Optimizing Segment Based Document Protection

Mirosław Kutyłowski   Maciej Gębala

Wrocław University of Technology

SOFSEM 2012

Segment Based Document Protection

Motivation

- For confidentiality, data in a digital document may be encrypted
  – then only a reader with the decryption key may read it.
- However, not all information are equally confidential: for each part there is a different subset of people that have the right to read it.
- Auxiliary right management information can be nicely encoded in an XML-structure of a document.
- The things might become messy: many different document user profiles, each profile defining the read access right for all parts of the document.

Segment Based Document Protection

Trivial solution

use a different encryption key for each part (segment)

Disadvantage

if a user has the right to many segments, then many keys must be given to him.

1-A	2-AB
3-AC	4-ABD
5-ABC	6-AC
7-BCD	8-ABC

$$A = \{K_1, K_2, K_3, K_4, K_5, K_6, K_8\}$$

$$B = \{K_2, K_4, K_5, K_7, K_8\}$$

$$C = \{K_3, K_5, K_6, K_7, K_8\}$$

$$D = \{K_4, K_7\}$$

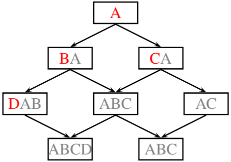
Segment Based Document Protection

access graph for a document

Access graph G

- Nodes of G - segments of the document.
- There is an arc AB in G, if all users having access right to A have also access right to B.
- Each node is labeled by the key used to encrypt that segment.

Well designed G is acyclic.
A user having access right to a segment A has also access right to every B such that there is a directed path from A to B.



Dags and Key Derivation Techniques

one-way derivation of keys

Goal

reduce the number of keys that need to be given to a user.

Linear scheme

If the access graph defines a linear ordering $A_1 \succeq A_2 \succeq \ldots \succeq A_m$, then the corresponding keys may be derive with any one-way (hash) function H. Choose a key K_1 at random and for $i = 2, \ldots, m$ derive

$$K_i \leftarrow H(K_{i-1})$$

Advantage

It suffices to give a user the key K_i, where i is the smallest number such that the user has the access right to A_i.

Tree schemes

If the access graph is a tree, then a similar solution applies:

If a node A has child nodes $B_1, \ldots B_k$ and a key K has been assigned to A, then to B_i ($i \leq k$) we assign the key

$$K_i \leftarrow H(i, K)$$

Dags and Key Derivation Techniques

schemes for arbitrary dag

Arbitrary dags - technique 1

For a dag P describing the access rights find a mapping $\rho : P \to N$, such that $A \succeq B$ iff $\rho(A)|\rho(B)$.
Use a multiplicative group $\mathbb{G}$ such that computing the roots in $\mathbb{G}$ is infeasible (e.g. use RSA Assumption).
Choose an element $g \in \mathbb{G}$, and compute $K_A \leftarrow g^{\rho(A)}$ as the key for node A.
If $\rho(A)|\rho(B)$, then $K_B \leftarrow K_A^{\rho(B)/\rho(A)}$.
Disadvantages: high computational complexity, low flexibility.

Dags and Key Derivation Techniques

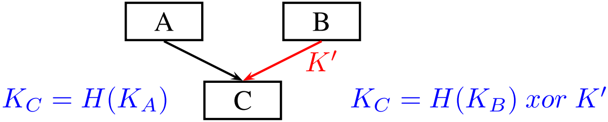
schemes for arbitrary dag

Arbitrary dags - technique 2

- Use an arbitrary tree scheme - this leads to conflicts when a node v has more than one incoming arcs.
- If the key of node C should be K_C and the computation for an arc (B, C) yields K_B, then define the offset for arc BC as K_C *xor* $H(K_B)$. Then:

$$K_C = H(K_B) \; xor \text{ offset for } BC$$

- The offset is the public information corresponding to the arc included in the document.



Problem Statement

In the segmented document we have to describe:

- key derivation method for each arc of the access graph,
- offset, if it is necessary for a given arc.

Our goal

Find solution with low space overhead of the document encoding and small computational requirements.

Eliminate as many offsets as possible!

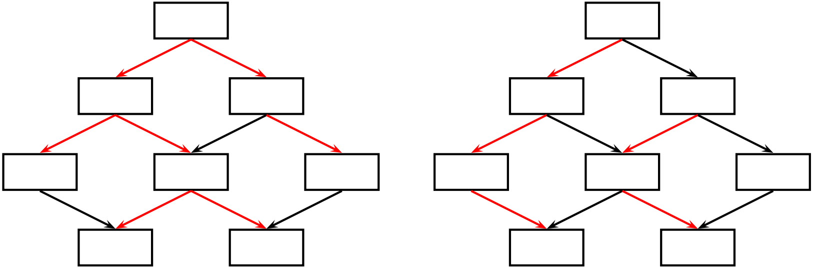
Problem Statement

Idea:

for tree method: embed a tree or disjoint trees in G

for path method: embed a path or disjoint paths in G

and use the offsets only for those edges that are not covered by the embedding.



Problem Statement

Problem 1

Given a dag *G*, embed some number of trees in G so that

- the embedded trees are node disjoint,
- the number of arcs that do not belong to any of embedded trees is minimal.

Problem 2

Given a dag G, embed some number of paths in G so that

- the embedded paths are node disjoint,
- the number of arcs that do not belong to any of embedded paths is minimal.

Tree-Based Key Derivation

algorithm

Input: a dag G
Output: a subgraph G' of G that consists a set of disjoint trees and containing the maximal possible number of arcs

Algorithm: Construct a reduced graph G', a subgraph of G, in the following way:
for each node $v \in G$ of indegree greater than 1 pick up an arbitrary arc with endpoint v and remove all other arcs with endpoint v.

Proposition 1

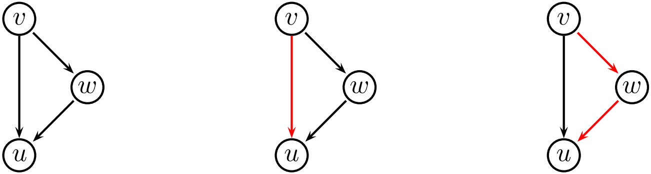
The number of arcs in G' does not depend on choices done during algorithm execution. Moreover, no forest embedded in G may contain more arcs than G'.

So for the tree method, finding an optimal embedding is straightforward.

Sequential Key Derivation

situation

for paths the greedy algorithm mail fail to find an optimal solution:



Sequential Key Derivation

algorithm

The problem can be solved by finding a **minimal vertex-disjoint path cover** in a directed acyclic graph.

A vertex-disjoint path cover in a DAG $G = (V, E)$ is a set of paths P such that every vertex in V is included in exactly one path in P. Paths may start and end anywhere, and they may be of any length, including 0. A minimum path cover of G is a path cover containing the fewest possible paths (i.e. maximum possible edges).

Solution by reduction to **maximum flow problem** thus complexity $O(|V||E|)$.

Final Remarks

open problems

Taking frequency into account

Known: the fraction of users that will have access to a given node of the access graph.

Process: for each user create the subdocument with only those segments that are accessible by him

Optimization: minimize the total volume of subdocuments

Tree-based scheme

The modified algorithm selects an arc with endpoint *v* which is used most frequently. **This constructs an optimal solution.**

Final Remarks
open problems

Sequential scheme - ~~open question~~

A similar argument cannot be used for finding optimal paths.

The reason: selecting an arc in our scheme does not change the number of arcs in the optimal solution, but changes a lot the shape of all paths in the final solution. As each decision has global consequences, it is unclear how to make an optimal choice.

Solution

Solution by reduction to **maximum weighted bipartite matching** thus complexity $O(|V|^2|E|)$.

Thanks for your attention!