Purdue University

Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

2007

This message will self-destruct: Self-easing covert communication

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Mercan Topkara

Umut Topkara

Report Number:
07-011

Atallah, Mikhail J.; Topkara, Mercan; and Topkara, Umut, "This message will self-destruct: Self-easing covert communication" (2007). *Department of Computer Science Technical Reports.* Paper 1675.
https://docs.lib.purdue.edu/cstech/1675

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

THIS MESSAGE WILL SELF-DESTRUCT: SELF-ERASING COVERT COMMUNICATION

Mikhail J. Atallah
Mercan Topkara
Umut Topkara

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #07-011
May 2007

This message will self-destruct: Self-erasing covert communication*

Mikhail J. Atallah Mercan Topkara Umut Topkara
Department of Computer Sciences
Purdue University
West Lafayette, IN, 47906, USA
mja,mkarahan,utopkara@cs.purdue.edu

May 11, 2007

Abstract

The WWW increasingly allows people to create and update content for public access. Some of this information is collaboratively owned (created and maintained), while other information is privately owned and maintained (but still publicly accessible). Whereas it is unethical to modify the former for covert communication, it is quite legitimate to do so with the latter, and this paper gives a design for doing so while achieving both plausible deniability and automatic perishability of the covert message (the message disappears unless periodically refreshed by the encoder). Traditional information-hiding has looked at the problem of embedding a message in a static version of an online document, the problem of doing so for rapidly evolving document collections has not been considered in the past. This paper shows that it is possible to do so, and in a manner that actually makes use of the rapidly evolving nature of the documents to achieve the above-mentioned property of evanescence: That the message decays over time and eventually becomes completely erased unless it is refreshed. Therefore the mark needs to be continuously maintained as the document evolves, in a manner that prevents the adversary from knowing who is doing the refreshing yet that allows the intended reader of the mark to recover it without any form of explicit communication. One advantage of our scheme is that the mark's reach is now unbounded: It can be read by any authorized entity on the web (anyone with the secret key), and the reading of it is indistinguishable from normal web access patterns. Another advantage is the "hiding in the crowd" effect: Many people are updating the documents, thereby providing a cover for the one person surreptitiously injecting and refreshing the mark, or replacing it with another mark message. We have also demonstrated the feasibility of the proposed technique, and shown that remarkably little effort is required to implement our scheme over today's web.

*Portions of this work were supported by Grants IIS- 0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security.

1 Introduction

Although messages that self-destruct were featured in spy movies, their potential usefulness is not limited to the original purpose of their self-destruction in such movies, which was to prevent their being read by a hostile adversary after they had served their useful purpose of communicating the next mission to Mr Phelps. The case for making such messages perishable includes many possible reasons: (i) Such messages can become stale and thereby convey misleading information to their intended recipient (e.g., the message "Alice and Bob are both doing fine" after one of them ceases to be doing fine) – more generally, because information is perishable and becomes stale, useless, or even dangerous as the world changes, it stands to reason that stealthy messages that convey that information should be similarly perishable and self-efface; (ii) if the message involves a secret key, then the longer it lingers, the more it is likely that it (and the key used to hide it) may eventually be compromised by an adversary who has enough computational resources (or will have such resources in the future - systems have to be resilient not only against the computational power of today but also against that of the future); (iii) the desired updating of the message (either refreshing it or replacing it with another message) by the person who wrote it may become infeasible through that person's accidental loss of the secret key, loss of access to the online world, or other physical inability to take such action; (iv) the automatic disappearance of the message can be used to communicate the very fact that (iii) has occurred; (v) the person in charge of removing the message may be negligent, or may erroneously believe that someone else was supposed to do the removal, etc.

Providing privacy preserving Web-based communication is an active research area [2, 7, 15]. Achieving this is hard because many players (such as authoritarian governments, aggressive marketers, etc) want to have a complete profile of Web users and a log of their actions on the Web. We propose a private communication channel, that ensures plausible deniability, and automatic perishability of the messages. We achieve this goal through the use of collaborative web content available on the Internet, without unethically interfering with the functionality of these valuable services, and without any need for modification of publicly available data (defined as data not owned by the sender of the message). In a nutshell, our scheme is based on pairing a privately owned web page with a collaboratively owned web page, and to use this pair as a cover document. The embedding changes are only performed on the privately owned web page. A more detailed summary of this scheme is given in Section 3.

There are many challenges involved in designing such a system:

- how to use, for marking purposes, the content that we cannot modify either because we have no control over it (e.g., news portals), or because it is unethical to use the ability to modify it for marking purposes (e.g., wikis, forums)
- heterogeneous content; most published marking schemes assume one type of content, whereas we are now faced with a semi-structured collection of different content types (text, images, audio, video, annotations, etc)
- not interfering with the proper functioning of the publicly owned covers used
- providing controlled perishability
- being stealthy while using a publicly accessible cover document (as is customarily required in information hiding)
- providing plausible deniability of the covert communication
- providing high covert communication bandwidth (especially challenging when the document consists of data that has low embedding capacity, such as text)

The difficulty of these challenges is exacerbated by the increasing power that a potential adversary can muster – the repertoire of information sources for such an adversary now includes forum or blog boards (most or least accessed pages, most active member etc.), web-bots, ISP logs, search engines, web page tracking engines[9, 6], to mention a few.

Our system fulfills most of the requirements listed for a Web based publishing system listed by Waldman et al. [15]: censorship-resistant, tamper evident, source anonymous, updatable, deniable, fault tolerant, persistent (i.e., no expiration date), extensible, freely available. As will become clear later, the only requirement we do not provide is persistency over time (which is inherently incompatible with a self-destructing message).

Previously proposed private communication systems use a third party distributor (e.g., e-mail services [2]) to store and distribute the message to intended receivers. In our system, the sender does not use a third party distributor and therefore has a greater degree of control. The sender also has the option of privately storing the cover document (until it is cached by another Internet company [11, 8]) if she/he wishes to maintain the privately owned web page that is used as part of the cover document.

Refer to Section 5 for more information about available systems.

In Section 2, we introduce the model of Web content used in the communication channel described in this paper. The system overview is provided in Section 3. The experimental framework and results are discussed in Section 4.

2 Document Models

Before going into the details, we give a brief overview of how our system works. In what follows, by referring to something as "secret" we mean that knowing it requires the key that is shared by the encoder and decoder of the message.

Perishability is achieved by a secret pairing of every encoder-controlled document (say, d) with a document $p(d)$ that is outside of the encoder's control; we consider a document to be outside the encoder's control if it is unethical or against the accepted etiquette for the encoder to use it for stealthy communication (so although a user may, for example, physically be capable of modifying collaborative content, it would be inappropriate to modify such content for the purpose of encoding). A document d that is within the encoder's control contains elements that we refer to as $e_1, \ldots, e_{l_d}$; if d contains too many elements for us to use, then we choose only a number l_d of them (less than the total available). What determines an e_i's role (e.g., pair-selection, mark-encoding, etc) is the keyed hash of it, denoted as $H(e_i)$. Some types of elements e_i have a payload, e.g., for a mark-encoding e_i the payload consists of the mark's bits that it encodes. Let e_i' be to $p(d)$ what e_i is to d (actually there is more to e_i' than that, but we defer this discussion to the section 3.2 on implementation details). If e_i has a payload, then that payload is determined by the keyed hash of the concatenation of e_i with with e_i': $H(e_i, e_i')$. A change in enough of the elements of $p(d)$'s will, over time, erase the message. See Figure 1.

Encodability is achieved through the latitude we have to modify the individual e_i's so they encode both their intended functionality and (if applicable) their appropriate payloads. This latitude includes, for rich content, a simple selection and re-ordering of content types, that typically can achieve 22 bits of encoding even without any modification to any of these rich types (see Section 2.1.1 for more details on this). If subset selection and ordering information for rich types is not enough, or not feasible because some sites disallow rich content or enforce rigid templates for it, then we resort to modification of the e_i's actual contents rather then their presence and sequence order: We modify some pixels of an image [4], replace words by synonyms using robust synonym substitution [14], judiciously inject typos [12] in domains where they are common enough (blogs, newsgroups), etc. We will later discuss desirable properties for $p(d)$.

There are 2 models for obtaining the e_i's from d, each applicable in a different domain. In free-format

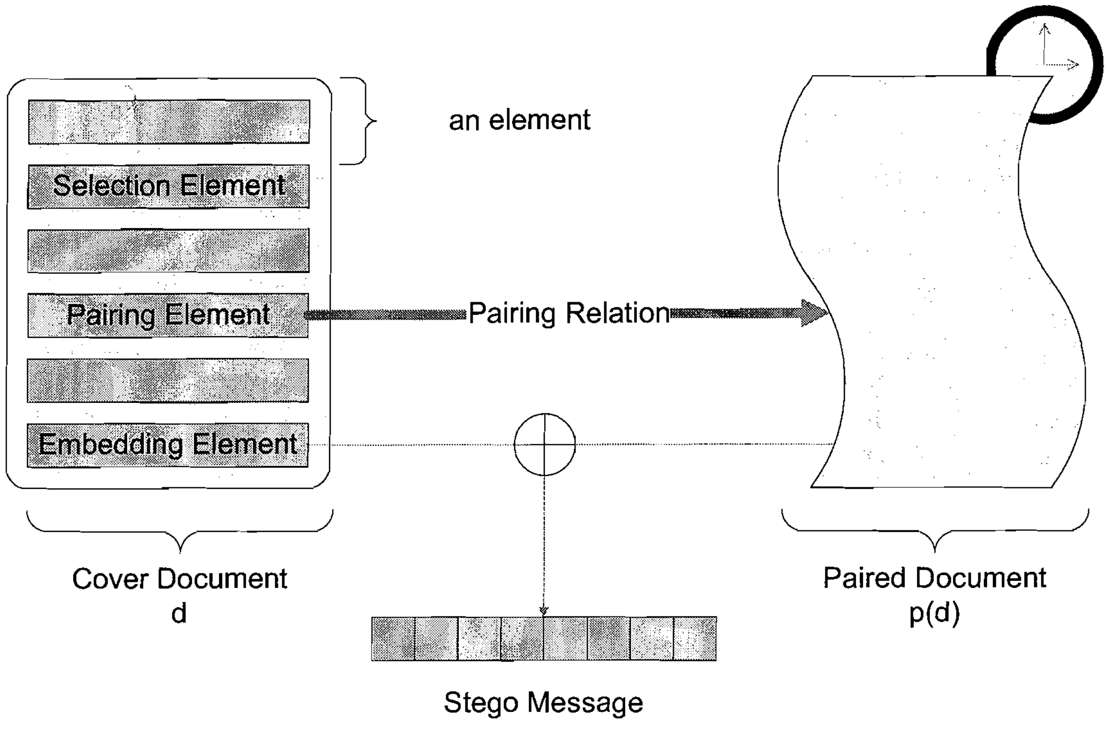


Figure 1: Pairing the elements of privately owned web page with the collaboratively updated web page for information hiding.

domains we can choose and modify the e_i's dynamically so as to maintain the desired pairing and encoding properties. But in append-only domains (e.g., forum posts, opinion pieces, etc) no modifications are allowed of the old versions of content: In these, we achieve the net effect of modifying a now-obsolete e_i roughly as follows. We append a new element e to d, making sure that $H(e)$ encodes for e the exact same functionality as e_i, thereby signaling to the decoder that the chronologically prior e_i is to be ignored and e used in its stead. That is, if successive hashes of all the e_j's of a particular d give more than one of the same functionality as another, the decoder always breaks the tie in favor of the more recent one.

As stated earlier, what constitutes the sequence of e_i elements is the chronological sequence of (possibly append-only, possibly update-able) posts, comments, essays, etc; so the next item posted would be any one of these logical units, let's call it e_i. Even in the most constrained case for us, when the e_i do not have rich content (e.g., only text and tags) we were always able to make that e_i (and, if applicable, its concatenation with $p(d)$) hash into what we want; in some sense, we "torture e_i until it confesses". In rare cases we resort to typos, but these are very common (almost expected) in such text-only informal material.

There might be cases (relatively rare) where the append operations have no provision for a chronological date and time cannot be handled as above; this occurs if, for example, free-flowing text is added without any information about which paragraph was added on which day (so that the web site shows only the current snapshot of the document). In that case we use a fixed number of items as our definition of what constitutes the next e_i, e.g., it could be the next k basic items from the available categories, where a basic item is a paragraph, or an image, etc. (so the next e_i could consist of the next k paragraphs of texts that we append, if only text is added). A too small value for k runs the risk of not being able to carry out enough modifications on e_i, whereas a too large value is wasteful.

2.1 Embedding Channels

As mentioned above, an e_i has an encoding capacity through making changes to it that modify $H(e_i)$ and (if it is used) $H(e_i, e_i')$. We now briefly review what kinds of changes to e_i are made. This depends on the domain, specifically whether it allows free-format rich content or not. We begin with the former.

2.1.1 Rich Content

Wikis are a popular way for collaborative content creation, which allow rich content creation. Inside a wiki website usually users can both modify the existing content of individual pages, and add new pages. The collective review and editing through such liberal content modification allows wiki sites to produce quality documents.

A rich content creation web site may offer a wide range of element types to contribute content:

- text in different styles
- headers of different levels
- figures
- quotes
- lists
- external links
- internal links
- tables
- references
- tags or categories

In high quality wikis there are style guides that sketch the general good editing practices. Clearly a wiki page that consists of several types of content can be typeset in a variety of ways while complying with the respective style guidelines. A good wiki page can consist of one of the more than 2^{22} different ordered permutations of the 10 types of content that were listed above. The 22 bits that were encoded while staying strictly within quality standards of the wiki can be used as a covert channel.

Another popular way of rich content creation is blogs. The blogs can be hosted by a large collection web site like wiki pages, but unlike wiki pages blogs are usually not strictly monitored for their content and style. Moreover individual blogs are usually under the control of one user.

This kind of content creation web sites provide an extra latitude for covert communication, since it is possible to communicate 22 bits of information just through the use of simple selection and re-ordering of content types even without any modification to any of these rich types.

2.1.2 Restricted Content

Some online content creation sites offer only one or a very limited number of different content types of contribution:

- tags (product tags)
- text (forum messages, comments, etc.)

- images (photographs)
- audio
- links (related links, bookmarks)

These sites are usually in the form of collections, and they restrict the users to appending new content and do not allow modification of existing content. Examples of these include online forums, newsgroups, product reviews, and product tags. History pages (pages that list the log of modifications on a page) of individual pages in rich content web sites (e.g. wikis) also exhibit an append-only characteristic, since older snapshots of the pages are fixed, and these history pages monotonically grow in size as new changes are made to the respective documents.

The append-only characteristic of these sites constitute an interesting challenge for covert communication. In Section 3 we explore this model further and give an algorithm for encoding and decoding using a covert channel based on it.

3 System Overview

The system consists of an algorithm that determines which of a document's n links are relevant to the mark and how they encode their relevant bit(s) of the mark (e.g. links could be different personal pages in a community web site). In our implementation, each message bit is encoded in a pair of key-selected links, *only one of which is controlled by the message encoder*: This pairing is necessary for the simultaneous achievement of the desired properties of encodability and perishability (self-effacing mark). Encoding a mark's bit using only a link under the control of the encoder would not achieve the desired perishability property, whence this "mixing" of what is under the encoder's control with a sibling that is beyond such control. Moreover, the rate of the perishability (does the mark vanish in days, or in weeks?) can be controlled by selecting a sibling that is rapidly changing (for faster decay) or slow changing (for slower decay). See Section 3.2 for further discussion on controlling the speed of message decay.

For the sake of definiteness we describe the algorithm for B bits per pair.

In this section, we will use the *append-only* content creation model to describe our system. However it is possible to generalize it to *free-form modification*, where the contributors are allowed to delete and modify existing content (see Section 3.4).

The algorithm takes as input (i) a document that has a set of n links (call this set U), m of which are in D and are designated (as being under the encoder's control), $m \leq n/2$; (ii) a keyed hash function H; and (iii) a r-bit mark message $w = b_1 \dots b_r$. Each link points to a document d with e_{l_d} elements (their tags, content or combination thereof). The algorithm outputs a judicious modification of the documents pointed to by designated links, such that a pairing operation using H only results in m pairs of selected links: One link of each pair is a designated link from our set, the other is its sibling, $p(d)$, chosen from one of the $n - m$ non-designated links. A decoder who has H but does not know the designated links can find out the same pairing, by applying H to all links in the document. The process of producing these pairs by the encoder takes place as follows (keep in mind that the pairing has to be producible by the decoder as well, who has the keyed hash function but otherwise has no a priori knowledge of which links are designated and which are not):

1. All of the designated links are made to acquire new elements from categories SELECTION, PAIRING, EMBEDDING and CANCEL according to their roles. The categorization of elements can be achieved by partitioning the 2 most significant bits of the hash values of the elements, e.g. 00 for SELECTION, 01 for PAIRING, etc.

In this section, for the sake of simplicity, we use a concatenation of the least significant bits of elements from the same type to generate larger bit strings. In practice we use more sophisticated information hiding methods to embed larger amounts of information into one element (e.g., image, audio, video [4], text [13, 12]).

2. The subset T of $|T| = r/B$ links out of D are marked by adding new SELECTION elements (until the probability that a non-designated link contains more SELECTION elements before the message decays is within acceptable limits). These new SELECTION elements signal the selected subset of links in U that encode the message to a decoder that only has the information of H. Let the set of non-selected links be $F = U - T$.

3. We modify PAIRING elements of each link l in T, and select a sibling for l from non designated links with the bit string they generate (i.e., by using the concatenation of the least significant bits of their hash value as a pointer or id). If a link in T is involved in a collision between its choice of sibling and the choice made by another link in T, we change some of the PAIRING elements until collision disappears and another desired sibling is selected.

4. The sorted order of the hash of concatenation of the SELECTION elements of links in T also provides a way of knowing which ones will encode which bits: The smallest in the sorted order will encode the first B bits of the mark, the second will encode the next B bits, etc, until all $B \times |T|$ bits of the mark are encoded (we consider the redundancy and error correction as being included in these $B \times |T|$ bits).

5. Go through each selected link in T according to the above-mentioned sorted order. For each of them do the following: add a set of EMBEDDING elements to the link so that the concatenation of the least significant bits of $H(e_i, e'_i)$ (i.e., hash of an EMBEDDING element e_i and a counterpart element e'_i of sibling) is equal to the B bits of the mark that it seeks to encode.

6. Later, if the sibling of a link l of T unexpectedly turns out to have the wrong mean-time-between-modifications characteristics (too small for a desired slow decay, or too large for a desired fast decay due to changes in public interest to the sibling document), then there are two options: i) the hash of the PAIRING elements are modified by appending new PAIRING elements such that their hash, as a seed, selects a suitable item from F (one that has the desired characteristic). Of course this has to be done subject to not creating a collision with the choice of another link in T. ii) if reusing l to accommodate a new pairing will cause exceeding a stealthiness threshold for its number of elements, l is de-selected using CANCEL elements. A new designated link can be selected to replace its place in encoding the bits of the message using the same SELECTION tags of l if desired.

The decoder operation consists of the following steps:

1. Compute T and F using the secret key and H, and then compute the sibling information (i.e pairing) $p(d)$ for each d in T.

2. Sort the documents in T according to the hash of their SELECTION elements.

3. Use the sorted order to read from each of the selected documents B bits that it encodes jointly with its sibling by computing $H(e_i, e'_i)$ for each EMBEDDING element.

SELECTION determines which of the given n links are used for encoding. PAIRING determines which one of the non-designated documents is paired with each document selected by SELECTION process. EMBEDDING determines the encoded message bit string by the selected link and its pair. CANCELING is used when there is a need for signaling the un-selected status of a previously selected document.

3.1 How the Mark Decays

Note the following characteristic of our system: Extensive changes to a non-designated link does not do more damage to the mark than more modest changes to that same link. Rather, the damage occurs from modifications done to many links (even if each of these modifications is modest) that, over time, eventually overwhelm the built-in redundancy and error-correction (which are needed for a practical operational consideration discussed below). The primary mechanism we use for controlling the lifetime of the mark (when it is not refreshed) is therefore not the amount of the redundancy and error-correction we introduce, but rather our choice of the characteristics of the sibling links – the frequency with which they are likely to be updated based on past observations of their behavior. What if that previously observed behavior changes against our expectation, e.g., a link that was rapidly changing when we selected it as sibling becomes largely dormant after a period of time. To avoid this damaging the perishability property of our mark, we refresh the mark so that a new sibling gets selected instead (one with a more appropriate mean-time-between-updates value).

3.2 Controlling the pace of perishability

It is quite likely that we may not find public documents that undergo changes at exactly the rate we need to achieve our targeted pace of perishability. This subsection discusses how we overcome a possible mismatch between our needs and what is available. We use two basic techniques: A *slowdown* technique for the case where the paired documents are too fast-changing for our needs, and a *speedup* technique for the opposite case where the paired documents are too slow-changing for our needs. We now discuss each of these in turn.

We made the assumption that changes to paired non-designated pages will localize at element level (e.g., a paragraph of text). This assumption is supported by analysis of Buriol et. al. [1] that the amount of changes to a page in Wikipedia have stayed roughly the same throughout the Wikipedia history at the level of one short paragraph at a time.

3.2.1 The slowdown technique

We temporarily assume that the number of embedding elements of d is no greater than $l_{p(d)}$ (the number of elements in $p(d)$); although this is a reasonable assumption in view of the fact that public documents tend to be larger than private ones (e.g., Wikipedia grows over time, whereas a student's web page tends to stay relatively small), we will later on discuss how to avoid this assumption.

Each of the (say, t) embedding e_is of d will be paired with the concatenation of one or more elements of $p(d)$, and it is this concatenation that we called e'_i in the previous sections. The number of elements (call it λ) of $p(d)$ that make up one e'_i, is determined by the desired slowdown factor s, by t, and also by the total number of elements $l_{p(d)}$ in $p(d)$. Specifically we choose

$$\lambda = \frac{l_{p(d)}}{ts}$$

By way of example, assume the number of embedding e_is in d is $t = 5$. Suppose we have a $p(d)$ that changes 10 times faster than we like, and that it consists of (say) $l_{p(d)} = 50$ elements (e.g., paragraphs). In that case we need a λ of

$$\lambda = \frac{50}{5 * 10} = 1$$

Using the keyed hash function of the 5 embedding e_i's in d, we select for each e_i a single element of $p(d)$ to act as its e'_i. As only 5 of the 50 elements of $p(d)$ impact the embedded message, the probability that a change in $p(d)$ causes damage to the message is only 0.1. Notice that the use of the keyed hash function to select the e_i's has the effect that, irrespective of the likelihood of change in the 50 elements of $p(d)$ (e.g., even if that distribution is nonuniform), the probability that a change in an element of $p(d)$ impacts our embedded message is now 0.1, i.e., we have achieved the desired slowdown of a factor of 10 not only through the judicious selection of λ, but crucially through the randomization done by the use of the keyed hash.

3.2.2 The speedup technique

Suppose that in the previous sections, instead of $H(e_1, e_1') \dots H(e_t, e_t')$ being used as the encoding mechanism for the pair $d, p(d)$, it was the following:

$$H(e_1, e_1', \dots, e_t'), \dots, H(e_t, e_1', \dots, e_t')$$

Although this would not change the probability that a modification in one of the elements of $p(d)$ will damage the mark, it does amplify t-fold the number of mark bits damaged in that case. This is indeed one mechanism for achieving a speedup of the decay.

The speedup can be further enhanced by increasing the size of the portion of $p(d)$ that can affect the mark until in eventually includes all $l_{p(d)}$ elements of $p(d)$.

The above mechanism is probably enough in many situations, but it could be the case that an even faster decay is needed than what can be achieved by the above damage-amplification process. In that case we resort to the following method. We use a one-to-many mapping for the pairing function $p(\cdot)$, i.e., $p(d)$ is no longer a single document but a concatenation of multiple documents. In fact this method can even be used in the slowdown technique in case the assumption of $t < l_{p(d)}$ does not hold: It can be forced to hold by using a large enough multiplicity for the $p(\cdot)$ function (although as we stated above, this will occur very rarely).

3.3 System Operation Issues

A burst of updates could be done to many nodes not under the control of the encoder, just prior to a read attempt by the decoder and before the encoder has a chance to react to the change and refresh the mark. Although redundancy and error-correction can mitigate the effects of this (i.e., the mark can survive), it is possible that the sudden updates are so extensive as to overwhelm the mark. What happens in such a case ? The reader (i) recovers the wrong mark, that he is therefore unable to decrypt into something that makes sense (it decrypts to random gibberish); then (ii) the reader backs off from the read attempt and tries to read again at a later time. We believe this to be preferable to a solution that involves a rendez-vous time between the encoder and the decoder (e.g., one refreshes at 4:55PM and the other reads at 4:57PM), which would be awkward and unnecessarily constrain the system's operation. Such a rendez-vous mechanism has its uses, however, if one wants to deliberately create an evanescent mark, which is a mark that fleetingly appears then is promptly read and erased immediately thereafter by the encoder. Such a rendez-vous mechanism also increases the communication bandwidth because the encoder relies on fleeting changes that will not have to stand up to the scrutiny of the other readers of the public information forum being used (the "problem" introduced in encoding gets fixed relatively fast, before the complaints roll in).

3.4 The Case of Free-Form Modification

Recall that this is when we are not constrained to operate in append-only mode when modifying d, i.e., we can modify the individual e_i's so as to carry out the modifications we seek.

Recall that the functionality of an e_i is governed by the two most significant bits of $H(e_i)$, whereas its payload is governed by the ℓ least significant bits of $H(e_i, p(d))$ where $p(d)$ is the sibling of d and ℓ depends on the functionality (e.g., if e_i encodes 20 bits of the mark then $\ell = 20$). We distinguish two cases.

The first case is when the update is not supposed to change the functionality of an e_i, only its payload. This is the most frequent update and happens when, e.g., we still want e_i's functionality to be an encoding one, but we want to change e_i so that the ℓ least significant bits of $H(e_i, e_i')$ are restored to their original value, which they now deviate from because of a modification that occurred to $p(d)$. Recall that such changes that occur to $p(d)$ are beyond the encoder's control – the encoder merely responds to them by modifying e_i so as to restore the correct value to the ℓ least significant bits of $H(e_i, e_i')$. The encoder will, on average, need to apply $2^{2+\ell}$ modifications to e_i before he manages to give their target value to the $2 + \ell$ bits that he is trying to control. The encoder achieves that capacity by a multiplicity of methods. One of these involves trying different subsets of rich content types for inclusion in the new e_i, and for each subset trying all possible orderings of these types. This usually gives the power to carry out the job, without the need to apply different modifications to these types. Specifically, if there are x types, then the number of different orderings of all their different subsets is

$$\sum_{i=1}^{x} x!/(x-i)!$$

For typical rich content, the $\log_2$ of the above gives 22.5, hence an encoding capacity of around 22 bits, making possible an $\ell = 20$ without the need to modify each individual type when refreshing the mark (capacity can be much increased if we are willing to make such changes to text, images, etc).

4 Experiments:Making of WaneMark

We have implemented our system to work as a plugin through a browser based interface, where the user is provided with a Javascript based code that can be run by clicking on a bookmark in the toolbar (See the bookmark labeled as "WaneMark" in the screen shot shown in Figure 2). When the bookmark is clicked a text area is dynamically created inside the browser window. The user enters the secret key, the paired URL (In the current implementation this is just a label of a Wikipedia page.), and a secret message. Encoding is performed in three steps. At every step user highlights a part of the text in the text preview window, and hits the "Try Encode" button. WaneMark modifies the highlighted text. The current system leaves the trailing piece of the highlighted text that does not contribute to the encoding intact; the remaining portion can be used for encoding other parts of the message. The user approves the changes by hitting the "Accept Changes" button and proceeds to the next step.

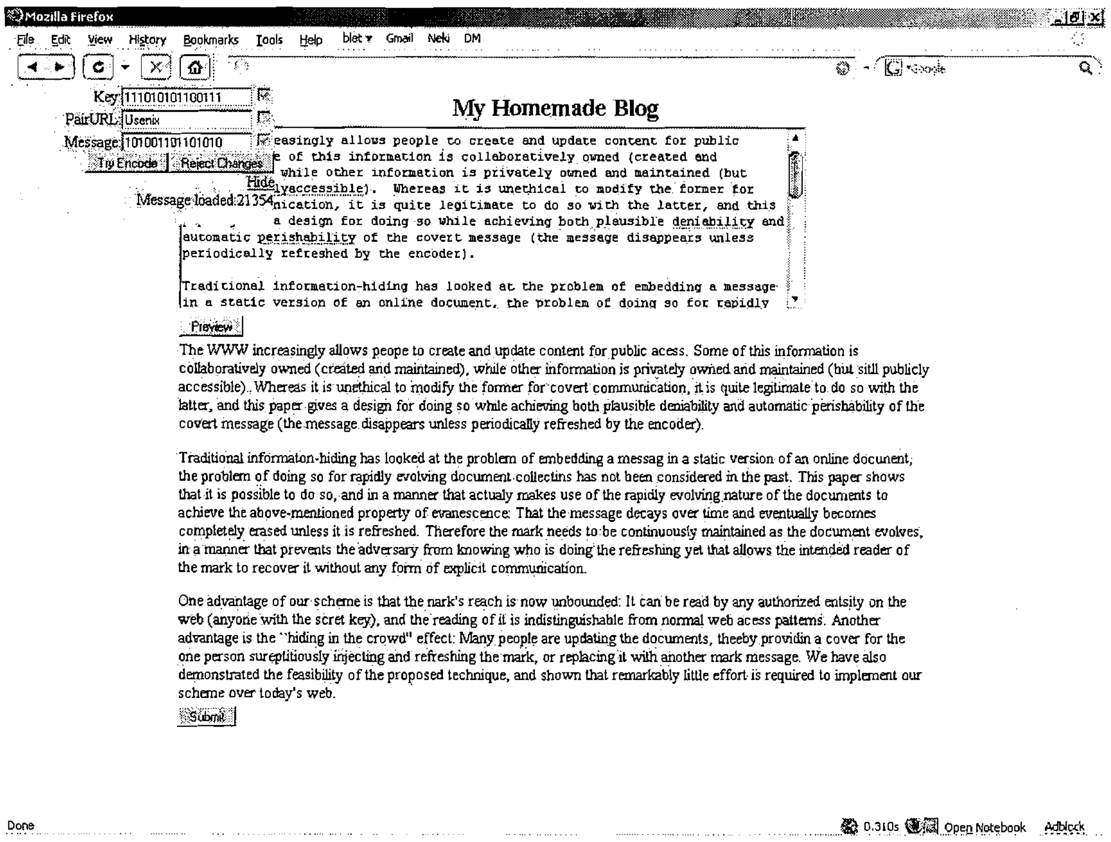
See Figure 2 for a screen shot of the current system operating on a third party blogging interface. [More detailed information about our experiments will be provided in the camera ready copy of our paper including the traffic analysis of dynamically changing pages. We can characterize the updates to non-designated links by their mean-time-between-updates.].

A demo of our implemented system is available at
http://www.cs.purdue.edu/homes/utopkara/wanemark.

Note that WaneMark can use different steganography algorithms as plugins. In our proof-of-concept implementation, we have used a typographical-error based information hiding technique that mimics errors of a human for modifying the text elements on web pages [12]. However, there are many other methods available for performing the same task [14, 13, 3, 10, 17, 16]. Non-text elements can be used for information carrying using image, audio and video steganography techniques available in the literature [4].

The current implementation is a bookmarklet, and does not have permissions to operate on sites other than the test site.

We use 16-bit labels to mark SELECTION, PAIRING and EMBEDDING elements.



Key: 111010101100111
PairURL: Usenix
Message: 101001101101010
Message loaded:21354

My Homemade Blog

```
easingly allows people to create and update content for public
e of this information is collaboratively owned (created and
 while other information is privately owned and maintained (but
lyaccessible).  Whereas it is unethical to modify the former for
nication, it is quite legitimate to do so with the latter, and this
 a design for doing so while achieving both plausible deniability and
automatic perishability of the covert message (the message disappears unless
periodically refreshed by the encoder).

Traditional information-hiding has looked at the problem of embedding a message
in a static version of an online document, the problem of doing so for rapidly
```

The WWW increasingly allows peope to create and update content for public acess. Some of this information is collaboratively owned (created and maintained), while other information is privately owned and maintained (but sitll publicly accessible). Whereas it is unethical to modify the former for covert communication, it is quite legitimate to do so with the latter, and this paper gives a design for doing so while achieving both plausible deniability and automatic perishability of the covert message (the message disappears unless periodically refreshed by the encoder).

Traditional informaton-hiding has looked at the problem of embedding a messag in a static version of an online docunent, the problem of doing so for rapidly evolving document collectins has not been considered in the past. This paper shows that it is possible to do so, and in a manner that actualy makes use of the rapidly evolving nature of the documents to achieve the above-mentioned property of evanescence: That the message decays over time and eventually becomes completely erased unless it is refreshed. Therefore the mark needs to be continuously maintained as the document evolves, in a manner that prevents the adversary from knowing who is doing the refreshing yet that allows the intended reader of the mark to recover it without any form of explicit communication.

One advantage of our scheme is that the nark's reach is now unbounded: It can be read by any authorized entsity on the web (anyone with the scret key), and the reading of it is indistinguishable from normal web acess patterns. Another advantage is the ``hiding in the crowd'' effect: Many people are updating the documents, theeby providin a cover for the one person sureptitiously injecting and refreshing the mark, or replacing it with another mark message. We have also demonstrated the feasibility of the proposed technique, and shown that remarkably little effort is required to implement our scheme over today's web.

Figure 2: Screen shot of the WaneMark system operating on a test interface. Each paragraph helps carrying 16 bits for SELECTION, PAIRING and EMBEDDING.

5 Related Work

Our system aims to mitigate the problem of establishing a truly private web based communication channel. As also mentioned by Butler et al., users of web based communication services have the right to expect that their messages will only be read by the intended recipients, and should be guaranteed of their privacy [2]. As a solution, they propose a scheme that uses several communication channels (different online e-mail services) simultaneously to deliver a steganographically marked text message that carries the secret message embedded in it and the cryptographic keys that are required to decode and decipher the message. Similarly, a form of private communication can be provided by the use of secure email tools such as Privacy Enhanced Mail (PEM), Secure Multipurpose Internet Mail Extension (S/MIME) and Pretty Good Privacy (PGP) [5]. But none of these tools can hide the fact that there is a communication going on between the sender and the receiver.

Waldman et al. [15] listed the required properties of a Web based publishing system as follows: censorship-resistant, tamper evident, source anonymous, updatable, deniable, fault tolerant, persistent (i.e. no expiration date), extensible, freely available. In the same paper, they introduce *Publius*, a web based publishing system that provides these requirements. Publius spreads the data of an anonymous publisher to several servers, such as each server receives an encrypted Publius content (some random looking data) and one of the key shares that was generated by splitting the publisher's key, K, into several keys. Any k of n key shares can reproduce the original K. The publishing process produces a special URL that is used to recover the data and the shares. To browse content, a retriever must get the encrypted Publius content from some server and k of the key shares. Any modification to the encrypted content or the special URL, that is cryptographically tied to the content, will make it impossible to retrieve the unencrypted data. Publius also provides a mechanism for the publishers to update or delete their content. Unlike Publius, where the persistence of the messages and reaching to public are driving goals, our system is based on the design goal of perishability of the messages, and providing a secure communication only with the intended recipient of the message.

Readers are referred to [7] for a survey of privacy enhancing technologies for the Internet.

6 Future Work

There is room for wringing some inefficiencies out of our scheme. We briefly discuss some of these below.

In the append-only mode, we need to design a finer granularity of modifications for the e_i that have an encoding functionality. The difficulty is that, if we were to do so in the natural way (by spreading the encoding task of e_i into a number of smaller elements) then we need a mechanism for specifying which one of the smaller elements (and possibly even which specific bit in it) we are modifying. This "random access" feature achieves a substantial saving only if the number of bits encoded is large enough. It also complicates implementation.

In the case of free-flowing text without chronological time-stamp information, the choice of k (how many basic units form an e_i) can be made more flexible and dependent on the nature of the basic units appended rather than their number. This would recognize the fact that some items (like images) have larger potential for modification than others (like tags or text). One could assign a set of weights to each such type and k would be the threshold accumulated weight at which an e_i is considered to have been created. In such a scheme an image could, literally, be worth a thousand words.

7 Conclusion

Steganograpy has a long and distinguished history, dating back to five centuries BC. It is useful whenever the very existence of the secret message is to remain hidden, as in authoritarian countries that severely limit and censor speech. This paper describes a keyed scheme for posting secret messages on the web such that anyone with the secret key can retrieve the messages, but even a computationally powerful adversary (e.g., a government with supercomputers) who does not have the key is not able to know (let alone prove) the message's existence: The message depends on web-wide content, not only on what is under the sender's control. If the sender is prevented from erasing the message, the actions of others on the web automatically take care of doing the erasing job on the sender's behalf. Moreover, the sender can control, a priori, the likely life of the secret message before it automatically disappears, by a judicious design of the "pairing" function with the web's sibling information.

Our scheme is primarily destined for thwarting censorship. There have been many rumors, none substantiated, of evil-doers using steganography for communication – investigations in that direction have turned up no evidence of this. This is probably because evil entities like organized crime (including drug cartels and terror networks) have other and more effective ways of communicating, ways that are not available to an isolated individual, with limited resources, living in a repressive country.

References

[1] L. Buriol, C. Castillo, D. Donato, S. Leonardi, and S. Millozzi. Temporal evolution of the wikigraph. In *Proceedings of Web Intelligence*, pages 45–51, Hong Kong, December 2006. IEEE CS Press.

[2] K. Butler, W. Enck, J. Plasterr, P. Traynor, and P. McDaniel. Privacy-preserving web-based email. In *International Conference on Information Systems Security*, 2006.

[3] M. Chapman and G. Davida. Hiding the hidden: A software system for concealing ciphertext in innocuous text. In *Proceedings of the International Conference on Information and Communications Security*, volume LNCS 1334, Beijing, China, 1997.

[4] I. Cox, M. Miller, and J. A. Bloom. *Digital Watermaking*. Morgan Kaufmann Publishers, 2002.

[5] C. Ellison and B. Schneier. Inside risks: Risks of PKI: secure email. *Communications of the ACM*, 43(1):160, 2000.

[6] Extreme Tracking. http://extremetracking.com.

[7] I. Goldberg. Privacy-enhancing technologies for the internet ii: Five years later. In *Workshop on Privacy-Enhancing Technologies*, 2002.

[8] Google. http://www.google.com/.

[9] Google Analytics. http://www.google.com/analytics.

[10] R. Stutsman, M. Atallah, C. Grothoff, and K. Grothoff. Lost in just the translation. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 338–345. ACM, 4 2006. steganography translation machine statistical information hiding text natural language.

[11] The Internet Archive. http://www.archive.org.

[12] M. Topkara, U. Topkara, and M. J. Atallah. Information hiding through errors: A confusing approach. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, 2007.

[13] M. Topkara, U. Topkara, and M. J. Atallah. Words are not enough: Sentence level natural language watermarking. In *Proceedings of ACM Workshop on Content Protection and Security (in conjuction with ACM Multimedia)*, Santa Barbara, CA, October 27, 2006.

[14] U. Topkara, M. Topkara, and M. J. Atallah. The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of ACM Multimedia and Security Workshop*, Geneva, Switzerland, September 26-27, 2006.

[15] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of 9th USENIX Security Symposium*, pages 59–72, August 2000.

[16] P. Wayner. Mimic functions. *CRYPTOLOGIA*, XVI(3):193–214, July 1992.

[17] K. Winstein. Lexical steganography through adaptive modulation of the word choice hash. In *http://www.imsa.edu/ keithw/tlex/*, 1998.