Optimizing Segment Based Document Protection*

Corrected Version

Mirosław Kutyłowski and Maciej Gębala

Faculty of Fundamental Problems of Technology,
Wrocław University of Technology
{miroslaw.kutylowski, maciej.gebala}@pwr.wroc.pl

Abstract. In this paper we provide a corrected and generalized version of the scheme presented at SOFSEM'2012 in our paper "Optimizing Segment Based Document Protection" (SOFSEM 2012: Theory and Practice of Computer Science, LNCS 7147, pp. 566-575).
We develop techniques for protecting documents with restricted access rights. In these documents so called *segments* are encrypted. Different segments may be encrypted with different keys so that different user may be given different *access rights*. Hierarchy of access rights is represented by means of a directed acyclic *access graph*. The segments are encrypted with keys - where each key corresponds to one node in the access graph. The main feature of the access graph is that if there is an arch $\overrightarrow{AB}$ in the graph, then all segments labelled with B can be decrypted with the key corresponding to node A.
We show how to minimize the space overhead necessary for auxiliary keying information stored in the document. We provide an algorithm based on node disjoint paths in the access graph and key derivation based on one-way functions. Our current solution, based on maximal weighted matchings, provides an optimal solution for creating subdocuments, in case when frequency of creating each subdocument is known.

Keywords: document protection, access rights, key management, key hierarchy, directed acyclic graph

1 Introduction

We consider securing digital documents from unauthorized access. Since digital documents can be copied and cryptographic mechanisms secure their integrity the ease of creating the copies is not only an advantage, but also creates new challenges: namely it is hard to protect the document from unauthorized access.

If the document has to be presented as a whole or not presented at all, then protection via encryption is enough: the document becomes readable only for a party that gets the decryption key (this approach is followed in particular in

* This research has been partially supported by Foundation for Polish Science, Programme "MISTRZ"

broadcast systems). The situation is different, if certain parts of a document have to be disclosed selectively. In order to catch this situation the model of segment based protection has been introduced.

Segment Based Document Protection Securing multiple parts of a document in different ways is a concept that appeared many times the literature; for recent application oriented work see [1],[2] and in particular the former version of this paper [3]. The idea is the following:

- The document contents is encoded together with some auxiliary information for management of access rights; in particular a standard XML-encoding can be used.
- The documents contents is divided into *segments*, a segment is a part of the document that is dedicated to the readers with exactly the same access rights.
- Access to a segment is protected by encryption with a symmetric key that is available only to the parties that are entitled to read this segment.
- Dependencies of access rights between segments are described by a directed acyclic graph (dag) called *access graph*. Its nodes are segments of a document. Access graph defines a partial ordering of segments: we say that $A \succeq B$ if there is a directed path from node A to node B. The edges describe inherited access rights. Namely, if $A \succeq B$, then a user having the right to read segment A should also be given possibility to decrypt and read segment B.
- A document stores public *key information*. A user having a key corresponding to a segment A can use this meta-data to derive the key of any segment B such that $A \succeq B$.

Obviously, the access graph should be acyclic. The relation $\succeq$ need not to be a linear ordering.

Our main goal is to minimize the size of auxiliary key information to be stored in the document. However, computational complexity, flexibility, scalability and simplicity of the solution are also important factors.

1.1 Preliminary Techniques

Linear schemes. Let the access graph define a linear ordering $A_1 \succeq A_2 \succeq \ldots \succeq A_m$. The keys may be derived using a one-way function F (see e.g. [4]):

- K_1 is chosen at random,
- for $i = 2, \ldots, m$:

$$K_i := F(K_{i-1}) \tag{1}$$

Then K_i is assigned to A_i, for $i \leq m$.

Formula (1) can be used by a holder of K_i to derive the keys $K_{i+1}, \ldots, K_m$. However, deriving $K_1, \ldots, K_{i-1}$ is infeasible due to one-wayness of F.

Tree schemes. A similar solution can be applied if access graph is a tree ([5]): if a node A has child nodes $B_1, \ldots B_k$, and K has been assigned to A, then for $i \leq k$ the node B_i gets the key

$$K_i := F_i(K) \tag{2}$$

where $F_1, \ldots, F_k$ are distinct and unrelated one-way functions. For instance, we can use a cryptographic hash function H and set: $F_i(x) \stackrel{def}{=} H(i, x)$. Another solution would be that

$$K_i := H(K, A, B_i) \tag{3}$$

In both cases we require strong cryptographic properties of H - immunity against all kinds of *related preimage attacks*. For instance, given K_i and K_j, it should be infeasible to derive K satisfying simultaneously $K_i = F_i(K)$ and $K_j = F_j(K)$, as well as to derive K_t for $t \neq i, j$. Therefore the tree scheme requires stronger functions than the linear scheme, where one-wayness seem to be sufficient.

Arbitrary posets. The interesting case is when the access graph is a poset (see e.g. [6] and [2]). If we apply the same method as for trees, then we get into troubles for each node B that have more than one incoming arcs. In this case the key for B is derived in many conflicting ways. Designing F such that we get the same values is generally hopeless, as this would mean creating conflicts of F almost on demand.

A simple solution is to store an offset for the arcs of the access graph. Namely, if a node C has two parent nodes A and B holding, respectively, K_A and K_B, then the key K_C is derived as $K_C := F(K_A, A, C) \oplus \Delta_{AC}$ and $K_C := F(K_B, B, C) \oplus \Delta_{BC}$, where Δ_{AC}, Δ_{BC} are the public offset stored for the arcs $\overrightarrow{AC}$ and $\overrightarrow{BC}$. Obviously, Δ_{AC} can be derived as $K_C \oplus F(K_A, A, C)$. If a node C has more parent nodes, then we define nonzero offsets for all but one incoming arcs.

In order to save space, we can set (at most) one of the offsets to a string of zeroes. Which offsets should be set to the zero string is the main topic of this paper.

1.2 Document Encoding and Problem Statement

Let us restate our goals from [3]. We concern creating offsets in two cases:

linear scheme: in this case we use a number of node disjoint paths and assign the keys along the paths according to Equation (1) – i.e. with the zero offsets.

tree scheme: in this case we embed a number of node disjoint trees and assign the keys within each tree according to Equation (2) – i.e. with the zero offsets.

In both cases the remaining offsets are determined as $\oplus$ of the keys on the ends of the arch (if both are already determined).

If we decide to use a tree scheme or a linear scheme for key derivation in a dag, we are faced with the following problems:

Problem 1 ([3]). 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 ([3]). 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.*

2 Tree-Based Key Derivation

In [3] we have proposed the following greedy algorithm:

Algorithm 1

Input: dag G
Output: subgraph G' of G that consists of a set of disjoint trees and contains 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.

The construction has the following simple properties:

- G' is a forest,
- the number of edges in G' does not depend on the random choices,
- every forest which is a subgraph of G is a subgraph of some forest generated by Algorithm 1.

For details see [3].

3 Sequential Key Derivation

The situation for node disjoint paths is more complicated. The example from Fig. 1 shows that a greedy procedure may lead to suboptimal solutions.

Error in [3]. Unfortunately, Algorithms 2 (and Algorithm 3 as well) presented in our paper [3] does not necessarily provide optimal solutions. The simplest example is the access graph from Fig. 1. Algorithm 2 from [3] selects an arbitrary node with indegree 0 and selects one of the outcoming edges for inclusion in the final solution. In our case the algorithm must select v and then may choose either arc $\overrightarrow{vu}$ or $\overrightarrow{vw}$. Then the algorithm is called recursively for a reduced graph. In the

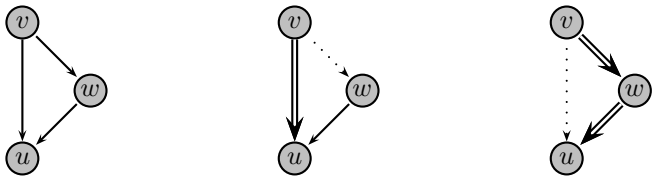

Fig. 1. ([3]) Let us consider a dag consisting of arcs (v, u), (v, w) and (w, u) - a triangle with source v and sink u. If during a greedy procedure we take the arc (v, u) (or remove (v, w)), then the final solution will contain just one path with a single arc. On the other hand, the optimal solution is the path $(v, w), (w, u)$.

first case, no edge will be added in this stage and the final output will contain only arc $\overrightarrow{vu}$, which is not optimal.

This example also shows that Lemma 1 from [3] is false. The gap in the proof is disregarding the case when an optimal solution already contains an arc starting from a point v. In this case it might be impossible to remove this arc from the solution and to include another arc starting from v.

New Algorithm. Below we present a simple Algorithm 2 that solves the problem of finding the maximal set of node disjoint paths. Execution of the algorithm for an example dag is shown in Figure 2.

Algorithm 2

Input: dag $G = (V, E)$
Output: subgraph S of G that consists of node disjoint paths and contains the maximal possible number of arcs

Algorithm:

1. construct bipartite graph $H = (V', E')$ where
   $V' = \{\ x_i\ :\ \forall\ i \in V\ \} \cup \{\ y_i\ :\ \forall\ i \in V\ \}$ and
   $E' = \{\ \langle x_i, y_j \rangle\ :\ \forall\ (i, j) \in E\ \}$
2. find maximal matching for graph H
3. for each edge $\langle x_i, y_j \rangle$ that belongs to the maximal matching, add arc (i, j) to S

Now we argue that Algorithm 2 outputs an optimal solution.

Lemma 1. *Output S of Algorithm 2 defines a family of disjoint paths in G that contains the maximal possible number of arcs.*

Proof. Consider a node i. Since at most one edge of the form $\langle x_i, y_j \rangle$ belongs to the matching, there is at most one arc in S starting in i. So the outdegree of

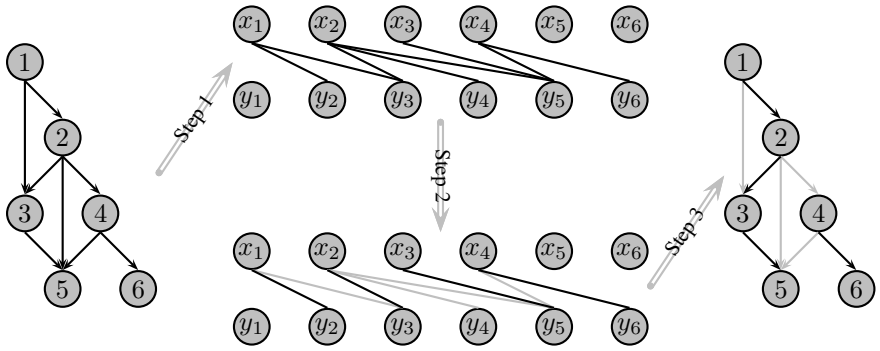

Fig. 2. Execution of the Algorithm 2 for an example dag.

each node in S is at most 1. Similarly, the matching contains at most one edge of the form $\langle x_k, y_i \rangle$. Therefore, there is at most one arc in S with the endpoint i. It follows that the indegree of each node in S is at most 1. As both indegree and outdegree of each node in S is at most 1, it defines a set of node disjoint paths.

On the other hand, it is easy to see that a set of node disjoint paths in G defines a matching consisting of the same number of edges in the graph H defined by Algorithm 2. Therefore any set of node disjoint paths in G does not have more arcs than the number of edges in the maximal matching in G. □

Let us observe that Algorithm 2 not only derives the optimal solution, but it is also efficient – its time complexity is $O(\sqrt{|V|}|E|)$ (see [7]).

4 Weighted Graphs

In this section we show that Algorithm 2 can be generalized to solve the open problems formulated in [3]. Namely, in some application areas it might be useful to generate subdocuments of a given document truncated to those segments that are readable by some specific group of users. In this case we consider a subgraph $\bar{G}$ of access right graph G with the following property: if v is a vertex in $\bar{G}$, then all arcs of the form (v, u) from G are also arcs of $\bar{G}$. Therefore, a node v from $\bar{G}$ has the same ancestors in $\bar{G}$ as in G. If D is an encoded document with access graph G, we construct a new document D' such that:

- D' contains only those segments that correspond to the nodes of $\bar{G}$, all remaining segments are removed,
- D' contains only those key information fields that correspond to the arcs of $\bar{G}$.

When truncated versions of a document D are concerned, then optimization of key derivation scheme has to be reconsidered. We assume that for each subdocument D' of D we know fraction $fr(D')$ of users that obtain D'. Therefore

the goal is to maximize the number of arcs in all documents, for which the offsets need not to be stored:

$$\sum_{D' \text{ subdocument of } D} fr(D') \cdot \text{gain}(D') \tag{4}$$

where $\text{gain}(D')$ denotes the number of arcs in the access graph corresponding to D', for which there is no offset information.

Tree-based scheme. Given a node v in G of indegree higher than 1, Algorithm 1 selects one of arcs with endpoint v. For the generalized scheme we need to optimize the choice taking into account frequency data about access rights. Namely, to each node w in G we associate frequency parameter $fr(w)$ which describes the fraction of users that are given the access rights corresponding to w. We can easily see that the arc (u, v) is contained in the fraction $fr(u)$ of all documents assigned to the users. Therefore in order to minimize the value (4) the modified Algorithm 1 for a node v selects an arc (u, v) such that

$$fr(u) = \max\{fr(u') | (u', v) \text{ is an arc in } G\} .$$

This minimizes the number of offset information regarding the arcs pointing to v in all documents given to the users. As the choices done by the algorithm at different nodes are not dependent, these leads to an optimal solution when regarding the whole access graph.

Sequential scheme. For the sequential scheme we modify Algorithm 2 to solve *Maximum Weighted Bipartite Matching Problem* instead of *Maximum Bipartite Matching Problem*. The weight of an edge $\langle x_i, y_j \rangle$ is defined as $fr(i)$. It is easy to see that weight of the of a matching equals to the gain defined by (4).

As the time complexity of solving Maximum Weighted Bipartite Matching Problem is $O(|V||E| \log |V|)$ [8], finding the optimal solution is possible for access graphs of realistic sizes.

References

1. Qiu, R., Tang, Z., Gao, L., Yu, Y.: A novel xml-based document format with printing quality for web publishing. In: Imaging and Printing in a Web 2.0 World; and Multimedia Content Access: Algorithms and Systems IV. Proc. SPIE 7540, Society of Photographic Instrumentation Engineers (2010) 0J (Cited on page 2.)
2. Xu, D., Tang, Z., Yu, Y.: An efficient key management scheme for segment-based document protection. In: Consumer Communications and Networking Conference (CCNC), 2011 IEEE. (2011) 896 –900 (Cited on pages 2 and 3.)
3. Kutylowski, M., Gebala, M.: Optimizing segment based document protection. In Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G., eds.: SOFSEM. Volume 7147 of Lecture Notes in Computer Science, Springer (2012) 566–575 (Cited on pages 2, 3, 4, 5, and 6.)

4. Levin, L.A.: The tale of one-way functions (2003). Available from: `http://arxiv.org/abs/cs.CR/0012023` [retrieved on May 20, 2011] (Cited on page 2.)
5. Hassen, H.R., Bouabdallah, A., Bettahar, H.: A new and efficient key management scheme for content access control within tree hierarchies. In: AINA Workshops (1), IEEE Computer Society (2007) 551–556 (Cited on page 3.)
6. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. **12**(3) (2009) (Cited on page 3.)
7. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing **2**(4) (1973) 225–231 (Cited on page 6.)
8. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM **19** (1972) 248–264 (Cited on page 7.)