# Watermarking PDF Documents using Various Representations of Self-inverting Permutations

Maria Chroni and Stavros D. Nikolopoulos
*Department of Computer Science & Engineering, University of Ioannina, GR-45110 Ioannina, Greece*

Keywords: Watermarking Techniques, Text Watermarking Algorithms, PDF Documents, Self-inverting Permutations, Representations of Permutations, Embedding Algorithms, Extracting Algorithms.

Abstract: Portable Document Format (PDF) documents are extensively used over the internet for information exchange and due to the ease of copying and distributing they are susceptible to threats like illegal copying, redistribution, and plagiarism. This work provides to web users copyright protection of their PDF documents by proposing efficient and easily implementable techniques for PDF watermarking; our techniques are based on the ideas of our recently proposed watermarking techniques for software, image, and audio, expanding thus the digital objects that can be efficiently watermarked through the use of self-inverting permutations. In particular, we present various representations of a self-inverting permutation $\pi^*$ namely 1D-representation, 2D-representation, and RPG-representation, and show that theses representations can be efficiently applied to PDF watermarking. Indeed, we first present an audio-based technique for marking a PDF document $T$ by exploiting the 1D-representation of a permutation $\pi^*$, and then, since pages of a PDF document $T$ are 2D objects, we present an image-based algorithm for encoding $\pi^*$ into $T$ by first mapping the elements of $\pi^*$ into a matrix $A^*$ and then using the information stored in $A^*$ to mark invisibly specific areas of PDF document $T$. Finally, we describe a graph-based watermarking algorithm for embedding a self-inverting permutation $\pi^*$ into the document structure of a PDF file $T$ by exploiting the RPG-representation of $\pi^*$ and the structure of a PDF document. We have evaluated the embedding and extracting algorithms by testing them on various and different in characteristics PDF documents.

## 1 INTRODUCTION

Information age has altered the way people communicate by breaking the barriers imposed on communications by time, distance, and location and has undoubtedly impact not only human activities but also global industry and economy.

An electronic document is an extensively used medium traveling over the internet for information exchange and due to the ease of copying and distributing they are susceptible to threats like illegal copying, redistribution of copyrighted documents, and plagiarism. Subsequently, it has become more important to protect the electronic documents from any malicious user while existing in the digital world. Copyright protection of digital contents is such a need of time which cannot be overlooked. In past, various methods like encryption, steganography and watermarking has been used to solve these problems. However, digital watermarking is the better solution for copyright protection than encryption and steganography. It is well known that digital watermarking methods are efficient enough to identify the original copyright owner of the contents. Note that there are many reasons why you would want to use watermarks in digital documents: as a copying deterrent, as a means of identifying the source of a printed document, as a means of determining whether a document has been altered, etc.

Any action that a user can perform on a text that can affect the watermark, or its usefulness, is called attack. In (Zhou et al., 2009) existing attacks on text watermarking can be classified into three main categories: *watermark attacks*, *geometric attacks*, and *system attacks* (Collberg and Nagra, 2010).

Text watermarking is the area of research that has emerged after the development of internet and communication technologies; we mention that the first reported effort on marking documents dates back to 1993. Previous work on digital text watermarking is based on several techniques among which image-based approach (Brassil et al., 1995; Huang and Yan, 2001; Low et al., 1998; Low and Maxemchuk, 2000; Maxemchuk and Low, 1997; Maxemchuk and Low, 1998), syntactic approach (Atallah et al., 2003; Meral

Chroni M. and Nikolopoulos S..
Watermarking PDF Documents using Various Representations of Self-inverting Permutations.
DOI: 10.5220/0005445800730080
In *Proceedings of the 11th International Conference on Web Information Systems and Technologies* (WEBIST-2015), pages 73-80
ISBN: 978-989-758-106-9

et al., 2009), and semantic approach (Lu et al., 2008; Vybornova and Macq, 2007; Topkara et al., 2007; Sun and Asiimwe, 2005). Recently, a significant number of techniques have been proposed in the literature which use Portable Document Format (PDF) files as cover media in order to hide data (Bindra, 2011; Liu et al., 2012; Liu et al., 2008; Liu et al., 2006; Lee and Tsai, 2010; Zhong et al., 2007).

In this paper, in order to provide to web users copyright protection of their digital documents, we present techniques for watermarking PDF documents by exploiting several representations of a self-inverting permutation $\pi^*$, i.e. the 1D-representation, the 2D-representation, and the RPG-representation. Our main contribution is a graph-based watermarking algorithm for embedding a self-inverting permutation $\pi^*$ into the document structure of a PDF file $T$ using the RPG-representation of $\pi^*$ and the structure of a PDF document.

## 2 BACKGROUND RESULTS

In this section we give some definitions and the theoretical background we use towards the watermarking of Portable Document Format (PDF) documents.

**Self-inverting Permutations.** Let $\pi$ be a permutation over the set $N_n = \{1,2,\ldots,n\}$. We think of permutation $\pi$ as a sequence $(\pi_1,\pi_2,\ldots,\pi_n)$, so, for example, the permutation $\pi = (1,4,2,7,5,3,6)$ has $\pi_1 = 1$, $\pi_2 = 4$, etc (Golumbic, 1980).

The *inverse* of $\pi$ is the permutation $\tau = (\tau_1,\tau_2,\ldots,\tau_n)$ with $\tau_{\pi_i} = \pi_{\tau_i} = i$. Clearly, every permutation has a unique inverse, and the inverse of the inverse is the original permutation.

A *self-inverting permutation* (or, for short, SiP) is a permutation $\pi = (\pi_1,\pi_2,\ldots,\pi_n)$ that is its own inverse, i.e., $\pi_{\pi_i} = i$, for $i = 1,2,\ldots,n$.

The definition of the inverse of a permutation implies that a permutation is a self-inverting permutation iff all its cycles are of length 1 or 2.

**1D-representation of SiP.** In our 1D-representation (Chroni et al., 2014), the elements of the permutation $\pi$ are mapped in specific cells of an array $B$ of size $n^2$ as follows:

- number $\pi_i \longrightarrow$ entry $B((\pi_{\pi_i}^{-1}-1)n+\pi_i)$

or, equivalently, the cell at the position $(i-1)n+\pi_i$ is labeled by the number $\pi_i$, for each $i = 1,2,\ldots,n$.

In our 1DM representation, a permutation $\pi$ over the set $N_n$ is represented by an $n^2$ array $B^*$ where the cells at positions $(i-1)n+\pi_i$ are marked by a specific symbol, say, the asterisk character "*".

**2D-representation of SiP.** In (Chroni et al., 2013), we have defined the 2D-representation of a SiP as the representation where the elements of the permutation $\pi = (\pi_1,\pi_2,\ldots,\pi_n)$ are mapped in specific cells of an $n \times n$ matrix $A$ as follows:

- number $\pi_i \longrightarrow$ entry $A(\pi_i^{-1},\pi_i)$

or, equivalently, the cell at row $i$ and column $\pi_i$ is labeled by the number $\pi_i$, for each $i = 1,2,\ldots,n$.

In 2DM-representation the cell at row $i$ and column $\pi_i$ of matrix $A$ is marked by a specific symbol, for each $i = 1,2,\ldots,n$.

**RPG-representation of SiP.** We have also presented an efficient and easily implemented algorithm for encoding numbers as reducible permutation graphs (or, for short, RPG) through the use of self-inverting permutations (Chroni and Nikolopoulos, 2012). In particular, we have proposed the algorithm `Encode_SiP.to.RPG` which applies to any permutation $\pi$ and relies on domination relations on the elements of $\pi$.

Figure 1 summarizes by an example the representations of the permutation $\pi^* = (4,7,6,1,5,3,2)$.

### 2.1 Structure of a PDF Documents

The Portable Document Format (PDF) (Adobe, 2006) is an open standard (defined in ISO 32000) which facilitates device and platform independent capture and representation of rich information such as text, multimedia and graphics, into a single medium. Thus the PDF format enables viewing and printing of a rich document, independent of either application software or hardware. In this section we present a structural analysis of a PDF file and give its basic components.

**Object.** An object is the basic element in PDF files, in which eight kinds of objects, namely Boolean, Numeric, String, Name, Array, Null, Dictionary and Stream are sustained. Objects may be labeled so that they can be referred to by other objects. A labeled object is called an indirect object.

**File Structure.** The PDF file structure determines how objects are stored in a PDF file, how they are accessed, and how they are updated. The file structure (see, Figure 2) includes the following:

- an one-line header identifying the version of the PDF specification to which the file conforms,
- a body containing the objects that make up the document contained in the file,

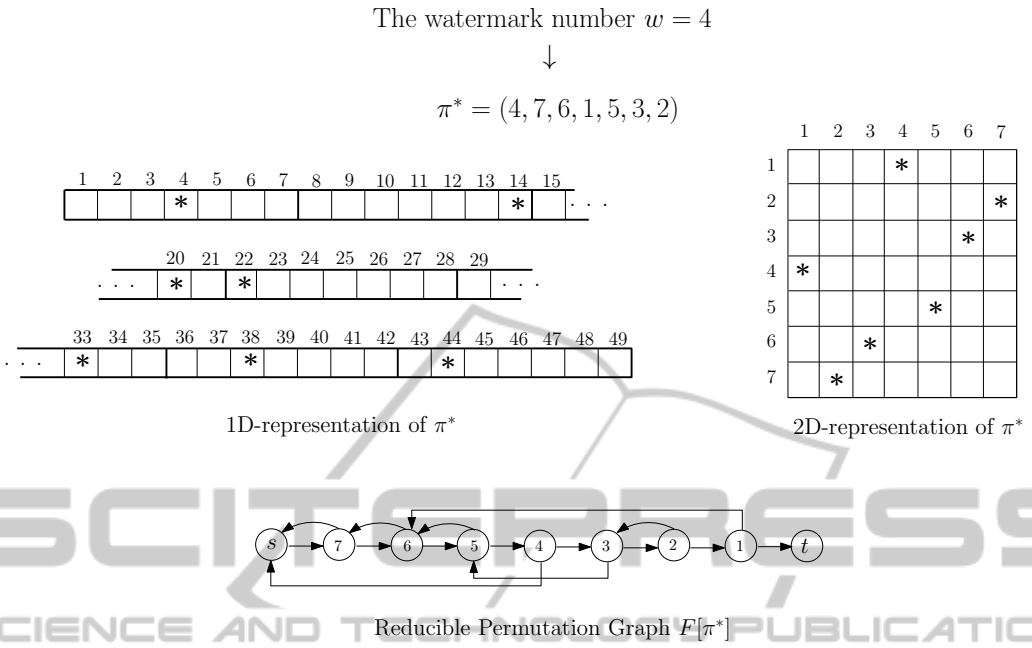




Figure 1: Three different representations of permutation $\pi^* = (4,7,6,1,5,3,2)$.

- a cross-reference table containing information about the indirect objects in the file, and
- a trailer giving the location of the cross-reference table and of certain special objects within the body of the file.

Figure 2 shows an example of a PDF file and its internal file structure.

**Document Structure.** The PDF document structure specifies how the basic object types are used to represent components of a PDF document: pages, fonts, annotations, and so forth. The document structure of a PDF file is organized in the shape of an object tree topped by Catalog, Page tree, Outline hierarchy and Article thread included. The Outline hierarchy is the bookmarker of PDF, and Page tree includes page and Pages which in turn includes the total page number and each page marker. Page, the main body of PDF file, is the most important object which involves the typeface applied, the text, pictures, page size, etc.

# 3 WATERMARKING PDF DOCUMENTS

In this section we describe embedding algorithms for encoding a SiP $\pi^*$ into a digital document $T$. More specifically, we embed the permutation $\pi^*$ into a PDF document by exploiting the 1D, the 2D, and the RPG-representation of the permutation $\pi^*$.

## 3.1 Embed Watermark into PDF - I

Our embedding algorithm watermarks a PDF document by exploiting the 1D-representation of $\pi^*$; the marking is performed by increasing the space between two consecutive words in a paragraph of $T$.

Let $B^*$ be the 1D array of size $n = n^* \times n^*$ which represents the permutation $\pi^*$ of length $n^*$, and let $(w_1, s_1)$, $(w_2, s_2)$, ..., $(w_n, s_n)$ be $n$ pairs of type "word-space" of a paragraph *par* of the input PDF document; recall that the entry $B^*((i-1)n^*+\pi_i^*)$ contains the symbol "*", $1 \le i \le n^*$. The algorithm increases by a small value "c" the $i$-th space of the pair $(w_i, s_i)$ if $B^*((i-1)n^*+\pi_i^*) = "*"$; our embedding algorithm works as follows:

**Algorithm** `Embed_SiP.to.PDF-I`

1. Compute the 1DM representation of the permutation $\pi^*$, i.e., construct the array $B^*$ of size $n = n^* \times n^*$ where the $(i-1)n^*+\pi_i^*$ entry of $B^*$ contains the symbol "*", $1 \le i \le n^*$;
2. Select an appropriate paragraph *par* on a page $P$ of PDF document $T$ to embed the self-inverting

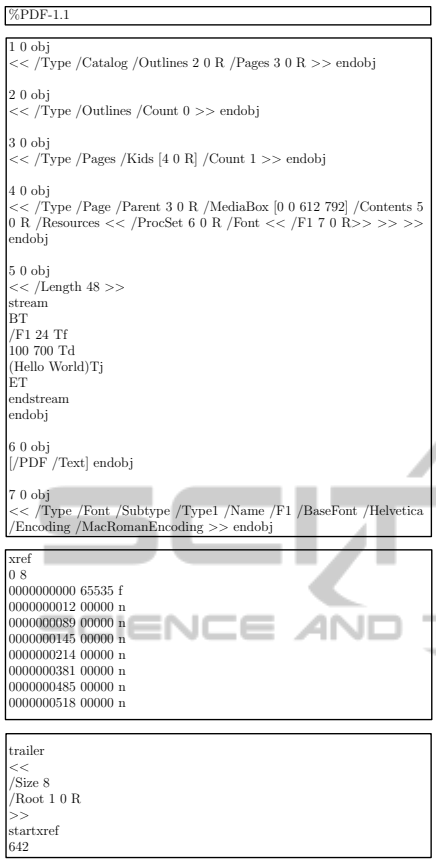


Figure 2: The structure of a PDF file along with its code containing, in object `5 0 obj`, the text "Hello World".

permutation $\pi^*$;

3. Partition the paragraph *par* into $n$ pairs $(w_1,s_1),(w_2,s_2),\ldots,(w_n,s_n)$, where $w_i$ and $s_i$ are the $i$-th word and space, respectively, in selected paragraph *par*, $1 \le i \le n$;
4. For each pair $(w_i,s_i)$ s.t. $B^*((i-1)n^*+\pi_i^*) = "*"$, increases the space $s_i$ or, equivalently, distance $d(w_i,w_{i+1})$ between words $w_i$ and $w_{i+1}$, by a relative small value $c$, $1 \le i \le n$;
5. Return the watermarked PDF document $T_w$.

**Extraction.** The extraction algorithm, which we call `Extract_PDF.from.SiP-I`, operates as follow: it takes as input the watermarked PDF document $T_w$, locates the paragraph *par*, and computes the permutation $\pi^*$ by finding the positions of the words $w_i$ such that:

- $d(w_i,w_{i+1}) > d(w_{i-1},w_i)$, or
- $d(w_i,w_{i+1}) > d(w_{i+1},w_{i+2})$

where, $d(w_i,w_j)$ is the distance between words $w_i$ and $w_j$ in a paragraph *par* of $T_w$, $1 \le i \le n$; note that, an appropriate paragraph *par* contains more that $n$ words.

## 3.2 Embed Watermark into PDF - II

In this section we describe an algorithm of embedding a self-inverting permutation $\pi^*$ into a digital document $T$ by exploiting the two-dimensional representation of permutation $\pi^*$.

The main idea behind the embedding algorithm, which we call `Embed_SiP.to.PDF-II`, is similar of that of algorithm `Embed_SiP.to.Image-F`; see, (Chroni et al., 2013). The most important of this idea is the fact that it suggests a way in which the permutation $\pi^*$ can be represented by a 2D-matrix and, since pages of a PDF document $T$ are two dimensional objects, such a representation can be efficiently used for embedding $\pi^*$ into $T$ resulting thus the watermarked PDF document $T_w$; in a similar way as in our image watermarking approach, such a 2D-representation can be efficiently extracted for a watermarked PDF document $T_w$ and converted back to the self-inverting permutation $\pi^*$.

Let $A^*$ be the 2D-matrix of size $n^* \times n^*$ which represents the permutation $\pi^*$ of length $n^*$. The marking of the input PDF document $T$ is performed by selecting an appropriate page $P$ of $T$ and setting $n^*$ objects (e.g., characters, symbols, images) in a specific positions on page $P$, $1 \le i \le n^*$. In fact, we set an object $O_i$ in position with $(x'_i,y'_i)$ coordinates on page $P$ if $A^*(x_i,y_i) = "*"$, where $1 \le x_i,y_i \le n^*$ and $0 \le x'_i,y'_i \le size(P)$; note that, $(0,0)$ is the lower-left point (or, equivalently, the bottom-left corner) of the page $P$.

The algorithm takes as input a SiP $\pi^*$ and a PDF document $T$, and returns the watermarked PDF document $T_w$; it consists of the following steps.

**Algorithm** `Embed_SiP.to.PDF-II`

1. Compute the 2DM representation of the self-inverting permutation $\pi^*$, i.e., construct an array $A^*$ of size $n^* \times n^*$ s.t. the entry $A^*(i,\pi_i^*)$ contains the symbol "*", $1 \le i \le n^*$;
2. Select an appropriate page $P$ to embed the permutation $\pi^*$ and compute the size $size(P)$ of the page $P$, say, $N \times M$;
3. Segment the PDF page $P$ into $n^* \times n^*$ grid-cells $C_{ij}$ of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$, $1 \le i,j \le n^*$;

4. For each grid-cell $C_{ij}$ s.t. $A^*(i,j) = "*"$, mark the cell $C_{ij}$ by setting a symbol, with an appropriate color, in any position inside $C_{ij}$ of $P$, $1 \leq i,j \leq n^*$, resulting thus the marked document $T_w$;

5. Return the watermarked PDF document $T_w$.

**Extraction.** The algorithm which extracts the permutation $\pi^*$ from the watermarked PDF $T_w$ operates in a similar way as the corresponding extraction algorithm for images: it takes the input watermarked image $I_w$, locate the marked page $P$, computes its $N \times M$ size, and segments $P$ into $n^* \times n^*$ grid-cells $C_{ij}$ of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$; then, it computes the permutation $\pi^*$ by finding the coordinates $(x_i, y_i)$ of the $n^*$ symbols in the page $P$, $1 \leq i \leq n^*$; we call it `Extract_PDF.from.SiP-II`.

## 3.3 Embed an RPG into a PDF

We next describe a watermarking algorithm for embedding a self-inverting permutation $\pi^*$ into a PDF document $T$, by exploiting the RPG-representation of $\pi^*$ and the structure of a PDF document $T$.

Indeed, we have recently proposed an algorithm, namely `Encode_SiP.to.RPG` (Chroni and Nikolopoulos, 2012), for encoding SiPs $\pi^*$ as reducible permutation graphs $F[\pi^*]$. Moreover, in this paper we have described the document structure DS($T$) of a PDF document $T$ (see, Subsection 2.1); note that, the document structure of a PDF file always contains a node, namely `Document-catalog`, and a page tree PT($T$) rooted at node `Page-tree`, denoted by `root(pt)`; see, Figure 3.

In light of algorithm `Encode_SiP.to.RPG`, we next present an algorithm for embedding the watermark graph $F[\pi^*]$ into a PDF document $T$. The main idea behind the proposed embedding algorithm is a systematic addition of appropriate object-references in selected nodes of the page-tree PT($T$) of the document structure DS($T$), through the use of entries of type `/Kye(·)`, so that the graph $F[\pi^*]$ can be easily constructed from the page-tree PT($T_w$) of the resulting watermarked document $T_w$.

Let $F[\pi^*]$ be a reducible permutation graph produced by one of our two encoding algorithms and let $u_{n+1}, u_n, \ldots, u_1, u_0$ be the nodes of the graph $F[\pi^*]$; note that, $F[\pi^*]$ does not contain the back-edge $(u_0, u_{n+1})$. In order to simplify the extraction process, the graph $F[\pi^*]$ which is embedded into a PDF document $T$ contains one extra back-edge, i.e., the edge $(u_0, u_{n+1})$.

The algorithm for embedding a reducible permutation graph $F[\pi^*]$ into a PDF document $T$ is called `Encode_RPG.to.PDF` and is described below.

**Algorithm** `Encode_RPG.to.PDF`

1. Compute the document structure DS($T$) of the input PDF document $T$ and locate its page-tree PT($T$); let `node(dc)` be the document catalog node of structure DS($T$) and `root(pt)` be the root node of the page tree PT($T$);

2. Compute a path $O(T) = (v_{n+1}, v_n, \ldots, v_1, v_0)$ on $n+2$ nodes (i.e., objects) of the page-tree PT($T$) s.t. $v_{n+1} = root(pt)$, and set $s = v_{n+1}$ and $t = v_0$;

3. Assign an exact pairing (i.e., 1-1 correspondence) of the $n+2$ nodes of path $O(T)$ to the nodes $u_{n+1}, u_n, \ldots, u_1, u_0$ of the watermark graph $F[\pi^*]$;

4. For each back-edge $(u_i, u_j)$ of the graph $F[\pi^*]$ (i.e., $u_j > u_i$), add the forward-edge $(v_j, v_i)$ in page-tree PT($T$) by adding in object [v$_j$ 0 obj] an entry of type /Key(v$_i$ 0 R); add in object [v$_{n+1}$ 0 obj] an entry of type /Key(v$_0$ 0 R);

5. Return the modified PDF document $T$ which is the watermarked document $T_w$;

Let us briefly discuss the way we add forward-edge in the page-tree PT($T$); recall that, in Step 4 of the previous algorithm `Encode_RPG.to.PDF` we add the forward-edge $(v_j, v_i)$ in page-tree PT($T$) by adding in object [v$_j$ 0 obj] an entry of type /Key(v$_i$ 0 R). The entry /Key(v$_i$ 0 R) may be of various types; note that, /Key(·) is used as parameter in our algorithm's description.

In our implementation, for the forward-edge $(v_j, v_i)$ such that the object [v$_j$ 0 obj] is not the roodnode root(pt) of the page-tree PT($T$), we always chose the entry /Key(v$_i$ 0 R) which we add in object [v$_j$ 0 obj] to be of the same type of object [v$_i$ 0 obj]. In the case where $v_j = $ root(pt), we chose the entry /Key(v$_i$ 0 R) to be of type /Kids(·).

For example, in Figure 3 we have added forward-edges from object [29 0 obj] to object [3 0 obj], from object [29 0 obj] to object [24 0 obj], from object [3 0 obj] to object [13 0 obj], etc. Thus, in our implementation we have added in the root-node object [29 0 obj] the entries /Kids(3 0 R) and /Kids(24 0 R), in object [3 0 obj] the entry /XObject(13 0 R), while in object [13 0 obj] the entries /ColorSpace(6 0 R) and /R9(5 0 R).

**Remark 3.1.** Let $T$ be a PDF file and let PT($T$) be a page-tree of the document structure DS($T$). A node of the page-tree PT($T$) may contain several entries /Key(·) of various types. We mention that some types are required for entries in specific nodes of PT($T$); for example, the required entries in the root-node root(pt) of the page-tree PT($T$) are the /Type(·),

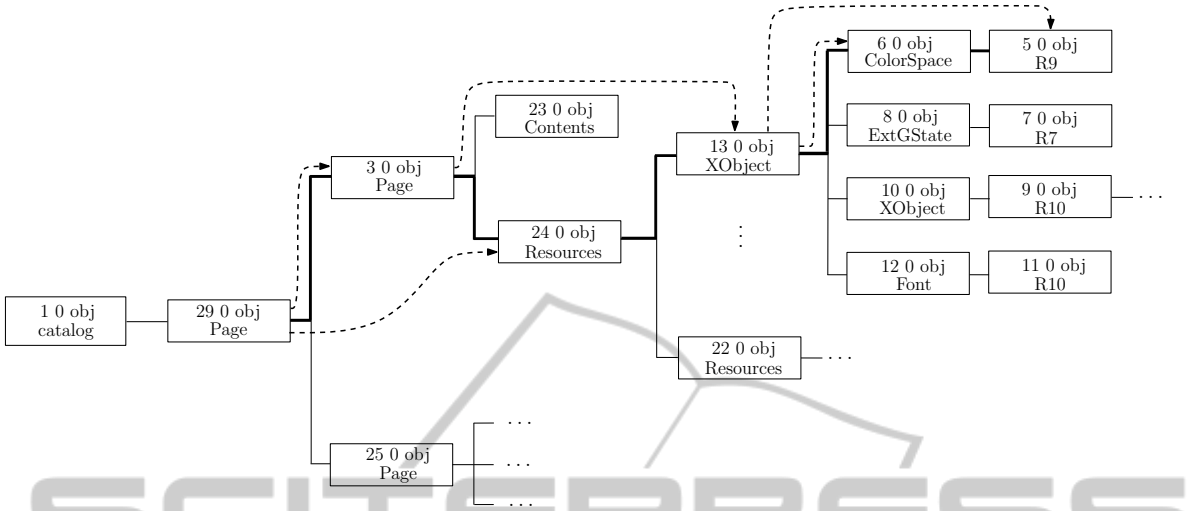




Figure 3: The watermarked $DS(T_w)$ which encodes the RPG of $\pi^* = (4,5,3,1,2)$.



/Parent(·), /Kids(·), and /Count(·).

**Extraction.** We next describe the corresponding extraction algorithm which extracts the graph $F[\pi^*]$ from the PDF document $T_w$ watermarked by the algorithm Encode_RPG.to.PDF; the algorithm, which we call Extract_RPG.from.PDF, works as follows:

- Take first as input the PDF document $T_w$, compute its document structure DS($T_w$), and locate its page tree PT($T_w$); then, find in object $root(pt)$, where $root(pt)$ is the root of the tree PT($T_w$), the entry /Kids($v_k$ 0 R) s.t. $v_k$ is not a child of $root(pt)$, and set $v_{n+1} = root(pt)$ and $v_0 = v_k$;
- Compute the path $O(T) = (v_{n+1}, v_n, \ldots, v_1, v_0)$ of PT($T_w$), from node $root(pt)$ to $v_0$, and assign an exact pairing (i.e., 1-1 correspondence) of the $n+2$ nodes of path $O(T)$ to the nodes $u_{n+1}, u_n, \ldots, u_1, u_0$ of a graph $F[\pi^*]$; initially, $E(F[\pi^*]) = \emptyset$;
- Add edges $(u_{i+1}, u_i)$ in $F[\pi^*]$ for $i = n, n-1, \ldots, 0$, and the edge $(u_i, u_j)$ iff $(v_i, v_j)$ is a forward edge in the page tree PT($T_w$);
- Delete the edge $(u_{n+1}, u_0)$ from the graph $F[\pi^*]$;
- Return the graph $F[\pi^*]$;

It is easy to see that, by construction the returned graph $F[\pi^*]$ is a reducible permutation graph produced by the algorithm Encode_SiP.to.RPG (Chroni and Nikolopoulos, 2012). Thus, $F[\pi^*]$ has the following property: the structure which results after deleting

(i) all the forward edges $(u_{i+1}, u_i)$ of $F[\pi^*]$, $0 \le i \le n$, and

(ii) the node $u_0$

is either the tree $T_d[\pi^*]$ or the tree $T_s[\pi^*]$ produced during the execution of the decoding algorithm Decode_RPG.to.SiP; see, (Chroni and Nikolopoulos, 2012). Thus, we can efficiently extract the self-inverting permutation $\pi^*$ embedded into a PDF document $T$ by algorithm Encode_RPG.to.PDF.

# 4 DISCUSSION

In this section we discuss the performance of the proposed watermarking algorithms after applying them on various PDF documents. We implemented the algorithms Encode_SiP.to.PDF-I, -II, and Encode_RPG.to.PDF and tested them on documents that have the same basic file structure (see, Subsection 2.1).

There are three main characteristics which we usually take into account in order to describe and evaluate a digital watermarking system: *fidelity*, *robustness*, and *capacity* (Cox et al., 2008).

Fidelity refers to the perceptual similarity between watermarked and original document. Concerning our watermarking systems, it seems to be of high fidelity as both algorithms Encode_SiP.to.PDF-II and Encode_RPG.to.PDF do not alter the PDF document display, whereas the algorithm Encode_SiP.to.PDF-I, although it modifies directly the text of the PDF

Table 1: Performance results of our graph-based algorithm `Encode_RPG.to.PDF` with respect to other similar methods.

| **Algorithm Performance** | *Our RPG Algorithm* | *Algorithm of (Simin et al., 2011)* | *Other similar algorithms* | *Algorithm based on PDF structure* |
|---|---|---|---|---|
| *Fidelity* | high | high | high | high |
| *Embedding Capacity* | based on file structure | infinite in theory | 67% of text size | infinite in theory |
| *Robustness* | best | better | worse | fragile |

document, ensures that its modification is not perceived by the human visual system.

We mainly focused on our graph-based watermarking algorithm `Encode_RPG.to.PDF` and evaluated its robustness under general manipulations of Adobe Acrobat Professional, such as addition of text, comments, stamp, signature, as well as optimization. The aforementioned operations although they added new objects in the PDF document, they did not altered the content of the original objects. The experimental results show that proposed algorithm is robust against editing and optimization attacks, since the entries inserted in PDF's objects does not affect their functionality, and the watermark graph $F[\pi^*]$ can be successfully extracted from the PDF document.

The embedding capacity essentially depends on the size of the watermark $w$ or, in our case, of the size of the embedding watermark RPG graph $F[\pi^*]$; note that, the size of the watermark graph is the number of vertices that it contains. In order to measure the embedding capacity, we calculate the ratio $|w|/|T|$, where $|w|$ is the size of the watermark RPG graph $F[\pi^*]$ and $|T|$ is the size of the original PDF document $T$ which is measured by counting the number of objects it contains; we use the number of objects since in our algorithm we assign an exact pairing of the nodes of $F[\pi^*]$ to the objects of $T$. We claim that our algorithms have high embedding capacity for large PDF documents since in such document we are able to encode a watermark graph less than or equal to document's size. Recall that, our recently proposed algorithms for encoding a number $w$ as reducible permutation graph $F[\pi^*]$ encode a relatively large graph into a large number of different integers (authors' papers). Additionally, the embedding of the watermark $w$ didn't increased the size of the PDF file.

We next select our graph-based encoding algorithm `Encode_RPG.to.PDF`, compare it with similar methods and evaluate its performance. More precisely, we compare the algorithm `Encode_RPG.to.PDF` with the algorithm recently proposed by the authors of (Simin et al., 2011), since both algorithms use similar watermarking technique, i.e., both algorithms modify content on specific objects of a PDF document.

In Table 1, we present the performance comparison of our algorithm `Encode_RPG.to.PDF` with respect to the algorithm proposed by authors of (Simin et al., 2011). In particular, in (Simin et al., 2011) authors presented their results on the robustness of their algorithm under specific attacks, such as addition of text content, postil, stamp, signature, background and delete text content of the PDF document. We extended these attacks by applying optimization to the watermarked PDF document $T_w$ produced by our algorithm and we show that the watermark $w$ can be successfully extracted by the document $T_w$.

For the sake of completeness, in Table 1 we also show performance results of the algorithm presented in (Simin et al., 2011) with respect to other similar methods (Gu and Yang, 2009; Liu et al., 2006; Kankanhalli and Hau, 2002; Wang and Liu, 2009), as well as with respect to a method based on the structure of PDF document (Zhong et al., 2007).

## 5 CONCLUDING REMARKS

In this paper we presented embedded algorithms, along with their corresponding extraction algorithms, for watermarking PDF documents using three different representations of a self-inverting permutation $\pi^*$, namely 1D-, 2D-, and RPG-representations.

In light of our graph-based embedding algorithm `Encode_RPG.to.PDF` it would be very interesting to investigate the possibility of altering other components of the document structure of a PDF file in order to embed the graph $F[\pi^*]$; we leave it as a direction for future work.

Moreover, an interesting open question is whether the embedding approaches and techniques used in this paper can help develop encoding algorithms having "better" properties with respect to text attacks.

## REFERENCES

Adobe (2006). Adobe systems incorporated, adobe portable document format version 1.7. In *Website http://www.adobe.com*.

Atallah, M., Raskin, V., Hempelmann, C., Karahan, M., Sion, R., Topkara, U., and Triezenberg, K. (2003).

Natural language watermarking and tamperproofing. In *LNCS 2578, Springer*, volume 5, pages 196–212.

Bindra, G. (2011). Invisible communication through portable document file (pdf) format. In *Proc. 7th Int'l Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'11)*, pages 173–176.

Brassil, J., Low, S., Maxemchuk, N., and Gorman, L. (1995). Hiding information in document images. In *Proc. of the 29th Annual Conference on Information Sciences and Systems*, pages 482–489.

Chroni, M., Fylakis, A., and Nikolopoulos, S. (2013). Watermarking images in the frequency domain by exploiting self-inverting permutations. In *Proc. 9th Int'l Conference on Web Information Systems and Technologies (WEBIST'13)*, pages 45–54.

Chroni, M., Fylakis, A., and Nikolopoulos, S. (2014). From image to audio watermarking using self-inverting permutations. In *Proc. 10th Int'l Conference on Web Information Systems and Technologies (WEBIST'14)*, pages 177–184.

Chroni, M. and Nikolopoulos, S. (2012). An efficient graph codec system for software watermarking. In *Proc. 36th Int'l Conference on Computers, Software, and Applications (COMPSAC'12)*, pages 595–600.

Collberg, C. and Nagra, J. (2010). *Surreptitious Software*. Addison-Wesley.

Cox, I., Miller, M., Bloom, J., Fridrich, J., and Kalker, T. (2008). *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd edition.

Golumbic, M. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, Inc., New York.

Gu, Y. and Yang, Y. (2009). A text digital watermarking algorithm for pdf document based on scrambling technique. In *Journal of Foshan University (Natural Science Edition)*, volume 2, pages 43–46.

Huang, D. and Yan, H. (2001). Interword distance changes represented by sine waves for watermarking text images. In *IEEE Trans. Circuits and Systems for Video Technology*, volume 11(12), pages 1237–1245.

Kankanhalli, M. and Hau, K. (2002). Watermarking of electronic text documents. *Electronic Commerce Research*, 2(1-2):169–187.

Lee, I. and Tsai, W. (2010). A new approach to covert communication via pdf files. In *Signal Processing*, volume 90(2), pages 557–565.

Liu, H., Li, L., Li, J., and Huang, J. (2012). Three novel algorithms for hiding data in pdf files based on incremental updates. In *Digital Forensics and Watermarking, Springer Berlin Heidelberg*, pages 167–180.

Liu, X., Zhang, Q., Tang, C., Zhao, J., and Liu, J. (2008). A steganographic algorithm for hiding data in pdf files based on equivalent transformation. In *Int'l Symposiums on Information Processing (ISIP'08)*, pages 417–421.

Liu, Y., Sun, X., and Luo, G. (2006). A novel information hiding algorithm based on structure of pdf document. In *Computer Engineering*, volume 32(17), pages 230–232.

Low, S. and Maxemchuk, N. (2000). Capacity of text marking channel. In *IEEE Signal Processing Letters*, volume 7(12), pages 345–347.

Low, S., Maxemchuk, N., and Lapone, A. (1998). Document identification for copyright protection using centroid detection. In *IEEE Transactions on Communications*, volume 46(3), pages 372–381.

Lu, P., Lu, Z., Zhou, Z., and Gu, J. (2008). An optimized natural language watermarking algorithm based on tmr. In *Proc. 9th International Conference for Young Computer Scientists*, pages 1459–1463.

Maxemchuk, N. and Low, S. (1997). Marking text documents. In *Proc. of the IEEE Int'l Conference on Image Processing*, pages 13–16.

Maxemchuk, N. and Low, S. (1998). Performance comparison of two text marking methods. In *IEEE Journal of Selected Areas in Communications*, volume 16(4), pages 561–572.

Meral, H., Sankur, B., Özsoy, A., Güngör, T., and Sevinç, E. (2009). Natural language watermarking via morphosyntactic alterations. In *Computer Speech and Language*, volume 23(1), pages 107–125.

Simin, H., Xingming, S., and .Zhangjie, F. (2011). A novel information hiding algorithm based on page object of pdf document. In *10th IEEE Int'l Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES'11)*, pages 266–270.

Sun, X. and Asiimwe, A. (2005). Noun-verb based technique of text watermarking using recursive decent semantic net parsers. In *LNCS 3612*, volume Part III, pages 968–971.

Topkara, M., Topraka, U., and Atallah, M. (2007). Information hiding through errors: a confusing approach. In *Proc. of SPIE, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 1–12.

Vybornova, O. and Macq, B. (2007). A method of text watermarking using presuppositions. In *Proc. of SPIE, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 1–10.

Wang, Q. and Liu, X. (2009). A new watermarking algorithm of pdf document based on correct coding. In *Computing Technology and Automation*, volume 28, pages 137–141.

Zhong, S., Cheng, X., and Chen, T. (2007). Data hiding in a kind of pdf texts for secret communication. In *International Journal of Network Security*, volume 4(1), pages 17–26.

Zhou, X., Zhao, W., Wang, Z., and Pan, L. (2009). Security theory and attack anlysis for text watermarking. In *Int'l Conference on E-Business and Information System Security (EBISS'09)*, pages 1–6.