

Portable and Flexible Document Access Control Mechanisms

Mikhail Atallah, Marina Bykova

Department of Computer Sciences and CERIAS
Purdue University

9th European Symposium on Research in Computer Security
(ESORICS'04)
September 2004

Outline

- Problem description
- Solution for unstructured data
- Solution for hierarchies
- Conclusions

Problem Description

- The model
 - We are given a very large data repository
 - Access is payment-based
 - Each customer can request a subscription to any subset of items
- Becomes important as the number and the level of maturity of on-line document collections grow
- Might not be challenging to solve without additional constraints

Problem Description (cont.)

- Constraints
 - For customer privacy subscriptions are not stored at the server
 - the design makes customer profiling difficult
 - Limited-capacity storage devices are used for policy configurations
 - important in the case of smart cards
 - results in inability to precisely represent all subsets
- The goal: minimal cost
 - The cost associated with “false positives” should be as small as possible

Problem Specification

- Repository contains n elements $1, \dots, n$.
- Customer can subscribe to any subset of $m \leq n$ items and is charged accordingly
- Access rights are stored on a card with k cells of $O(\log n)$ space each
 - $k \log n < n$ and $k < m$

Problem Specification (cont.)

- Design requirements:
 - Low rate of false positives
 - Transaction untraceability
 - Transaction unlinkability
 - Unforgeability
 - Unique policy representation
 - No additional sources of information
 - Fast access verification
 - Forward compatibility
 - In-house card generation

The Algorithm

- Given an order $r = \{r_1, \dots, r_m\}$ and stopping criteria $\tau = \{\tau_1, \dots, \tau_t\}$
 - Select a random seed s
 - $\forall r_i \in r$, permute the document $p_i = \pi_s(r_i)$
 - Sort p_i 's and compute the cost of permutation C
 - Apply evaluation criteria τ and stop if a sufficient subset of them $\tau' \subseteq \tau$ is satisfied
- One iteration can run in $O(m \log m)$ time

Producing a Permutation

- Any permutation satisfying these properties can be used:
 - It can be specified by a seed (and, given a seed, can be later reproduced)
 - A mapping for a single document can be generated independently from others
- The seed's length is $O(k \log n)$
 - gives n^k possibilities
 - limits the number of permutations that we can use

Computing the Cost of a Permutation

Input A sorted set of elements $\{p_1, \dots, p_m\}$

Output k intervals of the smallest cost

Algorithm steps :

- For $i = 1, \dots, m - 1$, compute $c_i = p_{i+1} - p_i - 1$
- In $O(m)$ time, select $(k - 1)^{\text{th}}$ largest among c_i 's (say, c_j)
- Select $k - 2$ largest among c_i 's, which will be the “gaps” between the intervals
- The cost is
$$C = \sum_{i=1}^{m-1} \{c_i \mid c_i < c_j\}$$

Card Operation

- Card is tamper-resistant
- Given a user request for document i , the card:
 - computes permutation $p_i = \pi_s(i)$
 - searches the k intervals for p_i
 - if successful, authenticates to the server and requests the document i .
- Shared keys or another form of anonymous authentication to the server is used

“Feasibility” Analysis

- The approach can be used in many settings and under many policies
- Alternatives for stopping criteria include:
 1. Threshold for number of false positives m'
 - is used to limit absolute or relative value of m'
 - $f(n, m, m') \leq m'_{max}$, for some function f
 2. Limited gain from cheating
 - each card keeps a count t' of requests that were denied
 - if t' reaches a threshold t , the card is deactivated
 - let n' denote the number of documents in which an attacker is interested

“Feasibility” Analysis (cont.)

- Alternatives for stopping criteria (cont.)
 2. Limited gain from cheating (cont.)
 - if $t' \leq t$, then $E(\text{gain}) \simeq t' \cdot \frac{c(m')}{n-m} \cdot \frac{n'}{n-m}$, where $c(m')$ is the monetary cost of m' documents
 - a possible policy can be $\frac{t \cdot c(m')}{n-m} \leq \alpha \cdot c_{card}$
 - can be satisfied with a small number of algorithm iterations
 3. Timeout rule
 - If no suitable permutation is found during a fixed interval, select the best permutation encountered

Structured Data

- Now instead of specifying intervals, we place records on nodes
- Two types of records: *positive* and *negative*
- A record placed on a node can have *local* or *recursive* effect
- Dynamic programming techniques are utilized to compute the cost of a tree in bottom-up fashion
 - each node is considered in *positive* and *negative* contexts
 - cost of an n -node binary tree can be computed in $O(n \cdot k^2)$ time
 - for t -ary trees, complexity becomes $O(n \cdot k^t)$

Structured Data (cont.)

- Extension: a record can cover a subtree of a variable height
 - all records are replaced with a single type
 - algorithm for computing the cost of a subtree takes height as an argument
 - this extension lowers the number of false positives
 - adds only an additional factor of h , complexity is $O(n \cdot k^t \cdot h)$
- Tree nodes cannot be easily permuted
 - uniqueness can be introduced through randomization of false positives

Conclusions

- Our scheme:
 - provides fine-grained document access control under space restrictions
 - respects customer privacy
 - contains efficient algorithms
- Future directions:
 - more thorough (formal and/or empirical) analysis
 - other data structures (e.g., grids for GIS)
 - other techniques for data compression
 - perfect minimal hash functions are under current consideration