

Alma Mater Studiorum - Università di Bologna

DIPARTIMENTO DI SCIENZE

Corso di Laurea Triennale in Informatica - 8009

Color Watermarking Techniques for Text-based Media

Google Workplace add-ons for watermarking

Relatore:

Prof. Danilo Montesi

0000838367:

Simone Branchetti

Correlatore:

Dr. Flavio Bertini

Anno Accademico 2020-2021

*To Bianca, for keeping my mind sane and to Davide and Luca, for filling it with
the things I needed to get this far.*

Summary

1	Introduction	1
2	Text-based watermarking techniques	3
2.1	State of the art for text based watermarking techniques	3
2.1.1	Zero Watermarking	4
2.1.2	Image-based methodologies	5
2.1.3	Syntactic methods	5
2.1.4	Semantic methods	6
2.1.5	Structural methods	6
2.1.6	Homoglyphs based watermarking	7
3	Watermarking techniques developed for this thesis	9
3.1	Grayscale based watermarking	10
3.1.1	GBW's Performance	11
3.2	Space Coloring based watermarking	11
3.2.1	SBW's performance	12
4	End user grayscale perception test	13
4.1	Population	13
4.2	Results	13
4.3	Considerations	16
5	Add-ons for Google Documents and Google Slides	18
5.1	Google Workplace and its add-ons	18
5.2	How the Google Documents Add-on works	19
5.3	How the Google Slides Add-on works	24
5.4	Performance and Portability tests	26
5.4.1	Performance	26
5.4.2	Portability for text editors	27
5.4.3	Portability for presentation editors	29
6	Digital Object Identifier metadata embedded Zero-watermarking	30
6.1	The Digital Object Identifier System	30
6.2	Structure of DOI's metadata	31
6.3	DOIs and Watermarking	34
6.3.1	Location of the string	34

7 Conclusions	37
Appendix A Survey for grayscale perception	38
Appendix B Add-ons code version	48

List of Figures

3.1	The first two paragraphs of our example text without any watermark.	9
3.2	The first two paragraphs of our example text watermarked with GBW using the lowest setting in our add-on (#070707).	10
3.3	The first two paragraphs of our example text watermarked with GBW using the highest setting in our add-on (#272727).	10
3.4	The first two paragraphs of our example text watermarked with SBW.	12
4.1	An example of a survey question from the first section.	14
4.2	An example of a survey question from the second section.	14
4.3	Population of the survey graphed for age groups and gender.	15
5.1	"Watermark" sidebar for the Documents add-on as it appears when it's just loaded.	21
5.2	"Display Watermarking" sidebar for the Documents add-on as it appears when it's just loaded.	21
5.3	The extra buttons in the DOI integration version of the Google Documents Add-on.	23
5.4	A comparison between a normal string (above) and its watermarked version (below).	24
5.5	The highlights from the second sidebar (below) help the user to identify the watermarks applied by the first sidebar (above).	24
5.6	"Watermark" sidebar for the Slides add-on as it appears when just loaded.	25
5.7	A comparison between a normal slide (above) and its watermarked version (below). The yellow highlight on the top element signifies that it wasn't long enough to hide the watermark at least once.	26
5.8	The resolution process of a slide leaves every element highlighted green if the resolution is successful, yellow if there weren't enough bits recovered to cover the watermark and red if the resolution was unsuccessful.	27
6.1	Visualization of the process of including a string generated by the add-on within a DOI's metadata.	35
6.2	Visualization of the process of resolving a watermark with the options being provided by a string in a DOI's metadata.	36

1	Text #1. Non-bold, font size 12	38
2	Text #2. Bold, font size 12	38
3	Text #3. Non-bold, font size 14	39
4	Text #4. Bold, font size 14	39
5	Text #5. Non-bold, font size 20	40
6	Text #6. Bold, font size 20	40

List of Tables

3.1	Table showing the details of the algorithm's performance when hiding a 64 bit payload.	11
4.1	Results to the first phase of the survey.	16
5.1	Table for text editors portability: which fonts hide homoglyphs in different text editors?	28
5.2	Table for presentation editors portability: which fonts hide homoglyphs in different presentation editors?	29
6.1	The basic elements of the IDF's Metadata Kernel.	33
1	First text results.	41
2	Second text results.	42
3	Third text results.	43
4	Fourth text results.	44
5	Fifth text results.	45
6	Sixth text results.	46
7	All of the texts combined.	47

Chapter 1

Introduction

Within the ever changing landscape of the internet certifying the ownership of a piece of media can be a challenging task. Media can be shared across various platforms like social media, cloud file sharing services, and websites and for this reason watermarking is becoming an ever increasing problem for people and companies worldwide.

Digital watermarking can be defined as the process of digitally embedding a certain amount of information (the watermark) into a piece of multimedia content such as a picture, a video or a written document. The simplest form of watermark to recognize is the photographer's name written inside the picture so that, if one were to use it improperly, the picture would carry the watermark and subsequently its paternity. Not every piece of media can be watermarked using the same method and the same piece of content can be watermarked in different ways, depending on the needs of the user. Watermarking a piece of text is especially difficult, because text as a medium does not provide a lot of avenues to be watermarked: for example, anyone would immediately be able to notice if a whole new word was to be added in the middle of a sentence. This means that watermarking text is done through other, more discreet, means like converting text to an image and watermarking that, using synonyms or, with the help of structural watermarking methods like the ones developed for this thesis, changing certain aspects of the text keeping the content intact. The related work of this thesis is a pair of add-ons (customized applications that integrate with Google Workspace productivity applications such as Gmail, Google Sheets, and Google Docs to provide extra functionalities) for Google Workplace (ex Google Suite) that watermark Google Documents and Google Slides respectively and resolve the watermarks. These add-ons use three structural watermarking techniques based on Homoglyphs, Space Coloring and Grayscale. Google Documents and Google Slides are two widely used tools for editing documents and presentations and add-ons provide a fast solution to adding features to them: in our case watermarking capabilities. Portability of the watermarking techniques presented for this thesis in both text editors and presentation editors is a pressing concern because of the large number of editors present on the market. In this thesis we

will explore our watermarking techniques' portability towards six different editors. The sixth chapter in this thesis is about the Digital Object Identifier (DOI) system and how, using metadata, we can apply this system, in combination with the Google Documents add-on, as an additional Zero-Watermarking technique. The DOI system serves as a way for people and/or organizations to certify one's paternity of an entity (books, documents, videos, pictures) by assigning a unique identifier and a set of metadata (additional information about the identified entity) to it.

Chapter 2

Text-based watermarking techniques

Watermarking a piece of text poses lots of different problems because text as a medium has low embedding bandwidth (meaning that there's not much room to hide information compared to an image, where every pixel can hide many bits of watermark) and only allows a restricted number of alternative syntactic and semantic permutations [25].

2.1 State of the art for text based watermarking techniques

According to [14], a watermarking method can be categorized using the following characteristics:

- *Readable or Detectable* - The watermarking is readable if the user can clearly read it. It's detectable if a function can be used to check for it, but cannot be otherwise read.
- *Visible or Invisible* - A visible watermarking method is visually perceptible by the user. On the contrary, the method is invisible if it is hidden in the original digital content and is not noticeable by the user. A visible watermark may be non-readable if a user can visually detect it but cannot read its content.
- *Blind or Non Blind* - If the original digital content is not needed in the extraction process, the watermarking is blind. Otherwise, the watermarking process belongs to the non blind category.
- *Simple or Multiple* - If a watermark can be applied only once the watermarking is simple. A multiple watermarking method means that a payload

of data can be embedded more than one time in the same document without affecting the whole process.

- *Fragile, Semi-Fragile, Robust* - A fragile watermark is detectable and can be altered or erased, thus, it is used for integrity authentication; a robust watermark is detectable and not erasable and it is most suitable for copyright protection. A semi-fragile watermarking is suited for content authentication.

Watermarking algorithms can also be classified in a few different categories based on how they watermark a text:

- *Zero Watermarking Techniques* - Instead of watermarking the text, some characterizing features of the text are stored on a third party authority server, such as an Intellectual Property Rights (IPR) database.
- *Image-based Techniques* - The text is transformed into an image, then the watermark is embedded into the image. Obviously, this approach modifies the nature of the original document so it cannot be considered a pure text watermarking method; however, it has some interesting features, such as length preservation and language independency.
- *Syntactic Techniques* - These methods transform the language depending structures in order to hide the watermark. Typically, the sentences have different language depending structures that make the process easier.
- *Semantic Techniques* - These methods use verbs, nouns, prepositions, spelling and grammar rules to permute the content and embed the watermark.
- *Structural Techniques* - These methods exploit double letters occurrences, words-shift and lines-shift encoding and the Unicode standard to embed the watermark. They are some of the most recent methodologies with which the original text is not altered.

This is just a basic description of the most important watermarking algorithms currently known. In order to better grasp the techniques presented in this thesis, we need to also look at the approaches actually used when embedding text. These approaches are generally classified in three main categories [14, 12]: image-based, syntactic and semantic. This classification leaves out the Zero marking approach as it doesn't actually hide any watermark in the text itself.

2.1.1 Zero Watermarking

We've already established that Zero watermarking techniques don't actually hide any information inside the content, but they aim to extract characterizing information from it, for example a picture or a song, and then store this information into an Intellectual Property Right (IPR) database [33]. Using this method the association between the content and the author does not rely on the watermark,

but on the proof from a trusted authority. Zero watermarking techniques can be seen as a form of dimensionality reduction, and in fact they are often based on well known dimensionality reduction techniques [19]. Using this technique the amount of data that the IPR has to store is greatly reduced if compared to a method that stores the entire text for paternity purpose. On the other hand, zero-watermarking has a clear weakness: the text and identity of the owner must be deposited to a third party and this can lead to privacy issues [25].

2.1.2 Image-based methodologies

Image-based text watermarking is the most researched approach to text watermarking and the earliest one to be investigated, with the first techniques dating back to the mid-nineties [9, 20]. In this approach a printed text is scanned as an image, or as a screenshot in the case of digital text, then a watermark is applied on this image. In gray-scale images of text documents, for example, the watermark payload is embedded by tuning the luminance of pixels according to the watermark data [6], or by modifying the edge direction histograms to carry the watermark signal [17]. A robust embedding can be obtained by slightly shifting the text elements horizontally or vertically: a text element can be a single word and the shift of a few pixels to the right or the left can hide information, or it can be a whole line of text, where shifting it up or down has the same purpose [9, 21]. Similar results can be obtained by altering the spaces between words to encode the watermark data [10, 16]. Other methods are based on the alteration of single characters [9], some focus on smaller details such as strokes and serifs of the characters and work by prolonging them [3], others, more simply, alter the character in their size by change the scale depending on the watermark content [32]. There are two important shortcomings of image based methods. The first is that text must be shared as an image (so, in an image file format e.g. PNG, JPEG or TIFF), as printed paper or through fax machines, which is nowadays less practical and not very common. The second is that text can be still reconverted to plain text by manual re-typing or through the use of an OCR software, losing any trace of watermarking in the process.

Overall, while it is a strong solution for printed papers and scanned documents, image-based text watermarking may become less and less relevant in the future because digital media is increasingly preferred to printed paper both for reading and sharing text contents.

2.1.3 Syntactic methods

Syntactic methods for text watermarking work on the syntax of natural language text, by altering its structure to embed a watermark. The first common step is to build the syntactic tree of a sentence, after which some syntactic operations such as clefting, passivization or activation are applied in order to encode the watermark bits [4]. Clefting is the process of transforming a simple sentence into a more, unnecessarily complex one: for example the simple sentence *"I like*

champagne" can be transformed into *"champagne is what I like"* (called **"what"** clefting) or into *"it is champagne that I like"* (called **"it"** clefting) [18]. Passivization is the transformation from the active to the passive form of verbs: for example, *"Tom kicked the bucket"*, to *"the bucket was kicked by Tom"*, while activization is the opposite process. There are also other morpho-syntactic transformations which can be also applied that are considered to preserve the original meaning: the linguistic notion of possession for instance can be written either with using the preposition 'of' or using the suffix '-s' [22]. Low embedding capacity, or the ratio between the text length and the length of the watermark that can be embedded, is a limit of these methods. Contexts of use where the length of the text is limited, such as mobile phones SMS texts or Twitter posts are inherently excluded. Other disadvantages of syntactic methods come from the alteration of the content; the assumption that different syntactic forms have the same meaning is not always true [11]: in the previous example, *"Tom kicked the bucket"* has an idiomatic interpretation, while its passive form only has a literal one.

2.1.4 Semantic methods

By exploiting the similarity of meaning of different words, it is possible to replace words with their synonyms [29]. The systematic substitution of words depending on the watermark data results in a non-blind watermark embedding. This semantic approach can be also mixed with syntactic approaches [28] to obtain an overall higher embedding capacity. Other semantic techniques work on sentence level semantic, leveraging the implicit presuppositions of each sentence [30, 31]. A presupposition is a sort of implicit information that follows directly from a sentence, usually a fact that must be true in order for the sentence to make sense. For example, in the statement *"Jane likes her white car"* the presupposition is that Jane has a car. By keeping the same meaning, the statement can be rephrased as *"Jane has a white car and she likes it."* It is therefore possible to add the presupposition explicitly, or in other cases to remove it, in order to encode watermark data. The semantic methods share some of the shortcomings of the syntactic methods. In the case of syntactic methods, the author's content can be strongly altered in order to embed the watermark. These methods also depend on the language and the correctness of written text.

2.1.5 Structural methods

Structural methods include all those methods that do not alter the text content but only its structure, intended as underlying representation or as features regarding visual rendering. These have recently emerged as methods that embed watermark or hidden payloads by changing the underlying encoding of symbols or adding invisible symbols, without actually altering the readable content of the text.

The Unicode standard has several different symbols for whitespaces, some of different width, others practically identical. By putting many of these whitespace

symbols at the end of a paragraph, or by filling an empty line, relatively long payloads have been hidden in Microsoft Word documents [24].

A similar technique has been effectively applied to watermark Arabic language text: by using a different Unicode whitespace between words depending on the bits of the watermark's binary representation a payload of data can be hidden in the text [2]. A more recent method uses multiple ASCII white spaces to embed a covert message [15] for PDF steganography instead. This last technique works on justified text and is able to embed 4 bits for each host line, where a host line is a line with at least 9 normal spaces and 3 wider spaces. Apart from whitespaces, the Unicode standard also provides some totally invisible symbols, which are coded as zero-width whitespaces. These symbols, together with whitespaces, have been exploited to hide hidden messages in text [5] and to watermark HTML pages [23, 27].

As mentioned earlier, these methods have the important advantage of keeping the original content unaltered without transforming the text to an image, or relying on an external database. The above structural methods are blind, meaning that the original text is not needed in order to extract the watermark. This, together with the easiness of removing multiple whitespaces, makes these approaches fragile in both malicious and benign attacks. This is particularly true for methods that use consecutive whitespaces and whitespaces before or after the whole text, because it has been proven that many digital platforms and social media automatically remove them [26]. A similar problem can also occur through selection for copy and paste: selection may easily exclude the white portion where the watermark is embedded. Apart from whitespaces, homoglyphs and invisible characters, some image based techniques where lines or words are slightly shifted without altering the text content [8, 7] have been also considered as structural methods [13].

2.1.6 Homoglyphs based watermarking

In Homoglyphs based watermarking (HBW), a particular type of structural watermarking, a payload of bits is embedded in a piece of text by replacing symbols and whitespaces with visually equivalent symbols, according to the watermark binary data: these symbols are called homoglyphs. This creates a technique that is invisible (the content is kept exactly the same), detectable, fragile and non blind (knowing the homoglyphs set it's possible to detect the watermark and expose the embedded data. This data is useless though, unless used in combination with the password the owner chose and the full text). HBW preserves the content and its length because every character is changed with another single character, thus preserving every aspect of the text. This is particularly relevant in the instances where there's a limit to the number of characters that can be used, like in social media posts, and other Unicode based techniques are not ideal because they embed the watermark by adding whitespace characters to the text. HBW has no overhead, is simpler to implement than NPL techniques and can be applied to every piece of text that is Unicode compatible: this includes most software and web platforms because because it does not depend on a particular file format [24] or markup language [23].

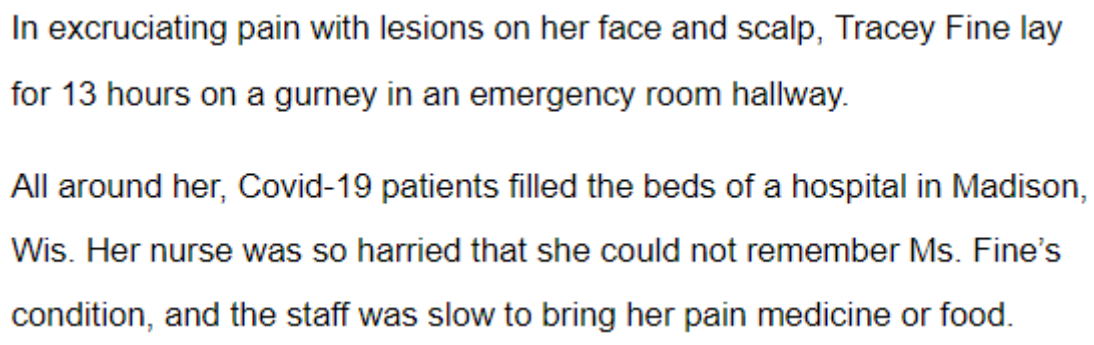
For a more comprehensive look at this watermarking method, its advantages and disadvantages see [25].

Chapter 3

Watermarking techniques developed for this thesis

After this lengthy explanation about different watermarking techniques we can focus on the ones developed for our project. We employed three different watermarking techniques: Homoglyph-based watermarking, Space coloring-based watermarking and Grayscale-based watermarking. The latter two are new watermarking techniques developed for this thesis. These methods are all structural methods of watermarking because they don't alter the text, but only visual aspects of it; these methods are also detectable, invisible, blind, simple and fragile and they can be used on their own or by combined, resulting in a total of 7 possible watermarking methods.

To better explain how these new techniques work we will take the first 5 paragraphs of an article written by Reed Abelson for The New York Times on the 27th of November 2020 titled ***Covid Overload: U.S. Hospitals Are Stretched Way Too Thin*** and use them as an example text on which we will apply our watermarking techniques. The full article can be read [here](#). Our example text is composed of 5 paragraphs or 1508 characters.



In excruciating pain with lesions on her face and scalp, Tracey Fine lay for 13 hours on a gurney in an emergency room hallway.

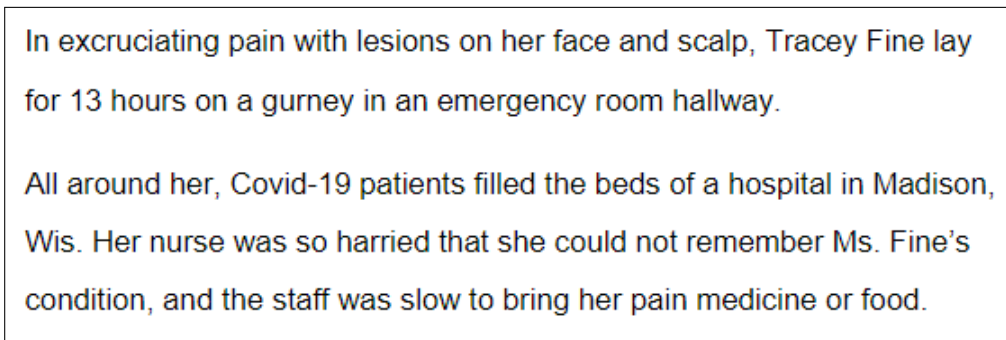
All around her, Covid-19 patients filled the beds of a hospital in Madison, Wis. Her nurse was so harried that she could not remember Ms. Fine's condition, and the staff was slow to bring her pain medicine or food.

Figure 3.1: The first two paragraphs of our example text without any watermark.

3.1 Grayscale based watermarking

Grayscale-based Watermarking (GBW) works by embedding the watermark into the color of characters (excluding whitespace and some punctuation signs). The RGB color of a character is split into three components (Red, Green, Blue) then for each component a small amount of bits (the number is chosen so that it can't go over the limit for any one component) is taken from the watermark and converted into an hexadecimal number. The three components are then joined together to create a single hexadecimal color for the character.

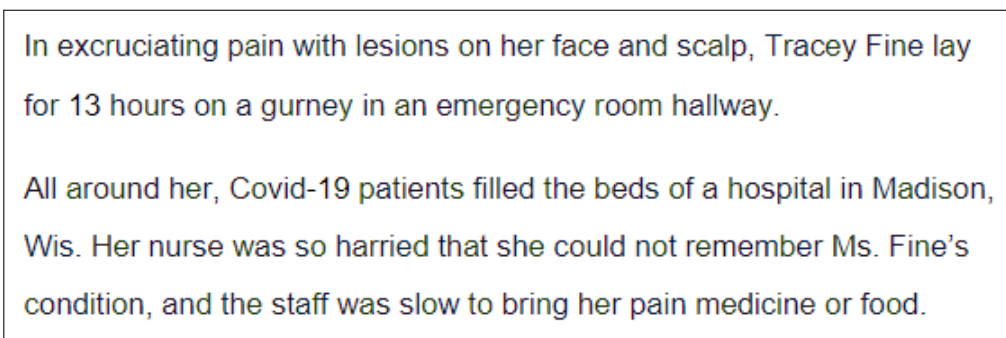
Coloring a character with a hex color is an easy way to hide a great number of bits into a single character but it also has its flaws; using GBW we can hide between 8 and 15 bits of watermark into every character, compared to the one to three bits per character embeddable with Homoglyphs based Watermarking. With the full hex scale we could embed more bits into every character but that would invalidate the invisibility of the method by making some of the characters stand out with color. This method is invisible, detectable, fragile (very susceptible to recoloring: if someone colors the whole text with any color every trace of the watermark is gone) and non blind because a user-inputted password is always required for the resolution process.

A rectangular box containing two paragraphs of text. The text is watermarked with a light blue color. The first paragraph reads: "In excruciating pain with lesions on her face and scalp, Tracey Fine lay for 13 hours on a gurney in an emergency room hallway." The second paragraph reads: "All around her, Covid-19 patients filled the beds of a hospital in Madison, Wis. Her nurse was so harried that she could not remember Ms. Fine's condition, and the staff was slow to bring her pain medicine or food." The watermarking is subtle, with the blue color being barely noticeable against the black text.

In excruciating pain with lesions on her face and scalp, Tracey Fine lay for 13 hours on a gurney in an emergency room hallway.

All around her, Covid-19 patients filled the beds of a hospital in Madison, Wis. Her nurse was so harried that she could not remember Ms. Fine's condition, and the staff was slow to bring her pain medicine or food.

Figure 3.2: The first two paragraphs of our example text watermarked with GBW using the lowest setting in our add-on (#070707).

A rectangular box containing two paragraphs of text. The text is watermarked with a light blue color. The first paragraph reads: "In excruciating pain with lesions on her face and scalp, Tracey Fine lay for 13 hours on a gurney in an emergency room hallway." The second paragraph reads: "All around her, Covid-19 patients filled the beds of a hospital in Madison, Wis. Her nurse was so harried that she could not remember Ms. Fine's condition, and the staff was slow to bring her pain medicine or food." The watermarking is more prominent than in Figure 3.2, with the blue color being more visible against the black text.

In excruciating pain with lesions on her face and scalp, Tracey Fine lay for 13 hours on a gurney in an emergency room hallway.

All around her, Covid-19 patients filled the beds of a hospital in Madison, Wis. Her nurse was so harried that she could not remember Ms. Fine's condition, and the staff was slow to bring her pain medicine or food.

Figure 3.3: The first two paragraphs of our example text watermarked with GBW using the highest setting in our add-on (#272727).

These pictures show that this method is invisible even with relatively high font size (the pictures show the text in Arial font size 14). The different colors start to become more apparent the higher the font size gets, of course. Another factor that facilitates the visibility of the different colored characters is if the character is bold or not because with bold characters it's easier to discern whether its color is black or not.

3.1.1 GBW's Performance

GBW Performance			
Hue	#Bits hidden	Time (seconds)	#Times hidden
#070707	8	15	188
#0f0f0f	11	19	251
#171717	13	25	301
#1f1f1f	14	14	301
#272727	15	20	301

Table 3.1: Table showing the details of the algorithm's performance when hiding a 64 bit payload.

When testing the performance of this algorithm we used our add-on on the example text to hide a watermark composed of 64 bits. The results are compiled in this table: the first column represents the maximum hue possible for every character (these are the same five hues we use as our options in the add-on, from very weak to very strong), the second one shows how many bits of watermark are hidden in one character, the third one tells us how much time the process took and lastly, the fourth column shows how many times we managed to hide our 64 bit watermark inside the text. Please keep in mind that every function in every add-on for Google Workplace is executed on Google servers, so execution times may vary and are out of our control.

3.2 Space Coloring based watermarking

Space coloring-based watermarking (SBW) works similarly to the Grayscale-based technique but focuses on whitespaces instead of characters. It retains the same weakness to recoloring as the previous method but boasts an increased bit embedding per character rate with 24 bits of watermark per single whitespace character. This is possible thanks to the fact that a whitespace character doesn't render its color on screen but remains transparent. This makes this method invisible, detectable, fragile and non blind (the watermark can be extracted from the colors but without the user-selected password it can't be resolved). SBW is fully compatible with Homoglyphs-based Watermarking because every whitespace character can still be colored like every other character.

3.2.1 SBW's performance

Our example text was watermarked with the same 64 bit watermark to test for performance. The process took 13 seconds and we managed to hide the watermark in the text 10 times. This shows that, while SBW has a greater embedding capacity per character compared to its grayscale counterpart, whitespaces are way less common compared to normal characters; for instance in our example text out of the total 1508 characters only 252 (17%) are whitespaces. This of course can be remedied by combining this technique with other non-structural watermarking techniques that add whitespaces to the text but, because the focus in this thesis is on structural watermarking techniques we have not tested that possibility. This

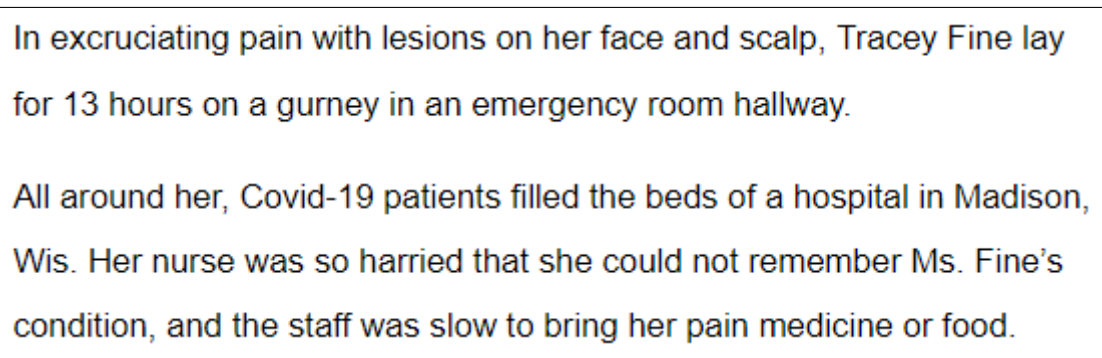


Figure 3.4: The first two paragraphs of our example text watermarked with SBW.

picture shows clearly one of the main strengths of this watermarking technique: invisibility. This technique is only visible when the document is still in the Google Documents Editor and with the cursor on a whitespace the user notices that its color is not the same as that of the rest of the text.

Chapter 4

End user grayscale perception test

Color perception is a vital part of grayscale based watermarking, because to guarantee its invisibility property we have to make sure that fewest people as possible can distinguish between black text and the grayscale value used. In order to determine the optimal range on values that are indistinguishable from black and let us hide the most bits possible per character, Prof. Montesi and Dr. Bertini created a Google Survey that we presented to 162 people. This survey is composed of two sections: in the first one participants are asked if two squares on the screen are filled with the same color or not; this tests grayscale in a standard environment, unrelated to text. The second section is composed of a series of texts where some words are colored with different grayscales, ranging from barely noticeable to very apparent: the user is then asked to write the word where he first noticed a different colored letter in. The characters in a lighter shade of gray are able to be embedded with more bits than darker shades but they are more noticeable when side by side with black characters. These results can give us insight into what hex values are not easily distinguished from pure black.

4.1 Population


Even though researchers are still unsure if men and women perceive grays differently [1], a diverse population is still preferred in every study. Our survey was answered by 88 males (54.3%) and 74 females (45.7%). The most represented age group is people from 20 to 29 years old and the least represented one is formed by people from 70 to 79 years old.

4.2 Results

We organized the 162 results to better visualize how the polled population perceives different shades of gray. Results for the first part show that, predictably, as hues get increasingly more distant from black, more and more people

Text Watermarking

*Campo obbligatorio



Le due immagini sono uguali? *

Si

NO

Figure 4.1: An example of a survey question from the first section.

Text Watermarking

Ora ti verranno proposti diversi testi. Dovrai riportare la prima parola, seguendo l'ordine nel testo, in cui noti una differenza di colore tra le lettere che la compongono. Ad esempio, nel testo che trovi qui sotto la prima parola in cui è stata notata la differenza è "primo", dovrai quindi indicare "primo" nella casella di testo disponibile.

Il signor Trelawney, il dottor Livesey e gli altri gentiluomini mi hanno chiesto di mettere per iscritto tutti i dettagli riguardanti l'Isola del Tesoro, dal primo all'ultimo, senza omettere nulla salvo la posizione dell'isola,

Figure 4.2: An example of a survey question from the second section.

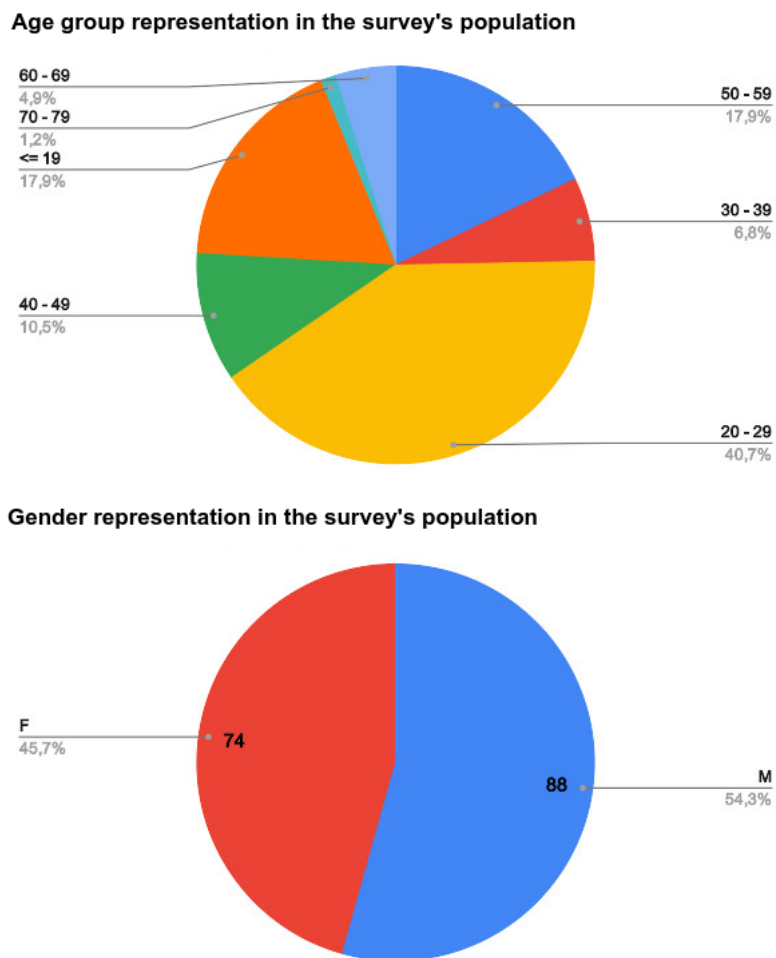


Figure 4.3: Population of the survey graphed for age groups and gender.

Are the two squares colored with the same shade of gray?		
Squares' hues	YES	NO
#000000 / #070707	118 - (72.8%)	44 - (27.2%)
#000000 / #0F0F0F	113 - (69.8%)	49 - (30.2%)
#171717 / #000000	68 - (42%)	93 - (58%)
#1F1F1F / #000000	45 - (27.8%)	116 - (72.2%)
#000000 / #272727	33 - (20.4%)	128 - (79.6%)
#2F2F2F / #000000	19 - (11.7%)	143 - (88.3%)
#000000 / #373737	10 - (6.2%)	152 - (93.8%)
#000000 / #3F3F3F	8 - (4.9%)	154 - (95.1%)
#474747 / #000000	5 - (3.1%)	157 - (96.9%)
#000000 / #4F4F4F	3 - (1.9%)	159 - (98.1%)
#575757 / #000000	3 - (1.9%)	159 - (98.1%)
#000000 / #5F5F5F	4 - (2.5%)	158 - (88.3%)

Table 4.1: Results to the first phase of the survey.

can tell the difference between the two squares. We see an increase in the number of people who correctly recognize the difference in the squares from a 26% when the colors are black and #070707 to a 97.5% with black and #5F5F5F.

Thanks to this table that shows every answer from the first part of the survey we can see that more than 70% of people can, for example, see the difference between #1F1F1F and #000000. This tells us that, for our purposes, we should aim for a lower maximum hue to minimize the watermark's visibility. While it is a good approximation, if it's not applied to text it's not as useful for our purposes; that's why the second section focuses on this exact problem by having the people taking the text write the first word where they notice a different colored letter. Every piece of text for the second part of this survey and its results are in the first appendix, so as to not bog down the main text, but the conclusions will be here.

4.3 Considerations

The point of this study was to understand what range of grayscales is the least noticeable to the human eye, while maximizing bits to watermark each character with. We're especially interested in the second part because it shows people's reaction to actual text. From the data we gathered across all of the texts it's clear that after #272727 too many people recognize the difference (>10%) so the final range of grayscales we used in the addons is from #070707 to #272727. This interval ensures that the range of people that can recognize the watermark is from 1.57% to 7.57%, which is an acceptable range of detection. Some things

to remember are that some of the texts in the survey are bold and that enhances the grayscale detectability, and that this is a cumulative measure, meaning that 1.57% of the interviewed people recognized one word with some letters colored #070707 in all of the texts.

Chapter 5

Add-ons for Google Documents and Google Slides

5.1 Google Workplace and its add-ons

Google Workplace (previously known as Google Suite) is Google's solution to an editing environment for documents, presentations, spreadsheets, surveys, events and e-mails whose focus is customization, speed, scalability and accessibility. This service includes many Google editors such as Google Documents, Google Slides and Google Forms while also containing tools to help people and businesses organize and collaborate when editing and planning, such as Gmail, Google Drive and Google Calendar. These webapps are, of course, packed with features that make it easier to create and edit documents collaboratively, but what happens when a feature you need, for example watermarking capabilities, isn't included in an editor? Fortunately, every webapp included in the Google Workplace package can be extended using add-ons. Add-ons are customized applications that integrate with Google Workspace productivity applications such as Gmail, Google Sheets, and Google Docs to provide extra functionalities. They are built using Apps Script, Google's own application development platform that is based on JavaScript. Using add-ons you can:

- Create customized user interfaces that are directly integrated into Google Workspace applications. These interfaces can display information to the user and provide user controls;
- Boost workflow efficiency when working with Google Workspace by automating or streamlining tasks;
- Easily control and move data between Google applications with Apps Script services;

- Remove the need for browser switching by providing the user everything they need within Google Workspace;
- Connect to non-Google services within Google Workspace applications, allowing you to retrieve or upload data from those services into and from Google Workspace.

There are two ways to add an add-on to your editors: making it yourself or adding it to your editor from the Google Workspace Marketplace. While the former is the way we chose to follow in this thesis, the latter is certainly the more common of the two because the Marketplace is a collection of ready to use, google-approved add-ons made by other users and/or companies that are free to use and easy to install. If there is no add-on on the Marketplace that implements the desired features, it is always possible and encouraged to make one and publish it by accessing the "External Components" tab and clicking the "Script Editor" button; this will open the App Script editor and let you start developing your own add-on.

Building an add-on from scratch can be a daunting task, but Google provides plenty of resources to help aspiring add-on creators get going. Since Apps Script has easy built-in access to every API from the Workplace, Google provides documentation for all of them plus documentation for App Script itself on Google Developers. Both of the add-ons presented in this thesis are made following directions from Google Developers regarding both the APIs and Apps Script itself, and feature, between the both of them, three sidebars, 8 different possible watermarking/resolving techniques, six hashing functions to generate a watermark and 5 levels of grayscale strength. One thing to keep in mind with both of the addons is that homoglyphs are supported only in a select number of fonts, defaulting to a different style if not properly mapped. This means that this watermarking techniques is truly hidden only in documents written in fonts like Times New Roman, while sans-serif fonts like Arial usually accentuate the presence of homoglyphs in the text instead of hiding it. This concern to doesn't apply to whitespace homoglyphs, but only to the non-whitespace ones. The whitespace homoglyphs sometimes expand when the text is justified, so right-aligned is recommended.

5.2 How the Google Documents Add-on works

After installing the add-on, it's possible to use it like any other: with any Google Documents document open, after clicking on the "Add-ons" tab in the top part of the screen and selecting our add-on, two buttons will appear. The two of them open up one sidebar each. Every functionality of this add-on is contained in one of these sidebars. The "Watermark" sidebar handles watermarking, watermark resolution, watermark deletion and can also tell you how many characters will hide the watermark you selected. The "Display Watermarking" sidebar contains functionalities for highlighting a watermark in your text, checking for signs of possible watermarks and cleanup of the aforementioned highlights. Scattered

around the sidebars are little blue symbols that, when hovered over with the cursor, offer useful information about many of the settings in both of the sidebars.

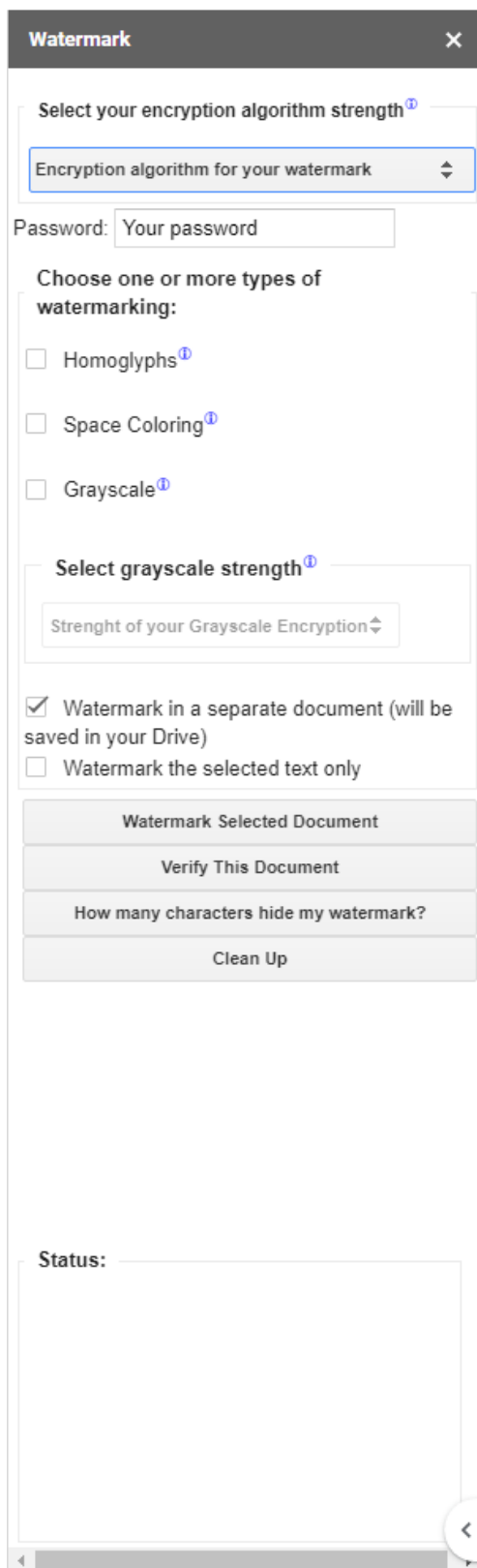


Figure 5.1: "Watermark" sidebar for the Documents add-on as it appears when it's just loaded.

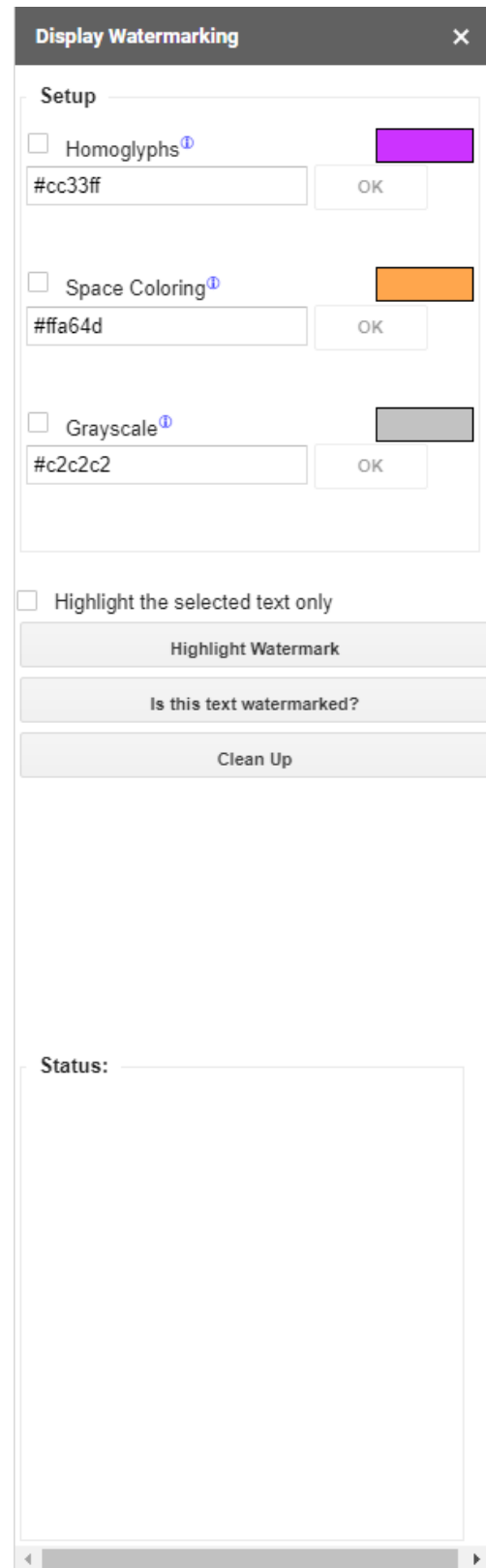


Figure 5.2: "Display Watermarking" sidebar for the Documents add-on as it appears when it's just loaded.

The components for the **Watermark** sidebar from top to bottom are:

- A dropdown menu to pick the algorithm you prefer to create the watermark: this will affect watermark length, with shorter watermarks requiring less characters to hide
- A textfield to insert your password: this password will be used to create the watermark and is required to positively recover your watermark from a text.
- Three checkboxes to select the type of watermark, allowing for 8 total watermarking techniques by combining the three main ones (Homoglyph-based, Space Coloring and Grayscale).
- When the checkbox for Grayscale-based watermarking is checked, the dropdown menu to specify the level of grayscale strength desired gets enabled. The stronger the grayscale the greater the number of watermark bits that we can hide in a single character but, at the same time, more people can possibly see the difference in color.
- One checkbox for controlling if the watermark is to be applied in the same document the add-on is open on or in a new document, saved in the same Google Drive folder, with the same name as the original but with the suffix "-WATERMARKED" added.
- One checkbox to control the range of the other functions. If it's checked most of the other functions only work in the user selected part of the text.
- Four buttons: the first one starts the watermarking process with the info from previous components, the second one starts the resolution process with the same components, the third one checks how many characters it would take to hide a watermark with the parameters specified (doesn't actually watermark the text) and the final one simply cleans up any trace of a watermark, returning homoglyphs to their respective ASCII counterpart and resetting the color of the text to black.
- A status box that displays messages with timestamps at the beginning and at the end of functions. These messages show what function started, when it started and its results.

The components for the **Display Watermark** sidebar from top to bottom are:

- Three selectors for the types of watermark to highlight. When one of these selectors is selected it's possible to change the color with which that technique will be highlighted by writing a new HEX code and pressing OK. Note that when highlighting Homoglyph-based watermarking in combination with other techniques the first one takes priority but that doesn't mean that the character isn't watermarked with both techniques. The rectangle next to the name of the technique shows a preview of the color

- A checkbox to select if you want the highlighting to happen on the whole text or just the selected part of the text.
- Three buttons: one for highlighting the text with the colors you chose, one for highlighting every paragraph in which a watermark is present in yellow and one to clean up any and all highlights.
- A status box that displays messages with timestamps at the beginning and at end of functions. These messages show what function started, when it started and its results.

Every button checks for related parameters before starting its function so that the script has every information it needs. For example a watermarking attempt won't go through without specifying the algorithm that you want to use. If two people want to watermark the same text one can use Grayscale-based watermarking and the other can use Space coloring-based watermarking. This is possible because grayscale watermarking doesn't check whitespaces when resolving a watermark, checking the color of other characters instead and vice versa for the Space Coloring technique. When watermarking a text the actual watermark is determined by hashing the entire text with the password and the algorithm provided. This poses some problems when trying to recover a watermark because, when homoglyph-based watermarking is used, some characters won't be the same as when the text was watermarked, thus requiring an additional step in the computation that restores the text to its version without homoglyphs. The API for Google Documents handles user selections like a list of elements, some of which can be non textual (for example pictures); since we are only interested in text for our purposes, a list of all the selected elements that have text in them is created to act as the *text* when the "Watermark selected text" checkbox is checked. When the same checkbox is not checked this is not an issue because, for whole texts the API has a function to return a list of all the paragraphs in a text and paragraphs only have text in them. Our watermarking and resolving algorithms can therefore also be used with documents that contain pictures. If you're using the version with DOI integration, you'll have access to a couple new options on top of all of the others.

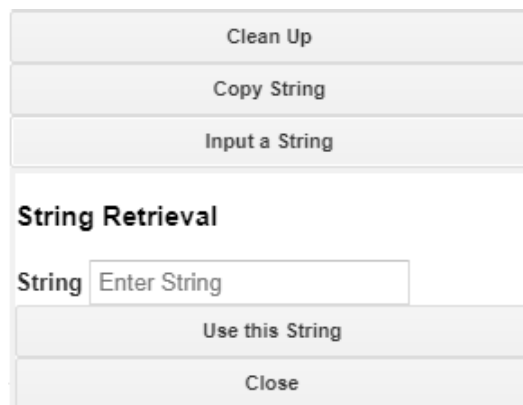
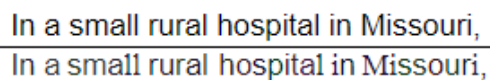


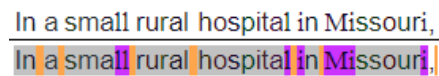
Figure 5.3: The extra buttons in the DOI integration version of the Google Documents Add-on.

These extra controls appear in the form of two buttons located just below the "Clean up" button in the **Watermark** sidebar. The button **Copy String** launches a function that transforms the settings of your watermark (password excluded) into a string that is ready to be hidden inside a DOI's metadata and automatically puts it in your clipboard. The button labeled **Input a String** opens the panel just under it, as shown in the picture. It contains a textbox to put the string retrieved from your DOI in, a button to use that string to load the settings to your watermark and a button to close the panel. Our string is simple by design, being composed of the values of the components separated by %. This allows ease of access and readability without compromising security because without a password, these settings are not useful.



In a small rural hospital in Missouri,
In a small rural hospital in Missouri,

Figure 5.4: A comparison between a normal string (above) and its watermarked version (below).



In a small rural hospital in Missouri,
In a small rural hospital in Missouri,

Figure 5.5: The highlights from the second sidebar (below) help the user to identify the watermarks applied by the first sidebar (above).

These images show how a full watermark (a watermark with all three techniques combined) looks on a string of text taken from our example article from Chapter 3.

5.3 How the Google Slides Add-on works

The Google slides add-on work similarly to the Google Add-on's one with some key differences to accommodate for the difference in the form of media treated. The most glaring difference is the presence of one sidebar instead of two. Because Google Slides works with presentations instead of documents, it handles text and selections differently: Google Slides lets the user select a wide range of elements, from whole slides to a part of a text inside of a slide; since it's hard to know what elements contain text and how long it is from the selection alone, our algorithm handles text watermarking element by element, leaving behind those that don't have any text. When going on a case by case basis, a possible problem is the shortness of the text: if a text is shorter than the minimum number of characters necessary to embed the watermark at least one time the element will be watermarked partially, meaning that no watermark can be recovered from that element. This problem is partly solved by highlighting every element with a different color depending on the result of the watermarking and resolving processes. The elements in this sidebar are very similar to the ones in the Google Documents version, and are only different in the way this add-on handles functions. The checkbox to switch to and from selection mode is gone and in this add-on every function is applied to the selected elements only. The filter that checks that the appropriate options for each function are valid is still present but

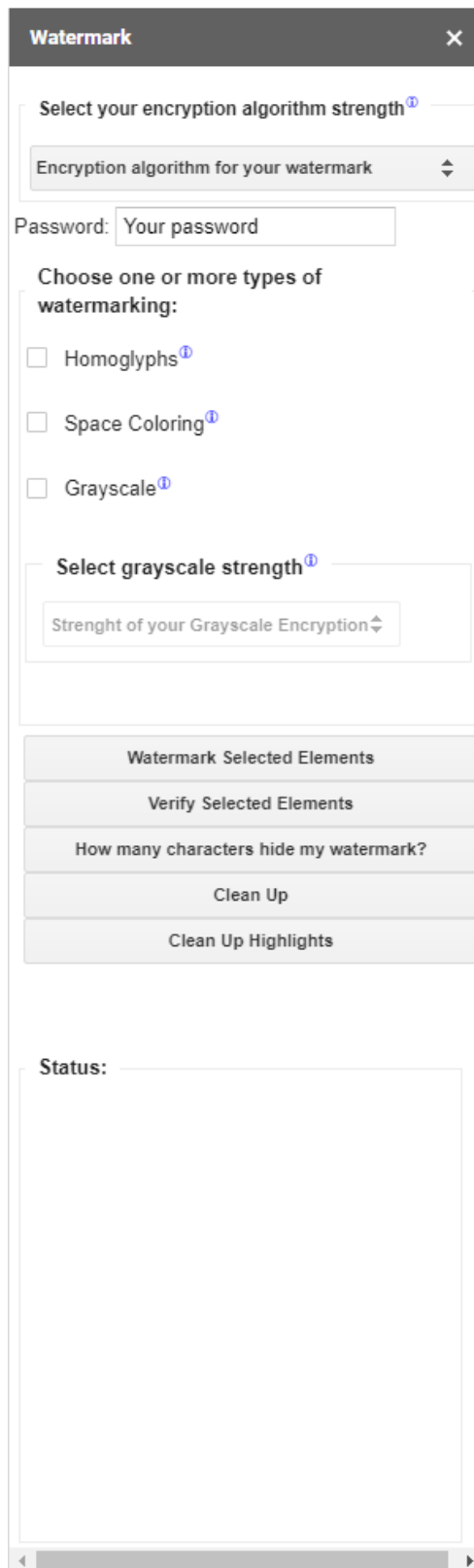


Figure 5.6: "Watermark" sidebar for the Slides add-on as it appears when just loaded.

there is no good way to check if a selection is empty, so if a selection is empty, the function will run like normal but change nothing in the presentation. The function that shows you how many characters hide your watermark has also been overhauled and it now highlights parts of text with three colors: each colored part represents one fully hidden watermark. If the last part of the text can't hide a full watermark it's highlighted in red to show it. This function doesn't actually watermark the text in any way. Another new button on this sidebar is the **Clean Up Highlights** one that, as the name suggests, gets rid of every highlight from the selected elements. Due to the fact that we are using highlights to let the user know basic information about the result of his operations, we felt like the second sidebar, whose focus has always been letting the user know about his watermark, had no place in this add-on and we got rid of it. Figure 5.7 and 5.8 are taken using a font that shows homoglyphs clearly. This is done on purpose to highlight them, because if they were hidden the comparisons wouldn't work as well.



Figure 5.7: A comparison between a normal slide (above) and its watermarked version (below). The yellow highlight on the top element signifies that it wasn't long enough to hide the watermark at least once.

5.4 Performance and Portability tests

5.4.1 Performance

Of course, when using an add-on for Google Workplace, a serious concern is time: because every script function is run on Google servers, normal end users have a maximum execution time of 6 minutes. At the same time, unfortunately, our

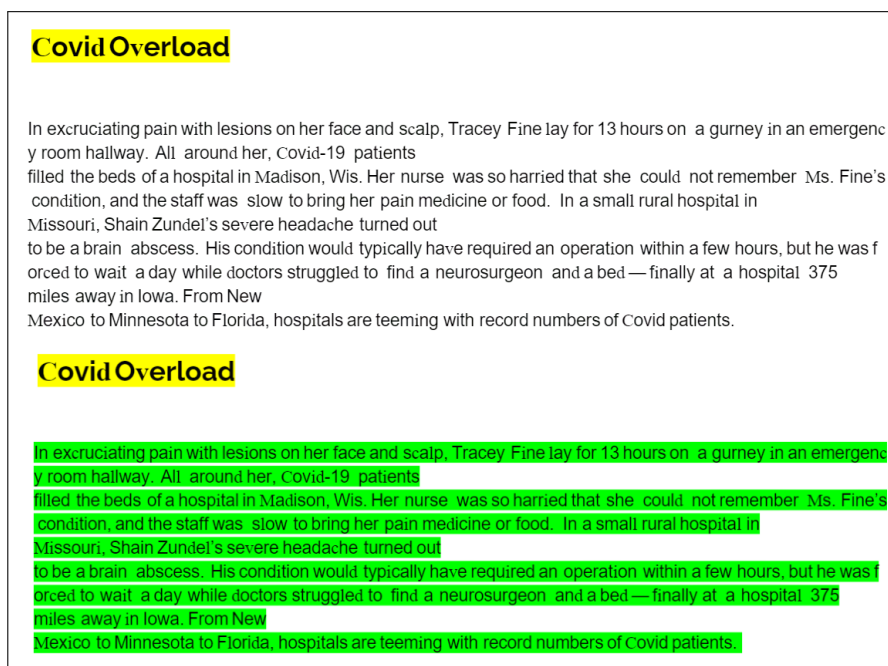


Figure 5.8: The resolution process of a slide leaves every element highlighted green if the resolution is successful, yellow if there weren't enough bits recovered to cover the watermark and red if the resolution was unsuccessful.

functions need many updates to the text to work and the Workplace API is built to handle few updates to large amount of text. This means that our functions have a longer execution time than normal. This time of course depends on many factors like length of the text, font size, if there are images in it or not and number of paragraph. Watermarking a full page typed in Arial 10 font with the Google Documents add-on takes one minute. This is a good baseline measurement for the performance of this add-on. The measurements for the Slides Add-on are slightly more complicated because Google Slides is fundamentally different from Google Documents. A slide of a presentation can contain many elements, many of which may not have any text in it. Watermarking the text in a single element is not difficult and it takes between 5 and 30 seconds depending on the length of the text. The time consuming part is finding the elements with text in a selection. Too many elements selected may run out the timer before the operation is fully completed but, unlike the Documents add-on, because the watermarking function is applied to every element, the first elements in the selection will be watermarked.

5.4.2 Portability for text editors

Portability plays a major role in the development of text-based watermarking techniques because of the many different text editors existing today. While Space coloring-based watermarking and Grayscale-based watermarking don't suffer from the shift in text editors, Homoglyphs-based watermarking may have some problems depending on how every other editor handles homoglyphs. For portability

in this case we mean the process of copying and pasting watermarked text from Google Documents (where it originated) to another text editor and back again; if the watermark is kept and can be retrieved, our watermarking method is portable towards that particular text editor. We tested our full watermarking technique (all three of the techniques combined) with a 64 bit watermark on the first two paragraphs of our example text from chapter 3. The other editors tested were Microsoft Office Word, LibreOffice Writer and OpenOffice Writer. All of these pieces of software were using their latest version possible as of the 28th of November 2020 and were installed on a Windows 10 system. Future updates or a different OS may skew these results. The first thing we noticed was that, while Google Documents handled homoglyphs in fonts that did not support them with minimal difference, in other editors (especially LibreOffice and OpenOffice) the difference is much more stark and noticeable. This calls for a list of **safe fonts** that guarantee that the watermark is properly hidden in every editor. These fonts must hide homoglyphs in both Google Documents and any other editor being tested; we will also only take into consideration fonts that are shipped with the products, discarding any custom font for ease of testing. A great number of fonts are not fit for our purposes, especially sans-serif fonts like Arial or Calibri. On the other hand, serifed fonts like Times New Roman or Georgia hide homoglyphs in Google Documents perfectly. This is the full list of fonts that hide homoglyphs in Google Documents:

- Times new Roman
- Georgia
- EB Garamond
- Spectral

Out of the four fonts listed here the only two present in every editor considered are Times new Roman and Georgia. Now these fonts have to hide the same watermark in the other editors. As we can see in the table, even though the fonts

Does the font hide the watermark in this editor?		
Text Editor	Times New Roman	Georgia
Microsoft Word	YES	YES
LibreOffice Writer	YES	NO
OpenOffice Writer	YES	NO

Table 5.1: Table for text editors portability: which fonts hide homoglyphs in different text editors?

are the same, only Microsoft Word correctly hides all of the homoglyphs with both fonts. With LibreOffice and OpenOffice, only Times new Roman hides the homoglyphs. The reason why some fonts hide homoglyphs and others don't is that all fonts define a series of glyphs for letters they intend to cover with different size, weight and style from one another, so, because of size and complexity concerns,

not every Unicode character is mapped when creating a new font. Many fonts only map ASCII characters. This means that other characters need to default to a different font when typed (this applies to our homoglyphs because they are non-ASCII characters). If the default font is similar in style to the original one or if the original font defines glyphs for our homoglyphs and those glyphs are the same as their corresponsive characters, the homoglyphs are hidden in that font. These problems are of course only present when using Homoglyph-based watermarking and the simplest fix to them would be to just use the two other techniques instead that are completely portable with every other text editor tested. Grayscale-based watermarking and Space coloring-based watermarking were correctly preserved, and a watermark was extracted successfully after the tests, by all of the editors and had no problem related to fonts that were not already discussed in chapter 3.

5.4.3 Portability for presentation editors

We repeated the same tests for presentation editors, this time to check the portability of watermarks applied by the Google Slides add-on. The editors considered for these tests are Microsoft PowerPoint, LibreOffice Impress and OpenOffice Impress. All of these pieces of software were installed on a Windows 10 system and had their version updated to the latest available as of the 28th of November 2020 before the start of the tests. We copied the contents of an element from a Google Slides presentation into a new element of the same type in a presentation from another editor, checked for invisibility and then repeated the process in the opposite direction to check if the watermark was still valid. The fonts tested for homoglyph invisibility are the same ones we tested for text editors for the same reasons as stated previously: Times new Roman and Georgia. The only differ-

Does the font hide the watermark in this editor?		
Presentation Editor	Times New Roman	Georgia
Microsoft PowerPoint	YES	NO
LibreOffice Impress	YES	NO
OpenOffice Impress	YES	NO

Table 5.2: Table for presentation editors portability: which fonts hide homoglyphs in different presentation editors?

ence in these results is that that Georgia on Microsoft PowerPoint does not hide homoglyphs. This is probably due to a different version of the font or a different default font for this software compared to Microsoft Word. Another problem present with presentation editors that text editors don't have is that color is not preserved between copy and paste operations. This means that unfortunately, Grayscale-based watermarking and Space coloring-based watermarking are not portable to any presentation editor tested. A watermark was successfully recovered and extracted using only Homoglyph-based watermarking with all three of the editors tested, so this technique is portable to any of them.

Chapter 6

Digital Object Identifier metadata embedded Zero-watermarking

6.1 The Digital Object Identifier System

A Digital Object Identifier, or DOI, is an identifier of an entity on the net. It provides a system for persistent identification and exchange of information and it can be assigned to any entity (physical, digital or abstract). Its main purposes are the sharing and managing as intellectual property of the associated entity. DOI names can also be expressed as URIs. This system is already widely used with more than 230 million DOI names assigned and over 5 billion DOI resolutions per year, according to the official International DOI Foundation's website. The system was initiated by the International DOI Foundation (IDF) in 1998 and later standardised as ISO 26324. The DOI system's main component is the DOI name; this name is composed of two parts: the prefix and the suffix. Let's take 10.1000/182 as an example: this is the DOI for IDF's latest applicable version of their handbook, which contains useful information about the whole system and its workings. This name is composed by:

- A prefix (**10.1000**) - This prefix is composed by a 10 (which signifies that this string is a DOI) and a user-selected prefix (in this case 1000), separated by a period
- A suffix (**182**) - The suffix is user-chosen and must not be repeated in any other DOI registered by the same user.

This combination of suffix and prefix ensures that no two DOIs have the same name as each other. These names are granted by a DOI Registration Agency (RA), that's a separate company from the IDF that provides services to whoever wants to register a DOI. Registration Agencies provide services like allocating prefixes, registering DOI names and providing the necessary infrastructure to maintain and store metadata. An RA can be considered like a module to the

DOI system. New RAs can be added at any time, allowing for the system's modular growth by incorporating new communities of users. Currently there are 10 Registration Agencies: Airiti, Crossref, CNKI, DataCite, EIDR, ISTIC, JaLC, KISTI, mEDRA and the Publications Office of the European Union. Every RA has a different business model and operates in a different part of the world; some are located in Asia, some in Europe, some in the United States of America. The IDF's main purpose is DOI name resolution, that is a process in which an identifier is inputted and one or more pieces of information related to the identified resource are returned, like an URL. Using this kind of *multiple resolution*, a single DOI name can be resolved to an arbitrary number of associated resources like URLs, other DOI names or other data types representing metadata items.

6.2 Structure of DOI's metadata

Metadata are a vital part of the DOI system because an identifier is of no value without some related metadata describing what it is that is being identified. Metadata are defined as data that provides information about other data, such as a title, author or anything that can help identify a piece of data. It can be stored in different format but every RA uses an XML model personalized to fit the RA's needs, using the custom tags that XML provides to the fullest. Every RA has its own set of metadata defined in a schema (or scheme) that a person or company can fill and update for free when he registers a DOI but the IDF defines a basic Kernel of mandatory metadata to identify a registered entity. The purpose of this kernel is to provide a minimum set of metadata that different RAs can build upon based on the preference of their audience. This minimum set of metadata has two aims: **recognition** and **interoperability**

Recognition means that the kernel metadata should be sufficient to show clearly what kind of thing our DOI name refers to, and allow a user to identify this particular thing. These are complementary, for it is possible to know that something is a book without knowing that it is "The Da Vinci Code" and vice versa. *Interoperability* means that kernel metadata from different RAs may be queried by the same software without requiring semantic mapping or transformation. The kernel provides this principle directly because every implementation of a schema from a different RA must contain this kernel.

Elements of the DOI Kernel according to the IDF		
Kernel element(s)	occurs	Description
DOI name	1	Specific DOI name allocated to the identified referent.
referentIdentifier(s)	0-n	Other identifier(s) referencing the same referent (e.g. ISAN, ISBN, ISRC, ISSN, ISTC, ISNI). This element contains a type element appropriate to the primaryReferentType. The schema at present recognises a creationIdentifierType and partyIdentifierType, which are open lists for which new allowed values may be registered.
referentName(s)	0-n	Name(s) by which the referent is usually known (e.g. title). This element contains a type element appropriate to the primaryReferentType. The schema at present recognises a creationNameType and partyNameType, which are open lists for which new allowed values may be registered. This element also contains a language element, for which the allowed value list is the ISO 639-2 code list.
primaryReferentType	1	The primary type of the referent (e.g. creation, party, event). This is an open list; new primaryReferentTypes may be registered.
structuralType	1	The primary structuralType of a referent. For creations, there are four mutually exclusive creationStructuralTypes (physical, digital, performance, abstraction) that allow classification according to overall form. Where structuralTypes may be contained within one another, the referent's structuralType is defined by the overall form [e.g. a CD (physical) may contain files (digital) which contain recordings of performances of songs (abstractions)], and elements of content can be further classified if necessary under referentType. For parties there are three mutually exclusive partyStructuralTypes (person, animal, organization). These lists are closed.
mode	0-n	For creations only, the principal sensory mode(s) by which a referent is intended to be perceived (audio, visual, tangible, olfactory, tasteable, none). Mode only identifies the principal intended modes of perception. This list is closed.
character	0-n	For creations only, a fundamental form of communication in which the content of a referent is expressed. There are four values: music, language, image, other. This list is closed.

referentType	0-n	Specification of type(s) of referent for parties: author, book publisher, library, university, film studio. It is typically described by creationType, which may be extended as needed to include format and genre elements (for example: audio file, book, PDF). This is an open list; new referentTypes may be registered.
linkedCreation	0-n	For creations only. Another creation with which a referentCreation is associated. This element contains a creationRoleToCreation element, which is an open list for which new allowed values may be registered.
linkedParty	0-n	For parties only. Another party with which a referentParty is associated. This element contains a partyRoleToParty element, which is an open list for which new allowed values may be registered.
principalAgent	0-n	For creations only, the entity or entities principally responsible for the creation or publication of the referent. This element contains an agentRole element which specifies the particular role played (for example: Creator, Author, BookPublisher). This is an open list for which new allowed values may be registered.
dateOfBirthOrFormation	0-1	For parties only, the date of birth (for an individual or animal) or formation (for an organization) of the referentParty.
dateOfDeathOrDissolution	0-1	For parties only, the date of death (for an individual or animal) or dissolution (for an organization) of the referentParty.
associatedTerritory	0-n	For parties only, a territory with which the referentParty is associated (for example, a territory of birth, nationality or residence). The allowed value list is the ISO 3166a2 territory code list.
registrationAuthorityCode	1	Code assigned to denote the name of the agency (authorized by the ISO 26324 Registration Authority) that issued this DOI name.
issueDate	1	Date when this DOI name was issued.
issueNumber	0-1	Number or other designation associated with the specific version of the DOI Kernel Metadata Declaration

Table 6.1: The basic elements of the IDF's Metadata Kernel.

6.3 DOIs and Watermarking

With this introduction to the DOI system it is clear that it has points in common with other watermarking techniques we described earlier in this thesis. The whole DOI system can be equated to a Zero-Watermarking technique. Both of them are methods of protecting a document's paternity without altering anything about it, instead opting to store information related to the text on a service provided by a third party. The proposed way of integrating our watermarking methods and tools with the DOI system is to include in the DOIs metadata a string that represents the settings used to watermark the document using the add-on (without the password, of course). This string is constructed in a way that makes it easy to read and short but without compromising security, because a watermark can't be resolved without the correct password (which is not included in any way in the string). An apposite function in the add-on converts the string back to its settings and the user only has to insert the correct password and start the normal function to resolve the watermark. One of the major weaknesses of a zero-watermarking technique (the fact that data given to a third party can't be certified) is circumvented because the only person that can resolve the watermark with the settings of the string embedded in the metadata is the same person that compiled the metadata for the DOI. Advantages to this technique include security and scalability: the process is secure because every RA only lets the user who registered a DOI change its metadata, so the only person who can modify the string is the owner itself. This technique is also scalable because if multiple people watermark different parts of the document the process allows for multiple different string to be hidden in the same metadata.

6.3.1 Location of the string

An important role in this method is given to the location of our famous string inside the metadata schema. Because the DOI system was created without this watermarking method in mind, its definition doesn't allow for much room to hide non required data in. Ideally, the best spot to hide our string in would be the IDF's Kernel to make it available to every RA but, as stated before, the kernel contains only the minimum metadata categories to keep it simple but still descriptive so there is no official place to put our string in. Using annotations (XML's comments) we can hide our string in the tag for the name of an author, so as to keep the scalability properties of this method. Another possibility is including our string into a metadata tag that an RA has left appositely for miscellaneous purposes, like Crossref's "Custom Metadata". Since annotations are still stored by the RA and our resolving method only needs the string pasted and nothing more, the two methods of hiding are equivalent.

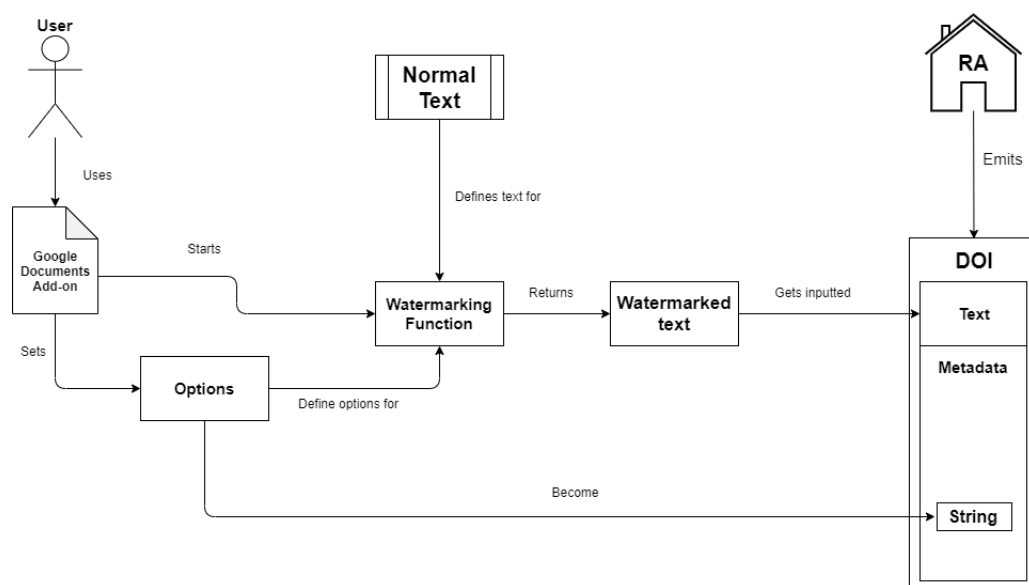


Figure 6.1: Visualization of the process of including a string generated by the add-on within a DOI's metadata.

In this picture we can see that a User is using the Google Documents add-on with the intent of watermarking a text, inserting it into a DOI and protecting it further by inserting the string formed by the options of the watermarking function into the DOI's metadata. In his first operation he takes a normal text and watermarks it by setting the options he wants in the add-on and using a password. The watermarking function returns a watermarked text and the options used can be turned into a string by a function inside the add-on. The second part of this process is actually registering a DOI with the document being the watermarked text and, when filling in the related metadata, inserting your options string in an annotation or in a "Custom Metadata".

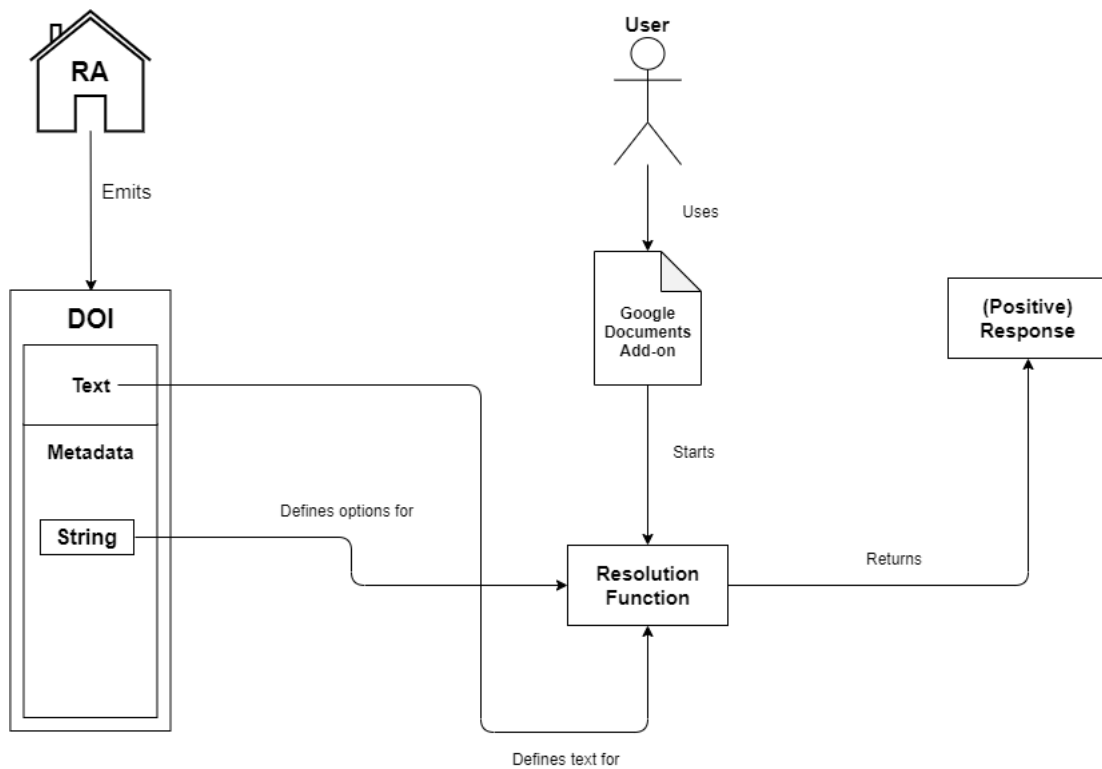


Figure 6.2: Visualization of the process of resolving a watermark with the options being provided by a string in a DOI's metadata.

This picture illustrates the inverse process to the first image. The same user wants to confirm his watermark from the DOI he registered. His first operation is using the text saved in the DOI to start a resolving watermarking function with the options recovered from the DOI's metadata that only he can access and the password that he used when watermarking the text. At this point the add-on returns a positive response if the watermark is recovered successfully or a negative one if it isn't.

Chapter 7

Conclusions

The protection of intellectual property of digital contents from plagiarism and unauthorized copy has become a challenging research problem, worsened by how easy it is selecting, copying and sharing other people's content. Text Watermarking techniques must, in this light, evolve and mutate in order to facilitate the user in the act of watermarking and discouraging potential attackers. An attack that only captures a small part of the text (a partial copy) is particularly insidious because watermarking techniques that watermark the whole text are not as effective as the techniques presented in this thesis, that watermark every letter of a document, often resulting in a full watermark being hidden in just a few words. All the techniques present in this thesis share a weakness to re-typing that is common in structural watermarking techniques. Grayscale-based watermarking and Space coloring-based watermarking are weak to re-coloring as well. The add-ons are a tool to combine these watermarking techniques in an automated and easy to use way, in order to bridge the gap between big corporations that watermark documents daily and new users that have never watermarked a document before. Our watermarked documents are also portable to other text editors, with some restrictions on fonts (only Times New Roman hides homoglyphs in every editor tested). Portability to presentation editors is instead limited to Homoglyph-based watermarked presentations in Times New Roman font. Simpler tools and safer techniques are required to keep the barrier of entry into the world of watermarking as low as possible and these add-ons are a step in that direction. The addition of the Digital Object Identifier support as a form of Zero watermarking to be used in conjunction with our techniques is another way in which text watermarking can evolve in the future.

A - Survey for grayscale perception

This appendix contains all the results to the second part of our survey on grayscale perception. This part was composed of six texts in total, and for every text participants were asked to report the first word they noticed was written with a different shade of gray. What follows is a look at all the texts from one to six with any special notes written in the caption.

First Text

Alice cominciava a sentirsi mortalmente stanca di sedere sul poggio, accanto sua sorella, senza far nulla: una o due volte aveva gittato lo sguardo libro che leggeva ma non c'erano immagini nè dialoghi, "e serve un libro," pensò Alice, "senza e dialoghi?"

Figure 1: Text #1. Non-bold, font size 12

Second text

Quando le gambe gli si furono sgranchite, Pinocchio cominciò a camminare da sè e correre per la stanza; finché, infilata porta di casa, saltò nella strada dette scappare. E il povero Geppetto correrli dietro senza poterlo raggiungere

Figure 2: Text #2. Bold, font size 12

Third text

Davvero non c'era troppo da meravigliarsi di ciò, nè Alice pensò che fosse cosa stravagante sentire parlare il Coniglio, quale diceva fra sè "Oimè! Oimè! ho fatto tardi!" (quando se lo rammentò in seguito s'accorse avrebbe dovuto meravigliarsene, ma allora le sembrò una assai naturale)

Figure 3: Text #3. Non-bold, font size 14

Fourth text

Alla fine, e per buona fortuna, capitò un carabiniere il quale, sentendo tutto quello schiamazzo, credendo si trattasse di puledro che avesse levata la mano al padrone, piantò coraggiosamente a gambe larghe in mezzo alla strada

Figure 4: Text #4. Bold, font size 14

Fifth text

L'idea era allarmante, sicché egli tiratosi fuori del letto andò brancolando verso la finestra. Fregò con manica della veste da camera sui vetri per veder qualche cosa; ma un gran che non arrivò a vedere. Vide nebbia fitta e sentì freddo indiavolato;

Figure 5: Text #5. Non-bold, font size 20

Sixth text

Più ci pensava, più s'imbrogliava; e si sforzava di non pensare, forte pensava. Lo spettro Marley lo turbava assai. Quante volte, dopo maturo esame, risolveva in mente sua che tutto era stato un sogno, subito, come una molla scattasse, il pensiero tornava indietro gli ripresentava stesso problema da sciogliere

Figure 6: Text #6. Bold, font size 20

These texts are created to highlight different situations regarding font size and boldness of the text. Following this is a series of tables that represent the results of the survey and are constructed as follows: the first column is composed of the words with letters of different shades of gray in the text, the second one refers to the colors with which the first column's word was painted and the last column represents the percentage of people that recognized the change in color with that hue **or less**. An hexadecimal number between square brackets represents a hue with every component (red, green, blue) being the number; for example [27] represents the color #272727. We used a second horizontal line to highlight the range of grayscale that is present in our add-ons, with every shade of gray above the double line being present in the add-ons. This is useful to show why we chose the hues we did and how they performed in the public's eye. The sixth text had a problem where two of the words with letters changed were the same so it's impossible to distinguish which one people voted for. This has been corrected in the final table.

Text 1		
ALICE	[07]	2,11%
COMINCIAVA	[0f]	4,23%
SENTIRSI	[17]	6,34%
MORTALMENTE	[1f]	6,34%
STANCA	[27]	7,75%
SEDERE	[2f]	8,45%
POGGIO	[37]	9,16%
ACCANTO	[3f]	9,16%
SORELLA	[47]	13,38%
SENZA	[4f]	22,54%
NULLA	[57]	45,77%
VOLTE	[5f]	80,99%
AVEVA	[67]	90,84%
GITTATO	[6f]	92,96%
SGUARDO	[77]	99,30%
LIBRO	[7f]	99,30%
LEGGEVA	[87]	100,00%
C'ERANO	[8f]	100,00%
IMAGINI	[97]	100,00%
DIALOGHI	[9f]	100,00%

Table 1: First text results.

Text 2		
QUANDO	[07]	0,00%
GAMBE	[0f]	0,00%
FURONO	[17]	0,00%
SGRANCHITE	[1f]	2,75%
PINOCCHIO	[27]	8,26%
COMINCIÒ	[2f]	9,17%
CAMMINARE	[37]	14,68%
CORRERE	[3f]	23,85%
STANZA	[47]	67,89%
FINCHÈ	[4f]	70,64%
INFILATA	[57]	90,83%
PORTA	[5f]	92,66%
CASA	[67]	97,25%
SALTÒ	[6f]	97,25%
NELLA	[77]	98,16%
STRADA	[7f]	100,00%
DÈTTE	[87]	100,00%
SCAPPARE	[8f]	100,00%
POVERO	[97]	100,00%
GEPPELTO	[9f]	100,00%

Table 2: Second text results.

Text 3		
DAVVERO	[07]	2,84%
TROPPO	[0f]	3,55%
MERAVIGLIARSI	[17]	6,38%
ALICE	[1f]	6,38%
PENSÒ	[27]	6,38%
FOSSE	[2f]	8,51%
STRAVAGANTE	[37]	21,99%
SENTIRE	[3f]	24,11%
PARLARE	[47]	46,10%
CONIGLIO	[4f]	67,38%
DICEVA	[57]	81,56%
FATTO	[5f]	93,62%
QUANDO	[67]	97,16%
RAMMENTÒ	[6f]	97,16%
SEGUIDO	[77]	97,16%
S'ACCORSE	[7f]	97,16%
AVREBBE	[87]	97,16%
DOVUTO	[8f]	97,16%
MERAVIGLIARSI	[97]	100,00%
ALLORA	[9f]	100,00%

Table 3: Third text results.

Text 4		
ALLA	[0f]	2,13%
FINE	[0f]	3,55%
BUONA	[17]	4,26%
FORTUNA	[1f]	4,26%
CAPITÒ	[27]	4,96%
CARABINIERE	[2f]	11,35%
SENTENDO	[37]	31,21%
TUTTO	[3f]	47,52%
QUELLO	[47]	65,96%
SCHIAMAZZO	[4f]	87,94%
CREDENDO	[57]	92,20%
TRATTASSE	[5f]	93,62%
PULEDRO	[67]	97,16%
AVESSE	[6f]	99,29%
LEVATA	[77]	100,00%
MANO	[7f]	100,00%
PADRONE	[87]	100,00%
PIANTÒ	[8f]	100,00%
CORAGGIOSAMENTE	[97]	100,00%
GAMBE	[9f]	100,00%

Table 4: Fourth text results.

Text 5		
IDEA	[07]	1,77%
ALLARMANTE	[0f]	2,66%
SICCHÉ	[17]	2,66%
TIRATOSI	[1f]	2,66%
LETTO	[27]	5,31%
ANDÒ	[2f]	13,28%
BRANCOLANDO	[37]	29,20%
VERSO	[3f]	29,20%
FINESTRA	[47]	67,26%
FREGÒ	[4f]	81,42%
MANICA	[57]	97,35%
VESTE	[5f]	97,35%
CAMERA	[67]	98,23%
VETRI	[6f]	98,23%
VEDER	[77]	98,23%
QUALCHE	[7f]	99,12%
COSA	[87]	100,00%
GRAN	[8f]	100,00%
ARRIVÒ	[97]	100,00%
VEDERE	[9f]	100,00%

Table 5: Fifth text results.

Text 6		
PENSAVA	[07]	16,08%
IMBROGLIAVA	[0f]	16,78%
SFORZAVA	[17]	17,48%
PENSARE	[1f]	19,58%
FORTE	[27]	27,27%
PENSAVA	[2f]	43,36%
SPETTRO	[37]	63,64%
MARLEY	[3f]	79,02%
TURBAVA	[47]	86,01%
ASSAI	[4f]	93,71%
QUANTE	[57]	99,30%
VOLTE	[5f]	99,30%
DOPO	[67]	99,30%
MATURO	[6f]	100,00%
ESAME	[77]	100,00%
RISOLVEVA	[7f]	100,00%
MENTE	[87]	100,00%
STATO	[8f]	100,00%
SOGNO	[97]	100,00%
SCATTASSE	[9f]	100,00%

Table 6: Sixth text results.

A combined look at all the answers

For obvious reasons, this table won't contain a column with words associated to the different hues, leaving only two columns.

Combined Texts	
[07]	1,57%
[0f]	2,61%
[17]	3,79%
[1f]	4,57%
[27]	7,57%
[2f]	13,58%
[37]	26,76%
[3f]	34,33%
[47]	55,74%
[4f]	69,32%
[57]	83,29%
[5f]	92,56%
[67]	96,48%
[6f]	97,39%
[77]	98,83%
[7f]	99,22%
[87]	99,48%
[8f]	99,48%
[97]	100,00%
[9f]	100,00%

Table 7: All of the texts combined.

B - Add-ons code version

The code for both of the add-ons and the tests related to them were performed on the latest versions of the appropriate softwares as of November 2020. Results are subject to change.

Bibliography

- [1] I. Abramov, J. Gordon, O. Feldman, and A. Chavarga. Sex and vision ii: color appearance of monochromatic lights. *Biology of sex differences*, 3(1):21, 2012.
- [2] R. A. Alotaibi and L. A. Elrefaei. Improved capacity arabic text watermarking methods based on open word space. *Journal of King Saud University-Computer and Information Sciences*, 30(2):236–248, 2018.
- [3] T. Amano and D. Misaki. A feature calibration method for watermarking of document images. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, pages 91–94. IEEE, 1999.
- [4] M. J. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *International Workshop on Information Hiding*, pages 185–200. Springer, 2001.
- [5] S. S. Baawi, M. R. Mokhtar, and R. Sulaiman. Enhancement of text steganography technique using lempel-ziv-welch algorithm and two-letter word technique. In *International Conference of Reliable Information and Communication Technology*, pages 525–537. Springer, 2018.
- [6] A. K. Bhattacharjya and H. Ancin. Data embedding in text for a copier system. In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 2, pages 245–249. IEEE, 1999.
- [7] J. Brassil, S. Low, N. Maxemchuk, and L. O’Gorman. Hiding information in document images. In *Proc. Conf. Information Sciences and Systems (CISS-95)*, pages 482–489, 1995.
- [8] J. T. Brassil, S. Low, and N. F. Maxemchuk. Copyright protection for the electronic distribution of text documents. *Proceedings of the IEEE*, 87(7):1181–1196, 1999.
- [9] J. T. Brassil, S. Low, N. F. Maxemchuk, and L. O’Gorman. Electronic marking and identification techniques to discourage document copying. *IEEE Journal on Selected Areas in Communications*, 13(8):1495–1504, 1995.

-
- [10] D. Huang and H. Yan. Interword distance changes represented by sine waves for watermarking text images. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12):1237–1245, 2001.
- [11] R. Huddleston. *Introduction to the Grammar of English*. Cambridge University Press, 1984.
- [12] Z. Jalil and A. M. Mirza. A review of digital watermarking techniques for text documents. In *2009 International Conference on Information and Multimedia Technology*, pages 230–234. IEEE, 2009.
- [13] N. S. Kamaruddin, A. Kamsin, L. Y. Por, and H. Rahman. A review of text watermarking: theory, methods, and applications. *IEEE Access*, 6:8011–8028, 2018.
- [14] M. Kaur and K. Mahajan. An existential review on text watermarking techniques. *International Journal of Computer Applications*, 120(18), 2015.
- [15] B. Khosravi, B. Khosravi, B. Khosravi, and K. Nazarkardeh. A new method for pdf steganography in justified texts. *Journal of information security and applications*, 45:61–70, 2019.
- [16] Y.-W. Kim, K.-A. Moon, and I.-S. Oh. A text watermarking algorithm based on word classification and inter-word space statistics. In *ICDAR*, pages 775–779. Citeseer, 2003.
- [17] Y.-W. Kim and I.-S. Oh. Watermarking text document images using edge direction histograms. *Pattern Recognition Letters*, 25(11):1243–1251, 2004.
- [18] K. Lambrecht. A framework for the analysis of cleft constructions. *Linguistics*, 39(3):463–516, 2001.
- [19] R.-z. Liu and T.-n. Tan. Svd based digital watermarking method. *Acta Electronica Sinica*, 29(2):168–171, 2001.
- [20] S. H. Low, N. F. Maxemchuk, J. T. Brassil, and L. O’Gorman. Document marking and identification using both line and word shifting. In *Proceedings of INFOCOM’95*, volume 2, pages 853–860. IEEE, 1995.
- [21] S. H. Low, N. F. Maxemchuk, and A. M. Lapone. Document identification for copyright protection using centroid detection. *IEEE Transactions on Communications*, 46(3):372–383, 1998.
- [22] H. M. Meral, B. Sankur, A. S. Özsoy, T. Güngör, and E. Sevinç. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23(1):107–125, 2009.
- [23] N. Mir. Copyright for web content using invisible text watermarking. *Computers in Human Behavior*, 30:648–653, 2014.

- [24] L. Y. Por, K. Wong, and K. O. Chee. Unispach: A text-based data hiding method using unicode space characters. *Journal of Systems and Software*, 85(5):1075–1082, 2012.
- [25] S. G. Rizzo, F. Bertini, and D. Montesi. Fine-grain watermarking for intellectual property protection. *EURASIP Journal on Information Security*, 2019(1):10, 2019.
- [26] S. G. Rizzo, F. Bertini, D. Montesi, and C. Stomeo. Text watermarking in social media. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 208–211, 2017.
- [27] M. Taleby Ahvanooy, H. Dana Mazraeh, and S. H. Tabasi. An innovative technique for web text watermarking (aitw). *Information Security Journal: A Global Perspective*, 25(4-6):191–196, 2016.
- [28] M. Topkara, C. M. Taskiran, and E. J. Delp III. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452. International Society for Optics and Photonics, 2005.
- [29] U. Topkara, M. Topkara, and M. J. Atallah. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174, 2006.
- [30] O. Vybornova and B. Macq. A method of text watermarking using presuppositions. In *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, page 65051R. International Society for Optics and Photonics, 2007.
- [31] O. Vybornova and B. Macq. Natural language watermarking and robust hashing based on presuppositional analysis. In *2007 IEEE International Conference on Information Reuse and Integration*, pages 177–182. IEEE, 2007.
- [32] X. Wang. Digital watermarking research based on text. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*, pages 433–436. IEEE, 2013.
- [33] Y. Zhou and W. Jin. A novel image zero-watermarking scheme based on dwt-svd. In *2011 International Conference on Multimedia Technology*, pages 2873–2876. IEEE, 2011.