Natural Language Watermarking and Tamperproofing[1]

Mikhail J Atallah, Victor Raskin, Christian F. Hempelmann, Mercan Karahan,
Radu Sion, Umut Topkara, Katrina E. Triezenberg

CERIAS,
Center for Education and Research in Information Assurance and Security,
Purdue University, West Lafayette, IN 47904, USA

{mja, mkarahan, utopkara, sion}@cs.purdue.edu, {vraskin,
hempelma, kattriez}@purdue.edu

Abstract. Two main results in the area of information hiding in natural language text are presented. A semantically-based scheme dramatically improves the information-hiding capacity of any text through two techniques: (i) modifying the granularity of meaning of individual sentences, whereas our own previous scheme kept the granularity fixed, and (ii) halving the number of sentences affected by the watermark. No longer a “long text, short watermark” approach, it now makes it possible to watermark short texts like wire agency reports. Using both the above-mentioned semantic marking scheme and our previous syntactically-based method hides information in a way that reveals any non-trivial tampering with the text (while re-formatting is *not* considered to be tampering—the problem would be solved trivially otherwise by hiding a hash of the text) with a probability $1–2^{-\beta(n+1)}$, n being its number of sentences and □ a small positive integer based on the extent of co-referencing.

1 Introduction

This paper reports a significant development in digital natural language (NL) text watermarking. It continues in the direction established in [1] in that it also:

- operates with the text *per se* rather than its printed or displayed image;
- embeds the watermark in the underlying structure of the text rather than in the surface elements of the text, such as words (cf. [2]);
- manipulates the text with the help of a small number of well-defined transformations (although, as mentioned, a transformation may now substantially modify the meaning of a sentence while preserving the meaning of the overall text and—unlike in our previous scheme—each transformation now has multiple ways in

[1] Portions of this work were supported by Grant EIA-9903545 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, and by sponsors of the Center for Education and Research in Information Assurance and Security (CERIAS). An online demo can be found at http://www.cerias.purdue.edu/homes/wmnlt/semdemo.html.

which it can be used to modify a given sentence, thus resulting in a much higher information-hiding capacity for a given text);
- does it all with the one secret key.

It improves that scheme considerably, however, by dramatically expanding the bandwidth and thus relieving the method of the long-text limitation.

While [1] establishes the basic technique for embedding a resilient watermark in NL text by combining a number of information assurance and security (IAS) techniques with the advanced methods and resources of natural language processing (NLP), it also faces the limitation of a narrow bandwidth: It was best applicable to long texts because it assured the embedding of just one bit of the watermark bitstring in each sentence and required a marker sentence for each watermark-bearing sentence, thus effectively lowering the bandwidth to .5 bit per sentence. And because both the resilience and low probability of false positives depended on a small ratio of the number of markers plus the number of watermark-bearing sentences to the total number of sentences, the longer the text the better were the results. The reason for the narrow bandwidth in [1] is that it manipulates the syntactic trees representing the syntactic structures of the text sentences: Such trees are not very large, and the possibilities for transforming them so that they embed the necessary portion of the watermark bitstring are limited.

This paper uses the same algorithm on text-meaning representations (TMRs) of the text sentences, which are much larger and richer trees that, like the syntactic trees, are automatically generated by the analyzer (see [3], Section 6.2, for a detailed and accessible example of TMR production) and that allow multiple semantic transformations of a large number of elements in them. This allows us to raise the bandwidth to around 8 bits per typical watermark-bearing sentence, although in practice we use 4 bits per sentence and reserve one for a special usage. It also allows, if needed, dispensing with marker sentences, thus enabling the technique to deal with such short texts as wire agency reports. And because the meaning of the text remains essentially the same, the current technique watermarks a text as well as many possible paraphrases of it, including its translations to any other NL.

This paper also describes how the semantic marking scheme presented in this paper, as well as the syntactic one in [1], can be used together to design a system that tamperproofs text; here we (mis)use "tamperproofing" in the sense of "making tamper-evident," i.e., any tampering with the text can be detected from the (corrupted) text itself, without the use of any other outside information. The probability that tampering with a sentence goes undetected is $2^{-\beta n+1)}$ (where n is the number of sentences in the text and b is the number of watermark bits per watermark-bearing sentence). The scheme uses, in two separate passes, both the semantic watermarking approach and the syntactic one (that leaves the TMR unchanged), thus overcoming circularity and exposure to the last sentence. In the first pass over the text, the semantic approach is used (which also modifies the syntactic trees). The second pass needs to perform something similar to the first pass but in a reverse order of the sentences, and has to also "respect" (i.e., not undo) what the first pass did: This is precisely what the syntactic pass does (works with syntactic trees without modifying the TMRs). Note that we cannot do it the other way around, because the semantic approach modifies both TMRs and syntax and, if used as the second pass, would undo what the first pass did. All this will be made more precise later in the paper, but for now we should stress that here we do not consider minor re-formatting of the text (like changing the line breaks or the spaces) to be tampering, otherwise the problem of tamperproofing text would be trivial ("store the keyed hash of the text in the formatting information"). Not surprisingly, our tamperproofing scheme works equally well for short texts as for long ones (contrast this with the fact that our watermarking scheme favors longer texts, in the sense that the watermark for them is more resilient than for short texts).

2   State of the Art

NL watermarking, at least as practiced here, abides by the same principles as image watermarking: The watermark should be resilient, undetectable to anybody but the author/owner of the text, easily and fully automatically produced by the dedicated software, etc. The crucial difference, making NL watermarking more difficult, is that "[u]nlike noisy data [in the images], written text contains less redundant information which could be used for secret communication" [4: 36], such as in steganography or watermarking. Naturally, the first attempts in text watermarking attempted to treat text as image [5-7] or to manipulate the external formatting properties and parameters of LaTeX, HTML, or PostScript [4: 36-37].

Attempting to embed watermarks in texts themselves, various groups have deliberately inserted spelling, syntactic, punctuation or even content errors. Synonym substitution has never lost its appeal (cf. [2]), but none of these methods prove to be very resilient, and they do degrade the quality of the text (an inessential deliberate distortion in the data may gain significance under special circumstances). Another technique that has been tried in text watermarking mimics, statistically or syntactically but never semantically, the properties of a NL text and generates a cover text around a secret message that may look like a regular text to a computer but never to a human, because it is, basically, meaningless—at least at the paragraph, if not the sentence level (see, for instance, [8-10]).

[1] is the first approach to aspire to the same principles and requirements as the best work in image watermarking while preserving the meaning and the quality of the text. The basic premises of this approach are shared by this paper. It should be noted that the approach follows the now pretty standard method of dividing the bits of the watermark's (hashed) bitstring among the text sentences, first introduced apparently in the work of Anderson and Petitcolas [11, 12].

3 Basic Premises

In this section we briefly review the framework and the basic elements of the scheme introduced in [1]. This is in preparation for the main novel ideas in the paper, which are contained in sections 4—6.

Watermarked Text. Watermark *W* is inserted in text *T*, resulting in text *T'*, which preserves the meaning of *T*. *W* is not readable from *T'* without knowledge of the secret key used to introduce *W*. With the secret key, one does not need *T* to produce *W* from *T'*. Without the key, it is very hard to remove *W* from *T'* without drastically changing its meaning and thus destroying the identity of the text. Only the key is secret while the process of introducing *W* into *T* is not.

Adversary. Interested in removing, destroying, or at least damaging *W* without destroying the identity of *T*, the adversary will perform meaning-preserving transformations on the text, well beyond re-formatting and other appearance-related tinkering, which he is actually allowed to do, including inter-language translation; perform meaning-modifying transformations on a small number of sentences (a large number of such transformation will modify the overall meaning and identity of the text); insert new sentences, move sentences and blocks of sentences around. While the adversary knows what our scheme does he does not know where in the text it has been applied, and, of course, no amount of paraphrasing, including sentence and paragraph substitution, will remove the watermark.

Building Blocks. We use k to denote the (secret) watermark-insertion key, which is also used at watermark-reading time. The first building block we need is a facility for using k to read a number (say, β) of secret bits that are in a sentence s (this is what the sentence “secretly says” to someone who knows k). If the watermark *W*’s length w is longer than β then it will be stored in $\alpha = w / \beta$ selected sentences: How these sentences are selected using k is the third building block, described last in this section. The second building block we describe explains how a particular selected sentence can be modified until it secretly says the right thing, where “the right thing” means that the β bits it secretly says equal the portion of the watermark that this sentence is supposed to store. The reading of what a sentence secretly says can be done in any of a number of ways. We describe some below, beginning with one that has drawbacks but that will serve as an introduction for the later (better) one. Let $H_k(s)$ be a keyed function of s when s is viewed as a bitstring (by reading the characters that make up s and recording the binary representation of each character); for the sake of definiteness, we assume $H_k(s)$ is a keyed hash of s. The β bits secretly hidden within s are the leftmost β bits of $H_k(s)$ (or, alternatively, its rightmost β bits, or its middle β bits—any consistent choice will do). One drawback of this scheme is that the slightest change to s (e.g., synonym substitution, replacing one article by an equivalent one) is likely to destroy the watermark bits in it, i.e., make it say something that no longer equals the portion of the watermark that the sentence is supposed to store. This drawback is remedied in the next technique. Let $T(s)$ be the tree structure that represents either the syntactic structure of s (in the syntactic version of our scheme), or the meaning of s (in the semantic version of our scheme, in which case the tree is the “text meaning representation,” a.k.a., TMR tree). (See the Appendix for examples of $T(s)$ in each case.) We use β bits of $H_k(s)$ to store the watermark, i.e., it is β bits of $H_k(s)$ (not of $H(s)$) that the sentence s “secretly says.” Using $H_k(T(s))$ rather than $H_k(s)$ has the advantage that minor modifications to s leave $T(s)$ unchanged, hence s more resiliently retains the β watermark bits in it, when subjected to simple modifications to s (for example, synonym substitutions do not change $T(s)$ when that tree’s representation captures the details of its branching structure but ignores the specific contents of individual nodes). If a selected sentence does not secretly say the bits we need it to say, we attempt to make it say the correct bit sequence by transforming the sentence without any serious meaning change to the overall text. The approach is to cause a change in $T(s)$ and re-calculate what the modified sentence secretly says, until it ends up secretly saying the desired b bits. The syntactic transformations are described in detail in [1]; the semantic ones are introduced in the next section.

Our scheme makes use of the notion of a secret ranking of the n sentences to determine which sentences will carry watermark bits. Let the text to be watermarked consist of n sentences $s_1 , \dots , s_n$. For each such tree T_i we obtain a binary string B_i, and the secret ranking of the sentences is that of the lexicographic ordering of their B_i’s (with ties broken according to the sentence’s position in the original text). There are many ways in which such a B_i can be obtained from the tree T_i. One example is $B_i = H_k(T_i)$ where T_i is a representation of the tree T_i. (There are many possible ways to represent a tree T_i, including using a listing of the pre-order numbers of the tree’s nodes according to a post-order traversal of the tree, or an “adjacency lists” representation—for each node use a list containing the node’s children, etc.) The smallest-ranked $\alpha = n / \beta$ sentences (in the secret ranking) are *markers* and it is the sentences that follow the markers that are watermark-carrying. (Actually there could be slightly more than α markers—see [1].) Why, though, not use the markers themselves (instead of their successors in the text) for storing the watermark? Because the modifications, needed to insert watermark bits in what a sentence secretly says, would change that sentence’s B_i and hence its secret ranking (that sentence would then almost surely no longer be a marker, and even if it remained one it would be in the wrong secret order relative to the other markers). One way of avoiding markers is described later in the paper.

Validation and Evaluation. We will, obviously, feel more confident about the proposals after we run the systems on a large number of texts. So far, the proof-of-concept system has run well on a small number of texts, as per the demo (see fn. 1), and the system time has been 3-8 msec per transformation performed. A test watermark has been inserted successfully in texts ranging from 12-36 sentences resulting in 3-6 transformations per text. These data should be considered very preliminary as we are planning a massive evaluation, validation, and improvement of the schemata within a much larger research frame.

4 TMR Trees and Semantic Transformations

4.1 Arborization

For our watermarking scheme, we use a tree built out of the TMRs provided by ontological semantics and obtained fully automatically in the analysis of the sentences of a text. The TMR is a list of propositions describing the events and concepts that represent the meaning of a text. For the purpose of generation of sentences and other issues we have to interface with the main ontological semantic application. This means that we need a reversible method of translation between the TMR proposition lists and our TMR trees. Such a method is described in this section.

For the building of TMR trees, the “arborization” if you will, we take the propositions as material and a small set of principles, many of which are already explicitly realized in the TMR, as tools. Generally, one such tree represents one sentence. A prominent exception is the co-reference list, a separate tree that establishes the identity of concepts throughout a text and will require special attention.

The main principles for turning a set of TMR proposition into a tree are:

1. The event proposition of the (often implicit) speech act of every sentence is the root of its TMR tree.
2. Filled slots of a concept are suspended from it as branches.

These simple principles cover most of the arborization issues, as in this straightforward example, slightly abbreviated as indicated by quotation marks to save horizontal space:

(1) The EU ministers will tax aviation fuel as a way of curbing the environmental impact of air travel.

```
author-event-1--|--author--unknown
                |--theme--levy-tax-1--|--agent--set-4--|--member-type--geopolitical-entity
                                      |                |--cardinality--unknown
                                      |                |--members--(set| “EU nations”)
                                      |--theme--kerosene-1
                                      |--purpose--regulate-1--|--agent--unknown-1
                                                              |--theme--effect-1--|--caused-by--flight
```

In case two or more propositions of one sentence share a concept, we have decided to suspend the second and later propositions from the first one. In the following example, the theme of the goods for which the manufacturing capacity is expected to be expanded are the same as would otherwise have to be imported. Hence, in the TMR propositions the themes of the concepts representing the expansion manufacturing event, MANUFACTURE-1, and the import event, IMPORT-1, are identical (see a detailed explanation of how this particular TMR is produced automatically in [3], Section 6.2).

(2) Dresser Industries said it expects that major capital expenditure for expansion of U.S. manufacturing capacity will reduce imports from Japan.

In TMR proposition lists the co-reference is represented as a separate parameter at the end of the list:

(3) co-reference-2
    import-1.theme manufacture-1.theme

In the arborization of the TMR list, the co-reference that pertains within one sentence will result in the latter proposition, IMPORT-1, to be suspended from the theme (as the shared concept) of the earlier one, MANUFACTURE-1.

(4)

```
...--purpose--import-1--|--agent--unknown
                        |--theme--manufacture-1.theme--manufacture-1--|--agent--unknown
                        |                                             |--theme--unknown
                        |                                             |--location--USA
                        |--source--Japan
                        |--destination--USA
```

In short, the third principle of arborization is

3. Propositions with co-referenced concepts that are not branches of the TMR tree through principle 2 are branches of the concept that is first used within the tree.

If full sentences are conjoined through coordination (“and,” “or,” “but”) they are linked as two nodes under the same AUTHOR-EVENT. Similarly, if there is a temporal relation between two sentences (“before,” “after,” “during,” etc.), they are ordered in actual event occurrence under a node indicating the temporal relation directly under the AUTHOR-EVENT node. Modalities of events (formality, politeness, respect, etc.) are final branches under the concept they take scope over, with the type, value, and attribution as the subbranches of that branch.

If there is no ordering imposed on nodes on the same level from the meaning of the text, we found it desirable to have “less important” ones at the bottom, where they will be the first to be targeted by our scheme. A first proposal for a hierarchy as suggested by the ontology browser is the following. Note, of course, that not all slots of a concept are filled where this concept occurs in the TMR of a text, but if several slots are filled, their order will follow this hierarchy: case roles >> specifically restricted slots (not inherited) >> cause-effect >> composition >> inheritance (should be rare in TMRs).

4.2 Accommodating the Watermark Bits

We have devised three general methods for changing the text in such a way that the TMR tree, and consequently our reading of it and the resulting bitstring for the watermark, will be affected by this change.

1. *grafting*: cutting/copying of information in one sentence and pasting it into another
2. *pruning*: cutting of information that is repeated
3. *substitution*: replacement with equivalent information

In general, the information that can be used for these schemes is chosen by two criteria. The first, most important, is that information that is repeated, established in the TMR through co-reference, can be safely removed, or repeated again. The second is that there is additional information available from the fact database of ontological semantics, and this additional information can be substituted for part of or added to the tree. We will discuss these criteria and their application in depth below.

4.2.1 Co-Reference

In order to be coherent, every well-formed text has cohesion, that is, it is about something. This theme will be established early in the text, which then proceeds to add more information about it. The new information will be in relation to the old information already given, and is often explicitly established in this way. If the text we have to watermark is, for example, about the United States bombing Afghanistan, we will expect it to make reference throughout to the concepts "United States," "Afghanistan," and "bombing." These instances of the concepts will be interrelated, because it is, for example, always the same "Afghanistan" the text will refer to. That is, the instances are co-referential.

In non-technical terms, for our sample text, Afghanistan is the main theme, or at least one of the main themes. Accordingly, many sentences will be about this nation itself or several of its slot fillers resulting in a rather extensive co-reference of "Afghanistan." The following are the established co-reference relations for the sample text (see appendix) with the portion of the text that contains the proposition which instantiates "Afghanistan" on the right:

(5)

co-reference-4

bomb-1.target	"carpet-bombed Taliban front lines in Afghanistan"
victory-1.theme	"With no visible victory so far in Afghanistan"
assault-1.theme	"The United States has been attacking Afghanistan"
fly-air-vehicle-1.path	*the occurrence in the example sentence*
carry-2.destination	"fly additional ... troops into the country"
assault-4.theme	"U.S. strikes on Afghanistan" (assault-1 = assault-4)
assault-6.location	"In Afghanistan, U.S. planes stepped up strikes"
...	

It is safe to assume that the web of co-references woven throughout a text is very tight. To illustrate this fact, in reverse, it is very exceptional that a sentence will not have any co-reference to its surrounding context. In a text about the United States bombing Afghanistan, there will not be a sentence like (6).

(6) Today, coffee is the second most popular beverage in the world, after water.

For the use in our watermarking scheme, every set of co-references is given with the full subtrees of the co-referenced concepts as they appear in each context. This way, the meaning-manipulating operations of our system are optimally facilitated. It should be added that co-reference detection, essential for meaning processing aside of any IAS concerns, is very reliable.

4.2.2 Fact Database

The second basic tool is the use of information available from the fact database resource of ontological semantics. The fact database entry for Afghanistan would yield the following additional information, conveniently structured in a tree-like hierarchy:

(7)

Afghanistan (nation-1)

borders-on	China, Iran, Pakistan, Tajikistan, Uzbekistan
has-currency	afghani
has-member	Pashtun, Tajik, Hazara, Uzbek
has-representative	Mullah Mohammad Omar
...	

4.3 Manipulation of the TMR Tree

In general, concepts that have greater co-reference will be better candidates to be changed. This selection principle to determine the possible cutting or pasting points of the tree is only outranked by the following one: The lower a concept is in the TMR tree, the less important we assume the information it represents to be. The search for a candidate to change will thus start from the lower end of the tree and proceed until it identifies a concept for which co-reference is established.

Through the co-reference relations computed for the whole text we know additional slot fillers for several of the ontological concepts instantiated in the TMR of the sentence to be changed. This includes, for example, MINISTRY-1 “Pentagon,” but most prominently NATION-4 “Afghanistan.”

4.3.1 Pruning of TMR Trees

In case several candidates present themselves for the manipulation based on repeated information as witnessed by co-reference, the pruning method will prefer the one for which more information is repeated. Although we won’t completely lose any information through pruning, because we know it is repeated elsewhere in the text, the assumption is that it is less likely to be a salient loss in the position where we cut it, if we have more than two uses of the concept.

In our example, co-reference tells us that NATION-4 is abundantly repeated and can be pruned more safely than other concepts that are less often repeated, like, for example, CITY-1, “Washington,” which occurs five times in the text. We will not clear the first use of a concept for pruning, but any of the subsequent ones und the assumption that the first mention is privileged. This would make the omission of NATION-5 in the TMR trees of these and any following co-referenced instances possible: VICTORY-1.THEME, ASSAULT-1.THEME, FLY-AIR-VEHICLE-1.PATH.

The following are the respective sentences with the words in italics cleared for pruning:

(8) a. With no visible victory so far *in Afghanistan*, President Bush asserted that the campaign he launched in reprisal for September’s mass killings on U.S. soil was going well, and he urged Americans to remain patient.

b. In Pakistan, which is backing U.S. strikes *on Afghanistan*, a minister said official tests confirmed that at least one suspicious letter received there contained anthrax spores.

c. The Pentagon ordered two new spy planes, including the unmanned “Global Hawk”, to the region to start flying *over Afghanistan*.

When a co-referenced instance of a concept has been used for any manipulation, grafting, pruning, or substitution, it gets a flag that will prevent its repeated use in the same cycle. Only if at a later point nothing but the use of a flagged concept could tweak a sentence to read the desired bit, will such concepts be considered again.

4.3.2 Grafting of TMR Trees

If we decide to pursue the scheme that would add additional information for an instance of a concept, our analyzer will pick one of the other instances and “litter” it with information about it that could be gleaned from the instance of the concept’s mention at hand. If it would choose, for example, the third co-reference, ASSAULT-1.THEME, the information in (9) from the TMR tree will be grafted onto the tree of sentence (10) after it has been copied or cut from its original occurrence, resulting in tree (11)

(9)

```
assault-1--|--agent--nation-1--“United States”
           |--theme--nation-4--“Afghanistan”
```

(10) The Pentagon ordered two new spy planes, including the unmanned “Global Hawk”, to the region to start flying over Afghanistan.

(11)

```
purpose--fly-air-vehicle-1--|--agent--unknown
                            |--path--assault-1.theme--assault-1--|--agent--nation-1--“United States”
                                                                 |--theme--nation-4--“Afghanistan”
```

In order to keep the tree branching strictly downward the new proposition is simply grafted onto the concept of the main tree that is co-referential to the other concept. Since this is a reversible process, the generator of ontological semantics will be able to generate the following new sentence out of the grafted TMR tree (11):

(12) The Pentagon ordered two new spy planes, including the unmanned “Global Hawk”, to the region to start flying over Afghanistan, which they are attacking.

4.3.3 Adding/Substitution

The methods described in this section can, of course, be combined, and we can utilize additional information ontological semantics provides through its fact database. For the example, we find the database entry given in subsection 3.2.2 above and can glean from it the fact that Afghanistan was at the time of writing of the text ruled by the politician Mullah Mohammed Omar. Thus, we can add to the final node this piece of information as in (13), or we can substitute it as in (14), yielding the sentences in (15) and (16), respectively.

(13)

```
assault-1--|--agent--nation-1--“United States”
           |--theme--geopolitical-entity--|--has-representative--politician-6
```

(14)

```
assault-1--|--agent--nation-1--“United States”
           |--theme--nation-4--|--has-representative--politician-6
```

(15) The United States are attacking the country ruled by Mullah Mohammed Omar.

(16) The United States are attacking Afghanistan, which is ruled by Mullah Mohammed Omar.

4.3.4 Summary

The three methods of TMR tree manipulation make use of the resources provided by ontological semantics in the following way:

pruning	co-reference
grafting	co-reference
substitution	fact database

If necessary they can be combined to more than one concept in most sentences, namely any concept that has co-reference, accounting for the high bandwidth of this scheme (see above).

5 Putting Large Watermarks in Short Texts

If the number of sentences n is small enough that $n\beta$ is not much larger than the number of bits (= w) in the watermark, as happens with short communiqués or newsclips, then we cannot afford to “waste” sentences by using them for markers: Every sentence is needed for carrying watermark bits. In that case we do not use markers, we simply use the first sentence s_1 for the first β bits of the watermark, s_2 for the next β bits of the watermark, etc. We may even be interested in choosing a rather large β, but for a large enough β there is then a danger that we may be unable to insert the next β watermark bits in a particular sentence. This failure has probability $(1-2^{-\beta t}$ of occurring in a sentence that we can “torture” in t different ways. While this is not a concern if β is small or when t is large, an ambitious enough choice for β would make the failure likely for at least some of the sentences that have a low t (even though t is exponential in the number of co-references for that sentence, that number of co-references may be small for some sentences). One way around this is to choose a large β anyway, but to provide a recovery mechanism in case the failure does happen. The mechanism is simply to sacrifice one (say, the first) of the β bits secretly said by each sentence, i.e., to no longer use that particular bit to store a watermark bit, but rather to use that bit as an indicator of whether that particular sentence is watermark-carrying or not (hence a watermark-carrying sentence now effectively carries $\beta–1$ rather than β watermark bits, but we are now free to choose a large β). In case of failure for a particular sentence, it is practically always possible to make the "indicator" bit 0, in which case the remaining $\beta–1$ bits secretly said by that sentence are ignored at watermark-reading time (for the watermark-carrying sentences that indicator-bit is 1). Assuming, for the sake of an approximate quantitative discussion, the same t for all sentences, the expected number of watermark bits successfully inserted is then

$$n(\beta–1)(1–(1–2^{-\beta t}$$

Compare the above quantity to the deterministic $n\beta'$ capacity that would have resulted had we used a β' that is smaller than β. Such a β' would have had to be small enough to practically guarantee that every sentence would be watermark-carrying. Sacrificing n bits to enable a larger β, in the manner described above, is a better design than getting stuck with a low β' whose value is determined by the "weakest" sentence (the one with smallest t), especially since most sentences will have a substantially higher t (and therefore higher watermark-carrying capacity) than the weakest sentence. Of course the "long watermark in a short text" framework of this section results in a watermark that is less resilient than in the "short watermark in a long text" case when we could afford the luxury of markers.

The above way of avoiding markers may be attractive even in the "short watermark in a long text" case: We could simply repeat the watermark, effectively using a longer watermark of length n that consists of repetitions of the watermark string as many times as needed to use all of the sentences. But the attacker who knows we are doing this then immediately knows there is a periodicity involved and may be able to selectively damage everywhere the same fragment of the watermark (the fragment she dislikes) without damaging its other fragments. Another drawback is that an attack that consists of changing the order of some sentences now becomes effective, whereas it had a low probability of success when markers were used. (The probabilities of success of various attacks are, for the version of our scheme that uses markers, the same as in [1], although an attacker must now change the TMR, a more tricky proposition for her than modifying the syntax tree.)

The main other advantages of the scheme presented in this section are that:

- Watermark-carrying capacity of a sentence is much improved, because a typical sentence involves many co-references and the number of possible ways we can "torture" a sentence (to make it secretly say what we want) is exponential in the number of co-references for that sentence. Specifically, if ρ is the number of co-references and we are using t modification mechanisms then the number of ways is t^{ρ}. Contrast this with the t possibilities we could play with in the syntactic approach: no exponent in that approach, the number of ways was simply t (see [1]).

- Watermark-carrying capacity of the whole text is liberated from the straightjacket of "weakest sentence determines bits of watermark per sentence." This is achieved through a choice of watermark bits per sentence (a number we call β) that is so high that many of the weaker sentences will fail to accommodate the β bits, in which case we "bypass" them by sacrificing one watermark bit and using it as an "indicator" of whether the sentence is watermark-carrying or not (the weaker sentences will not—but they no longer force upon us a low $\tilde{\beta}$
- The scheme deals with collusion tolerance by providing a mechanism for creating deliberate noise in the text; however, that as well as the interesting topic of error correction and post-attack restoration are subjects for future research.

6 Tamperproofing

Our scheme of meaning-based text marking and manipulation lends itself not only to watermarking, but also to tamperproofing. As stated earlier, here we use "tamperproofing" in the sense of "making tamper-evident," i. e., any tampering with the text can be inferred from the corrupted text itself (without the use of any other outside information). The problem of tamperproofing text is easier if one considers trivial formatting modifications (such as re-formatting the text, like inserting new line breaks or blank spaces in it) as tampering to be detected, than when one is supposed to be forgiving of such changes, as we indeed are. The reason the former is easy is because one can then compute some kind of keyed hash of a format-independent version of the text, and hide that hash value in the formatting information—any change to either the text or the formatting would be detectable because they would no longer be "tuned" to each other. Here we consider the harder version of the problem, where trivial formatting changes are not considered to be tampering, and in fact they are specifically allowed because different people in the organization use different word processors, etc. It is hard because we run into a circularity problem: By embedding the hash in the text as a watermark, we change the text, so that it is inevitably no longer represented by that hash. Consequently, it is not tamperproof, because a change of the text by an attacker is in principle indistinguishable from the change resulting from the embedding. Note also that we reject, as many researchers do, making the hash obvious: among other things, it is rejected in business models as the visible expression of distrust. Making the hash obvious would, of course, make our and most other tamperproofing proposals redundant.

One straightforward way to use our marking scheme for tamperproofing, is to simply manipulate the text so that every sentence says the same secret bitstring (e.g., β zeroes for some small β). While easily achievable with our marking method, this scheme has two major drawbacks: It is impervious to the deletion of whole sentences, as the remainder will still appear tamperproof; and a modification of a sentence has a $2^{-\beta}$ probability of succeeding in being undetected. In the design we give below, the probability that removing or modifying a sentence goes undetected is $2^{-\beta(n+1)}$. Here β is chosen to be fairly small, so we are practically certain of being able to put β watermark bits in a sentence (even $\beta=2$ is fine, as the probability of an undetected modification of a sentence is then $2^{-2(n+1)}$, which is one in a million even in a nine-sentence text).

As mentioned earlier, we make two passes over the sentences. The first pass examines the sentences in the same secret ordering of them that we described earlier. The second pass examines them in the reverse order of the first. To avoid cumbersome notation, we assume in what follows that we have re-numbered the sentences according to their secret ordering, so that s_i is the *i*th sentence according to the secret ordering and T_i is the representation of its associated tree (a TMR tree in the first pass, a syntax tree in the second pass). Note also that this takes care of the risk of last-sentence tampering.

Implementation note: In what follows, when we refer to semantic or syntactic watermarking schemes, we mean a deliberately “fragilized” version of each—for example, in the syntactic scheme a syntax tree T_i’s leaves now do contain the exact words associated with these leaves, so that synonym substitution is detected.

First pass: The first pass is a semantic marking scheme. We need to state precisely which β bits are to be the inserted as a “mini-watermark” in each sentence. Let H denote a keyed hash function. We do the following:

We compute $x_1 = H_k (1\ldots1)$ = the keyed hash of all 1s (e.g., 100 ones).

We insert (as watermark) in s_1 the leftmost β bits of x_1.

Then for $i = 2, \ldots , n$ we do the following:

We compute $x_i = H_k (x_{i-1} , T_{i-1})$

where x_{i-1} , T_{i-1} denotes the concatenation of x_{i-1} and T_{i-1}, and T_{i-1} is the TMR tree obtained from the already marked version of sentence s_{i-1}.

We insert (as watermark) in s_i the leftmost β bits of the just-computed x_i.

The verification phase that corresponds to the first pass is a similar pass, except that instead of inserting β watermark bits in an s_i we instead read them and compare them to the leftmost β bits of x_i. Because of the “forward chaining” from 1 to n, the probability that a modification of s_i goes undetected by this “first pass verification” is

$$2^{-\beta(n-I+1)}.$$

Second pass: The second pass is a syntactic marking scheme (so it does not change any of the TMRs resulting from the first pass). We need to state precisely which β bits are to be the inserted as a “mini-watermark” in each sentence. As before, H_k denotes a keyed hash function. We do the following:

We compute $x_n = H_k (1\ldots1)$ = the keyed hash of all 1s (e.g., 100 ones).

We insert (as watermark) in s_n the leftmost β bits of x_n.

Then for $i = n–1, \ldots , 1$ we do the following:

We compute $x_i = H_k (x_{i+1} , T_{i+1})$.

where x_{i+1} , T_{i+1} denotes the concatenation of x_{i+1} and T_{i+1}, and T_{i+1} is the syntax tree obtained from the already marked version of sentence s_{i+1}.

We insert (as watermark) in s_i the leftmost β bits of the just-computed x_i.

The verification phase that corresponds to the second pass is a similar pass, except that instead of inserting β watermark bits in an s_i we instead read them and compare them to the leftmost β bits of x_i. Because of the "backward chaining" from n to 1, the probability that a modification of s_i goes undetected by this "second pass verification" is

$$2^{-\beta\square} .$$

The probability that a modification to s_i escapes detection by both the first-pass verification and the second-pass verification is therefore:

$$2^{-\beta(n-i+1)}\,2^{-\beta\square} = 2^{-\beta(n+1)}$$

which is as we claimed it to be.

References

[1] Atallah, M. J., V. Raskin, M. Crogan, C. F. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik 2001. Natural Language Watermarking: Design, Analysis, and a Proof-of-Concept Implementation. In: I. S. Moskowitz (ed.), Information Hiding: 4th International Workshop, IH 2001, Pittsburgh, PA, USA, April 2001 Proceedings. Berlin: Springer, 185-199.

[2] Atallah, M. J., C. J. McDonough, V. Raskin, and S. Nirenburg 2001. Natural Language Processing for Information Assurance and Security: An Overview and Implementations. In: M. Shaeffer (ed.), NSPW '00: Proceedings of Workshop on New Paradigms in Information Security, Cork, Ireland, September 2000. New York: ACM Press, 51-65.

[3] Nirenburg, S., and V. Raskin 2003. Ontological Semantics. Cambridge, MA: MIT Press (forthcoming). Pre-publication draft, http://crl.nmsu.edu/Staff.Pages/Technical/sergei/ /book/index-book.html.

[4] Katzenbeisser, S. C. 2000. Principles of Steganography. In: S. Katzenbeisser and F. A. P. Petitcolas (eds.), Information Hiding. Techniques for Stenography and Digital Watermarking. Boston: Artech, 17-41.

[5] Brassil, J., N. F. Maxemchuk, and L. O'Gorman 1994. Electronic Marking and Identification Technique to Discourage Document Copying. Proceedings of INFOCOM '94, 1278-1287.

[6] Maxemchuk, N. F. 1994. Electronic Document Distribution. AT&T Technical Journal, September/October, 73-80.

[7] Low, S. H., N. F. Maxemchuk, and A. M. Lapone 1998. Document Identification for Copyright Protection Using Centroid Detection. IEEE Transactions on Communication 46(3), 372-383.

[8] Wayner, P. 1992. Mimic Functions. Cryptologia XVI(3), 193-214.

[9] Wayner, P. 1995. Strong Theoretical Steganography. Cryptologia XIX(3), 285-299.

[10] Chapman, M., and G. Davida 1997. Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text. Proceedings of the International Conference on Information and Communication Security. Lecture Notes in Computer Sciences 1334. Berlin: Springer, 333-345.

[11] Anderson, R. J., 1996. Stretching the limits of steganography. In: R. Anderson (ed.), Information Hiding. First International Workshop. Cambridge, UK, May June 1996. Proceedings, Lecture Notes in Computer Science 1174, Berlin: Springer, 39-48.

[12] Anderson, R. J., and F. A. P. Petitcolas 1998. On the limits of steganography. IEEE Journal of Selected Areas in Communications 16:4, 474-481.

Appendix: Sample Text

U.S. Carpet-Bombs Taliban; Western Bridges Guarded / Last Updated: November 02, 2001 03:51 PM ET
By Alan Elsner and Mike Collett-White

WASHINGTON/RABAT, Afghanistan (Reuters) - The United States on Friday carpet-bombed Taliban front lines in Afghanistan and dispatched two new spy planes to pinpoint targets, while at home troops guarded California bridges against new terror attacks. // The anthrax scare spread abroad. One letter in Pakistan was confirmed to contain spores of the deadly bacteria but initial fears that the germ warfare weapon had also spread to Germany appeared to be a false alarm. // "We're slowly but surely tightening the net on the enemy. We're making it harder for the enemy to communicate. We're making it harder for the enemy to protect themselves. We're making it harder for the enemy to hide. And we're going to get him and them," Bush said. // The United States has been attacking Afghanistan for almost four weeks to root out the ruling Islamic fundamentalist Taliban and their "guest", Saudi-born militant Osama bin Laden, whom Washington accuses of masterminding the Sept. 11 attacks on New York and Washington that killed almost 4,800 people. // The Pentagon ordered two new spy planes, including the unmanned "Global Hawk", to the region to start flying over Afghanistan. // But Navy Rear Adm. John Stufflebeem said freezing rain was hampering efforts to fly additional elite U.S. special forces troops into the country to join the handful already there.

The United States added 22 groups including Hamas and Hizbollah, which have taken responsibility for suicide bombings against Israeli civilians, to the list of "terrorist" groups under tight financial controls introduced after Sept. 11. // Americans were on top alert on Friday, after warnings of new terror attacks in the coming week, and California beefed up security around several bridges, including San Francisco's landmark Golden Gate Bridge in the light of what it called credible threats of a rush hour attack on November 2.

[...]

ATTACKS "SPOT ON"

An opposition commander, Mustafah, watched from his roof near the front as B-52s unleashed their second day of carpet-bombing this week. He said he saw flames and smoke rise from positions used by the Taliban to shell the opposition-held Bagram air base. // "These attacks are spot on," another opposition commander, Rellozai, said from a rooftop vantage point. // Washington's strategy has focused on promoting a broad-based alliance incorporating the Northern Alliance, the majority Pashtun, from which the Taliban draw their support, and other ethnic groups across the impoverished country. // The effort suffered a blow last week when the Taliban captured and executed one prominent Afghan opposition leader, Abdul Haq, who had sneaked into the country to organize resistance to the Taliban. // In southern Afghanistan, Taliban fighters chased Hamid Karzai, a supporter of ex-King Zahir Shahon, on a mission to rally opposition to the Muslim militia. The former minister fled to the hills after his base was overrun.

Hamid's brother, Ahmed Karzai, said Hamid was fine. He said his brother had been holding a meeting with tribal leaders when he was attacked. // The brother said Hamid had more than 100 fighters with him on his mission that he said was trying to find support for a broad-based government to be formed after a Loya Jirga, or grand council, of Afghans in a post-Taliban Afghanistan. // The Taliban, who have imposed their own strict interpretation of Islam on Afghanistan, say 1,500 people have been killed since the U.S.-led air campaign began. There is no independent confirmation of the figure, which Washington says is grossly exaggerated.

LEADERS SAFE

The Taliban consul in Karachi said on Friday that the movement's leader, Mullah Mohammad Omar, as well as bin Laden were safe. // Moulvi Rahamatullah Kakazada told Qatar's al-Jazeera television that "thank God all brothers inside are protected from any harm especially the prince of believers (Mullah Omar) and Sheikh Osama bin Laden."

The governor of the war-battered city of Kandahar, where rubble from flattened houses litters the streets, said the Taliban movement would survive even if U.S. forces killed Mullah Omar. // "We are Muslims, we are an organization. These things do happen, but one person can be replaced by others," he said.