Censorship-Resilient Communications through Information Scattering

Stefano Ortolani[1], Mauro Conti[1], and Bruno Crispo[2]

[1] Vrije Unversiteit Amsterdam, The Netherlands
{ortolani,mconti}@few.vu.nl
[2] University of Trento, Italy
crispo@dit.unitn.it

Abstract. The aim of this paper is to present a new idea on censorship-resilient communication Internet services, like blogs or web publishing. The motivation of this idea comes from the fact that in many situations guaranteeing this property is even matter of personal freedom. Our idea leverages: i) to split the actual content of a message and to scatter it through different points of retrieval; ii) to hide the content of a splitted message in a way that is clearly unidentifiable—hence involving encryption and steganography; iii) to allow the intended message recipient to correctly retrieve the original message. A further extension on this idea allows the recipient of the message to retrieve the message even if: i) some of the retrieval point are not available; ii) some retrieved data have been tampered with—their integrity has been violated.

1 Introduction

Allowing people to communicate privately is important in many context. Guaranteeing the anonymity of the source of a message is a way to obtain communication privacy. The concept of anonymity refers to the fact that a given subject is concealed among a set of subjects—the anonymity set—as taking part of an action. However, anonymity is not enough if the communication channel is censored. The act of publishing information establishes a direct communication between the publisher (e.g. `http://www.blogspot.com`) and the user (e.g. the corresponding IP address).

Different works addressed this problem by loosing the correlation between the IP address used by the user and the user itself. For instance, in Tor [1], the user accesses the World Wide Web by means of multiple identities provided by a network of nodes. Tor is able, to some extent, to anonymise the user and even the publisher (by means of Tor hidden services) within the Tor network. Hence, it seems that the problem of guaranteeing the anonymity of an individual has been already addressed and solved. We observe that the solution provided by Tor works only if the information are retrieved from within the Tor network. However, not all the users are now using the Tor network—and we cannot force them to do it. Moreover, if someone wants to publish something using Tor he has to deploy a server to employ these hidden services.

B. Christianson and J. Malcolm (Eds.): Security Protocols 2010, LNCS 7061, pp. 82–89, 2014.

As an example, let us consider a non democratic authority censorship. It might i) identify the publisher and subsequently ban it—forcing the publisher to stop its activity, or ii) check the user traffic and filter the one directed to the publisher. In the remaining part of the paper, we describe our solution with respect to a blog post service. It is however straightforward to extend the described solution to other Internet services, such as other type of web sites.

Our solution counters censorship by dividing the content of the message into chunks, then publishing them in multiple locations as comments to blogs. In such a way, censoring a publisher does not prevent the content published through other publisher to be retrieved. The scattered content is shared by means of a link including all the necessary information to retrieve all the post fragments. An application running inside the browser of the client (e.g. a Firefox extension) provides then the logic to i) re-assemble the content and ii) display it as a web page inside the user browser.

Moreover, the content can be concealed by means of steganography. Hence, it will not be easy to identify a publisher as the one publishing a specific content. However, any steganography technique requires the definition of what is known as the cover signal. In other words, we have to provide the message acting as the envelope for our secret message. Since such envelopes shall resemble already published content, we retrieve the cover signal from website such as wikipedia: the meta keywords included in the publisher's site can be easily used as query terms.

Finally, we consider only blogs that are not moderated and do not require any registration. This would pose another defense to the users privacy in the case the publisher collude with the censorer.

Organization. The rest of the paper is organized as follows. Section 2 reports on the related work in the area. Section 3 presents a brief overview of our approach. We introduce some preliminaries in Section 4, while our solution is described in Section 5. Section 6 gives a brief discussion on the key points of the proposal. Finally, Section 7 concludes the work.

2 Related Work

A wide range of solutions have been published on the anonymity of communications. Among these, Tor [1] is probably the most practical one. While Tor could partially solve the problem we address, it would require the following. One one side, it would require all interested users to use Tor. On the other side the Tor node that wants to act as publisher must manage a server in order to offer a Tor *hidden service* [2]. Furthermore, we observe that the *directory servers* used to announce the hidden service represent points of failure and a weakness by itself—identifying the *introduction points*.

In our solution, we leverage on the information scattering concept [3]. Furthermore, we make use of the text steganography [4] to hide the message M intended to be sent within other text. As a property of the steganography, it is not possible to detect that the resulting text contain an hidden message. A further extension

of the proposed solution could also make use threshold cryptography [5]. This would make the solution more efficient and resilient: the message M could be splitted and scattered into n different points and retrieved contacting just $t < n$ of these points.

3 Our Approach

We are interested in publishing a message in a way such that it is not easy to censor the message itself, or even censor the publisher that keeps the message on line and available to visiting clients. The main idea is to split the message M intended to be sent into n message chunks m_i $(1 \leq i \leq n)$, and publish these separately. Furthermore, each single message chunk m_i can be published hidden with steganographic techniques. We recall that we assume the publication is done through posting on blogs. A censorer would not be able to get the real meaning of an actually posted blog's comment as well as it will not be able to reconstruct the intended message M. In fact, only the intended message recipient will be provided with the actual information required to retrieve all the required published comments, and get out from them (hidden with steganography) the actual chunks for M. Hence, only the intended recipient will be able to reconstruct the message M, while the censorship will not be able to selectively ban a message or the publisher of a specific message.

The message chunks m_i are scattered to different n publishers p_i $(1 \leq i \leq n)$. In our example-driven discussion a publisher is a blog server and the publication is made throughout comments to pre-existing blogs. We consider blogs that do not require any registration, nor a moderator approving the comment itself. In the following, we refer to a publisher also as a repository. Furthermore, we assume that the number of comments are not bounded and that a specific blog's post is identifiable by the combination of URL (Uniform Resource Locator) and comment id. Without loss of generality, we also assume that no CAPTCHA mechanisms are in place on the publisher. However, we can relax this assumption considering that for our scenario a user already qualifies as motivated. Hence, solving a set of CAPTCHAs does not pose any hindrance to our system. The effort of solving a set of a CAPTCHAs is therefore considered acceptable.

The basic approach of publishing just a "piece" of the given text can be enhanced by means of steganographic techniques. In this case we aim to choose as an "envelope"—the object that hides and carries the actual "piece" of message—a text that matches the blog's topic. This process can be automatic, since a web site provides the meta keywords. Upon hiding through steganography we can look in a texts source website (e.g. wikipedia) the topic of the web page and retrieve a consistent text snippet. Upon decryption we just need the key.

In order for a user that wants to get the actual message scattered and hidden through steganography among different unknown sources, the user need the appropriate "link", that is a set of information (URL, comment id, key) for each of the used repository. All these information can actually be assembled in a link in a way similar to how it happens for P2P protocols (e.g. eMule).

4 Preliminaries and Notation

In this section, we present some preliminaries required for the rest of the paper. In particular, we give the definition of a publisher and some other related concepts. Table 1 summarizes the notation used in this paper.

Table 1. Notation Table

M	Message that is intended to be sent.
n	Number of chunks a message is divided into.
m_i	A chunk of message M, $(1 \leq i \leq n)$.
p_i	The publisher (repository) of chunk m_i.
url_i	URL that identifies publisher p_i.
m'_i	Message that hide (through steganography) the message m_i.
ID_i	Identifier of blog's comment m'_i.
$meta_i$	The first meta keyword in the HTML page of the blog of publisher p_i.
E_k	Steganographic function with key k.

Definition 1. *A publisher p_i is a blog server where we can publish information. The blog is not moderated and it does not have bound on the number of posts. Any publisher p_i is univocally identified by its URL url_i. Given a publisher p_i, we identify with $meta_i$ the first meta keyword included in the HTML page of the blog.*

A blog is composed by a set of posts (or comments), i.e. pieces of text.

Definition 2. *A blog's post (or comment) for a publisher p_i is a text message bounded by the max length accepted by p_i.*

Definition 3. *We define as $Cover(meta_i) = t_i$ the function that, given a keyword $meta_i$, it returns a text t_i from a texts source website, where the topic of t_i is in accordance with the keyword $meta_i$.*

An example of a texts source website is Wikipedia. One other component that our system requires is the steganographic primitive.

Definition 4. *Given a message m_i, and a meta keyword $meta_k$, we conceal m_i as follows: $E_k(m_i, Cover(meta_i)) = E_k(m_i, t_i) = m'_i$, where E_k is a steganographic function with key k.*

Intuitively, the message $m'_i \approx t_i$ conceals the message m_i. Since E_k can be implemented through encryption followed by the application of steganography, in the following we also refer to E_k as just encryption. Publishing the obtained message m'_j is defined as follows.

Definition 5. *Given a message m'_i and a blog url_i, we define the function $Publish(m'_i, url_i) = ID_i$, as the function that publish m'_i in url_i. The value ID_i is the identifier of the resulting added blog's comment.*

To ease the retrieval of the scattered information we define a URI (Uniform Resource Identifier) resembling the one adopted in P2P systems, such as eMule.

Definition 6. *The URI of our protocol, namely* `info-URI`*, is defined accordingly to the following ABNF notation:*

```
info-URI     = info-scheme "::" info-publish [ "||" info-publish ]
info-scheme  = "info"
info-publish = http-URI "|" info-cid "|" info-key
http-URI     = The publisher URL
info-cid     = The comment ID
info-key     = The key of the encrypted content
```

A message scattered among two different publishers, p_1 and p_2, respectively in two comment ID_1 and ID_2, has the following URI:

$$info : url_1|ID_1|k_1||url_2|ID_2|k_2.$$

The last corner-stone of our approach is the definition of the function in charge of retrieving the so-scattered content. A browser extension is enough to carry out this task. The assembling phase is defined as follows.

Definition 7. *Given a URI $info : url_i|ID_i|k_i$ the decryption part works as follows: $D_{k_i}(Retrieve(url_i, ID_i)) = D_{k_i}(m'_i) = m_i$ where the function $Retrieve(url_i, ID_i)$ retrieves the comment ID_i from the blog at url_i.*

5 Our Solution

In Section 3 we gave an overview of our approach. In this section, we describe its architecture.

The overall architectural design is depicted in Figure 1. We immediately identify the two main actors interested in the communication: the publisher (repository) and the visiting client. The three repositories are in turn identified by url_1, url_2, and url_3. Even if the figure depicts just one visiting client, the same mechanism applies for any number of them. Last but not least, the source of cover signal is represented by the Wikipedia web site.

Let us now assume that a client wants to publish a message M in a way such that the message will not be censored. First, the client selects the repositories—i.e. blog sites—satisfying Definition 1. These are identified by url_1, url_2, and url_3 in Figure 1. The client hence splits M into message chunks, one for each selected repository. In the example, we have three chunks: m_1, m_2, and m_3. For each of the selected repository, the client gets the meta keywords from the blog HTML page and, depending on these, it selects a text of the same topic from the texts source (e.g. Wikipedia). Let us call this texts "envelops" (t_1, t_2, and

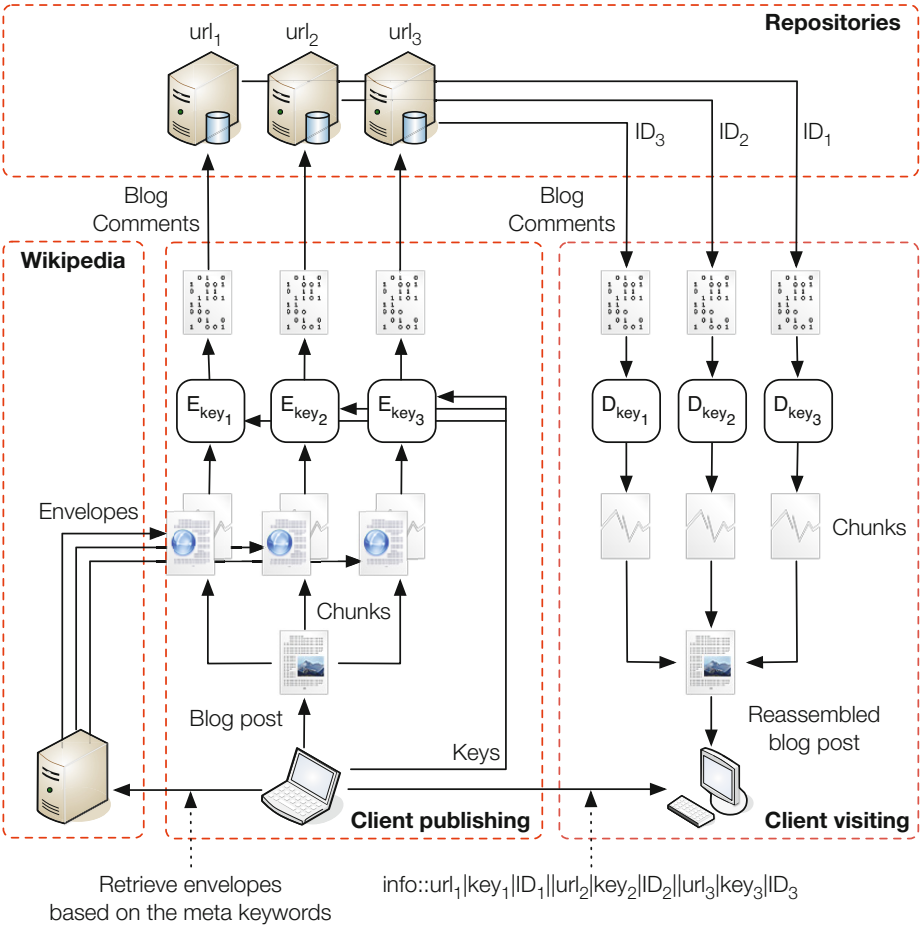

Fig. 1. Architecture of our Information Scattering solution

t_3). Now the client selects a different key k_i for each message chunk m_i. A key k_i is used to hide by mean of steganography the message m_i into envelope t_i (text steganography tools [6] are used in this step). The resulting text m'_i is then posted as a comment on the repository url_i. Finally, the publishing client uses a side channel (e.g. email) to send the information required to a visiting client to correctly retrieve the original intended message M. In particular, for each message chunk m_i, the publishing client will state the URL of the repository url_i, the blog's comment id ID_i within the given blog, and the key k_i required to hide m_i within the comment ID_i. The overall resulting string will be a string accordingly to Definition 6, $info :: url_1|key_1|ID_1||...||...$.

Once the URI string is received (by means of the side channel), the visiting client acts as follows. For each element (url_i, key_i, ID_i), it retrieves from the repository url_i the blog's comment ID_i. Then it uses the key key_i to get the

message chunk m_i, out of the retrieved blog's comment. Once the visiting client has all the required chunks, it has just to re-assembly them to obtain the original message M the publisher intended to send.

Implementation. We are implementing the solution described in Section 5 as a Firefox extension. In particular, the expected browser plug-in is divided in two parts: (i) the part in charge of publishing the content; (ii) the part in charge of retrieving the content.

The publishing part:

- chooses the repositories;
- divides the text to be published into chunks;
- for each repository, it retrieves the meta keywords and downloads from the texts source website (e.g. wikipedia) a text about that topic;
- conceals the original text chunks in the text retrieved in the texts source website;
- publishes each part asking the user to solve CAPTCHAs, if needed.

The retrieving part:

- receives in input a link with all the required information on where the chunks are (Definition 6);
- retrieves the chunks and organize them in a pre-determined manner (it acts as a XSLT transformation).

6 Solution Analysis

In this section we evaluate the effectiveness of our solution. First, we recall that our solution relies upon the following points:

1. the original intended message M is not posted as it is, instead it is splitted in chunks m_i;
2. each chunk m_i is not posted as it is, but it is hidden by means of steganography within another text;
3. the text used to hide the original chunk m_i is chosen in a way such that the resulting post looks like a plausible blog's comment, related to the topic that is actually discussed in the blog—thanks to the meta keywords;
4. the information required to retrieve the chunks and compute the original message M are transferred to the visiting client throughout a side channel that is outside the distribution mechanism.

The point 1 is used for practical reasons. In fact, the message M intended to be sent could have an arbitrary length that is over the bound of a single post accepted by a blog server. Furthermore, the steganography function works better when the text to be hidden is small.

The steganography function (point 2) is used to avoid the censorer to identify a message chunk m_i as not desired and to ask the publisher p_i to ban it. Also hiding the chunk m_i within a text t_i with a similar topic (point 3) is done in

order to avoid the censorer to understand that a blog post might contain some hidden message.

The last point is used to transfer to the visiting client all the required information to get M. We assume that the side channel used for this purpose is secure, e.g. mail could be encrypted with public key crypto allowing confidentiality and authentication. While guaranteeing the security of this channel is out of the scope of this paper, one might argue that if such a channel exist the message M could be convoyed just using this channel. However, we observe that the information could be passed before the actual message is generated. Also, once some initial secret data are shared among the sender and the receiver, the information required for the messages sent after M, could be auto generated without requiring any further communication—e.g. using chain of hash function.

Furthermore, using techniques like threshold cryptography the original messages could be splitted in a number of chunks n such that a smaller number w out of n are sufficient to retrieve the original M. In this way, even if $n - w$ repositories are identified and blocked, the clients will still be able to retrieve M.

7 Conclusion

In this paper, we presented a new method to achieve censorship-resilient communications. We presented a solution for sending generic text messages, using existing server, thus not requiring any special software (e.g. Tor) at the server side. Furthermore, we depicted a way to hide these messages through blog posting. The presented idea can be easily extended and even optimized for specific type of messages (e.g. html pages or pictures).

References

1. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium, pp. 303–320 (2004)
2. Lenhard, J., Loesing, K., Wirtz, G.: Performance measurements of tor hidden services in low-bandwidth access networks. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 324–341. Springer, Heidelberg (2009)
3. Bhavnani, S.K.: Information scattering. In: ELIS, pp. 1–8 (2009)
4. Mansor, S., Din, R., Samsudin, A.: Analysis of natural language steganography. International Journal of Computer Science and Security (IJCSS) 3(2), 113–125 (2009)
5. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
6. Wayner, P.: Disappearing Cryptography: Information Hiding Steganography & Watermarking, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco (2008)