EURASIP Journal on Information Security

RESEARCH Open Access

Fine-grain watermarking for intellectual property protection

Stefano Giovanni Rizzo[1*], Flavio Bertini[2] and Danilo Montesi[2]

Abstract

The current online digital world, consisting of thousands of newspapers, blogs, social media, and cloud file sharing services, is providing easy and unlimited access to a large treasure of text contents. Making copies of these text contents is simple and virtually costless. As a result, producers and owners of text content are interested in the protection of their intellectual property (IP) rights. Digital watermarking has become crucially important in the protection of digital contents. Out of all, text watermarking poses many challenges, since text is characterized by a low capacity to embed a watermark and allows only a restricted number of alternative syntactic and semantic permutations. This becomes even harder when authors want to protect not just a whole book or article, but each single sentence or paragraph, a problem well known to copyright law. In this paper, we present a fine-grain text watermarking method that protects even small portions of the digital content. The core method is based on homoglyph characters substitution for latin symbols and whitespaces. It allows to produce a watermarked version of the original text, preserving the anonymity of the users according to the right to privacy. In particular, the embedding and extraction algorithms allow to continuously protect the watermark through the whole document in a fine-grain fashion. It ensures visual indistinguishability and length preservation, meaning that it does not cause overhead to the original document, and it is robust to the copy and past of small excerpts of the text. We use a real dataset of 1.8 million New York articles to evaluate our method. We evaluate and compare the robustness against common attacks, and we propose a new measure for partial copy and paste robustness. The results show the effectiveness of our approach providing an average length of 101 characters needed to embed the watermark and allowing to protect paragraph-long excerpt or smaller the 94.5% of the times.

Keywords: Digital text watermarking, Unicode characters, Copyright protection, Copyright enforcement, Tampering detection

1 Introduction

The last decades are characterized by the easy availability of millions upon millions of digital contents that meet several kind of users' needs both in professional activities and social interactions. An important reason for the proliferation of digital contents among users is the increase in the usage of online communication platforms, like websites, social media, and cloud file sharing services, to name a few. All these platforms have introduced changes in the user habits with respect to digital contents by increasing the copying and sharing of text, audio, images, and video, namely digital contents [1].

While the current digital technologies facilitate the copy and sharing of these digital contents, this is often an unattributed copy of others' work, resulting in a misappropriation of their intellectual property. In several contexts, such as for online newspapers and blogs, the contents' owners have solid interests in protecting their IP rights, in order to preserve their business. In particular, there are different illicit actions concerning these digital contents, like tampering, forgery, theft, and, more simply, making a copy of both the whole content or part of it. The problem is more meaningful with text since it is the main carrier of information (e.g., online news articles, scientific articles, e-mail, product catalogs) while being more prone to full or partial misappropriation. Moreover, findings in [2] demonstrate that the copy and paste function significantly increase the plagiarism attitude of the

*Correspondence: strizzo@hbku.edu.qa
[1]Qatar Computing Research Institute (QCRI) HBKU, Doha, Qatar
Full list of author information is available at the end of the article

users, that is 13% more likely when copy and paste is possible.

Up until now, the techniques used to protect IP and prevent illegal use, like digital rights management (DRM) and technical protection measures (TPM), also prevent legal or permitted uses of the copyrighted digital contents [3, 4], by representing a strong limit in terms of freedom of information and expression of the user. In [5], Lai and Graber point out the complexity to reach a fair balance among IP rights and freedom of expression and information. In particular, the authors compare the need of the IP owners and privacy and freedom of choice of the users. These classic digital rights protection techniques are based on hardware or firmware supports and proprietary encodings that prevent the user of making copies, reading unauthorized copies, or reproducing it on unauthorized supports. The total prevention of copy through cryptography and dedicated supports, such as the content scrambling system for DVD protection [6], reduces the ability of sharing and distributing the creative content.

In order to overcome the limits of the classic digital rights protection techniques and meet the various needs in IP protection field, different approaches are developed [7]. For instance, while *steganography* provides techniques to hide new information into the original digital content, *cryptography* produces an unreadable version of the document by applying a kind of permutation or substitution to the original information. *Watermarking* is the most balanced technique for sharing not obfuscated information while preserving the copyright [8, 9]. In particular, it ensures copyright protection by applying a mark to the original digital content, without showing such mark to the readers. Watermarking methods can be applied in innumerable contexts, such as identifying unauthorized users, establishing the authorship of a digital content, monitoring the broadcasting process, and distrusting a tampered digital content. Up to an acceptable distortion, watermarking can be also adopted to protect dynamically generated contents from databases [10].

Watermarking an intellectual property allows the free sharing of a digital content, while binding the artifact with the original author. When the authorship of a digital content is misattributed, the original author can claim his/her authorship or copyright. In this scenario, the author can extract and show the digital watermark as an irrefutable proof of authorship, avoiding costs and efforts of more elaborated and timestamped evidence. At the same time, the watermark exclude the possibility of unintentional plagiarism, in the case when the malicious user appeal to the lack of originality of the work, that may have lead to the unrelated creation of the same or very similar content.

1.1 Problem statement

Out of all digital content watermarking techniques, we focus on text watermarking. The reason behind our choice is that textual information represent one of the largest bunch of digital contents that people can daily share and explore online, for instance, online newspaper articles, manuals and guides, social media, and microblogging posts, to name a few.

Furthermore, text messages increase daily and are more often used for commerce, mobile banking, and government communications. In comparison with watermarking techniques for other digital contents, text watermarking is the most difficult task, presenting several challenges mainly because text is not noise-tolerant.

In particular, a text watermarking algorithm must work with some additional constraints, as short-length message, a limited set of transformations in order to preserve readability and a restricted number of alternative syntactic and semantic permutations [11]. In fact, one of the main problems concerning textual content, even short text message, is authorship verification, that is, to verify if a text has been actually produced by a given author, as he/she claims. If we exclude that a third-party guarantor is involved in the verification process, such as an IPR database with certified timestamps of deposited contents (also known as zero-watermarking), then some author-dependent data must be embedded in the text content, such as a unique code derived from the author's secret key.

Another peculiarity of text in the context of unauthorized copy is that, unlike images, any meaningful excerpt, like a paragraph, could be copied, and it is difficult to predict which one. While it is true that in the case of images, some partial cropping is often applied before unauthorized re-sharing, the unauthorized copy will still account for an important percentage of the original image (with some exceptions, for example in aerial photography). Instead, in the context of text, it is very common to copy only few sentences, which may not be subsequent in the original document and may account for a very small percentage of it (e.g., one paragraph from a book). This can be seen as a special case of a deletion attack, in which most of the watermarked document is deleted and only some paragraphs or sentences are left, motivating the need of a *fine-grain* approach able to embed the watermark in as many sub-portions of text as possible.

The concept of a *fine-grain* protection of text content is well known in copyright law: it is common to claim intellectual property rights on small portions of larger works, and there is a vast literature involving several trials and studies [12] trying to define at which fine-grain level an intellectual work can be copyrighted. This known scenario however has not been addressed so far in the text watermarking literature. It also makes the text length constraint even stricter, because the watermark has to be embedded in smaller parts of the text content. Additional issues arise if we must be able to verify a copied text that is straddling partially two watermarked portions.

1.2 Contribution

In this paper, we propose a structural text watermarking method for intellectual property protection[1]. The method protects the whole document as well as smaller excerpt of it, up to a minimum size of excerpt that depends on the specific characters of the text. Nevertheless, it is fair with respect to the concerns regarding communicative freedom and privacy of the users, without altering the content of the text or embedding explicit author-related data. More precisely, the proposed method is *invisible* and *content-preserving* and belongs to the *fragile* and *non-blind* classes. In practice, it is able to embed a password-based watermark without altering the content and preserving the length, ensuring data protection against the copy and paste of even small excerpt of text.

The embedding process consists of two phases. In the first one, the watermark is generated by applying a hash function that combines the user (author) password and the structural characteristics of the text. In the second phase, that is the core of our methodology, the watermark is embedded into the original text by exploiting homoglyph characters. Homoglyph characters, as symbols, numbers, and letters, look very similar on the screen and in print; nevertheless, their low-level encoding is completely different. More precisely, the Unicode confusable characters, namely the homoglyph characters, are listed by Unicode Consortium and look confusingly similar from each others [13]. In practice, we replace a subset of characters of the original text with an indistinguishable latin homoglyph symbol, with a substitution process driven by the watermark bits sequence. The password allows to verify the authorship since only the actual author of the text can correctly regenerate the watermark.

The proposed method has the following four new significant features:

1 It leaves visually indistinguishable original text, in other words, the watermark is not noticeable by the user.
2 The length of the original text is preserved, no matter how short is it.
3 It can be continuously applied to small excerpts of a longer text, protecting a document at a fine-grain level against the copy and paste of text portions
4 It allows to cryptographically bind each text excerpt to the original source document.

The visually indistinguishable features strongly depend on the font used. However, we will show in the evaluation section how the homoglyph characters allow to cover the most used font families. The length preservation feature is quite complex to ensure when the algorithm operates on short texts. The proposed method is able to embed a watermark while preserving the text length with very short texts (theoretically a minimum of 22 symbols).

The minimum length depends on the text content, as only a subset of characters can be substituted to embed the data. In order to establish the minimum length requirement on real text examples, we provide the results of an extensive experiments on 1.8 million of New York Times articles [14]. The results show that, on average, 101 characters are sufficient to embed the watermark preserving the length and visible aspects of the original text. Despite paragraphs can be very short or having few confusable symbols that can be replaced, the method allows to watermark very short excerpt, shorter than a single paragraph of New York Times articles for the 94.5% of the times, meaning that it ensures data protection when only a single paragraph or a smaller excerpt of it is copied and pasted. The combination of these two features allows to use our text watermarking method in several new contexts, for instance, word and pdf documents, online newspaper articles, short message communications, e-mails, microblogging platforms, and social networks posts.

The fine-grain watermarking method of the proposed approach allows for the first time to protect small excerpt of text, by repeatedly embedding the watermark across the document. This is made possible by the short length requirements of the approach and has two valuable consequences: (i) it is possible to extract the watermark even when it is "broken" between two watermark sequences, and (ii) each excerpt is bound to the source document and can be traced back to it.

In order to evaluate the fine-grain property of the method and compare it with current methods, we propose also a novel measure for the robustness to partial copy and paste.

The rest of the paper is organized as follows. In Section 2, we provide a small background in watermarking, in order to classify the methods and show the features usually required to a watermarking algorithm. In Section 3, we review the literature works related to text watermarking methods. In Section 4, we describe our text watermarking method, including watermark generation, embedding, extraction, and authorship verification. We discuss the evaluations of our method in Section 5. Some concluding remarks are made in Section 6.

2 Background in watermarking

In this section, we provide a small background in watermarking methods. This is important as it will help in understanding the reasons behind the design of our method.

In accordance with the literature [15], watermarking methods can be categorized as follows:

- *Readable or detectable*—The watermarking is readable if the user can clearly read it. It is instead detectable if a detection function can be used to check if a watermark exists or not, but it cannot be read.
- *Visible or invisible*—A visible watermarking is visually perceptible by the user. Contrary, the watermarking is invisible if it is hidden in the original digital content and it does not noticeable by the user. A visible watermark may be not readable, that is, 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 belongs to non-blind category.
- *Simple or multiple*—If a watermark can be applied only once the watermarking is simple. Otherwise, a multiple watermarking can be embedded more than one time without affecting the whole process.
- *Fragile, semi-fragile, and robust*—A fragile watermark is detectable and can be altered or erased; thus, it is used for integrity authentication. On the flip side, 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.

In [16], the researchers identify several features usually required to a watermarking method. *Verifiability* represents the ability to irrefutably prove the ownership of the digital content. *Data payload* represents the maximum number of bits of extra information that can be embedded in the original digital content. *Robustness* represents the ability to resist to processing operations and attacks, as *security* is the capacity to not be altered or removed without having full knowledge of the watermark or the embedding process. Finally, *computational cost* is the cost required in embedding and extraction process.

Out of all digital content watermarking techniques, text watermarking is the most challenging. Text has a low embedding bandwidth and allows only a restricted number of alternative syntactic and semantic permutations. Text watermarking algorithms can be classified as follows:

- *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*—Firstly, 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; in other words, it cannot be considered a pure text watermarking method. However, it has some interesting features, as length preservation and language independent.
- *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, and even spelling and grammar rules to permute the contents and embed the watermark.
- *Structural techniques*—These methods exploit double letter occurrences, word shift and line shift encoding, and Unicode standard to embed the watermark. They are one of the most recent methodologies with which the original text is not altered.

3 Related works

The text watermarking approaches with actual watermark embedding are usually classified into three main categories [15, 17]: image-based, syntactic, and semantic. In this categorization, the zero-watermarking approaches are often not considered as no watermark is actually applied; however, this alternative solution is getting more attention lately and it is important to understand the difference between the zero-watermarking and content-preserving methods. A recent survey [18] considers instead the structural, linguistic, and statistics as the three main categories. After highlighting the core ideas, advantages, and disadvantages of the mentioned approaches, we will focus on the structural methods. Unicode-based methods such as the proposed method belong to this latter class.

3.1 Zero-watermarking

The first important dichotomy in text watermarking works, and watermarking in general is the one between zero-watermarking techniques and the more common "non-zero" or embedding watermarking techniques. Zero-watermarking aims at extracting characterizing information from a digital content, for example, from a picture or a song, and then store this information into an Intellectual Property Right (IPR) database [19]. Embedding watermarking instead aim at embedding in the digital content a payload related to the author or to the content itself (e.g., the author name, a company logo, the keyed hash of the content with the author's password).

In the zero-watermarking process, no actual watermark is applied to the content or embedded in the content, which is left untouched. The association between the content and the author does not rely on the watermark, but on the proof from a trusted authority.

Copious literature has emerged in recent years proposing zero-watermarking techniques on text [20–23] as an alternative solution to cope with the two orthogonal challenges of text watermarking: hiding information in small, unnoisy data and keeping the content unaltered to the human eye. In zero-watermarking, this is addressed by avoiding the embedding of any watermark whatsoever.

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 [24]. The clear advantages of dimensionality reduction are that the performance for similar content search is improved and the storage needed on the IPR database is reduced. However, in terms of security and IP protection, the same result can be obtained by simply storing the original content as it is, without extraction, and then applying a similarity technique (like the SSIM method for images [25] or structure-level, word-level, and character-level similarities in text [26]) to efficiently identify duplicates when the format has been altered.

A collateral shortcoming of zero-watermarking is that the identity of the author of watermarked content must be preventively registered on a third-party authority, leading to privacy issues.

3.2 Image-based methods

The 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-1990s [27, 28].

In this approach, a printed text is first scanned as an image, or as a screenshot in the case of digital text, and then a watermark is applied on this image. For example, in grayscale images of text document, the watermark payload is embedded by tuning the luminance of pixels accordingly to the watermark data [29] or by modifying the edge direction histograms to carry the watermark signal [30]. A robust embedding can be obtained by slightly shifting the text elements horizontally or vertically: a text element can be a word, to which a shift of few pixels to the right or to the left can embed an information, or can be a text line or block, shifter up and down with the same purpose [31, 32]. Similar results can be obtained by altering the spaces between words to encode the watermark data [33, 34]. Other methods are based on the alteration of single characters [31], some focus on smaller detail such as strokes and serifs of the characters and work by prolonging them [35], and others, more simply, alter the character in their size by change the scale depending on the watermark content [36].

There are two important shortcomings of image-based methods. The first is that text must be shared as an image, in an image file format (e.g., PNG, JPEG, or TIFF), or as printed paper or through fax machines, which is nowdays less practical and not very common. The second is that text can be still reconverted to plain text by manual retyping or using an OCR software, leaving behind in the process any trace of the watermark.

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.

3.3 Syntactic methods

Syntactic methods for text watermarking works 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 like clefting, passivization, or activization are applied in order to encode the watermark bits [37]. Clefting is the process of transforming a simple sentence into a more, unnecessarly complex one, for example, the simple sentence *I like champagne* can be transformed into *champagne is what I like* (the *what* clefting) or into *it is champagne that I like* (the *it* clefting) [38]. Passivization is the transformation from the active to the passive form of verbs as from, 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 that 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" [39].

The low embedding capacity, that is 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 comes from the alteration of the content. The assumption that different syntactic forms have the same meaning is not always true [40]: in the previous example, *Tom kicked the bucket* has an idiomatic interpretation, while its passive form has only a literal one.

3.4 Semantic methods

By exploiting the similarity of the meaning of different words, it is possible to replace words with their synonyms [41]. 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 approach [42] to obtain an overall higher embedding capacity.

Other semantic techniques work on the sentence level semantic, leveraging the implicit *presuppositions* of each sentence [43, 44]. 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. Like in the case of syntactic methods, the author's content can be strongly altered in order to embed the watermark. Also, they depend on the language and on the correctness of written text.

3.5 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. They have more recently emerged 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 paragraphs, or by filling an empty line, relatively long payloads has been hidden in Microsoft Word documents [45].

A similar technique based on different Unicode whitespaces 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 [46].

A more recent method uses instead multiple ASCII whitespaces to embed a covert message [47] for PDF steganography. The techniques 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 provided as zero-width whitespaces. These symbols, together with whitespaces, have been exploited to watermark HTML pages [48, 49] and more generally to hide hidden messages in the text [50].

As mentioned earlier, these methods have the important advantage of keeping the original content unaltered, but 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 uses consecutive whitespaces and whitespaces before or after the whole text, because it has been shown that many digital platforms and social media automatically remove them [51]. This can also happen through selection for copy and paste: selection may easily exclude the white portion where the watermark is embedded.

Apart from whitespaces, homoglyphs ad invisible characters, some image based where lines or words are slightly shifted without altering the text content [52, 53] have been also considered as structural methods [54].

4 Our approach

The proposed method for text watermarking can be categorized as a structural method; therefore, it preserves both the appearance and the content without converting the text into image and without the need of a third-party IPR database. Our approach compute a watermark depending on the original text and a password, both given by the author, then it replaces the symbols and whitespaces with visually equivalent symbols, according to the watermark binary data.

Following the watermarking features and categorizations so far presented in related works, our approach is:

- Invisible (totally or partially): The readable content is kept, while the symbols can be replaced by their *homoglyph*. The changes applied to the text are not noticeable to the human reader, to a certain degree of analysis. Visibility depends strictly on the font used: some fonts may represent the similar symbol in the same exact way, thus making it impossible to visually distinguish the two, others may implement slightly different strokes or serifs or, in the case of whitespace, slightly different widths.
- Detectable, fragile, and non-blind: As for all the content-preserving watermarks in text, the content can be simply re-typed thus losing the embedded watermark. By knowing the set of homoglyphs used, it is also possible to detect the watermark and expose the embedded data. However, this data would be useless without the author's password and original complete text needed for verification (see Section 4.5), so the attacker cannot prove the ownership using the extracted data. The requirement for the original complete text makes our approach non-blind, but it hold only in the cases when an excerpt of text has been copied, and not the whole text.
- Content and length preserving: The whole embedding process works by replacing the symbols with visually indistinguishable homoglyphs; therefore, everything, from letters to sentences, is preserved. An important feature is that our method is length preserving: the related works on structural watermarking with Unicode add multiple invisible symbols. This results in an overhead in the original

text and can be an issue in context where the number of characters is limited. Moreover, this can make the watermark detection simpler by comparing the number of readable symbols with the size of the text. On the contrary, our approach preserves also the length of the text because, when a substitution happens, one symbols is replaced with another symbol, keeping the same total number of symbols.

Besides its general properties, our method show several advantages in comparison with other non-zero-watermarking techniques. It is more efficient because it implements a mapping symbol-to-symbol which is much simpler than NLP techniques where a syntax tree must be built or complex semantic analysis must be carried.

From the perspective of the range of application, it can be applied to most software and web platforms, because it does not depend on a particular file format [45] or markup language [48], but it is only dependent on Unicode support. The method does not use consequent whitespaces or rely on specific invisible symbols [48, 55]. This is a great advantage when the text is posted online, as most online platforms apply several filters to the incoming text. Moreover, it has been shown the set of symbols used in our method can pass through several instant messengers, webmail services, and social media without getting filtered, allowing robust online applications [51].

Lastly, it is more robust against the partial selection in *copy and paste*, because unlike other structural methods [45, 48] where the whitespaces symbols are appended between new lines, at the beginning or at the end of the text, in our approach, the watermark is embedded across all the text and is part of the text. This means that it is much more difficult to avoid copying also the watermark in the process.

4.1 Unicode confusables

The Unicode standard consists of more than 120 thousand symbols, among which some are very similar or totally indistinguishable. Despite these symbols have a different numerical code and different Unicode name, thus a separate purpose or meaning, the fonts with Unicode support depict them with the same aspect. These symbols are often called *homoglyphs*.

This similarity between symbols is a well-known security threat, because they may be used to deceive users into clicking on fake links or may avoid spam filters by altering the words with spurious symbols. For this reason, the Unicode Consortium maintain a list of the above *confusable* symbols [13], which is publicly available[2].

In our approach, we exploit the similarity of Unicode homoglyphs to seamlessly replace them accordingly to the bits of a payload. More specifically, the payload is the watermark of the text. To better clarify the approach, let us suppose that each symbol of the alphabet has a "common" version and a much less usual clone, with a different underneath Unicode value. Then, we could encode a binary string by using the common symbols to express "0" and the clone symbols to express "1." In this way, we would be able to encode 1 bit for each symbol in the text. After the encoding, we can also decode the binary string by looking at the Unicode: if the decoder finds a common symbol will produce a "0" ; otherwise, if the clone symbol has been used, it will produce a "1."

The real scenario is different from the above example, because only some symbols have a related homoglyph. We identified these duplicate symbols for some letters of the latin alphabet and for some punctuation in Table 1. Moreover, in Table 2, we consider the sets of whitespaces that the Unicode standard provides as homoglyphs, thus using them to encode bits.

In order to find out the most similar symbols, we tested their homoglyphs under the most used font families in modern desktop and web applications, obtaining imperceptible differences in most used sans-serif fonts. The reader can find the evaluation results on fonts in Section 5.

4.2 Watermark generation with password

Before going into details of the watermark embedding method, we first describe how the watermark is generated. We want the watermark to be a function of the original text and the author's identity so that we can (i)

Table 1 Encoding bits for latin letters and punctuation symbols

	Bit 0	Bit 1
Symbol	Original code	Duplicate code
-	0x002d	0x2010
;	0x003b	0x037e
C	0x0043	0x216d
D	0x0044	0x216e
K	0x004b	0x212a
L	0x004c	0x216c
M	0x004d	0x216f
V	0x0056	0x2164
X	0x0058	0x2169
c	0x0063	0x217d
d	0x0064	0x217e
i	0x0069	0x2170
j	0x006a	0x0458
l	0x006c	0x217c
v	0x0076	0x2174
x	0x0078	0x2179

Subset of confusable symbols used to encode the watermark. Original and duplicate code is shown for each symbol

Table 2 Encoding bits for whitespace symbols

Whitespace	Bits	Unicode
Space	000	0x0020
En quad	001	0x2000
Three-per-em space	010	0x2004
Four-per-em space	011	0x2005
Punctuation space	100	0x2008
Thin space	101	0x2009
Narrow no-break space	110	0x202f
Medium mathematical space	111	0x205f

Encoding for whitespace symbols and related Unicode value. In order to encode 3 bits in a single whitespace, 8 different whitespaces are used in total

prove that the watermark is related to the original text and (ii) assure that only the author who generated the watermark can verify it. These requirements can be satisfied by a cryptographic *keyed* hash function such as SipHash [56]. SipHash is a function that takes in input a variable-length message and a secret key and produces in output a binary string of a fixed size. This binary string is a message authentication code (MAC) in message exchange: only using the same secret key it is possible to recreate the same MAC and authenticate the message. We use the MAC as a watermark in order to add an additional security layer to the watermarking schema: while it may be possible for attackers to extract the watermark, they will not be able to prove the authorship of that watermark.

Other *non-keyed* hash function can be also used for the same purpose [57]; however, SipHash is specifically designed to securely authenticate short messages producing a small but robust code of 64 bit. This is particularly suitable in our context of fine-grain watermarking, as we want the watermark to be as small as possible to embed it in small excerpt of a text while retaining cryptographic robustness. Nevertheless, our embedding approach does not depend on a specific hash function or, more generally, watermark generation method; for this reason, in Section 5, we evaluate the embedding method using other hash functions of different MAC length.

The watermark generation is shown in Fig. 1, where the cryptographic keyed hash function takes in input the original text *t* and the password *k* producing in output the 64 bit string, representing the watermark. Only who owns the password used for generating the watermark and the original text will be able to prove the authorship by replicating the generation process. The robustness of this authorship verification process is ensured by the strength of the hash function.

4.3 Unicode watermark embedding

The watermark, generated using the keyed hash function, is then embedded through symbols replacement, following the proposed approach in Algorithm 1. By replacing original symbols and whitespaces with identical or almost identical Unicode symbols, the algorithm embeds the watermark binary string producing a new text which is indistinguishable from the original.

More specifically, the algorithm scans the text starting from the first character, looking for a confusable symbol, that is, a symbol or whitespace that has a duplicate in the Unicode standard. The mapping between confusable symbols and watermarked bit is shown for clarity in Fig. 2.

When a confusable symbol is found, this is replaced with its homoglyph or kept depending on the next bit of the watermark, starting from the leftmost bit. The Unicode codes of symbols used when the bit is 0 and when the bit is 1 are shown in Table 1. It must be noted that the usage of original symbols to embed the bit 0 and duplicate to

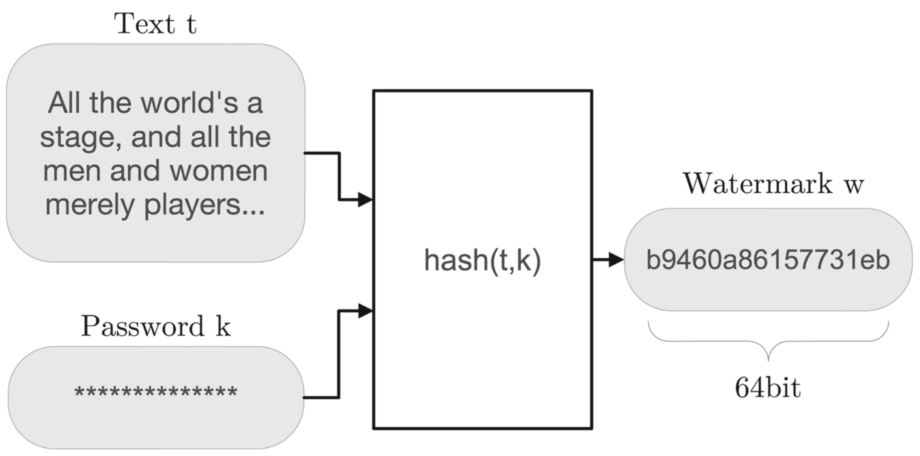


Fig. 1 Generation of the watermark bits. Given the original text *t* and a secret password *k*, the SipHash function generates a cryptographic hash, representing the watermark *w* to be embedded

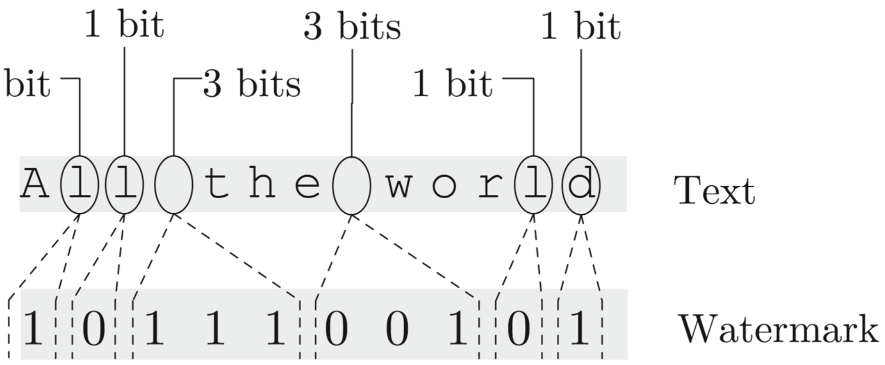


Fig. 2 Embedding watermark's bits by replacing confusable symbols. Only the symbols with a related *duplicate* in Tables 1 and 2 are used to embed the watermark bits: one bit can be embedded on latin letters, and three bits can be embedded on space characters

Algorithm 1 WATERMARK EMBEDDING

```
/* Text string of $n$ Unicode characters $c$ */
$T = \{c_1, ..., c_n\}$
/* List of confusable original symbols */
$Originals = \{U+002c, U+002d, U+002e, ...\}$
/* List of confusable duplicate symbols */
$Duplicates = \{U+a4f9, U+2010, U+a4f8, ...\}$
/* List of confusable white spaces */
$Spaces = \{U+0020, U+2002, U+2005, ...\}$
/* List of all confusables*/
$Confusables = Originals \cup Spaces$
$W = [b_1, ..., b_{64}]$ /* Watermark bit array */
$GetDuplicate : Originals \rightarrow Duplicates$
$GetSpace : \{000, ..., 111\} \rightarrow Spaces$
$WT = [\,]$ /* Watermarked text */
for all $c \in T$ do
    if $c \in Confusables \wedge |W| \neq 0$ then
        if $c \in Spaces$ then
            $bits = Pop(W, 3)$
            $lshift(bits, 3 - len(bits))$
            $c = GetSpace(bits)$
        else
            $bit = Pop(W, 1)$
            if $bit$=1 then
                $c = GetDuplicate(c)$
            end if
        end if
    end if
    $Append(WT, c)$
end for
return $WT$
```

embed the bit 1 is completely arbitrary and for the sake of simplicity. It is possible to make the opposite association or to choose a more elaborate scheme, for example, one in which the bit 1 is represented with the duplicate code for some symbols and with original code for others.

Similarly, when a whitespace is found, this is replaced with one of the 7 whitespaces in Table 2 or kept depending on the next 3 bits of the watermark. Specifically, it is kept when the bits are 000 while it is replaced with another whitespace for any other 3 bits combination. As for the symbols, the whitespace association table can be rewritten arbitrarily making a custom, less predictable scheme of embedding.

4.4 Unicode watermark extraction

The embedded watermark is invisible to the reader of the watermarked text, however can be detected in a technical analysis of the symbol encoding, noticing that unusual symbols have been used. Knowing the embedding algorithm and the mapping between confusable and bits, it is possible to also extract the watermark. The extraction algorithm (Algorithm 2) is in fact the opposite process of the embedding. As in the embedding process, it scans the watermarked text for confusable symbols. When a confusable symbol or whitespace is found, the association table (Table 1 or Table 2) is used to find the corresponding bit (or bits in the case of whitespaces). For every confusable symbol, the algorithm output in sequence the bits of the watermark, from the leftmost (most significant bit) to the rightmost (least significant bit).

4.5 Authorship verification

Suppose that an unattributed copy of a text, carrying a watermark, is shared by an attacker. The original author's will is to claim his/her IP rights on the watermarked text, but the attacker too may try to claim the authorship. The goal of the verification mechanism is to ensure that only the original author, who generated and embedded the watermark in the first place, will be able to prove the authorship.

In our approach, this goal is achieved through the regeneration of the same SipHash MAC. It is based on the assumption that, with limited computational resources, the attacker will not be able to generate the same MAC without having the password used in the generation phase. The verification process consists of the following 3 steps, also illustrated in Fig. 3:

Algorithm 2 WATERMARK EXTRACTION

```
/* Text string of $n$ Unicode characters $c$ */
$T = \{c_1, ..., c_n\}$
/* List of confusable original symbols */
$Originals$ = {U+002c,U+002d,U+002e,...}
/* List of confusable duplicate symbols */
$Duplicates$ = {U+a4f9,U+2010,U+a4f8,...}
/* List of confusable white spaces */
$Spaces$ = {U+0020,U+2002,U+2005,...}
$W$ =[ ] /* Watermark bit array */
$SpaceMap : Spaces \rightarrow \{000, ..., 111\}$ /* Reverse mapping function*/
for all $c$ in $T$ do
    if $c \in Spaces$ then
        $Append(W, SpaceMap(c))$
    else if $c \in Originals$ then
        $Append(W, 0)$
    else if $c \in Duplicates$ then
        $Append(W, 1)$
    end if
    if $|W| \geq 64$ then
        break
    end if
end for
return $W[:64]$
```

1. Knowing the associations tables, the author can extract the watermark w and the original text t from the watermarked text. The watermark is a MAC obtained from the original text and password using the keyed hash function. This step can be performed by an attacker with knowledge of the embedding algorithm and association tables.
2. The author applies the keyed hash function to the original text t using the same password k used in the watermark generation. The function produces the watermark w'. The attacker, as well, applies the same keyed hash function to the original text t using a different password k', thus obtaining a different watermark w''. Assuming that the author password k is different from the attacker password k', then the generated watermarks w' and w'', with a very high probability, are two distinct binary strings because of the collision-free property of hash functions.
3. The extracted watermark w is now compared with w' and w''. Because w and w' have been generated using the same text and password, they are equal, while w'' is different. The comparison proves the authorship of the watermarked text.

Despite the SipHash is considered as a secure keyed hash function, given its short MAC length, it may become less secure when more computational power becomes available to attackers. The proposed approach however does not rely on a specific keyed hash function, allowing any other hash function with stronger security to be applied instead. With this in mind, in Section 5, we provide evaluation of text length requirements using the hash functions MD5, SHA-1, and SHA-2.

It must be noted that, as for any other content-preserving method (i.e., all the image-based and structural methods), the attacker may still re-type the text and embed his own watermark. In this case, assuming that a digital sharing method has been used, it is usually possible to track the earliest version of the text and use this earliest version to track the original author, because it will carry the watermark firstly embedded by the author. The earliest watermarked text in fact can be verified only by the original author following the above verification process.

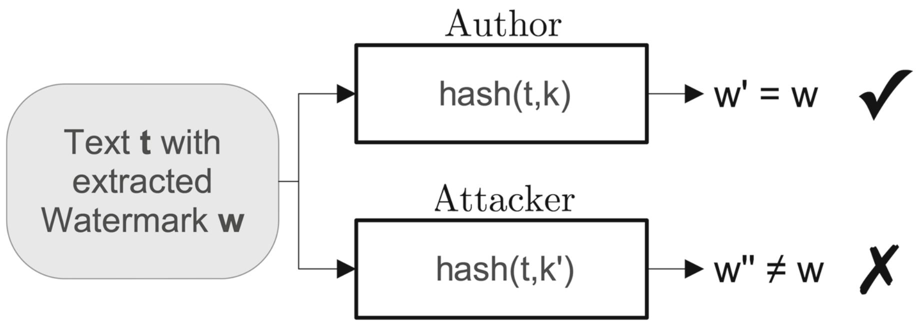


Fig. 3 Proof of authorship on watermarked text. The password k is a proof of authorship. Once the watermark is extracted, only the author with the original password k is able to reproduce it

4.6 Fine-grain watermarking

The presented watermark embedding method allows to watermark a text by embedding at the beginning of the text a unique information derived from the text itself together with a secret password. Considering that the underlying approach makes the method suitable for very short text, we extend the method to watermark longer text but in a fine-grained fashion. The ultimate goal is to keep authorship protection at paragraph or even lower levels, so that the text document can be protected from the copy of even a single sentence.

As before, the watermark is first computed uniquely from the content of the whole text document and the secret password, by means of a secure hash function. This guarantees that it is not computationally possible to generate the same hash using a (i) different key, (ii) a different document, or (iii) both, because any of the previous would imply finding collisions by purpose [58].

In the embedding, Algorithm 1 presented for the simple case the embedding process would stop once that all the bits in the watermark have been embedded one time in the original text. In the fine-grain watermarking, instead, we keep embedding repeatedly the document watermark until no document characters remain. The process of fine-grain watermarking is shown in Algorithm 3. Starting from the first character of the text, Algorithm 3 first checks if the current character can be replaced (that is, if it has a corresponding homoglyph). If this is the case, the current bit of the watermark (or 3 bits for whitespaces) is taken as a replacing condition. As in the original algorithm, it iterates over the text replaceable characters and the watermark bits in parallel. However, instead of consuming the bits of the watermark, which is usually way shorter than the text, watermark's bits are taken in circle, in a string rotation setting. This is accomplished by simply applying the modulus of the index by the watermark length.

Algorithm 3 Fine-grain Watermark Embedding

```
1: /* Text string of $n$ Unicode characters $c$ */
2: $T = \{c_1, ..., c_n\}$
3: /* List of confusable original symbols */
4: $Originals = \{$U+002c,U+002d,U+002e,...$\}$
5: /* List of confusable duplicate symbols */
6: $Duplicates = \{$U+a4f9,U+2010,U+a4f8,...$\}$
7: /* List of confusable white spaces */
8: $Spaces = \{$U+0020,U+2002,U+2005,...$\}$
9: /* List of all confusables*/
10: $Confusables = Originals \cup Spaces$
11: $W = [b_1, ..., b_{64}]$ /* Watermark bit array */
12: $GetDuplicate : Originals \rightarrow Duplicates$
13: $GetSpace : \{000, ..., 111\} \rightarrow Spaces$
14: $WT = [\,]$ /* Watermarked text */
15: $i = 0$ /* Watermark index pointer */
16: **for all** $c \in T$ **do**
17:     **if** $c \in Confusables$ **then**
18:         **if** $c \in Spaces$ **then**
19:             $bits = [\,]$
20:             **for** $ii = 0..2$ **do**
21:                 $append(bits, W[\,(i + ii)\%64]\,)$
22:             **end for**
23:             $c = GetSpace(bits)$
24:             $i+ = 3$
25:         **else**
26:             $bit = W[\,i\%64]$
27:             **if** $bit$=1 **then**
28:                 $c = GetDuplicate(c)$
29:             **end if**
30:             $i+ = 1$
31:         **end if**
32:     **end if**
33:     $Append(WT, c)$
34: **end for**
35: **return** $WT$
```

The complexity of the proposed algorithm is $\mathcal{O}(n)$, linear in the number of characters of the original text. In the external loop of Algorithm 3, we scan the original text for each of the n characters, while the internal loop (in row `20` of Algorithm 3) has constant complexity, appending the 3 bits for whitespace replacement. The append operation for a single character, the *GetSpace* function to get the Unicode value from the 3 bits from Table 2 and the *GetDuplicate* function to get the homoglyph given the original letter from Table 1, can all be executed in constant time.

In Fig. 4, we illustrate the watermarking process through a simplified example. Our document *doc* is a New York Times article that we want to fine-grain watermark. We use a toy 12-bit hash function, applying it to the concatenation of document and secret password. In Fig. 4, for easier readability, the 12-bit watermark is represented in hexadecimal notation by 3 digits, namely `0xABC`. The resulting 12-bit watermark is embedded repeatedly in the text, following the replaceable characters.

The watermarked excerpts of a text, alternately highlighted, have different lengths depending on the number and the type of replaceable characters found (e.g., spaces alone allow an embedding of 3 bits each). In fact, in order for a text excerpt to maintain the watermark, a variable length is required depending on the replaceable symbols in it: in the example of Fig. 4 with a 12-bit watermark, the portion "*ow republicans -*" is watermarked having only 17 characters, spaces included, while the portion "*was brewing as lawmakers*" is watermarked having 25 characters, spaces included. We will see in the evaluation section that the average symbols needed to embed a secure watermark of 64 bits is 101 characters.

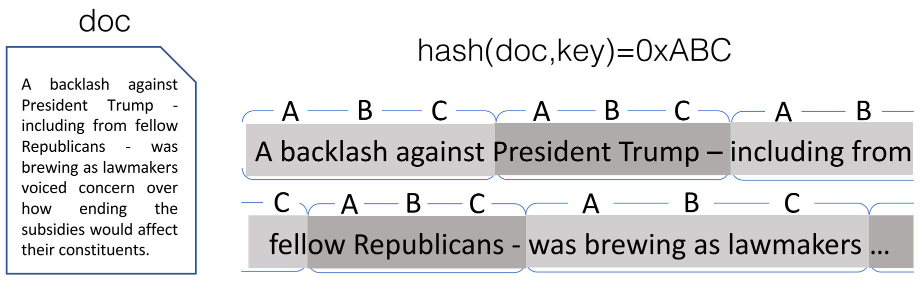

doc
A backlash against President Trump - including from fellow Republicans - was brewing as lawmakers voiced concern over how ending the subsidies would affect their constituents.
hash(doc,key)=0xABC
A B C A B C A B
A backlash against President Trump – including from
C A B C A B C
fellow Republicans - was brewing as lawmakers ...

Fig. 4 Fine-grain watermarking. The watermark, computed as the hash of the document and the secret key, is repeatedly embedded in smaller portions of text. For simplicity, a 12-bit watermark is shown, corresponding to the hexadecimal value `0xABC`. Watermarked portions are alternately highlighted

The sequential embedding of a watermark computed over the whole document in small excerpts of text confers two unique properties to excerpt watermarking: the watermarking is *continuous*, spanning across sub-portions of a text, and it is *part-of-whole*, because it binds each excerpt to the source document.

4.6.1 Continuous watermarking

Because the watermark is repeated sequentially across the document, it is possible to extract it from any sufficiently long excerpt of text, even if this is broken between two contiguous watermarks. For instance, in Fig. 4, the watermark `0xABC` is repeated in the order `A,B,C,A,B,C,` etc. Extracting the watermark `0xABC` from the first complete portion *"A blacklash against"* is straightforward using the presented Algorithm 2 for watermark extraction. However, this is also possible if the copied text portion is between two complete watermarks, as long as it has at least 12 bits of embedded watermarks: the watermark can be obtained by using a 12-bit window of embedded watermark and shifting the extracted binary string. In Fig. 5, a similar scenario is shown, in which a text *"against President Trump -'*) between two complete watermarked portions has been copied. Applying the extraction Algorithm 2, the shifted watermark `0xCAB` is obtained, which is then rotated until verification is reached or, after 11 rotation, verification is refused. Because any rotation of the original watermark is successfully verified, this verification process is shallower than the non-continuous version. Verifying as positive, any rotation of the hash reduces the search space for a verifying key; however, this is negligible for standard-sized hash.

Figure 6 shows a real example using a 64 bits watermark, one that can be obtained with SipHash, on the first paragraph of a New York Times article. Continuous lines define complete applications of a 64-bit watermark; however, any contiguous sequence with at least 64 bits is also watermarked, even if it overlaps two different complete watermarks. In the figure, the example excerpt *"Australia is late to the space party. The leader of its new space agency, Megan Clark, said"* is watermarked and can be verified even if it spans across two watermarks, because it has at least 64 bits embedded in it.

4.6.2 Part-of-whole watermarking

Each text portion is considered as a part of the source document. The unique hidden information is derived from the source document and the secret password, and it is the same for all the portions. For the purpose of authorship verification, the source document is also needed along with the secret password, but it will prove that the portion is coming from that particular document. The feature is illustrated in Figs. 4 and 5, and it is the direct consequence of the fact that, while the hash is computed over the whole document, the short-length requirement in fine-grain watermarking embed this information in each small text portion.

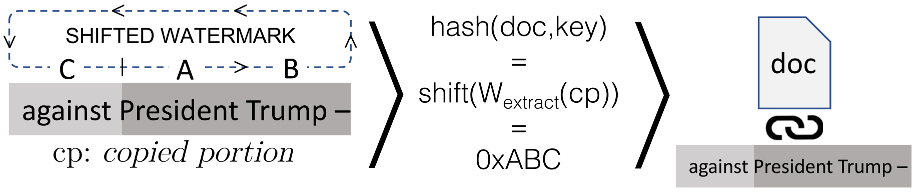
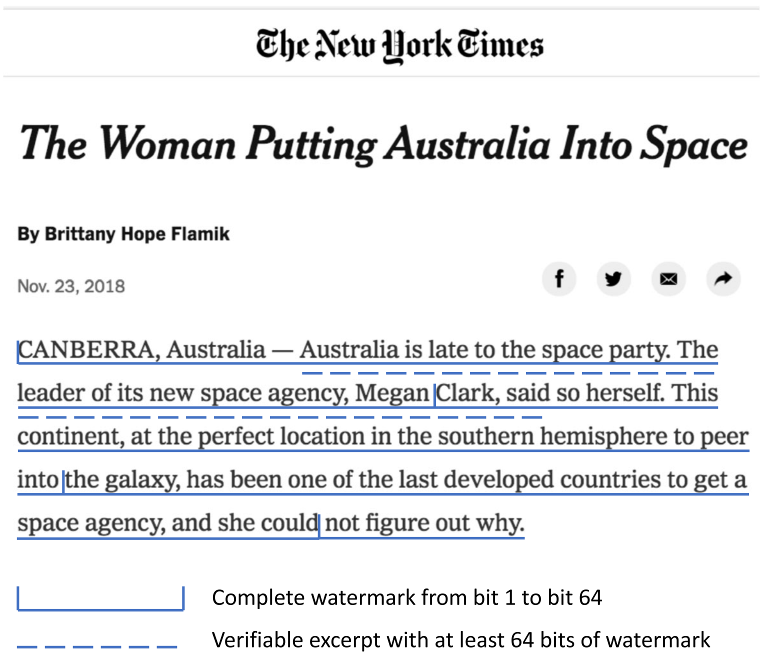


Fig. 5 Continuous part-of-whole watermarking. Any sufficiently long excerpt of text will keep the watermark when copied, independently from its position in the document. Because its watermark is derived from the whole origin document, any watermarked excerpt of sufficient length is bind to its source



The New York Times

The Woman Putting Australia Into Space

By Brittany Hope Flamik

Nov. 23, 2018

CANBERRA, Australia — Australia is late to the space party. The leader of its new space agency, Megan Clark, said so herself. This continent, at the perfect location in the southern hemisphere to peer into the galaxy, has been one of the last developed countries to get a space agency, and she could not figure out why.

Complete watermark from bit 1 to bit 64
Verifiable excerpt with at least 64 bits of watermark

Fig. 6 Excerpt example from NYT article. The continuous lines limit complete 64 bits watermarks. Underlined with a dashed line is an example of verifiable excerpt spanning across two complete watermarks

The described part-of-whole watermarking schema increase the security of the verification, as the original complete source is needed to verify the watermarked portion, resulting in a non-blind watermark.

5 Results and discussion

We conduct several experiments to assess the crucial properties of the proposed approach: the number of symbols required to embed a full watermark, the imperceptibility of changes in the watermarked text with respect to the original text, and the robustness of the watermark.

5.1 Embedding capacity

Because we can only embed watermark bits when a confusable symbol is found, the numbers of symbols needed depends on the number of confusables in the text. For this reason, this must be evaluated empirically on several texts. This symbol length requirement will directly affect the effectiveness of the fine-grain watermarking, because the shortest the length, the finest will be the watermarked portions of text.

In order to carry a realistic length requirement estimation, we take into consideration the articles from the New York Times Corpus [14], a collection of 1.8 million articles spanning from 1987 to 2007, appeared in the New York Times newspaper. News article are an example of authored text commonly subject to unattributed copies in blogs and social media. Moreover, because we want to show the fine-grain method capability at the paragraph level, we extract from each article only the lead paragraph[3].

In Table 3, we show the minimum and average obtained in the experiments results together with some ideal minimum boundaries: the most extreme case in which our original text has consequent whitespaces, allowing to embed 3 bits in each of them and resulting in 22 symbols needed on SipHash, and the less extreme but still uncommon scenario of one-character words separated by a whitespace, resulting in 33 symbols needed on SipHash. The average length obtain on NYT Corpus using the SipHash function is 101 characters, spaces included, while the minimum is 46 characters.

The embedding capacity is computed as the average ratio between the number of embedded bits and the number of characters in each document. Considering the average number of characters needed to embed a full

Table 3 Text length required to embed a watermark

	SipHash (64 bits)	MD5 (128 bits)	SHA-1 (160 bits)	SHA-2 (224 bits)
Spaces	22	43	54	75
1-char words	33	65	81	113
NYT MIN	46	93	116	163
NYT AVG	101.3	197.7	246.5	344

Minimum text length for length-preserving watermarking on SipHash and other well-known cryptographic hash functions watermark of 64 bits, the expected embedding capacity is $\frac{64}{101} = 0.632$ bits/character.

We also show ideal and real values for other hash functions, for which the length requirement grow linearly to the length of the produced MAC: using MD5, with a MAC of 128 bits, the average on NYT Corpus is 198 characters. The maximum length requirement is not shown because the maximum length is bound by the length of each paragraph. More meaningfully instead, we show how many times the paragraph was too short and lacking of confusable symbols for the embedding to be successful at paragraph-level.

In Fig. 7, we show what is the percentage of articles for which the watermarked excerpt is smaller than its lead paragraphs. Using a watermark of with a length of 64 bits, such as SipHash, the percentage stops at 94.51%, because in the remaining 5.49%, we are not able to successfully embed a watermark of 64 bits at paragraph-level; thus, the watermarked excerpt are bigger than a paragraph. The percentage of completely watermarked lead paragraphs decrease to 91.83% when we use a 128-bit watermark, 74.38% with a 160-bit hash and 63.97% using a 224-bit hash.

Figure 7 provides also a statistical estimation of the probability of watermarking at an excerpt of n characters. For example, it shows that an excerpt of 100 characters have a 50% probability of being successfully watermarked with SipHash that increases to 80% if the excerpt is 108 characters length.

We compare the embedding capacity of the proposed method with other recent steganography and text watermarking structural methods. In Table 4, the embedding capacity values for each of the considered methods are provided. Together with the capacity, we show also the overhead introduced by the embedding in terms of additional characters.

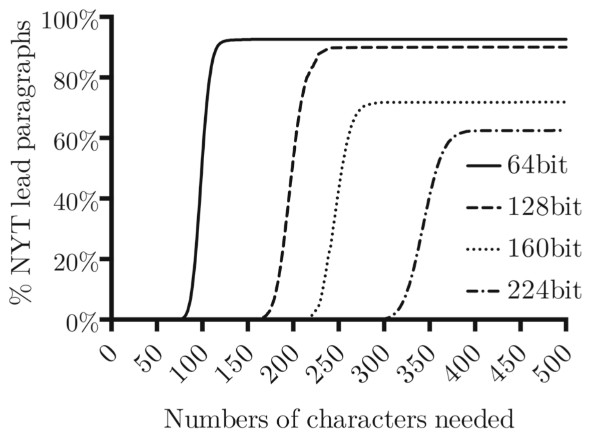


Fig. 7 Length requirements to successfully embed the watermark. Percentage of leading paragraphs in NYT Corpus that can be watermarked using only the first n characters. Results are shown for different hash lengths

Table 4 Embedding capacity and overhead comparison

Method	Embedding capacity	Overhead
Por et al. [45]	0.321	1.0
Taleby A. et al. [49]	0.063	1.0
Rizzo et al. [51]	0.632	0
Taleby A. et al. [59]	`Unlimited`	0.5
Khosravi et al. [47]	0.03	0.12
Fine-grain TW (proposed)	0.632	0

Embedding capacity in bits per character (the higher the better) for the proposed method and related structural methods for embedding data in text and overhead data introduced in additional characters per embedded bit (the lower the better)

Our method exhibits higher capacity with respect to all the other methods except for [59], where the capacity is only limited by the maximum length of the text and by the cover message, since an arbitrary number of invisible characters are inserted in the beginning of the text. However, our method is the only one that does not introduce any overhead of characters in the embedding, since each character is replaced with another character.

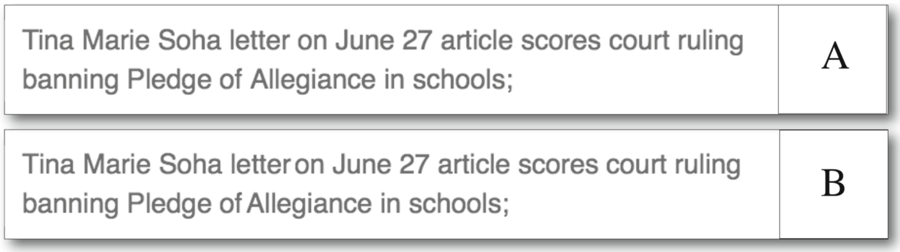
5.2 Indistinguishability of watermarked text

Our watermark embedding process works by replacing common symbols of latin alphabets, punctuations, and whitespaces with similar Unicode symbols. Because whitespaces have slightly different width and confusable symbols are not always identical to common ones in all Unicode-supporting fonts, it is crucial to assess in practice how much these differences are noticeable.

In Fig. 8, we show an example of a paragraph in its original and watermarked version. The watermarked version had several of the original symbols replaced by confusable symbols. Overall, the example does not show any noticeable differences to the human eye: main difference regards the horizontal spacing, because the width of the confusable whitespaces affect the words' position and ultimately the width of each line. Considering also that in a common scenario, the attacker does not have both original and watermarked version for comparison, we argue that the differences in paragraph level are not perceivable.

Besides paragraph-level differences, we compare each symbol with its related duplicate in well-known fonts. In Fig. 9, all the symbols are rendered in three sans-serif fonts used in the most known web platforms, including Facebook, Twitter, and Google. All the symbols show no noticeable differences in rendering when their duplicate is used instead of their common version. The reader can refer to Fig. 11 in the Appendix for a more extensive results on ten fonts.

It must be noted that changing the font of the watermarked text cannot affect its content (i.e., the Unicode



Tina Marie Soha letter on June 27 article scores court ruling banning Pledge of Allegiance in schools;	A
Tina Marie Soha letter on June 27 article scores court ruling banning Pledge of Allegiance in schools;	B

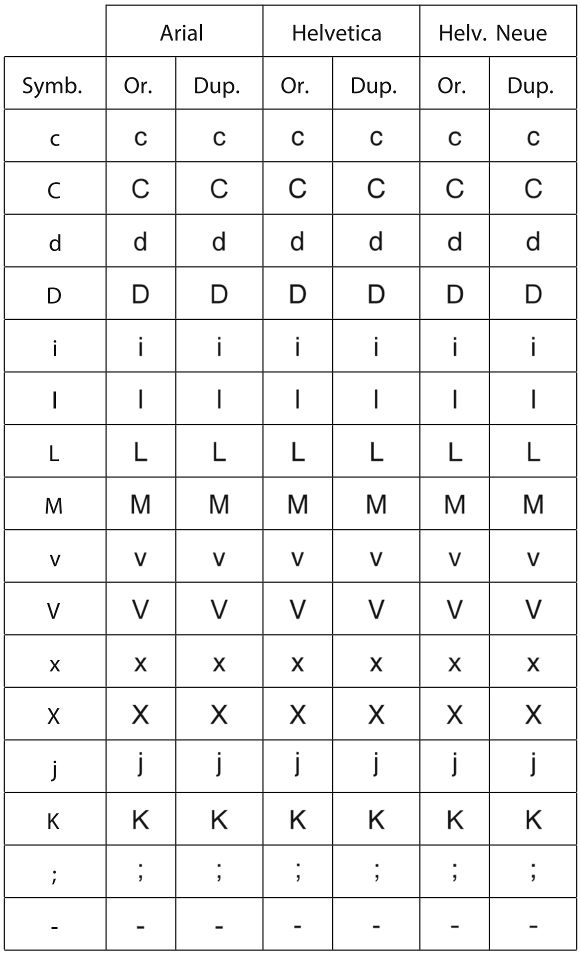
Fig. 8 Graphic rendering of original and watermarked. A lead paragraph from a New York Time article before watermarking (**a**) and after watermarking (**b**)

symbols) but only its appearance; thus, the watermark is retained with any change of font. In fact, the watermarked text—being a string of Unicode symbols—is not bound to a particular font and has no font embedded in it. If a malicious user copies the watermarked text in a word editor and changes the font to any other font, the watermark is still retained. On the other hand, if a font with a smaller or no support of the Unicode set is used, this is still rendered but the indistinguishability of the non-supported symbols is not guaranteed. This is because all modern software applications, including word processors and web browsers, automatically render the unsupported symbol using a similar font that supports it, in a cascading mechanism [60].

We evaluate the visual indistinguishability of the proposed method in comparison with two structural steganography techniques, Whitesteg [55] and Unispach [45]. Contrary to the proposed method, these two methods work by inserting additional invisible characters, so that while the readable content is preserved, they add a data overhead, resulting also in horizontal shifting due to whitespaces added between the words.

Figure 10 shows the overlapping between original text (in red) and watermarked text (in black) for a sample text. It is easily noticeable how the use of double spaces in Whitesteg and Unispach completely shift the alignment. Conversely, the proposed method does not use multiple, consequent whitespaces or append whitespaces at the end or beginning of the text.

We do not evaluate structural methods that are completely invisible [49, 59], as there is no visual change after the watermark is embedded. On the other hand, by adding new invisible symbols to the text instead of replacing the symbols, the invisible methods still introduce an overhead that increases the overall text size, a change that can be detected using any text editor or file manager.



	Arial		Helvetica		Helv. Neue	
Symb.	Or.	Dup.	Or.	Dup.	Or.	Dup.
c	c	c	c	c	c	c
C	C	C	C	C	C	C
d	d	d	d	d	d	d
D	D	D	D	D	D	D
i	i	i	i	i	i	i
I	I	I	I	I	I	I
L	L	L	L	L	L	L
M	M	M	M	M	M	M
v	v	v	v	v	v	v
V	V	V	V	V	V	V
x	x	x	x	x	x	x
X	X	X	X	X	X	X
j	j	j	j	j	j	j
K	K	K	K	K	K	K
;	;	;	;	;	;	;
-	-	-	-	-	-	-

Fig. 9 Confusable symbols rendering in web fonts. Original (Or.) and duplicate (Du.) symbols rendering for the three most known web fonts

5.3 Robustness

We evaluate the robustness of our approach together with a set of recent structural watermarking techniques for comparison. No exact means are available to measure the general robustness of text watermarks [54]; however, it is possible to measure the robustness against different attacks. The following attacks are commonly considered in assessing the robustness of text watermarking methods.

1 *Copy and paste.* A very common scenario where the content of a text file is copied and pasted into an attacker's file.

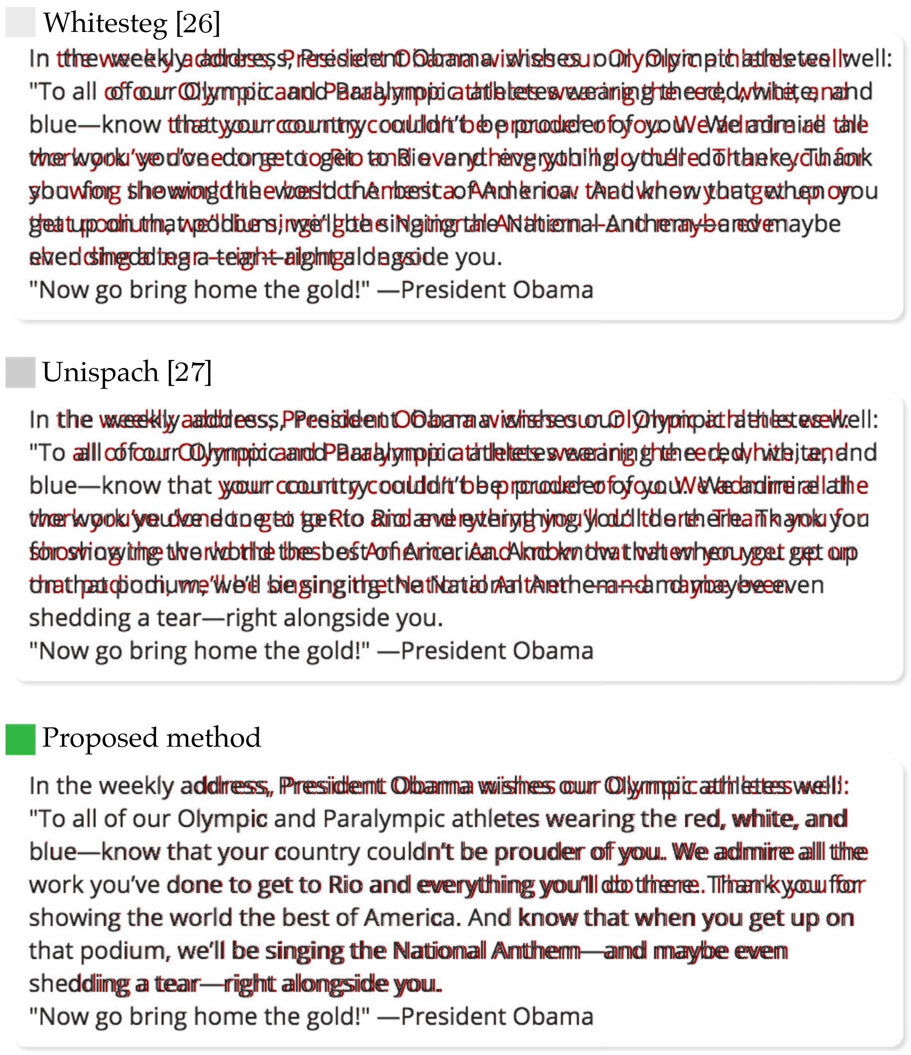


Fig. 10 Indistinguishability test. Overlapping original and watermarked text for content-preserving methods shows less noticeable changes with proposed method

2 *Reformatting.* Reformatting attacks including the change of formatting features of the text such as fonts or color. Copy and paste, retyping, and OCR have been also considered subtypes of formatting attacks [54]. Our method is robust against any change of font type, size, or color, because any formatting process leaves the Unicode content unaltered.
3 *Insertion.* Insertion attacks randomly new words in the watermarked text [61], with the goal of altering the watermark. We randomly insert the 10% of additional words, following a common attack size considered high [62].
4 *Deletion.* In this attack, some parts of the text are removed [54]. If the deleted portion is part of the watermark, the watermark may be destroyed or completely removed [61]. As for the insertion attack, we randomly remove the 10% of the words.
5 *Replacement.* In a replacement attack, a set of words in the watermarked text are replaced with other words [54, 61]. It can be considered as a deletion attack followed by an insertion attack in the same location. We test the robustness against a replacement attack [62], where 10% of the words in the watermarked text are replaced with other words. The word inserted as replacement is randomly chosen from a word list of 10,000 words.
6 *Retyping.* In a retyping attack, a malicious user retype the text in a different file or platform. Structural methods are all fragile to retyping by definition. In fact, because the structural methods embed the watermark by altering the layout such as formatting features, spaces, Unicode, and ASCII encodings without altering the content, retyping the content will always destroys a watermark.

5.3.1 Robustness to partial copy and paste

Selection of text is a natural—and in most software obligatory—step involved in the copy and paste process. In web pages, for example, users select specific sub-portions of the page to avoid copying advertisement images, logos, and navigation menus. More generally, it is common to select and copy only the portions of interests from a document. Copying a portion of a text document can be also considered a special case of *deletion* attack, in which all characters before and after the portion are removed. Fore example, in the case of attackers trying to remove the authors' names from a text [61], they may instead just select the text without the authors' information.

In order to measure the robustness against the partial copy attack, a scenario that also motivated the design of our fine-grain algorithm, we propose an additional measure. Note that it is not possible to know a priori which portion of a text the attacker will copy and paste nor it is possible to know a priori if a portion of some length will be protected: depending on the characters or spaces in the specific portion, a complete watermark may or may not be embedded in it. For these reasons, the only way to measure the robustness is empirically by assessing the robustness of each possible portion in a given text, for a large enough number of texts.

Because a digital text is a string of symbols, a naive way of measuring this robustness is by counting how many substrings out of all the possible substrings are still protected when extracted through copy and pasting. However, current laws do not allow to claim the copyright of small groups of words, let alone symbols, as this could set up barrier to expression [63], despite there are some cases where *microworks* of 16 words, or quotes from larger works (e.g., movie scripts) have been copyrighted [12] given their strong originality. Our metric therefore considers subsequences of words instead of substrings of characters and is parameterized on a number of words z.

Here, we use the indicator function $\mathbb{I}\{\cdot\}$ that returns 0 or 1 if the condition is false or true, respectively. Given that T is the whole text as a sequence of words, $S \subseteq T$ is a portion of T, and the number of protected portions of z words is:

$$T_{\text{protected}}(z) = \sum_{S \subseteq T, |S|=z} \mathbb{I}\{E_{T,k}(S) = E_{T,k}(T)\} \tag{1}$$

Then, the number of all possible subsequences with z words in a text of $|T|$ words is:

$$T_{\text{all}}(z) = |T| - z + 1 \tag{2}$$

Therefore, the robustness to partial copy and paste R_{PCP} is:

$$R_{\text{PCP}}(T, z) = \frac{T_{\text{protected}}}{T_{\text{all}}} \tag{3}$$

Less formally, the above definition of R_{PCP} measures the fraction of possible small words sequences with z words that are protected when isolated from their original content T. This definition also estimates the probability of having any subsequence S protected against partial copy. In fact, let a malicious user selects S among any of the sequences of the watermarked text T, where S is unknown to the author, and assuming that the selection happens with uniform probability, then R_{PCP} is also the probability of S to retain the watermark, that is, $P(E_{T,k}(S) = E_{T,k}(T))$.

In the evaluation, we use $z = 32$, as 32 words is considered a minimal group of words that can be copyrighted given enough originality [63].

Some of the evaluation tests are computationally intensive, such as counting word frequencies for the geometric advanced replacement or extracting all possible substring; therefore, we draw a random sample of 1000 articles from the NYT Corpus [14]. In the sample, the minimum character length in an article is 197, maximum is 57,136 and the average is 4213. We extracted words from the article using the regular expression `([\w][\w]*'?\w?)`, to also capture words with genitive such as `Tom's` as a single word, resulting in an average of 676 words per document, with a minimum of 31 and a maximum of 9375 words.

We implemented embedding and extraction for the methods under comparison following the original published descriptions. An exception regards the implementation of the algorithm in [49], where we repeatedly embed the watermark after each dot (.) only when this is followed by a new line; otherwise, the invisible control characters used have the effect of reversing the following string. For the steganography techniques considered, the results are computed by embedding a 64-bit message. For each method, the payload is embedded in the document using the implemented embedding method.

In the R_{PCP} evaluation, each possible portion of 32 words starting from the first in the article is copied. Then, the extraction method is ran on the portion: if the original payload is extracted, then $\mathbb{I}\{E_{T,k}(S) = E_{T,k}(T)\} = 1$, i.e., the portion will increase the number of $T_{\text{protected}}$ in Eq. 3; otherwise $\mathbb{I}\{E_{T,k}(S) = E_{T,k}(T)\} = 0$. This assessment is done for each portion until the end of the article. The R_{PCP} is computed for each article in the sample, and the average overall articles is shown in Table 5 together with the other measures.

In Table 5, we show the results of the evaluation. Overall, our method shows robustness results higher or similar to other methods for all the attacks. Methods which embed the watermark in non-word locations, such as whitespaces [45], after the dot (.) or in the end of the text [49, 59] show high or total robustness when words are attacked through insertion, deletion, and replacement. On the other hand, because the watermark is embedded in isolated locations outside the text, they are more easy to

Table 5 Robustness evaluation and comparison

Method	Copy and paste	Reformatting	Insertion (%)	Deletion (%)	Replace (%)	Retyping (%)	R_{PCP} (%)
Por et al. [45]	✓	✓	48.3	53.7	55.7	×	0.01
Taleby Ahvanooey et al. [49]	✓	✓	100	99.7	99.6	×	1.9
Rizzo et al. [51]	✓	✓	0	0	1.2	×	0
Taleby Ahvanooey et al. [59]	✓	✓	100	100	100	×	0
Khosravi et al. [47]	×	×	0	1.7	2.6	×	0
Fine-grain TW (proposed)	✓	✓	96.4	98.2	98.3	×	99.92

Robustness evaluation on 1000 news articles of recent, latin-based structural text watermarking and steganography methods. The results of the attacks involving insertion, deletion, and replacement of words are shown for 10% attack size, while 32 words are considered for partial copy R_{PCP}

lose when partial text is selected, thus performing poorly with respect to the proposed fine-grain method. The combination of a high embedding capacity, the repeated embedding, and the verification method through binary string shifting confers a very high copy and paste robustness to our method, which provides an almost total protection for groups of 32 words.

6 Conclusions

The protection of intellectual property of digital contents from plagiarism and unauthorized copy has become a challenging research problem, worsened by the ease of selecting, copying, and sharing other people's content. While text watermarking methods aimed at the protection of entire documents, the *fine-grain* protection of creative work from partial copy is a difficult and common phenomenon faced in copyright law.

In this paper, we have presented a fine-grain text watermarking method able to embed a password-based watermark in latin-based alphabet texts at paragraph level. By not relying on isolated text locations and having a discrete embedding capacity, the watermark is embedded repeatedly and thoroughly in a fine-grain fashion, so that it can be extracted even when only a small portion is copied.

The core of the method is the watermark embedding through confusable symbol replacement: Unicode symbols very similar or identical to common symbols are replaced following the content of the watermark. The watermark is generated using a keyed hash function, binding the watermark to the author's secret key and the original text. The method does not modify the readable content of the text, producing a watermarked text that is visually indistinguishable from the original text. Moreover, it is the first method able to embed data without introducing overhead.

From an extensive experiment on 1.8 million documents, the resulted average embedding capacity of the embedding method is 0.632 bits/character, while the fine-grain algorithm is able to protect text at paragraph level for the 94.5% of the times. This features allows the first fine-grain protection of text through continuous, part-of-whole watermarking. For any sufficiently long portion of a text, the proposed method allows authorship verification and source traceability, binding all the portions to its original source document, and protection against partial copy and paste.

Along with well-known attacks on words, we tested the algorithm for partial copy and paste, for which we proposed a novel measure of robustness. The algorithm has shown the highest robustness against partial copy and paste, protecting 99.92% of 32 words sequences, and high robustness to other attacks. Despite partial copy and paste is a natural activity, we have found that structural techniques are more fragile to it than to intentionally malicious attacks such as random insertion, deletion, and replacement of words. This is because, by keeping only a sequence of words, several locations commonly used for embedding are left out, such as whitespaces and hidden control sequences before and after punctuations. On the other hand, because our method embed the watermark in a fine-grained way thorough all the words, it is slightly less robust to word insertion, deletion, and replacement than other methods [49, 59]. In this context, we also noted how all structural watermarking methods are fragile to the simple retyping attack, allowing an attacker to remove the watermark with ease and without any knowledge of the underlying embedding algorithm or watermark embedding locations. This should be taken into account when assessing the robustness against more complex malicious attacks.

Endnotes

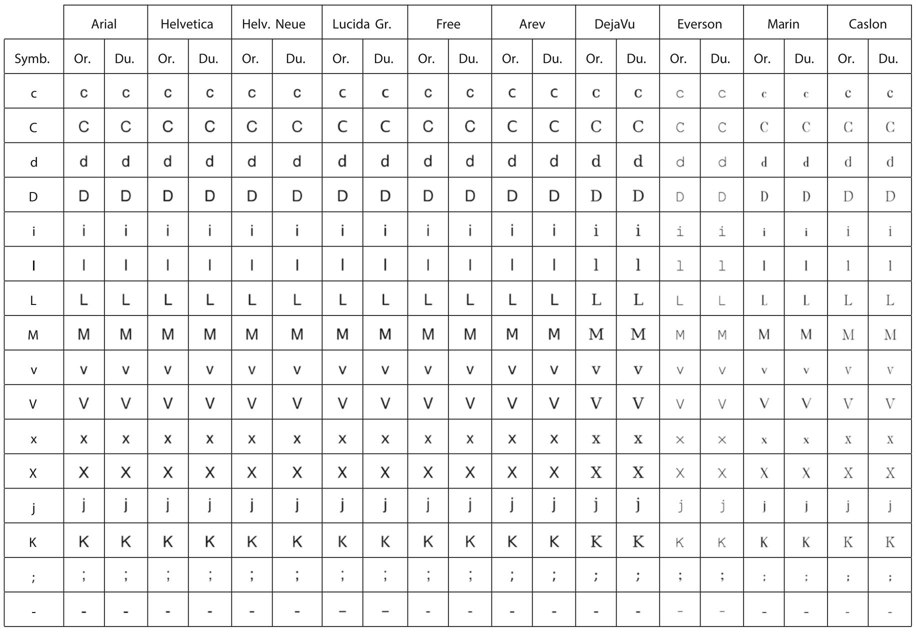
[1] A prototype that implements our method can be tested here: http://smartdata.cs.unibo.it/finegrain-watermark

[2] http://www.unicode.org/Public/security/8.0.0/confusables.txt

[3] The *lead paragraph* field of the article as defined by New York Times.

Appendix A

In Figure 11, for each of the 16 Unicode confusable symbols, we show both original and duplicate versions for ten font families.



	Arial		Helvetica		Helv. Neue		Lucida Gr.		Free		Arev		DejaVu		Everson		Marin		Caslon	
Symb.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.	Or.	Du.
c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i	i
l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l	l
L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v
V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j	j
K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;	;
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Fig. 11 Confusable symbols rendering. Original (Or.) and duplicate (Du.) symbols rendering for ten font families with Unicode support. The duplicate is indistinguishable for both serif and sans-serif fonts

Acknowledgements
This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 776848.

Authors' contributions
SGR is the corresponding author of the paper and is responsible for all aspects of the paper. FB did the part of the experiment. All authors wrote, read, and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The dataset supporting the conclusions of this article is available in the SmartData repository, http://smartdata.cs.unibo.it/datasets.

Competing interests
The authors declare that they have no competing interests and that there is no conflict of interest regarding the publication of this paper.

Author details
[1]Qatar Computing Research Institute (QCRI) HBKU, Doha, Qatar. [2]Department of Computer Science and Engineering, University of Bologna, Mura Anteo Zamboni, 7, 40126 Bologna, Italy.

Received: 19 December 2018 Accepted: 23 June 2019
Published online: 12 July 2019

References

1. A. Kapczynski, The access to knowledge mobilization and the new politics of intellectual property. Yale LJ. **117**, 804 (2007)
2. Y. Kauffman, M. F. Young, Digital plagiarism: an experimental study of the effect of instructional goals and copy-and-paste affordance. Comput. Educ. **83**, 44–56 (2015)
3. J. Grimmelmann, Regulation by software. Yale LJ. **114**, 1719 (2004)
4. S. K. Katyal, Filtering, piracy surveillance and disobedience. Colum. JL Arts. **32**, 401 (2008)
5. J. C. Lai, C. B. Graber, *Is digital text-watermarking the long-desired user friendly digital rights management? copyright and fundamental values from a comparative perspective. European Intellectual Property Review*. (Sweet & Maxwell, London, 2016)
6. A. M. Eskicioglu, E. J. Delp, An overview of multimedia content protection in consumer electronics devices. Signal Proc. Image Commun. **16**(7), 681–699 (2001)
7. S. Katzenbeisser, F. Petitcolas, *Information hiding techniques for steganography and digital watermarking*. (Artech house, Norwood, 2000)
8. X. Zhou, W. Zhao, Z. Wang, L. Pan, in *E-Business and Information System Security, 2009. EBISS'09. International Conference On*. Security theory and attack analysis for text watermarking (IEEE, New York, 2009), pp. 1–6
9. M. Durvey, D. Satyarthi, A review paper on digital watermarking. Int. J. Emerg. Trends Technol. Comput. Sci. **3**(4), 99–105 (2014)
10. D. Gross-Amblard, in *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. Query-preserving watermarking of relational databases and xml documents (ACM, New York, 2003), pp. 191–201
11. P. Verma, R. A. Shaikh, K. Deshmukh, A novel approach to angle based invisible text watermarking with ebcdic coding. Int. J. Comput. Appl. **61**(20) (2013)
12. J. Hughes, Size matters (or should) in copyright law. Fordham L. Rev. **74**, 575 (2005)
13. M. Davis, M. Suignard, Unicode security mechanisms. Unicode technical standard #39, Unicode. http://www.unicode.org/reports/tr39/. Accessed 02 Sept 2019
14. E. Sandhaus, The new york times annotated corpus. Linguistic Data Consortium Phila. **6**(12), 26752 (2008)
15. M. Kaur, K. Mahajan, An existential review on text watermarking techniques. Int. J. Comput. Appl. **120**(18) (2015)
16. R. Patel, P. Bhatt, A review paper on digital watermarking and its techniques. Int. J. Comput. Appl. **110**(1), 10–13 (2015)
17. Z. Jalil, A. M. Mirza, in *Information and Multimedia Technology, 2009. ICIMT'09. International Conference On*. A review of digital watermarking techniques for text documents (IEEE, New York, 2009), pp. 230–234
18. M. Taleby Ahvanooey, Q. Li, J. Hou, A. Rajput, C. Yini, Modern text hiding, text steganalysis, and applications: a comparative analysis. Entropy. **2019**, 350–381 (2019). https://doi.org/10.3390/e21040355
19. Y. Zhou, W. Jin, in *Multimedia Technology (ICMT), 2011 International Conference On*. A novel image zero-watermarking scheme based on DWT-SVD (IEEE, New York, 2011), pp. 2873–2876

20. M. Sabir, A. M. Mirza, Z. Jalil, Content based zero-watermarking algorithm for authentication of text documents. Int. J. Comput. Sci. Inf. Secur. **7**(2), 212–217 (2010)
21. Z. Jalil, H. Aziz, S. B. Shahid, M. Arif, A. M. Mirza, in *Educational and Information Technology (ICEIT), 2010 International Conference On, vol. 2*. A zero text watermarking algorithm based on non-vowel ASCII characters (IEEE, New York, 2010), pp. 2–503
22. K. Jaseena, A. John, An invisible zero watermarking algorithm using combined image and text for protecting text documents. Int. J. Comput. Sci. Eng. **3**(6), 2265–2272 (2011)
23. S. Kaur, G. Babbar, A zero-watermarking algorithm on multiple occurrences of letters for text tampering detection. Int. J. Comput. Sci. Eng. **5**(5), 294 (2013)
24. R.-z. Liu, T.-n. Tan, SVD based digital watermarking method. Acta Electron. Sin. **29**(2), 168–171 (2001)
25. Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
26. D. Bär, T. Zesch, I. Gurevych, in *Proceedings of COLING 2012*. Text reuse detection using a composition of text similarity measures (The COLING 2012 Organizing Committee, Mumbai, 2012), pp. 167–184
27. J. Brassil, S. Low, N. Maxemchuk, L. O'Gorman, in *INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE*. Electronic marking and identification techniques to discourage document copying (IEEE, New York, 1994), pp. 1278–1287
28. S. H. Low, N. F. Maxemchuk, J. Brassil, L. O'Gorman, in *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE, vol. 2*. Document marking and identification using both line and word shifting (IEEE, New York, 1995), pp. 853–860
29. A. K. Bhattacharjya, H. Ancin, in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference On, vol. 2*. Data embedding in text for a copier system (IEEE, New York, 1999), pp. 245–249
30. Y.-W. Kim, I.-S. Oh, Watermarking text document images using edge direction histograms. Pattern Recogn. Lett. **25**(11), 1243–1251 (2004)
31. J. T. Brassil, S. Low, N. F. Maxemchuk, L. O. Gorman, Electronic marking and identification techniques to discourage document copying. Sel. Areas Commun. IEEE J. **13**(8), 1495–1504 (1995)
32. S. H. Low, N. F. Maxemchuk, A. M. Lapone, Document identification for copyright protection using centroid detection. Commun. IEEE Trans. **46**(3), 372–383 (1998)
33. D. Huang, H. Yan, Interword distance changes represented by sine waves for watermarking text images. Circ. Syst. Video Technol. IEEE Trans. **11**(12), 1237–1245 (2001)
34. Y.-W. Kim, K.-A. Moon, I.-S. Oh, in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*. A text watermarking algorithm based on word classification and inter-word space statistics (IEEE, New York, 2003), pp. 775–779
35. T. Amano, D. Misaki, in *Proceedings of the Fifth International Conference on Document Analysis and Recognition*. A feature calibration method for watermarking of document images (IEEE, New York, 1999), pp. 91–94
36. X. Wang, in *Information Science and Technology (ICIST), 2013 International Conference On*. Digital watermarking research based on text (IEEE, New York, 2013), pp. 433–436
37. M. J. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, S. Naik, in *Information Hiding*. Natural language watermarking: design, analysis, and a proof-of-concept implementation (Springer, New York, 2001), pp. 185–200
38. K. Lambrecht, A framework for the analysis of cleft constructions. Linguistics. **39**(3; ISSU 373), 463–516 (2001)
39. H. M. Meral, B. Sankur, A. S. Özsoy, T. Güngör, E. Sevinç, Natural language watermarking via morphosyntactic alterations. Comput. Speech Lang. **23**(1), 107–125 (2009)
40. R. Huddleston, *Introduction to the Grammar of English*. (Cambridge University Press, Cambridge, 1984)
41. U. Topkara, M. Topkara, M. J. Atallah, in *Proceedings of the 8th Workshop on Multimedia and Security*. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions (ACM, New York, 2006), pp. 164–174
42. M. Topkara, C. M. Taskiran, E. J. Delp III, in *Electronic Imaging 2005*. Natural language watermarking (International Society for Optics and Photonics, Bellingham, 2005), pp. 441–452
43. O. Vybornova, B. Macq, in *Security, Steganography, and Watermarking of Multimedia Contents IX, vol. 6505*. A method of text watermarking using presuppositions (International Society for Optics and Photonics, Bellingham, 2007), p. 65051
44. O. Vybornova, B. Macq, in *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference On*. Natural language watermarking and robust hashing based on presuppositional analysis (IEEE, New York, 2007), pp. 177–182
45. L. Y. Por, K. Wong, K. O. Chee, Unispach: A text-based data hiding method using Unicode space characters. J. Syst. Softw. **85**(5), 1075–1082 (2012)
46. R. A. Alotaibi, L. A. Elrefaei, Improved capacity arabic text watermarking methods based on open word space. J. King Saud University-Computer Inf. Sci. **30**(2), 236–248 (2018)
47. B. Khosravi, B. Khosravi, B. Khosravi, K. Nazarkardeh, A new method for pdf steganography in justified texts. J. Inf. Secur. Appl. **45**, 61–70 (2019)
48. N. Mir, Copyright for web content using invisible text watermarking. Comput. Hum. Behav. **30**, 648–653 (2014)
49. M. Taleby Ahvanooey, H. Dana Mazraeh, S. H. Tabasi, An innovative technique for web text watermarking (AITW). Inf. Secur. J. Glob. Perspect. **25**(4-6), 191–196 (2016)
50. S. S. Baawi, M. R. Mokhtar, R. Sulaiman, in *International Conference of Reliable Information and Communication Technology*. Enhancement of text steganography technique using Lempel-Ziv-Welch algorithm and two-letter word technique (Springer, New York, 2018), pp. 525–537
51. S. G. Rizzo, F. Bertini, D. Montesi, C. Stomeo, in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. Text watermarking in social media (ACM, New York, 2017), pp. 208–211
52. J. T. Brassil, S. Low, N. F. Maxemchuk, Copyright protection for the electronic distribution of text documents. Proc. IEEE. **87**(7), 1181–1196 (1999)
53. J. Brassil, S. Low, N. Maxemchuk, L. O'Gorman, in *Proc. Conf. Information Sciences and Systems (CISS-95)*. Hiding information in document images (Johns Hopkins University Press, Baltimore, 1995), pp. 482–489
54. N. S. Kamaruddin, A. Kamsin, L. Y. Por, H. Rahman, A review of text watermarking: theory, methods, and applications. IEEE Access. **6**, 8011–8028 (2018)
55. L. Y. Por, T. Ang, B. Delina, Whitesteg: a new scheme in information hiding using text steganography. WSEAS Trans. Comput. **7**(6), 735–745 (2008)
56. J.-P. Aumasson, D. J. Bernstein, in *Progress in Cryptology - INDOCRYPT 2012*. Siphash: a fast short-input PRF (Springer, New York, pp. 489–508
57. H. Krawczyk, M. Bellare, R. Canetti, Hmac: Keyed-hashing for message authentication. Technical report (1997)
58. I. B. Damgård, *A Design Principle for Hash Functions*. (G. Brassard, ed.) Springer, New York, 1990), pp. 416–427
59. M. Taleby Ahvanooey, Q. Li, J. Hou, H. D. Mazraeh, J. Zhang, AITSteg: an innovative text steganography technique for hidden transmission of text message via social media. IEEE Access. **6**, 65981–65995 (2018)
60. M. J. Dürst, M.-A. Parent, *Font selection and font composition for Unicode* (Unicode Consortium, Mountain View
61. M. Taleby Ahvanooey, Q. Li, H. J. Shim, Y. Huang, A comparative analysis of information hiding techniques for copyright protection of text documents. Secur. Commun. Networks. **2018**, 5325040:1-5325040:22 (2018)
62. M. Bashardoost, M. S. M. Rahim, T. Saba, A. Rehman, Replacement attack: a new zero text watermarking attack. 3D Res. **8**(1), 8 (2017)
63. B. Kaplan, An unhurried view of copyright 45-46 (1967) and Robert Denicola, copyright in collections of facts: a theory for the protection of nonfiction literary works. Colum. L. Rev. **81**, 516–521 (1981)

Publisher's Note