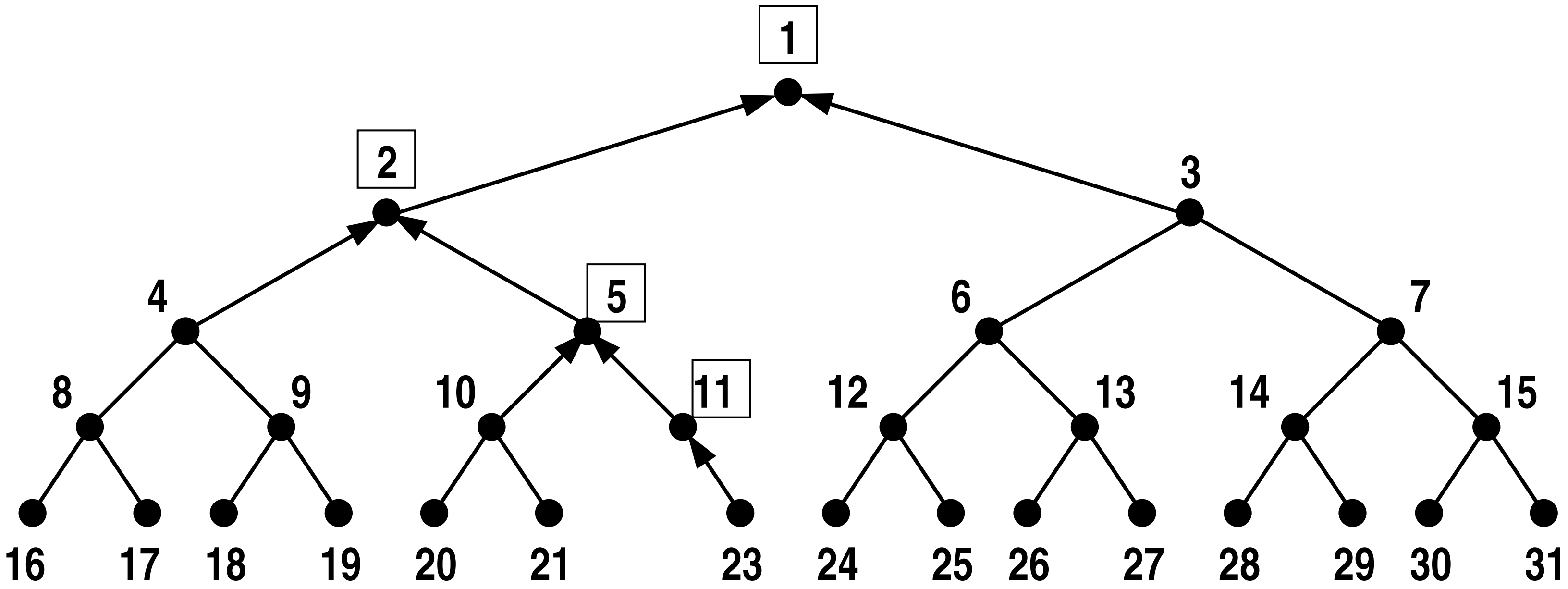
Example

- the broadcast allows any non-revoked user V to update all the keys in the intersection $\mathcal{P}(U) \cap \mathcal{P}(V)$
- for example, suppose the TA wants to revoke user $U = 22$
- the path $\mathcal{P}(U) = \{22, 11, 5, 2, 1\}$
- the TA creates new keys k'_{11}, k'_5, k'_2 and k'_1
- the siblings of the nodes in $\mathcal{P}(U)$ are $\{23, 10, 4, 3\}$
- the broadcast consists of

$$\begin{array}{cccc} e_{k(23)}(k'(11)) & e_{k(10)}(k'(5)) & e_{k(4)}(k'(2)) & e_{k(3)}(k'(1)) \\ & e_{k'(11)}(k'(5)) & e_{k'(5)}(k'(2)) & e_{k'(2)}(k'(1)) \end{array}$$

Example (cont.)

The labels of the nodes receiving new keys are boxed. Encryptions of new keys are indicated by arrows:



Comments

- every user stores $O(\log n)$ keys and the broadcast has size $O(\log n)$
- any number of users can be revoked over a period of time, without affecting the security of the system
- multiple (simultaneous) revocation can be done, but it is somewhat complicated
- new users can be added, whenever the current number of users is less than 2^d, by assigning the new user to an "unoccupied" leaf node of the tree
- if the number of users exceeds 2^d, then a new root node can be created
- this increases the depth d by one, and allows the number of users to be doubled

Copyright Protection

- protection against copyright violation is an important, but very difficult, challenge
- digital content can easily be copied and transmitted over computer networks
- content may be encrypted before it is transmitted
- for example, broadcast encryption protects encrypted content (i.e., unauthorized users cannot decrypt it)
- however, all content must eventually be decrypted in order to be useful
- after content is decrypted, it can potentially be copied
- "hardware-based" solutions provide a limited amount of protection
- other approaches include algorithms for *tracing*

Two Types of Copyright Violation

Here are two potential threats:

illegal content redistribution

Decrypted content can be copied and transmitted to others, for example in an illegal *pirate broadcast*.

illegal key redistribution

Here we are assuming that content is encrypted. The keys used to decrypt the content may be copied and distributed, or they may be used to create a *pirate decoder* which can subsequently be used to illegally decrypt encrypted content.

Fingerprinting

- suppose that every copy of some digital data, D, contains a different *fingerprint*, F
- for example, there might be 1 Mb of binary data, and the fingerprint might consist of 100 “special” bits “hidden” in the data
- the vendor can maintain a database of all the fingerprints, and the owners of the corresponding copies of the data
- then any exact copy can traced back to its owner
- potential problems include the following:
  - if the fingerprint is easily recognized, then it can be modified or destroyed, thus making the data impossible to trace
  - coalitions may be able to recognize fingerprints, even if individual uisers cannot do so

Mathematical Model

- for concreteness, suppose that (a copy of) the data, D, consists of L bits of *content*, say C, and an ℓ-bit *fingerprint*, F, i.e., $D = (C, F)$
- all copies of D have the same content but different fingerprints: $D_1 = (C, F_1), D_2 = (C, F_2), \ldots$
- the fingerprint bits always occur in the same (secret) positions in all copies of the data; e.g., bits $b_{i_1}, \ldots, b_{i_\ell}$ are fingerprint bits
- fingerprinting problems are usually studied assuming that a certain *marking assumption* holds:

  Given some number of copies of the data, say $D_1, D_2, \ldots, D_w$, the only bits that can be identified as fingerprint bits are those bits b such that $D_i[b] \neq D_j[b]$ for some i, j.

Possible Attacks

- the marking assumption implies that the content is irrelevant, and the problem reduces to studying properties of the fingerprints
- given w copies of the data, some bits can be identified as fingerprint bits
- a new "pirate" copy can be constructed, by setting values of these identified bits to be 0 or 1 arbitrarily
- the resulting data $D' = (C, F')$, where F' is a newly created *hybrid fingerprint*
- the fundamental question is whether a hybrid fingerprint can be "traced" if the fingerprints are constructed in a suitable way
- these concepts can be generalized to non-binary data (i.e., data consisting of strings over a specified *alphabet* of size q, say)

Hybrid Fingerprints

- an (ℓ, n, q)-*code* is a subset $\mathcal{C} \subseteq Q^\ell$ such that $|Q| = q$ and $|\mathcal{C}| = n$
- we have n *codewords* (i.e., fingerprints), each of which is an ℓ-tuple of elements from the alphabet Q
- let $\mathcal{C}_0 \subseteq \mathcal{C}$
- define $\mathsf{desc}(\mathcal{C}_0)$ to consist of all ℓ-tuples $\mathbf{f} = (f_1, \ldots, f_\ell)$ such that, for all $1 \leq i \leq \ell$, there exists a codeword $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathcal{C}_0$ such that $f_i = c_i$
- for example,

$$\mathsf{desc}(\{(1,1,2),(2,3,2)\}) = \{(1,1,2),(2,3,2),(1,3,2),(2,1,2)\}$$

- $\mathsf{desc}(\mathcal{C}_0)$ consists of all the hybrid fingerprints that can be constructed from the fingerprints in $\mathcal{C}_0$