Authorship Proof for Textual Document

J. Wu* and D. R. Stinson**

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada

Abstract. In this paper, we investigate the problem of how to prove the authorship of textual documents. First we define the basic functionalities of an authorship proof scheme (APS) based on natural language watermarking, and identify two essential security requirements for an APS to be secure against various attacks. We review existing natural language watermarking schemes, and we propose two new schemes with improved security.

1 Introduction

The Internet has provided a new publishing channel for writers. Now many writers put their work online before officially publishing them in traditional media. Such practice benefits the writers by allowing feedback from readers and by promoting their works to publishers. This creates a new problem about protecting authors' rights, specifically, their authorship. Usually authors do not care if their work is reprinted online elsewhere, as long as the correct authorship is stated. However, someone may copy a text and reprint it online, or publish it in official channels, in his or her own name. Such cases have been observed. When this happens, it is difficult for the true authors to dispute their authorship.

The same problem for digital multimedia such as image, audio, and video can be solved by using digital watermarking techniques. Some researchers have tried to solve the problem for textual documents with similar approaches. One approach is to transform a textual document to an image, and then apply image watermarks [5], [4], [9]. Another approach embeds information in a text by modifying the appearance of its elements such as the fonts, space between lines and words, etc. (see [7], [6], [11], [13]). A common problem of the above two approaches is that the watermark can be erased by reformatting or re-typing the text document.

In this paper we use natural language watermarks (see [18], [17], [2], [19], [1], [8], [10]) to solve the problem. Our contributions in this paper are as follows:

- Identify and define the properties of an authorship proof scheme (APS) (section 2).
- Investigate existing natural language watermark schemes that are possible candidates for APS (section 3).
- Propose two schemes that are robust against watermark erasure attacks (section 4).

In addition, we discuss remaining problems and further improvements for APS in section 5.

2 Authorship Proof Scheme

2.1 Functionalities and Security

First we identify the basic functionalities of an APS, the components it must have, the attacks it may be confronted with, and the security properties it has to provide. We assume that an author has several texts to protect, and the author has one master key, K. For each of the texts T, the author can generate a secret key K_T from K and T, e.g., $K_T = h(K||T)$ where $h()$ is a cryptographic hash function and $K||T$ is the concatenation of K and T. An APS enables the author to watermark T and prove his or her authorship on the watermarked text. The APS consists of two processes:

* research supported by an NSERC post-graduate scholarship
** research supported by NSERC discovery grant 203114-06

1. $E(K_T, T)$, the *embedding process*, which takes the *text* T and the *secret key* K_T as inputs, and outputs the *watermarked text* T_w which is embedded with information about K_T;
2. $V(K_T, T_w)$, the *verification process*, which takes T_w and K_T as inputs, and outputs *true* if T_w is generated by $E()$ using the same K_T, otherwise it outputs $false$.

Given the above APS, we identify several possible attacks. We assume an adversary with polynomially bounded computational ability, i.e., the adversary cannot break secure cryptographic hash functions. The adversary has several copies of watermarked texts from one author. The attacks are as follows.

1. The adversary may try to find out the author's master key, K.
2. The adversary may try to extract a specific secret key, K_T.
3. The adversary may try to declare authorship on a T_w by computing a K' such that $V(K', T_w) = true$ while $K' \neq K_T$ (we call this an *impersonation attack*).
4. The adversary may try to erase the watermark in T_w by making some changes on T_w to produce T'_w which has essentially the same content as T_w, but where $V(K_T, T'_w) = false$ (we call this an *erasure attack*).
5. The adversary may watermark a text T_w with his or her own key to produce a T'_w and declare authorship on T'_w (we call this a *double watermark attack*).

For the above APS, K is secure since it is only used as a seed to compute some hash values K_T using a cryptographic hash function. Since each text T will be different, the keys $K_T = h(K||T)$ can be regarded random and independent of each other. It is not difficult to show that knowing some other watermarked texts does not help to attack a certain T_w or its K_T. Therefore we only need to analyze, when an adversary is given one T_w, if he or she is able to find out the corresponding K_T, or succeed in an impersonation attack, erasure attack, or double watermark attack on the given T_w.

Next we show that if an APS satisfies the following two basic security requirements, then it is secure against all the attacks described above:

1. Low false negative probability
   A false negative occurs if an adversary can edit T_w by inserting/deleting/modifying some sentences to produce T'_w such that $V(K_T, T'_w) = false$. False negative probability indicates robustness of the embedded watermark.
2. Low false positive probability
   A false positive is the event that $V(K', T_w) = true$, while $T_w = E(K_T, T)$ and $K' \neq K_T$.
   False positive probability must be very small to prevent impersonation attacks. In this paper, we stipulate that false positive probability is less than $\epsilon = 1/2^{56}$, which is the security level of a well-chosen password consisting of eight ASCII characters.

Low false negative probability implies the embedded watermark is secure (otherwise the adversary can easily erase the watermark with certainty), which in turn implies K_T is secure (otherwise the adversary can find out the watermark using K_T). In the case of a double watermark attack, the adversary produces $T'_w = E(K', T_w)$ using his or her own K'. The adversary can prove authorship on T'_w but not on T_w (because of the low false positive probability), while the true author can prove authorship on both T_w and T'_w (because of the low false negative probability). This indicates that T'_w is produced by modifying T_w, which resolves the dispute.

Since low false negative probability and low false positive probability indicate the security of an APS against all the attacks we have identified, in the following parts, we only need to analyze these two properties of an APS.

2.2 APS Evaluation

For an APS, its false positive probability must be lower than the pre-defined security level ϵ, but we do not mind if the false positive probability of one scheme is lower than that of another, as long as they are both

lower than ϵ. That is, as long as the schemes are secure, it does not matter so much if one is more secure. For false negative probability, we hope that it is as low as possible. Therefore, to compare two schemes, first we tune their settings so that their false positive probabilities are all close to and lower than ϵ, then we compare their false negative probabilities to see which one is more robust.

3 Related Works

3.1 Natural Language Watermarking

Natural language watermarking embeds information using linguistic transformations such as synonym substitution, syntactic transformations and semantic transformations (see [18], [17], [2], [19], [1], [8], [10]). It embeds a bitstream as a watermark in a text document, while preserving the meaning, context, and flow of the document. The modifications to the text are imperceptible. A natural language watermark scheme consists of two processes, a watermark embedding process and a watermark extraction process. The rightful owner has the original text and a secret key. The embedding process takes the text, the key, and some watermark information as inputs and produces a watermarked text, which is published. The watermark can be extracted from the watermarked text only with the knowledge of the key. The basic requirement of these schemes is that the watermark should not be perceivable, otherwise an adversary can easily erase the the watermark. As general-purpose watermark schemes, some other requirements, e.g., resistance to collusion attacks, are also considered in such schemes.

3.2 Previous Schemes

We briefly analyze the robustness of existing natural language watermarking schemes that can be used for authorship proof. There have only been a few such schemes, e.g., [2], [1] and [8], that are suitable for this purpose. In [2] Atallah *et al.* treat a text as a collection of sentences. The syntactic structure of a sentence is used to embed the watermark. Suppose a text T consists of m sentences, and an n-bit watermark is to be embedded in n sentences (i.e., we have one watermark bit for each of the *watermarked* sentences). A secret *rank* is computed for each sentence in T. The n sentences with least ranks are chosen as *markers*. Each sentence following a marker is transformed to carry one bit of the watermark. Both rank computing and watermark insertion are based on a secret key. Without the key, the probability that an adversary can forge a text with a valid n-bit watermark is 2^{-n}, i.e., the false positive probability of this scheme is 2^{-n}. When the adversary tries to erase the watermark, he or she cannot locate the markers or the watermarked sentences without the key. So the adversary can only randomly insert/delete/modify some sentences. When an inserted sentence happens to have a small rank, or it happens to be inserted right after a marker, the watermark will be damaged. When a deleted sentence happens to be a marker or a watermarked sentence, the watermark will also be damaged. When a modified sentence is a marker or a watermarked sentence, or the rank of the modified sentence becomes one of the n least ranks, the watermark will be damaged too. The probability that one random sentence insert/delete/modify operation can destroy the watermark is no more than $3n/m$. Therefore, when c sentences are inserted/deleted/modified, the false negative probability is $1-(1-\frac{3n}{m})^c$. For example for $n = 56$, $m = 2000$, when the adversary randomly edits eight sentences, the false negative probability is 0.5, i.e., there is a 50% chance the watermark will be erased.

A follow-up paper [1] improves upon [2] in terms of the number of watermark bits embedded in one sentence. The algorithms used in [1] are the same as in [2], but the algorithms are applied to the sematic structure of a sentence, instead of the syntactic structure, to embed the watermark bits. Since the semantic structure is much larger and richer than the syntactic structure, this allows transformations of a large number of elements they contain, and therefore it achieves more watermark bits in one sentence. In view of robustness, the probability that one sentence insert/delete/modify can destroy the watermark in [1] is the same as [2], therefore its false negative probability is also the same as [2].

In [8] Gupta *et al.* proposed a scheme to improve the robustness. The scheme treats the text as a collection of paragraphs, each of which consists of number of sentences. In the scheme, the paragraphs

are ordered according to the number of the sentences in it. Although the scheme is robust against various possible attacks, these attacks do not cover all possible attacks (from a cryptanalyst's point of view) which can effectively erase the watermark, e.g., insert/delete a paragraph, or insert/delete sentences in a paragraph such that its order is changed. In such cases, the false negative probability is rather high.

To the best of our knowledge, the most robust APS we can find in previous works are the ones from [2] or [1]. We use the scheme in [2] as an example and refer to it as APS1 hereafter.

4 Proposed Schemes

In this section we propose two new robust authorship proof schemes, APS2 and APS3. The building blocks of our schemes include:

- **Cryptographic hash functions.**
  A cryptographic hash function takes a variable length input and produces a fixed length output which can be considered to be randomly uniformly distributed on the function domain.
- **Text-meaning representation.**
  A *text meaning representation* (TMR) is a language-independent description of the information conveyed in natural language text. The theory and methodology of TMR have been studied and applied in machine translation systems. For more about TMR, we refer to [15], [14], [3], [16]. In this paper, we do not go into the details of TMR. We simply assume the existence of a TMR system $M()$ which takes as input a sentence s and outputs its *meaning representation* $M(s)$. For literally different sentences which have the same meaning, their meaning representations are the same, so when s is rewritten to s', which is literally different but has the same meaning, its meaning representation does not change, i.e., we have $s' \neq s$, but $M(s) = M(s')$.
- **Edit distance.**
  The *edit distance*, also named *Levenshtein Distance*, between two strings is the minimum number of character insert/delete/modify operations on one string to produce the other string. We use $D_{Lev}(S_1, S_2)$ to denote the edit distance between the strings S_1 and S_2. Given two strings, their edit distance can be computed using dynamic programming algorithms [12].

4.1 APS2

Following are the parameters and functions defined in APS2:

T: a text document consisting of m sentences $s_0, \cdots, s_{m-1}$.
$\mathcal{K} = \{0,1\}^k$: the key set.
$h_0 : \{0,1\}^* \rightarrow \mathcal{K}$, a hash function to generate a key.
$h_1 : \mathcal{K} \rightarrow \{0,1\}^n$, a hash function with n-bit output.
$h_2 : \mathcal{K} \times \{0,1\}^* \rightarrow \{0,1\}^{160}$, a keyed hash function with 160-bit output.
$h_3 : \{0,1\}^* \rightarrow \{0,1\}$, a hash function with 1-bit output.
d_{max}: a pre-defined threshold. If the edit distance between a damaged watermark and the original watermark is no greater than d_{max}, then the damaged one is considered valid.

We note that h_0, h_1, h_2 and h_3 can be easily constructed from standard cryptographic hash functions such as SHA-1.

APS2 Watermark Embedding. Given an original text document T, the author computes $K_T = h_0(K||T)$ where K is the master key, then performs the following embedding process:

1. Compute an n-bit pseudo-random string $W = h_1(K_T)$. $W = w_0 \cdots w_{n-1}$ is the watermark to be embedded.

2. Compute $o_i = h_2(K_T, M(s_i))$ for each sentence s_i. Here o_i is used as a secret *rank* of s_i. For simplicity, we assume each sentence has a different meaning, and therefore there will not be conflicting o_i values.
3. In the text T, rewrite the n sentences $\{s_{j_0}, \cdots, s_{j_i}, \cdots, s_{j_{(n-1)}}\}$ with least rank to $\{s'_{j_0}, \cdots, s'_{j_i}, \cdots, s'_{j_{(n-1)}}\}$ such that $h_3(s'_{j_i}) = w_{j_i}$ and $M(s_{j_i}) = M(s'_{j_i})$. The produced textual document T_w is then made public. After s_{j_i} is rewritten to s'_{j_i}, its literal expression is changed such that the watermark bit w_{j_i} is "embedded", meanwhile, its meaning does not change, and hence ranks of the rewritten sentences do not change, either.

APS2 Authorship Verification. To prove the authorship of T_w, the author presents K_T and runs the verification process $V(K_T, T_w)$ as follows:

1. compute $o_i = h_2(K_T, M(s_i))$ for each sentence in T_w.
2. compute an $(n + d_{max})$-bit string W' by hashing the $n + d_{max}$ sentences with least o_i values using h_3.
3. If there is a prefix W'' of W' such that $D_{Lev}(W'', h_1(K)) \leq d_{max}$, then output $true$, otherwise output $false$. Clearly $n - d_{max} \leq |W''| \leq n + d_{max}$.

The prefix search in step 3 guarantees that if the watermark is damaged by at most d_{max} insert/delete/modify bit operations, the verification process outputs *true*, otherwise it outputs *false*.

Example. Here we give a small toy example to show how APS2 works. In this example, $n = 4$ and $d_{max} = 1$. We use the first paragraph in Section 1 as the text T. There are eight sentences in the paragraph, which are denoted as $s_1, \cdots, s_8$ respectively. Let $h_1(K_T)$ be the last four bits of SHA-1(K_T), $h_2(K_T, s)$ be the last four hex digits of SHA-1(K_T, s) and $h_3(K_T, s)$ be the last bit of SHA-1(K_T, s).

First we define a TMR based on the following synonym dictionary:

1	provide	offer	2	new	novel
3	channel	outlet	4	nowadays	today
5	many	lots of	6	put	place
7	benefit	assist	8	feedback	response
9	promote	advertise	10	meanwhile	at the same time
11	problem	issue	12	specifically	especially
13	usually	generally	14	care	worry
15	correct	true	16	however	but
17	someone	somebody	18	official	formal
19	case	situation	20	observe	notice
21	happen	take place	22	hard	difficult
23	dispute	argue			

Each row of the table contains a set of two synonyms and a number as their meaning representation. The TMR of a sentence is defined as a string in which a word is replaced by its meaning representation. For example, the TMR of the sentence, *"The Internet has provided a new publishing channel for writers"*, is *"The Internet has 1 a 2 publishing 3 for writers"*.

Suppose K_T computed for this text is 123. Then the watermark is $W = 1111$.

Next the author computes the ranks of the sentences using h_2 and the TMR defined above. Suppose the results are (in hex format) dd21, dab6, 298c, 1b27, 3064, 2539, 04f0, and 6bc9 respectively. The seventh, fourth, sixth and third sentences are selected to embed the four watermark bits since they have the least ranks.

The author then rewrites the four sentences (if necessary) to embed the watermark. For example, the author checks $h_3(K_T, s_3)$. Since the result is 0, the author replaces *feedback* with *response*. Now the result is 1, which means the watermark bit is embedded. The outputted T_w is as follows:

(1) The Internet has provided a new publishing channel for writers. (2) Nowadays many writers put their works online before officially publishing them in traditional media. (3) Such practice benefits the writers by

allowing **response** *from readers and by promoting their works to publishers. (4) Meanwhile it incurs a new problem about protecting the right of the author, specifically, the authorship. (5) Usually the authors do not care if their works are reprinted online elsewhere, as long as the correct authorship is stated. (6) However, someone may copy the text and reprint it online, or publish it in formal channels, in his or her own name. (7) Such* **situations** *have been observed. (8) When this happens, it is hard for the true authors to dispute for their authorship.*

We include the numbers of the sentences in the above paragraph for easy reading. In the third sentence, *feedback* is replaced with *response*. In the seventh sentence, *cases* is replaced with *situations*. The resulting T_w is published.

Suppose the adversary rewrites the first sentence as *The Internet has offered a novel publishing channel for writers*, and deletes the seventh sentence.

In the verification process, the author supplies the key $K_T = 123$ to the versification process. Since $n + d_{max} = 5$, the five sentences in T_w with least ranks, i.e., sentences 4, 6, 3, 5 and 8, are selected. A five-bit string 111xx (x could be either 0 or 1) is extracted using h_3. A prefix, 111, and the watermark 1111 have an edit distance at most 1, so the verification process will output *true*.

Remark on the example. Note we provide a very simple TMR to make a concrete example of our scheme. TMR is a research topic in natural language processing and is not the focus of this paper. In practice, we may need TMR more sophisticated than that in the example in order to support desired transformations.

APS2 Watermark and Key Secrecy. Given T_w, the adversary can compute $h_3(s_i)$ for each sentence. The string $\{h_3(s_i)\}$ includes the watermark which is a pseudo-random string generated by $h_1(K_T)$. For the sentences not rewritten in the watermark embedding process, their h_3 values can be regarded as random bits. It is not difficult to show that, given T_w, it is infeasible for the adversary to tell what is the watermark, which sentences carry the watermark, nor can the adversary infer any information about the key K_T.

APS2 False Positive Probability. Since the adversary cannot infer any information about K_T, when trying to impersonate the true author, he or she cannot do better than present a randomly chosen K'. The adversary wins if $V(K', T_w) = true$.

Now we consider if the adversary presents a random K', when $K' \neq K_T$, what the probability is that $\Pr[V(K', T_w) = true]$. Let $\mathcal{A}_l = \{S : |S| = l, D_{lev}(S, W) \leq d_{max}\}$, the set of strings whose lengths is l, and whose edit distances from $W = h_1(K')$ are at most d_{max}. Any string in $\mathcal{A}_l$ can be produced by applying $d \leq d_{max}$ insertion/deletion/modifcation operations on W. We can assume these operations affect disjoint bits. Also it is easy to check that the order of the operations does not affect the result. Given n, l, and d_{max}, we have

$$|\mathcal{A}_l| \leq \sum_{d_i - d_d = l-n, d_i + d_d + d_m \leq d_{max}} \binom{n}{d_m + d_d} \binom{n - d_d + d_i}{d_i} 2^{di + d_d + d_m}. \quad (1)$$

In the above inequality, d_i can be interpreted as the number of insertions, d_d the number of deletions, and d_m the number of modifications. $\binom{n}{d_m+d_d}2^{d_m+d_d}$ is the number of ways to delete d_d bits and modify d_m bits in an n-bit string. $\binom{n-d_d+d_i}{d_i}2^{d_i}$ is the number of ways to insert d_i bits into an $(n - d_d + d_i)$-bit string.

If T_w is not produced by using K', then we can assume that $W = h_1(K')$ and the $n + d_{max}$ bits W' generated from T_w and K' are independent random strings. In the verification process, if any prefix of W' is in $\mathcal{A}$, the verification process will output *true*, and a false positive event happens. There are $2d_{max} + 1$ different prefixes of W' whose edit distances from W are possibly no more than d_{max}. The lengths of these prefixes range from $n - d_{max}$ to $n + d_{max}$. Therefore, the false positive probability is

$$P_{fp} \leq \sum_{n - d_{max} \leq l \leq n + d_{max}} \frac{|\mathcal{A}_l|}{2^l}. \quad (2)$$

APS2 False Negative Probability. Since given T_w, the adversary does not know which sentences carry the watermark nor what is the watermark, when the adversary changes T_w in order to erase the watermark, he or she can do it only in a random way . When the adversary inserts a new sentence, the probability that one bit is inserted into the watermark, which is the same as the probability that the rank of the new sentence is among the n least ranks of all the m ranks, is n/m. When the adversary modifies (without change the meaning[1]) or delete a sentence, the probability that one bit in the watermark is changed or deleted is n/m. So when c sentences are inserted/deleted/modified in T_w, the false negative probability of APS2 is

$$P_{fn} \leq 1 - \sum_{d=0}^{d_{max}} \binom{c}{d} \left(\frac{n}{m}\right)^d \left(1 - \frac{n}{m}\right)^{c-d}. \tag{3}$$

Other edit operations, such as moving/switching sentences, do not damage the watermark.

APS2 Parameters. In this part we use some concrete parameters to check the performance of APS2. Suppose $k = 100$ is fixed first. The false positive probability of APS2 is determined by n and d_{max}. Given m and c, the false negative probability of APS2 is determined by n and d_{max} too. Different pairs (n, d_{max}) may result in the same false positive probability and different false negative probability. So we hope to find the (n, d_{max}) such that false positive probability $\leq \epsilon$, while the false negative probability is the least.

We set $m = 2000, c = 200$, and compute the minimum n such that the false positive probability $\leq \epsilon = 2^{-56}$, and the corresponding false negative probability. The results are shown in Figures 1 and 2 respectively.

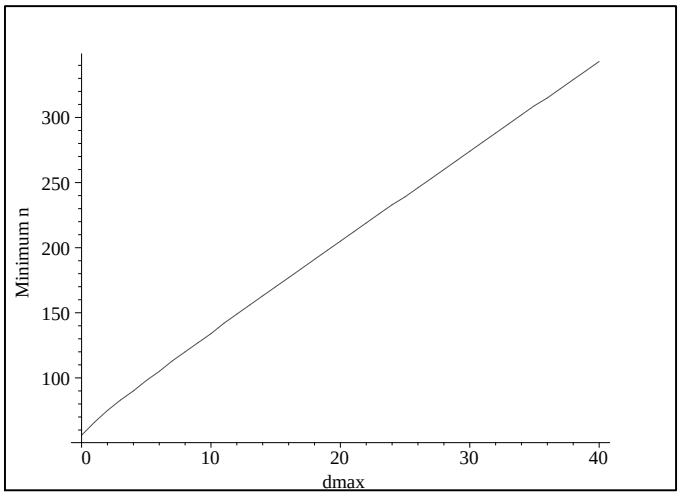

Fig. 1. APS2: minimum n that satisfies false positive probability $\leq \epsilon = 2^{-56}$ under different d_{max} when $m = 2000, c = 200$.

The results suggest that a large value of d_{max} is beneficial. In the above example, such an (n, d_{max}) pair, say (321, 40), results in both small false negative probability (0.06) and false positive probability ($< \epsilon$). We can use bigger d_{max} to achieve smaller false negative probability. The cost is a longer watermark, which means more effort is required to embed the watermark.

Now we set $n = 321, d_{max} = 40$ for APS2, and $n = 56$ for APS1. In such a setting, the false positive probabilities of both APS1 and APS2 are close and no more than ϵ. We compare their false negative probabilities as a function of c in Figure 3, which shows a significant improvement of APS2, as compared to APS1, in robustness.

[1] If a modification changes the meaning a sentence, it can be considered as one delete operation plus one insert operation.

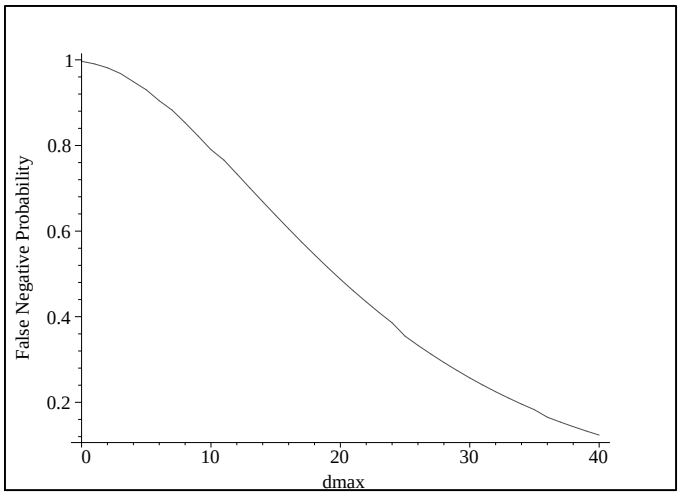

Fig. 2. APS2: false negative probabilities under different d_{max} when $m = 2000, c = 200$.

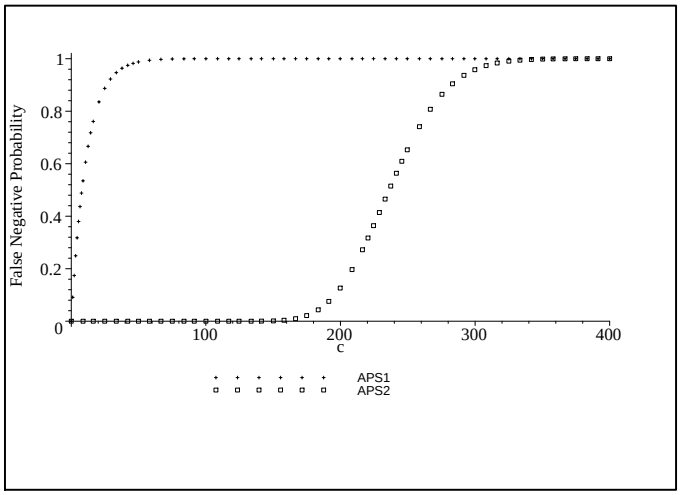

Fig. 3. False negative probabilities of APS1 and APS2.

Remarks on Design of APS2. The design of APS2 uses two approaches to improve the work in [2] and [1]. First, in the previous schemes, two sentences, a marker and a carrier, are necessary to carry one watermark bit. The reason that a marker itself cannot carry a watermark bit is that if it is changed to carry the watermark, its rank will be changed and is no longer a valid marker. We solve this dilemma by using TMR to compute the rank, and inserting the watermark bits in the marker sentences by changing their literal expression without changing their TMR. Thus, we not only lower the probability that the watermark is damaged when one sentence is inserted/deleted/modified (from $\frac{3n}{m}$ to $\frac{n}{m}$), but we also get rid of the watermark damage caused by moving/switching the sentences.

Another approach used in APS2 to improve the robustness is based on the simple idea of error-tolerance: when a small number of bits are damaged in a long watermark, the damaged watermark should still be regarded as valid. We use edit distance to quantify the extent to which a watermark is damaged. It turns out that this measure significantly improves the robustness of the scheme.

4.2 APS3

APS2 relies on TMR systems. In this section, we design a lightweight scheme, denoted APS3, without using TMR systems. APS3 uses the same building blocks as APS2, except that it does not use the functions $M()$ or $h_2()$.

APS3 Watermark Embedding. In APS3, the author first computes $K_T = h_0(K||T)$, then performs the following embedding process:

1. Compute an n-bit pseudo-random string $W = h_1(K_T)$, and choose a random number $p \in [0, m-1]$. $W = w_0 \cdots w_{n-1}$ is the watermark and p indicates where in T to insert W.
2. In T, starting from the p^{th} sentence s_p, rewrite the following n sentences such that $h_3(s_{p+i}) = w_i, 0 \leq i \leq n-1$. The resulting text T_w is then published.

Authorship Verification. To prove authorship on T_w, the author presents K_T and runs the verification process $V(K_T, T_w)$ as follows:

1. Compute an m-bit string S by hashing each sentence in T_w using h_3.
2. If S has a contiguous substring whose edit distance from $W = h_1(K_T)$ is no more than d_{max}, then output $true$, otherwise output $false$.
   Note that the value of p is not used in the verification process.

APS3 Watermark and Key Secrecy. When an adversary applies h_3 to each sentence in T_w, he or she gets an m-bit string S which contains the watermark W starting from the p^{th} bit in S. S consists of three parts: a p-bit random string (from bit 0 to bit $p-1$), followed by an n-bit pseudo-random string W, followed by an $m-p-n$ bit random string (from bit $p+n$ to bit $m-1$). Given such a string, assume h_1 is a secure pseudo-random bit generator, then it is infeasible for the adversary to determine which part is W, i.e., the watermark is imperceptible to the adversary.

In APS3, the watermark will be damaged only when the adversary deletes/modifies sentences in the n-sentence block that carries the watermark, or inserts sentences in that block. Since the adversary does not know where in T_w the block is, when the adversary tries to erase the watermark by editing T_w, he or she cannot do better than rewrite/delete randomly chosen sentences in T_w, or insert new sentences at random locations in T_w.

Note that unlike APS2, APS3 is not immune to sentence move/switch. Such attacks can be regarded as combination of multiple sentence insert/delete.

Since the adversary cannot find the watermark in T_w, he or she can compute nothing about the secret key K_T which is used to generate T_w (in fact, since h_1 is a secure hash function, even given the watermark $W = h_1(K_T)$, the adversary cannot find K_T).

APS3 False Positive Probability. Since the adversary knows nothing about K_T, to prove he or she is the true author of T_w, the adversary cannot do better than choosing a random K'. If $V(K', T_w)$ happens to output $true$, a false positive event happens and the adversary wins. Next we analyze the false positive probability of APS3.

In (1) we have computed $|\mathcal{A}_l|$, the number of strings whose lengths are l, and whose edit distances from W are at most d_{max} where $W = h_1(K')$. If any of the contiguous substrings of the m-bit string S is in any $\mathcal{A}_l$, a false positive event happens. Since there are at most m different contiguous substrings of any given length l in S, and only the substrings of length $n - d_{max} \leq l \leq n + d_{max}$ can possibly be in an $\mathcal{A}_l$, the false positive probability is

$$P_{fp} \leq \sum_{n-d_{max} \leq l \leq n+d_{max}} \frac{m|\mathcal{A}_l|}{2^l}. \quad (4)$$

APS3 False Negative Probability. As we have analyzed, the best way an adversary can damage the watermark is to delete/modify sentences randomly chosen from T_w, or insert new sentences at randomly chosen locations in T_w. Given m, n, d_{max}, and c, the false negative probability of APS3 is the same as that of APS2, i.e.,

$$P_{fn} \leq 1 - \sum_{d=0}^{d_{max}} \binom{c}{d} \left(\frac{n}{m}\right)^d \left(1 - \frac{n}{m}\right)^{c-d}. \tag{5}$$

APS2 and APS3 Comparison. We compare the minimum n to achieve false positive probabilities $\leq \epsilon$, and corresponding false negative probabilities of APS2 and APS3 in Figure 4 and 5 respectively. It shows with the same d_{max}, APS3 needs a larger value of n to achieve the same false positive probability, and the corresponding false negative probability is higher. It is consistent with the intuition that, since APS3 searches the watermark in a wider range, it needs a smaller d_{max} or a larger n to keep the same false positive probability. Then the smaller d_{max} or larger n results in a higher false negative probability.

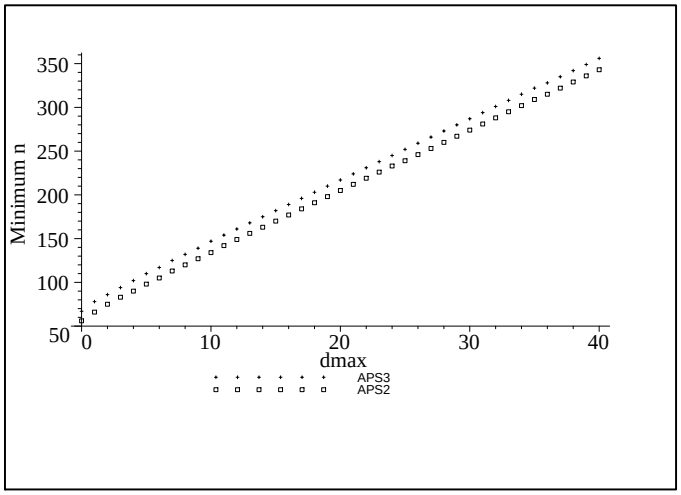

Fig. 4. Comparison of minimum n of APS2 and APS3 that satisfies false positive probability $\leq \epsilon = 2^{-56}$ under different d_{max} when $m = 2000, c = 200$.

Although APS3 is not as robust as APS2, it does not rely on TMR and is much simpler. It may be an alternative to APS2 in occasions where simplicity is more important.

5 Conclusion

In this paper, we investigated the problem of how to prove the authorship of a textual document. We identified and defined the general requirements for authorship proof schemes (APS). We studied existing natural language watermark techniques to solve the problem, and we proposed two new schemes which are more robust than the previous ones.

There are two techniques in our schemes which improved the robustness:

1. We used the meaning representation of a sentence to generate the rank, and we used the literal representation to embed the watermark bit. With this approach, we can use a sentence as both a marker and as a watermark carrier. Thus we reduced the probability that the watermark is damaged when one sentence is inserted/deleted/modified, and we eliminated the damage caused by sentence moving/switching.
2. We used edit distance for error-tolerant watermark search.

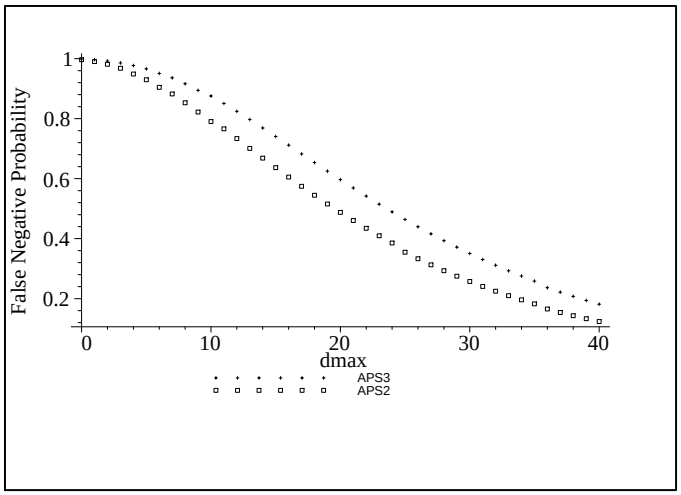

Fig. 5. Comparison of false negative probabilities of APS2 and APS3 under different d_{max} when $m = 2000, c = 200$.

These two techniques can also be applied to general purpose natural language watermark schemes to improve their robustness.

An issue for our scheme and most other natural language watermark schemes is that they are suitable only for long text. This is because the basic element to carry the watermark is a sentence. Designing schemes for shorter text would be of interest in further research.

References

1. M. Atallah, V. Raskin, C. F. Hempelmann, M. Karahan, R. Sion, U. Topkara, K. E. Triezenberg, Natural Language Watermarking and Tamperproofing, *Fifth Information Hiding Workshop, IHW 2002, LNCS 2578*, pages 196–212, Springer Verlag, Noordwijkerhout, The Netherlands, 2002.
2. M. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik, Natural language watermarking: design, analysis, and a proof-of-concept implementation. *In Proc. of 4th International Workshop on Information Hiding, IH 2001. LNCS, volume 2137*, pages 185-199. Springer-Verlag, Heidelberg, 2001.
3. S. Beale, S. Nirenburg, and K. Mahesh, Semantic Analysis in the MikroKosmos Machine Translation Project. *Proc. of the 2nd SNLP-95*, Bangkok, Thailand, 1995.
4. J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman, Marking text features of document images to deter illicit dissemination. *In Proc. of the 12th IAPR International Conference on Computer Vision and Image Processing*, volume 2, pages 315 - 319, Jerusalem, Israel, 1994.
5. J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Gorman, Hiding information in documents images. *In Conference on Information Sciences and Systems (CISS-95)*, 1995.
6. N. Chotikakamthorn, Electronic document data hiding technique using inter-character space. *In Proc. of The 1998 IEEE Asia-Pacific Conference on Circuits and Systems, IEEE APCCAS 1998*, pages 419-422, Chiangmai, Thailand, 1998.
7. N. Chotikakamthorn, Document image data hiding techniques using character spacing width sequence coding, *Proc. IEEE Intl. Conf. Image Processing*, Japan, 1999.
8. G. Gupta, J. Pieprzyk, and H.X. Wang, An attack-localizing watermarking scheme for natural language documents. *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. Taipei, Taiwan, 2006.
9. H. Ji, J. Sook, and H. Young, A new digital watermarking for text document images using diagonal profile. *In Proc. of Second IEEE Pacific Rim Conference on Multimedia, PCM 2001. LNCS, volume 2195*, pages 748 -, Beijing, China. Springer-Verlag, Heidelberg, 2001.
10. M.S. Kankanhalli and K.F. Hau, Watermarking of electronic text documents. *Electronic Commerce Research*, 2(1-2):169-187, 2002.

11. Y. Kim, K. Moon, and I. Oh, A text watermarking algorithm based on word classification and inter-word space statistics. *In Conference on Document Analysis and Recognition* (ICDAR03), 1995.
12. V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. Dokl. Akad. Nauk SSSR 163 845–848 (Russian); translated as Soviet Physics Dokl. 10 1965 707–710.
13. S. Low, N. Maxemchuk, J. Brassil, and L. O'Gorman, Document marking and identification using both line and word shifting. In *Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People, INFOCOM 1995*, volume 2, pages 853-860, Boston, USA, 1995.
14. K. Mahesh and S. Nirenburg, Meaning Representation for Knowledge Sharing in Practical Machine Translation. *Proe. the FLAIRS-96 Track on Information Interchange*. Florida AI Research Symposium, 1996.
15. S. Nirenburg, Application-Oriented Computational Semantics. *Computational Linguistics and Formal Semantics*, R. Johnson and M. Rosner (eds.), 1991.
16. B. Onyshkevych, An Ontological- Semantic Framework for Text Analysis. Ph.D. Diss., School of Computer Science, Carnegie Mellon University. 1997.
17. X. Sun, G. Luo , and H. Huang, Component-based digital watermarking of Chinese texts, *Proceedings of the 3rd international conference on Information security*, Shanghai, China, 2004.
18. M. Topkara, G. Riccardi, D. Hakkani-Tur, and M.J. Atallah, Natural Language Watermarking: Challenges in Building a Practical System, *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, 2006.
19. M. Topkara, C. Taskiran, and E.J. Delp, Natural Language Watermarking, *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, San Jose, CA, 2005.