13

Authentication of Script Format Documents Using Watermarking Techniques

Mario Gonzalez-Lee, Mariko Nakano-Miyatake and Hector Perez-Meana
National Politechnics Institute,
Mexico

1. Introduction

The electronic document authentication is a subject of active research because, with the release of very efficient program for documents, images and video processing, the manipulation of such digital content becomes easier. Then, the development of efficient methods allowing the protection of sensitive digital material, avoiding unauthorized manipulations, without degradation of the original materials is a very important task that has found application in the solution of many practical problems in the financial, banking, insurances, legal, and Government fields, among others.

Thus digital content authentication and protection algorithms, for using in several practical applications, have been proposed during the last decade some of them use fragile or semi-fragile watermarking algorithm, fingerprints for document leakage investigations and robust watermark for copyright protection.

Most of these schemes consider the document to be protected as an image, without taking in account that in a more natural scenario, a digital document is in fact stored using an electronic format such as PDF, postscript and word files, etc., especially with the increasing use of digital signatures.

This chapter presents an authentication scheme for script format digital documents using watermarking techniques that are capable to achieve an accurate verification that makes possible to detect malicious and unauthorized documents manipulations. The remaining of this chapter is organized as follows, first, a review of similar works for document watermarking, followed by detailed background in sections 2 and 3, then, the document watermarking approach is presented in section 4, the results are presented in section 5 and finally some conclusions where the main achievements of this watermarking approach will be discussed, and in the end, the references used in this chapter are listed.

1.1 Previous works

Several schemes have been developed to authenticate digital documents which embed invisible watermark into digital documents, most of them considering the digital documents as binary images. Yang and Kot proposed a document authentication scheme, in which an authentication code is embedded by changing the spaces size between consecutive words and characters (Yang & Kot, 2004). The main drawback of this scheme is its high computational complexity and vulnerability against noise.

Huang proposed an authentication method for binary images including text documents (Huang et al., 2004), in which firstly the binary image is segmented in blocks and then some pixels in each block are rearranged in order to enforce a given relationship between the total number of black and white pixels in it. During the authentication process, this relationship is verified for each block in order to authenticate the block. If this relationship is satisfied the block is considered as authentic, otherwise the block is considered as tampered. The principal disadvantage of this method is that a degradation introduced in the encoded binary image is noticeable.

Wu and Liu proposed binary image block-wise authentication scheme, in which flippable pixels in each block are manipulated in order to embed a watermark bit in the block (Wu & Liu, 2004). Here the embedded watermark is imperceptible, because fliping flippable pixels do not cause any distortion of the binary image. However, in general, the watermark embedding payload is very low compared with the number of flippable pixels into the image.

To improve the embedding payload, Gou and Wu introduced the concept of "super-pixels" and wet paper coding into the Wu and Liu's scheme (Gou & Wu,, 2007). The "Super-pixels" form a set of individually non-flippable pixels, which can be removed or added together without causing visual distortion. Also Wu and Liu reported that their authentication scheme is robust to printing and scanning operations. However during the scanning process, a rotation, even with angles smaller than one degree may results in an embedded watermark signal lost.

Document authentication schemes for formats such as Portable Document Format (PDF) or PostScript had received few attention among researchers although many official documents are stored using this type of formats. In (Zhu et al., 2007), a document authentication method using render sequence encoding is proposed, in which the encoding process is based on modulate the display sequences using a Document Description Language (DDL), such as PostScript, PDF, Printer Control Language, etc. In the render sequence, predefined characters are permuted by a user's secret key; and then during the authentication process, the document is considered as authentic if the permutation corresponds to the secret key used in embedded stage. This scheme determines correctly if a document is authentic or not, however there are two inconveniences that may limit its practical use. Firstly the size of the encoded document file is considerably increased compared with the original file size, and the second one is the fact that the structure of the encoded render sequence is unnatural, and as a consequence, it can be easily detected by an unauthorized person, doing it possible the used of reverse engineering to tamper the document.

To solve these problems, Gonzalez-Lee proposed a watermarking-based document authentication scheme, in which character metrics are used to embed a watermark sequence (Gonzalez-Lee et al., 2009). The advantage of proposed scheme is that the watermarked file size is not changed compared with original file size and also the watermarked file conserves its original appearance, enhances in this form its security because the watermark presence is not evident.

Finally, we would like to discuss the previous work in document security done by the main promoters of electronic document schemes, the PDF uses a scheme with several variants of permissions that allow user to do different tasks, for example, permissions for printing or even copy portions of the document (done by CTL+C, CTL+V shortcuts), a password protected document will ask for the password when one wants to perform one of the described task. Unfortunately, this scheme is tied to Acrobat Reader and the security can be override as easy as to use another PDF viewer, for example Gnome Document Viewer available in most Linux distributions, that viewer won't ask for any password for printing or to copy portions of the document. Another possibility is that the security relies on hiding the document contents; in this case, the viewer doesn't allow anyone to see the contents of the document unless the right password is given. Again this scheme can be easily broken with the use of free tools, for example PDFcrack (Noren, 2008); by using this tools, anyone can break the password within a couple of days with a consumer computer. Once Broken, the attacker will be able to view the document contents, and save an unprotected copy of the document which can be modified, and even saved with the same password so the legitimate document is replaced by the tampered document and the user is unaware of this. More on the security model of PDF can be read in (Adobe, 2006).

2. Document description languages

Computer languages such as C language are general propose, they can be used for developing a broad spectrum of applications; others like Fortan and Matlab are designed for numerical calculations so their respective instruction sets facilitate greatly calculations in engineering field. One can easily think on many useful instructions or functions that facilitate coding complex programs, for example, the function sin(x) is very useful in engineering computing programs but it is of little use in describing an electronic document.

In order to achieve an efficient description of the basic elements that allow the creation of a practical document, we need a proper computer language that meets the challenge of describing properly an electronic document, this computer language is called a Document Description Language or DDL for short, and thus a DDL is a computer language which instruction set is designed to contain commands for common tasks needed to draw a document.

A DDL is designed to facilitate the description of a document, in other words, their instruction set are very handy for common task such as to indicate where to draw a given set of characters (e.g. a row or a paragraph), which font size, and other properties according to the desired document layout. It is hard to imagine trying to describe a web page using C or Matlab instruction set, so, the scope and propose of DLL's is evident.

We can mention many implementations of practical DDL's, for example, for describing Web pages we can use the Hiper Text Markup Language (HTML), and for electronic documentation, we can choose among PostSript, Portable Document Format (PDF), Open Document Format (ODF) used by the OppenOffice.org and LibreOffice projects.

As discussed above, there are many DDL's, most of them are different radically, this difficult the development of a universal approach that can be used for every DDL. In most cases, a given watermarking approach can be adapted for several DDL's, but in other cases, we must to design a completely different paradigm.

Finally, we wish to point out that a DDL is like any other computer language, it provides an instruction set but those instructions must be properly structured, in next section, a discussion on this subject is carried out.

3. Document Description Scripts

In previous section, we discussed the scope of DDLs, in this section we'll introduce a new concept: the Document Description Script or DDS for short. Let's state this: a DDL is an instruction set, these instructions are unable to perform anything unless they are properly structured and proper parameters are given.

Most of the time, for any computer language, instructions are written in a file known as a sourcecode and then compiled in order to generate a computer program (sometimes, the sourcecode is not compiled but interpreted instead), sometimes these source code is also called a script; a DDS shares this concept, the DDS contain a set of instructions properly structured, they are written in a script what we call a document and this document is interpreted by a document viewer, so this viewer interprets how to draw a document in a computer screen or how to print it.

For example, in Fig. 1; a part of the DDS as used for the ODF, PostScript and PDF is shown. Of course, it lacks many essential elements, but the aim is to show the nature of those approaches.

In Fig. 1(a), we can see that the text "This is a text document showing a DDL with a xml approach" is to be drawn in the page, we can identify the special tags body to indicate that the body of the document is to begin, and then the special tag text indicates that the enclosed stream is the text of the document and furthermore, the special tag `text:p text:style-name="Standard"` indicates that the enclosed paragraph and this text has the style Standard (12 pt Times Roman font, normal weigth), usually a document has several paragraphs and several styles including user defined styles, for example bold letters with font size 14 pt and Arial font, and the way to define which parts of the whole text has to be in this style is by means of these command sequence.

In Fig. 1(b) the command sequence to draw the text "this is a text document showing a DDL with a PostScript approach" is illustrated, it is clear how different DDL's approach the same task in different ways, not necessarily better yet different. In this slice of code, one can identify a command used to position the text in a given point in the page ("`100 50 moveto`" positions the beginning of the text at the point (100,50)), and then, the character stream is given, note the special delimiters "`(`" and "`)`" which enclose the characters to be drawn and finally the instruction "`show`" that draws the given stream in the page. And in Fig. 1(c) it is shown the corresponding script slice to approach the same task, one can see that it is almost the same as done using the postscript approach, not surprisingly since it is know that PDF is an evolution from Postscript.

We would like to emphasize that not all DDL's use the same instruction set for document descriptions, furthermore, in most cases DLL's differ greatly, thus in the remaining of this chapter, we well focus in DDL in which character metrics are available so an automated system can locate an process them, and illustrative examples will be carried out using the postscript DDL because is better documented and easier to understand; since postscript is

```
<office:body>
      <office:text>
         <text:p text:style-name="Standard">
            This is a text document showing a DDL with a xml
            approach
         </text:p>
      </office:text>
  </office:body>
 </office:document-content>
```

(a)

```
100 50 moveto
(this  is  a  text  document  showing  a  DDL  with  a  PostScript
approach)
show
```

(b)

```
100 50 Td
(This a text document showing a DDL with a PDF approach) Tj
```

(c)

Fig. 1. Example of a DDS, one can notice how a Language is used to describe the structure of an electronic document. The same text was written with a) the ODF; b) the Postscript Language and c) the PDF.

considered the basis of PDF, it is feasible that if you understand the postscript it will be in fact easier to understand the PDF internals, conversely, it will be more difficult to proceed the other way.

A typical approach is depicted in Fig. 2. In this figure we can see that the most important parts of the script file are the header and the body. The former is called Encapsulated PostScript or EPS, it contains information about the version of the standard used in the document; in addition, it contains other useful data such as the number of pages, the bounding box, etc. The latter, that is to say, the body contains the whole contents of the document organized in pages (each one can be recognized easily by the special command

```
 %!PS-Adobe-2.0
 %%Pages:  2
 %%Creator: Txt2Ps
 %%Title: A Simple Document.
 %%PageOrder: Ascend
 %%BoundingBox: 0 0 615 792
 %%CreationDate: Fri Jul  9 17:31:33 2010
 %%BeginSetup
 %%PaperSize: Letter
 %%EndSetup

 /Times-Roman      findfont
 12         scalefont    setfont

 %%Page: 1 1
 %%               %%                 Page                   Contents
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 showpage


 .
 .
 .
 %%Page: N N
 %%               %%                 Page                   Contents
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
showpage
```

Fig. 2. Example of a basic DDS of PostScript.

showpage which is used to mark the end of a page and tell the document interpreter that the page must be drawn). In this example, the actual contents of the page is not shown, a comment is shown instead. The first lines illustrate a header, then, the marker %%Page: x x is used to begin the page x, and the command showpage marks the end of the page.

In the examples ahead, all this structure will be omitted and just the contents will be illustrated in order to keep the examples small and to focus in the parts of the script that are processed.

3.1 Character metrics

In last section, the basic concepts of DDS's and their role was described, in this section we will go deeper in the internals of the document description scripts.

Let's first introduce the character metrics concept.

A character metric is the distance between consecutive characters, another way to understand the character metrics is as the distance that "the cursor" must be advanced to place next character. A character has two metrics, called m_x and m_y, that are the distance in the x-axis and the y-axis where the next character must be placed (see Fig. 3). Since some languages have different writing styles, the metrics should agree with this, and thus we can have vertical documents, like Japanese in which $m_x=0$ and $m_y \neq 0$, and horizontal documents like in English in which $m_x \neq 0$ and $m_y=0$, and the seldom used, diagonal documents, which are mostly used in graphic design field, even when seems that this class apply only for line shapes, here consider that any text in which $m_x \neq 0$ and $m_y \neq 0$ holds is a diagonal document. Fig. 4 shows examples of each type of documents.

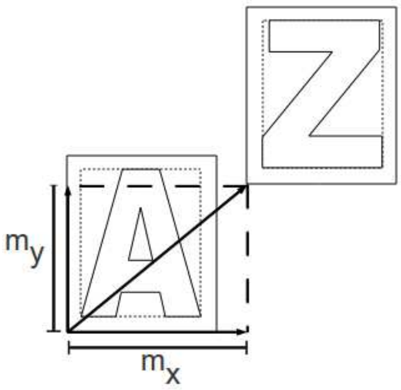


Fig. 3. The character metrics.

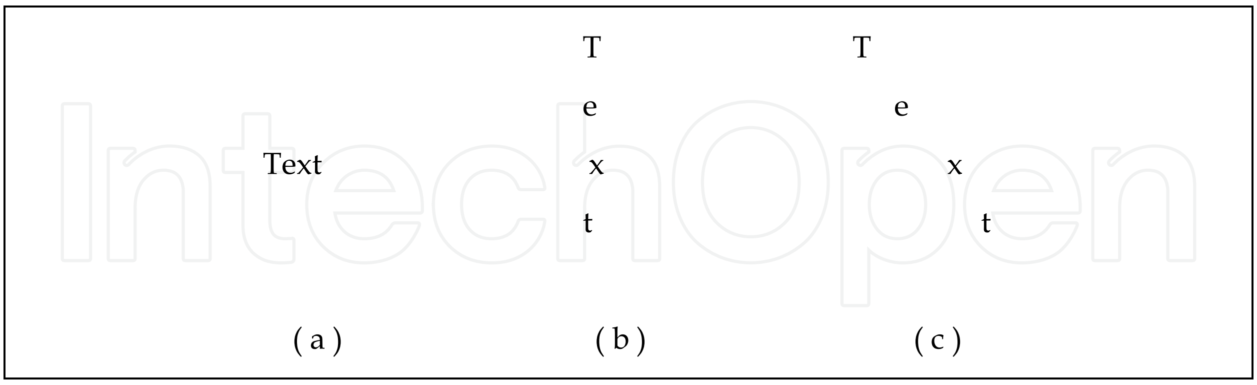


Fig. 4. Types of documents. a) Horizontal document, b) Vertical document and c) diagonal document.

More information on character metrics can be read in (Turner, 2000).

As mentioned above, the actual contents of a page is enclosed in special tags; for text documents, the text is organized in rows. In Fig. 5 it is shown an example of a simple row definition. Firstly, the position for the row within the page is set at (52,742) by the command

```
50     742  moveto  (C Language History)

[ 8.100947  3.930948     7.540798     5.871108     6.430798
  6.430798  6.430798     5.871108     6.430798     5.871108
  3.930948  8.650798     4.210798     5.320798     4.210798
  6.430798  4.761107     6.430798     ] xshow
```

Fig. 5. Example of an actual row definition.

`moveto` and then the text "C Language History" is the contents of the row and the following vector contains the metrics for each character in the row, generally, the characters does not full fill the page width, so a small constant should be added to each metric in order to fit the page width, that is to say, to left and right justify the text, next, the command `xshow` indicates that this row must be drawn with given metrics, however nothing is actually drawn until a `showpage` command is encountered.

As depicted in Fig. 5, we can find a rich source of data that can be modified in order to either hide information to implement a steganographic system or to embed digital watermarks. A natural question is that if such modifications could have side effects such as visual distortion, but consider that each unit of metrics is in fact 1/72 inches, that it to say, a metric of 1.0 = 1/72 inches, so the changes are mostly imperceptible. More about DDS languages can be read on (Adobe, 1999),(Adobe,2006) and (Reid, 1990).

In next section, we will discuss a watermarking system that uses character metrics in order to embed digital watermarks.

4. Document watermarking approach

Watermarking for authentication schemes differ from copyright enforcement schemes, in the latter, the watermark integrity is crucial, since no matter what attack is carried out on the protected material, the watermark should be still detected, of course damaged yet detectable. In authentication applications, the watermark should be fragile, any modifications should damage the watermark seriously so the system would be unable to detect the watermark, and in other words, any modification on the protected media would render the watermark undetectable by the system. These kinds of applications are intended to prevent frauds or moral damages.

4.1 Attack scenario to watermark

As stated in last section, in watermarking for authentication applications, a natural attack scenario is as follows: an attacker trying to modify a protected digital material in order to change the meaning of this material. An example of this is an electronic document that is modified to change the message contained in this document to commit fraud. Such attack is feasible due to the existence of free tools such as PDFedit, (Hocko, 2009).

In order to carry out a successful attack, the attacker must achieve the following goals:

- Change the meaning of the original message in the protected document so it matches some desired meaning, usually malicious, in a way that is not possible to figure the modification out.

- Preserve as much as possible of the watermark, so an automatic verification system still be able to detect it an thus to validate the document as a legitimate one.

From this situation is evident the need of a document authentication system based on fragile watermarking, so even if the modification of the document is small, the watermark shall be no detectable.

4.2 Watermarking using character metrics

In section 3.1, the metrics of characters were described, in this section; we discuss a model for watermarking using characters metrics. This model is depicted in Fig. 6. In this model, some edition software takes the raw text so it can build a well formed DDS from the input data; the edition software uses the instructions in a DDL data base so the resulting DDS follows the file standard. Then, the watermarking algorithm embeds a watermark generated using some secret key in the resulting script, the final product is a watermarked DDS.

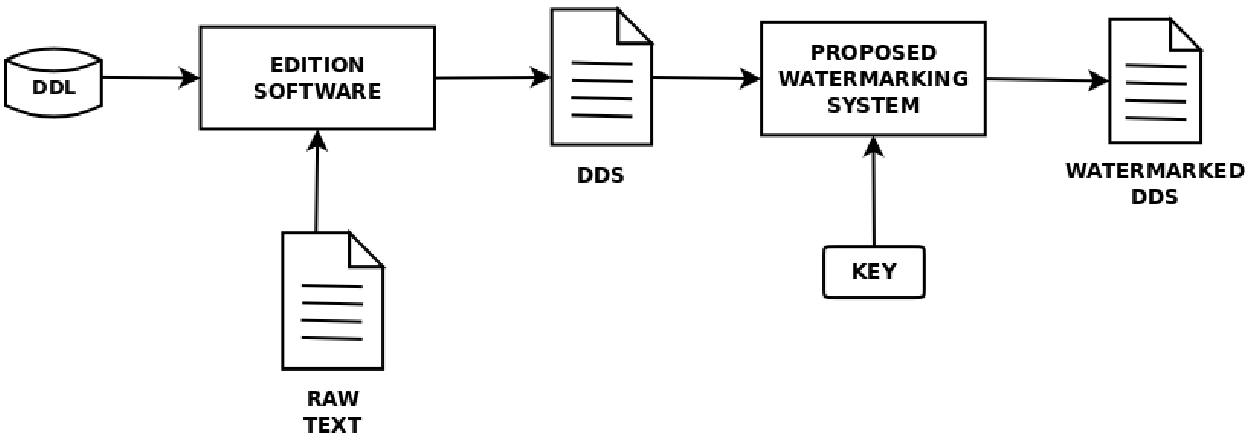


Fig. 6. Watermarking model for electronic documents in a DDS approach.

There are many software capable of producing high quality documents, we will assume that such software is provided by third party, yet the resulting documents follow some standard. So, the watermarking system has to be designed to interpret the input DDS in order to process it under this assumption.

Next, we will introduce a watermarking scheme which relies on the modification of character metrics for watermark embedding; a question might be arisen regarding the distortion caused by the metrics modification, in this subject, we must consider that a unit of metrics equals 1/72 inches, so small modifications should be negligible.

The watermark $W=[w_i], i=1,2,...,N$ is a binary (-1 or 1) pseudo random sequence with zero mean an variance 1. Without losing generality, we will assume that we are dealing with horizontal documents; the extension to vertical and diagonal documents is easily carried out.

The whole document is interpreted and then we can form two vectors named $C=[c_i], i=1,2,...,N$ and $M=[m_i], i=1,2,...,N$, the former is the vector of the characters of the document, and the latter is a vector of their metrics. The character metrics are firstly modified as follows:

$$m'_i = m_i + \frac{ASCII(c_i)}{1000} \tag{1}$$

Where c_i is the i-th character in the document and $ASCII(c_i)$ is the ASCII value of character c_i. For example, if $c_i = A$, $ASCII(c_i) = 097$.

The watermark is embedded using a multiplicative rule as follows:

$$M_i = m'_i(1 + gw_i) \tag{2}$$

where M_i is the watermarked metric corresponding to the i-th character, this is another vector named $M' = [M_i], i = 1,2,...,N$ and w_i is the i-th watermark bit, g is the gain factor; in experimental results, we found that a good value for g is one that just crosses the threshold as depicted in Fig. 7, that keeps a balance between the watermark imperceptibility and tamper detection capability.

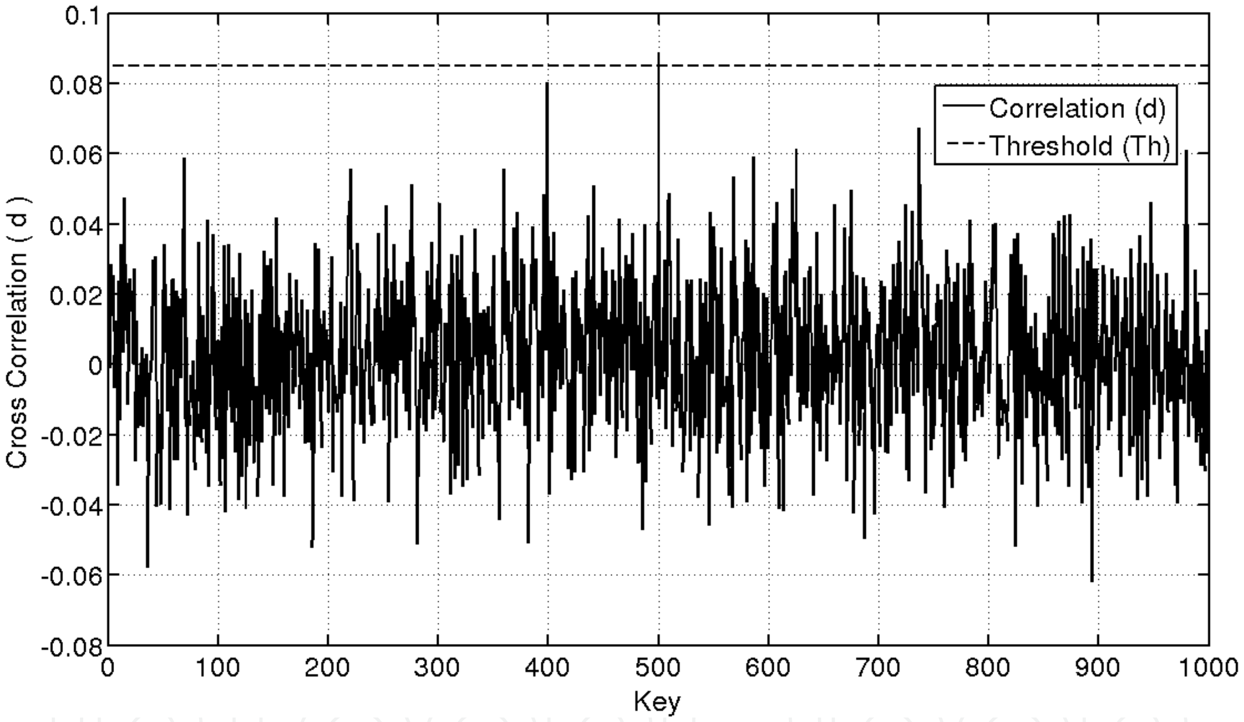


Fig. 7. Watermarking detection, the watermark was generated using key number 500. The use of a gain value that barely crosses the threshold is advised.

Then, the watermarked metrics vector M' replaces the original metrics vector M. Finally, the vectors C and M' are used to re-assemble the document, for better understanding see Fig. 8.

On the other hand, for detecting the watermark, we need to retrieve the watermarked metrics vector from the file, so we have the vector $\tilde{M} = [\tilde{m}_i], i = 1,2,...,N$. Where $\tilde{m}_i$ is the extracted metric. Then the presence of the watermark can be assessed by computing the Cross Correlation (d) between the retrieved watermark $\tilde{M}$ and the watermark W as follows:

$$d = \frac{1}{N}\sum_{i+1}^{N} \tilde{m}_i w_i \tag{3}$$

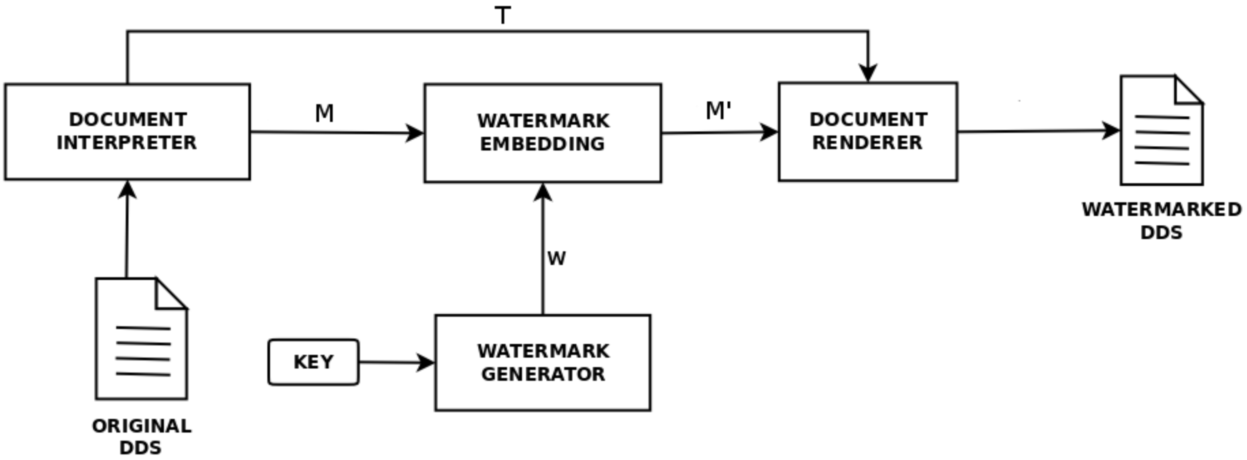


Fig. 8. Detailed block diagram of the Watermarking algorithm.

The value of d must be compared with the threshold Th and if $d \geq Th$ holds, then the watermark is present and thus the document is considered as authentic, otherwise, as tampered. The threshold is computed as:

$$Th = 2.8\sqrt{2\frac{\sigma^2}{N}} \tag{4}$$

Where σ^2 is the variance of the vector of metrics $\tilde{M}$.

Equations (4) is a modification from the one proposed by Piva as the optimal threshold for correlation-based detectors, and since proposed system holds the same asumptions as presented in (Piva, 1998), equation (4) holds, however, in order to achieve accurate results for the intended application, the value of '3.3' from the original equation was changed for '2.8' because in this way a lower value of embedding gain can be set, this helps to make the watermark very fragile, so a lower value of Th is desirable because it helps to reduce false positive error rate (a false positive is when the system decides that a tampered document is authentic; false negative occurs when the system decides that an authentic document is tampered). A block diagram for the watermark detection process is shown in Fig. 9.

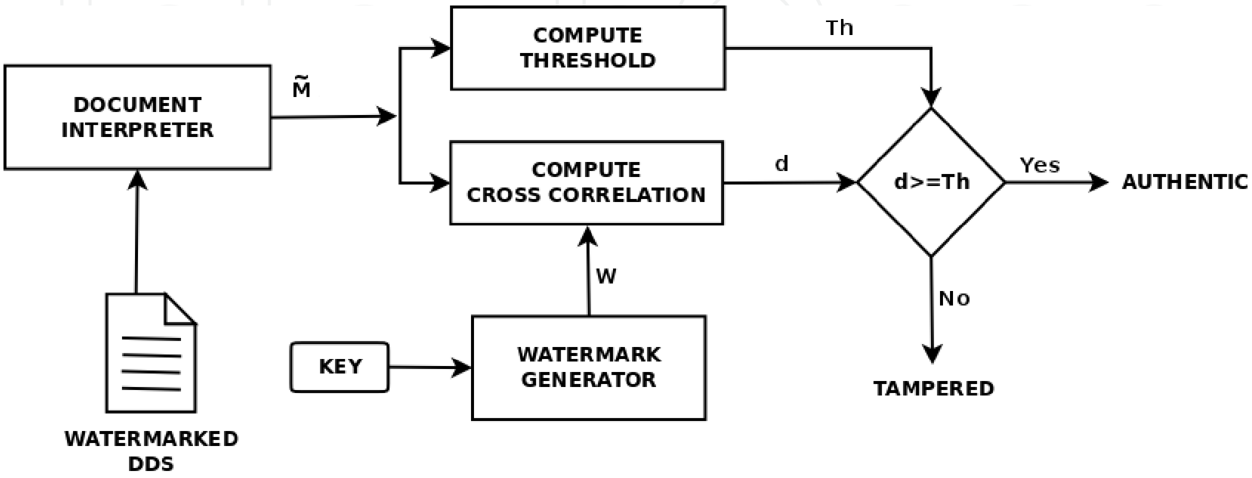


Fig. 9. Watermarking detection.

Experimental results and discussions will be carried out in next section.

5. Results and discussions

Although there is not a standard benchmark for document watermarking systems, we will present results for common concerns in watermarking electronic documents such as watermark imperceptibility, tamper detection capability and practical considerations.

5.1 Watermark imperceptibility

Since electronic documents are not images we cannot assess the distortion caused by the watermarking process using common distortion measures such as the Peak Signal to Noise Ratio (PSNR) or the Mean Absolute Error (MAE), because of this, the distortion assessment was carried out using a Mean Opinion Score (MOS) evaluation.

The MOS evaluation was set this way: twenty pair of different documents (each pair consisted of the original and the watermarked document) were shown to 100 observers whose gender and ages are distributed as described in Tab. 1.

Age (years)	Female	Male
20-30	33	32
30-40	4	10
40-50	2	7
50+	3	9

Table 1. Age and gender distribution of MOS observers.

The observers were asked to assess the difference between the original and watermarked documents, and to assign a score according to Tab. 2. And the average result of the MOS was a 4.6 which confirms the watermark imperceptibility. The observers argued the following reason to score other than 5:

- The ink of the letters is uneven.
- The text is misaligned to the paper sheet.
- The paper whiteness is slight different.

Since the observers were aware that they must find differences, they pointed out what they though could be the difference, and even when these differences in fact existed, they were caused directly either by the printer or by the composition of the paper.

Score	Meaning
5	There is not any perceptible difference
4	There is a slight difference that can be ignored
3	There is a slight difference which cannot be ignored
2	There is a noticeable difference
1	It is evident the difference between the two documents

Table 2. MOS evaluation criteria.

To further support the results of the MOS, we present a measure of the distortion of the metrics compared with the original metrics (see Fig. 10). It can be seen that when a character with high ASCII value appears in the document, the distortion becomes larger although it is too small to cause significant distortion.

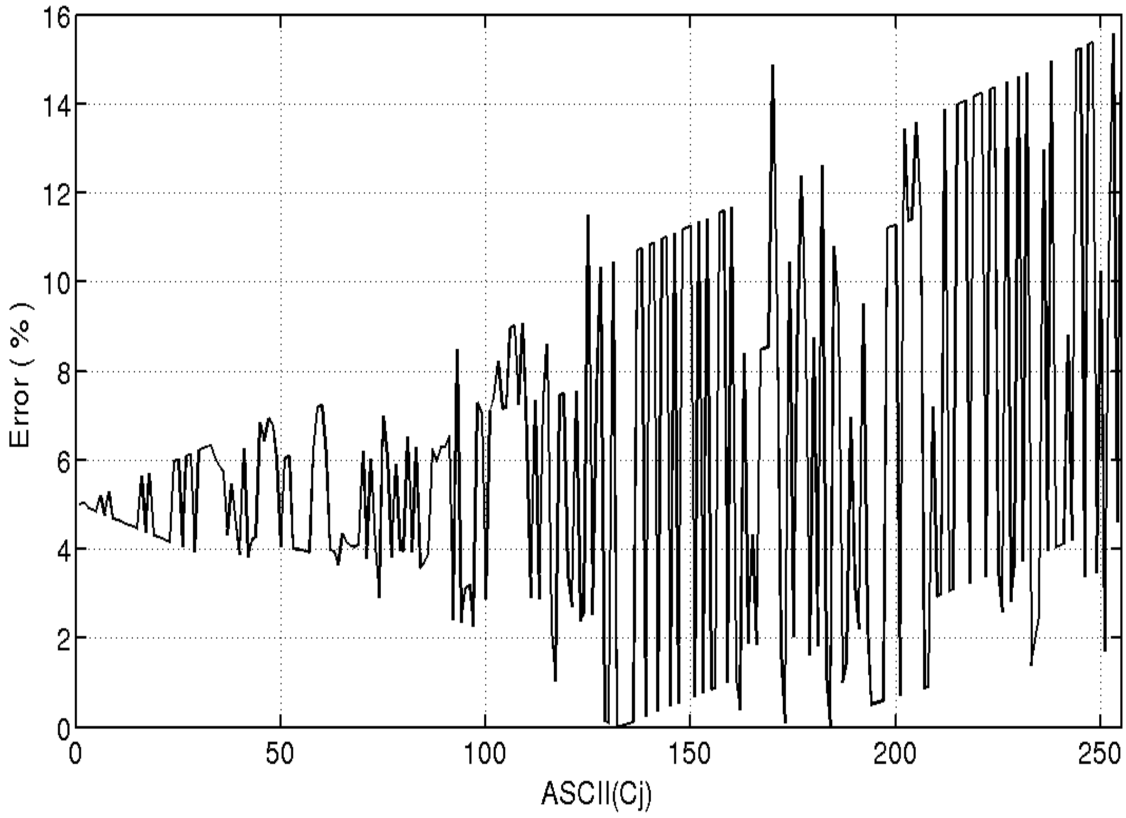
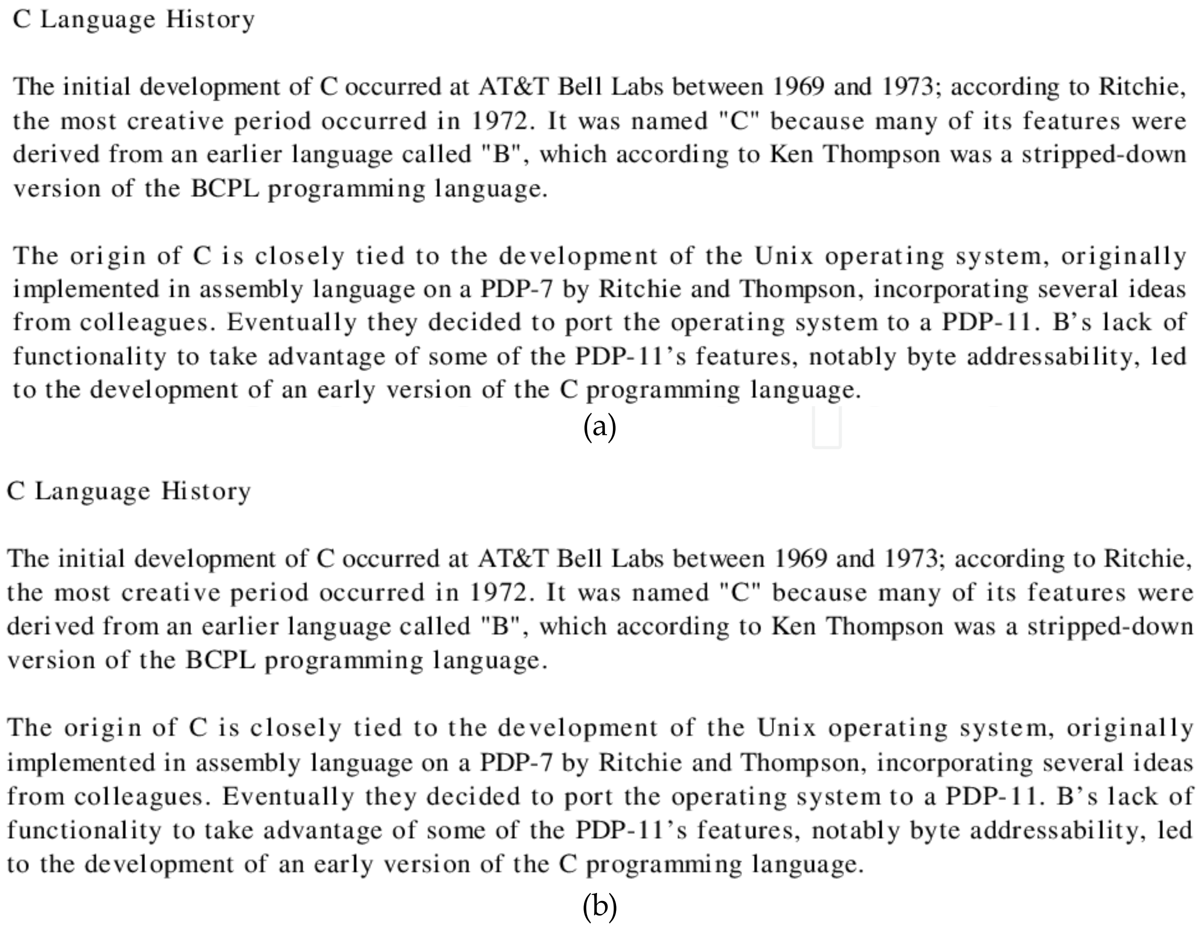


Fig. 10. Error percentage for each character in the ASCII code for some random watermark; the maximum distortion is about 16 %.

In Fig. 11 a pieces of a document and its watermarked version is shown.

5.2 Tamper detection capability

Let's consider two possibilities to tamper a document, in the first one, the attacker changes characters according to convenience without changing the metrics because he expects that this won't damage the watermark, if the attack is carried out this way, we can expect a document as shown in Fig. 12. It is quite evident that some modifications were made, so any human can easily detect the tamper even if the original document is not available for comparison. Now, consider another variant, the attacker have knowledge of the file standard so he has the needed skills to modify the document to preserve its natural look, to achieve this goal, the attacker must to re-compute the metrics related to the tampered characters, as expected, the more tampered characters, the more the damage to the watermark, in Fig. 13 we show a typical behaviour of this phenomena, we can see that once the correlation value d is below the threshold value, it never surpasses it again, furthermore,



C Language History

The initial development of C occurred at AT&T Bell Labs between 1969 and 1973; according to Ritchie, the most creative period occurred in 1972. It was named "C" because many of its features were derived from an earlier language called "B", which according to Ken Thompson was a stripped-down version of the BCPL programming language.

The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7 by Ritchie and Thompson, incorporating several ideas from colleagues. Eventually they decided to port the operating system to a PDP-11. B's lack of functionality to take advantage of some of the PDP-11's features, notably byte addressability, led to the development of an early version of the C programming language.

(a)

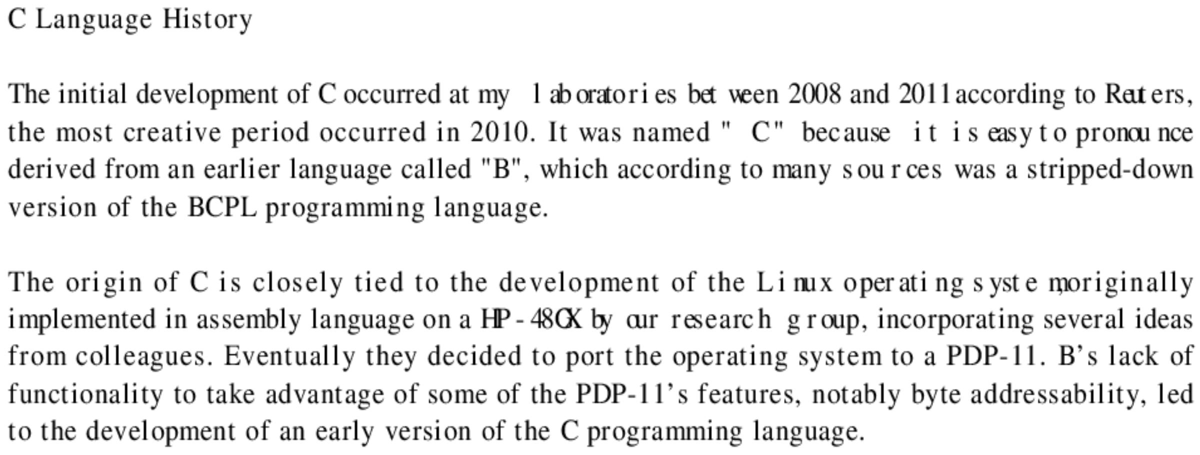
C Language History

The initial development of C occurred at AT&T Bell Labs between 1969 and 1973; according to Ritchie, the most creative period occurred in 1972. It was named "C" because many of its features were derived from an earlier language called "B", which according to Ken Thompson was a stripped-down version of the BCPL programming language.

The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7 by Ritchie and Thompson, incorporating several ideas from colleagues. Eventually they decided to port the operating system to a PDP-11. B's lack of functionality to take advantage of some of the PDP-11's features, notably byte addressability, led to the development of an early version of the C programming language.

(b)

Fig. 11. Sample documents. a) Original document. b) Watermarked document.



C Language History

The initial development of C occurred at my laboratories between 2008 and 2011according to Reuters, the most creative period occurred in 2010. It was named " C" because it is easy to pronounce derived from an earlier language called "B", which according to many sources was a stripped-down version of the BCPL programming language.

The origin of C is closely tied to the development of the Linux operating system,originally implemented in assembly language on a HP-48GX by our research group, incorporating several ideas from colleagues. Eventually they decided to port the operating system to a PDP-11. B's lack of functionality to take advantage of some of the PDP-11's features, notably byte addressability, led to the development of an early version of the C programming language.

Fig. 12. Example of a malicious modification; only the characters were changed whilst the metrics remain unchanged. The modifications can be easily spotted.

even when the threshold seems to possess a parabolic like shape and in some point it decreases, the correlation value is below the threshold. A close up of Fig. 13 is shown in Fig. 14, in this figure we can see the point in which the correlation goes below the threshold, in this case, when about 0.6% of characters are tampered

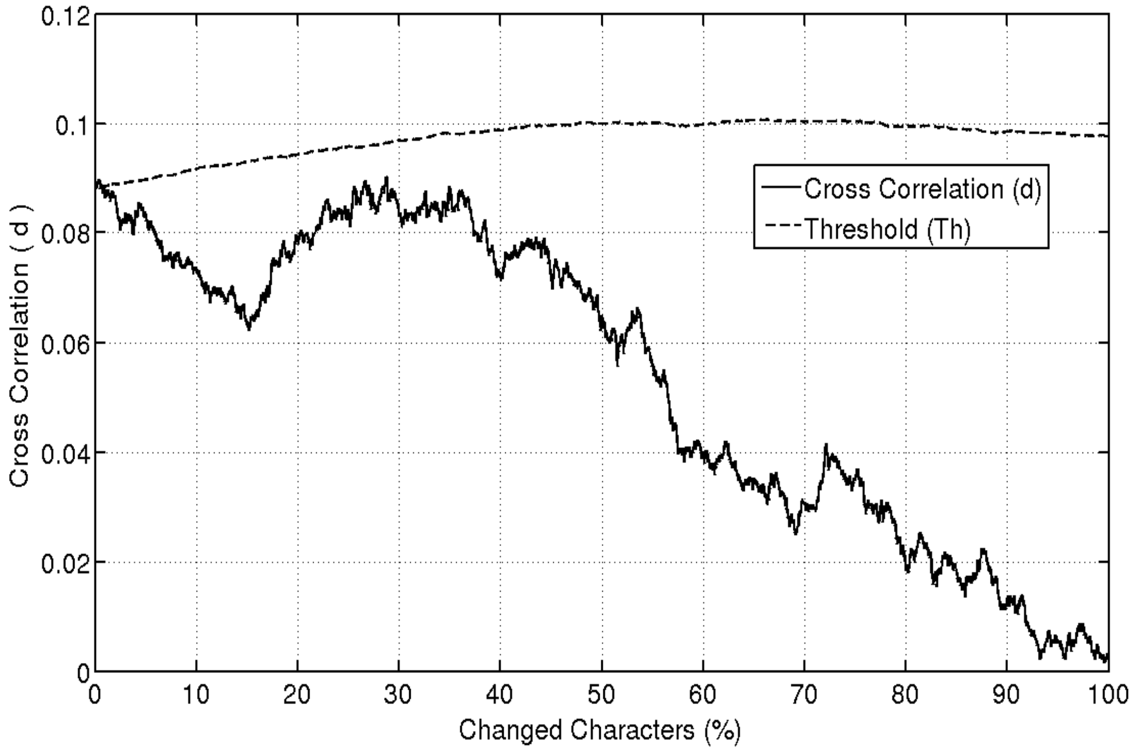


Fig. 13. System response as the percentage of tampered characters varies from 0% to 100%.

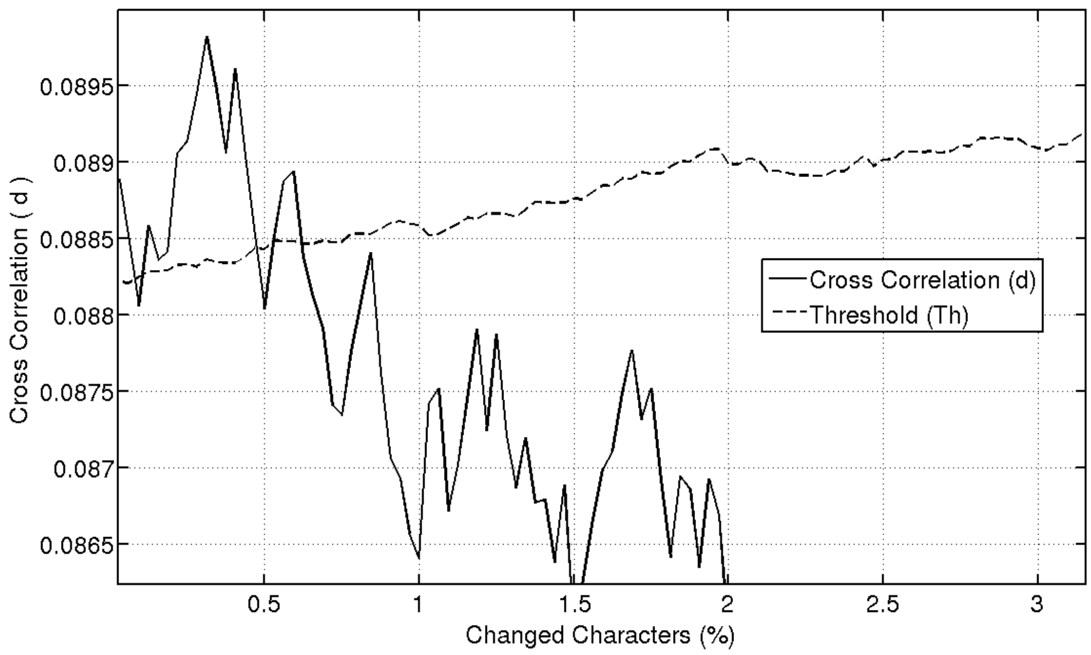


Fig. 14. System response as the percentage of tampered characters varies from 0% to 3.125%.

In Tab. 3 we present results for 10 different documents, showing the percentage of tampered characters that had to be tampered so the system considers them as tampered. High values in the table are explained as follows, as seen in Fig. 13 and Fig. 14, the correlation value does not decrease monotonically because the metrics are highly correlated to the watermark, this causes oscillations specially in low percentages of tampering, so the reported percentages are those in which the correlation don't crosses the threshold anymore.

Document Sample	Gain (g)	Altered Characters (%)
1	0.020	0.625
2	0.0140	1.570
3	0.0140	22.76
4	0.0190	2.510
5	0.0120	20.09
6	0.0135	2.003
7	0.0200	12.46
8	0.0160	6.308
9	0.0170	0.675
10	0.0175	0.453

Table 3. Percentage of minimum altered characters the system can determine that the document is tampered.

5.3 Practical considerations

The system described above has a very low complexity, for embedding a watermark of length N, 5N multiplications are needed, the average execution time in a consumer laptop is depicted in Fig. 15. It can be seen that the system clearly meets a wide spectrum of practical needs; one can ensure that the system can process a document with hundreds of pages in few seconds, which should be good enough for most practical scenarios.

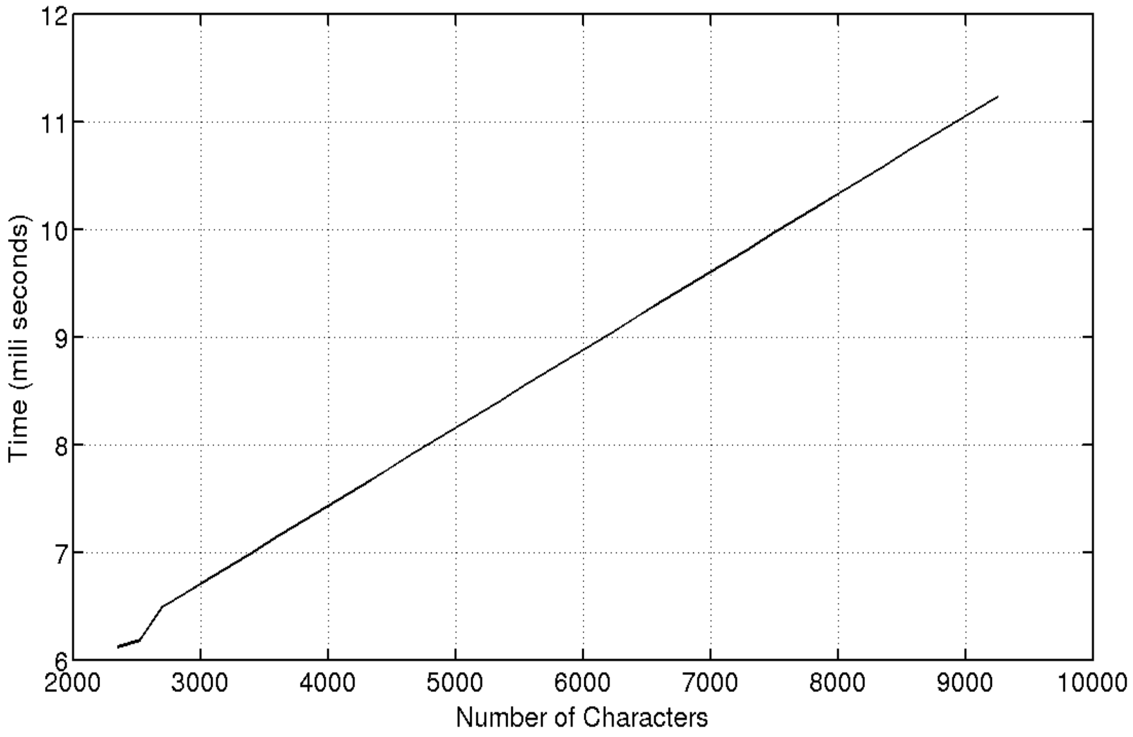


Fig. 15. Execution times for documents as the number of characters varies.

6. Conclusions

Through the development of this work, the following conclusions can be reached: Watermarking DDS format documents is a feasible and low complexity task that accomplishes a reliable electronic document authentication schemes with many desirable characteristics such as imperceptibility and very good tamper detection capabilities. Recall that many works in the field of document authentication are developed considering electronic documents as binary images, thus the development of watermarking systems in script format is a rich research field.

Results show that watermark imperceptibility is highly achieved as described in section 5.1, and considering the results of the MOS test, we can conclude that the proposed watermarking system will meet almost any imperceptibility requirements. Another important achievement is the tamper detection capability, that proved to be reliable even in the worst case of our tests, however, if this is a concern, a future work could perform verifications in smaller blocks, for example, the verification can be done in streams of 100 characters, so the 22.7% of characters that must be tampered, and 23 characters altered out of 100 is more likely to be a harmless modification since would be more difficult to have an attack useful to the proposes of any attacker.

Finally, the scheme discussed in this chapter is not intended to replace any security measures implemented in the different electronic document schemes such as the ones implemented in the ODF or in the PDF, but it would be advised to complement the current ones so a more secure electronic document model could be achieved.

7. Acknowledgments

The authors would like to thank the Council of Science and Technology (CONACYT) in Mexico and to the National Polytechnic Institute (IPN) of Mexico for support this work.

Examples in this chapter were chosen to mention C language in memory of its creator Dennis Ritchie, who passed away last October 12th, 2011. C language was extensively used during the development of this research.

8. References

Adobe, (1999). PostScript Language Reference, Third edition. Addison-Wesley Publishing Company Inc., ISBN 0-201-37922-8, U.S.A.

Adobe, (2006). PDF Reference: Adobe Portable Document Format Version 1.7, Sixth Edition. Adobe Press, ISBN 0-321-30474-8, U.S.A

Gonzalez-Lee, M.; Santiago-Avila, C.; Nakano-Miyatake, M. & Perez- Meana, H.; (2009) Watermarking based Document Authentication in Script Format. *Proc. 52th IEEE Midwest Symp. on Circuits and Systems*, ISBN 978-1-4244-4479-3. Cancun, Mexico. August, 2009.

Gou, H. & Wu, M. (2007) Improving Embedding Payload in Binary Images with Super-Pixels. Proc. *IEEE Int. Conf. Image Processing*, ISBN 1-4244-1437-7. San Antonio, U.S.A , September, 2007.

Hocko, M.; Mišutka, J. & Petříček, M.; (2009). *PDFedit. In PDFedit pdf manipulation library, gui, tools.* Available from: http://pdfedit.cz/en/index.html.

Huang, P.; Wu, D. & Tsai, W. (2004) A Novel Block-Based Authentication Technique for Binary Images by Block Pixel Rearrangements. *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME) 2004*, ISBN 0-7803-8603-5. Taipei, Taiwan. June, 2004.

Noren H. (2008) pdfcrack, *In PDFcrack – A Password Recovery Tool for PDF- Files. October 2011.* Available from: http://sourceforge.net/projects/pdfcrack/.

Piva, A.; Barni, M. & Capellini V. (1998). Threshold selection for correlation-based watermark detection. *Procedings of COST254 Workshop on intelligent communication. ISBN _____.* L'Aquila, Italy. April, 1998.

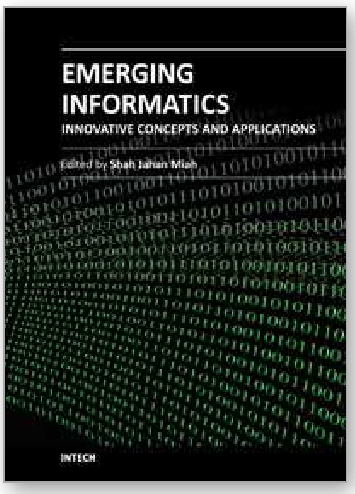
Reid, G.C.; (1990); Thinking in PostScript; Addison-Wesley Publishing Company Inc.; ISBN0-201-52372-8; U.S.A.

Turner D. (2000); Glyph , In: *Freetype Glyph Conventions.* October 2011. Available from: http://www.freetype.org/freetype2/docs/glyphs/index.html.

Wu, M. & Liu, B. (2004) Data Hiding in Binary Image for authentication and Annotation. *IEEE Trans. on Multimedia* Vol. 6 No. 4. April, 2004. pp. 528-538. ISSN 1520-9210.

Yang, H. & Kot, A.C. (2004). Text Document authentication by Integrating Inter Characters and Spaces Watermarking, P*roc. IEEE Int. Conf. On Multimedia and Expo (ICME) 2004.* ISBN 0-7803-8603-5. Taipei, Taiwan. June, 2004.

Zhu, B.; Wu J. & Kankanhalli, M.S. (2007) Render Sequence Encoding for Document Protection. *IEEE Trans. on Multimedia* Vol. 9, No. 1, January, 2007. pp. 16-24. ISSN 1520-9210.


EMERGING
INFORMATICS
INNOVATIVE CONCEPTS AND APPLICATIONS
Edited by Shah Jahan Miah
INTECH