Message In A Bottle: Sailing Past Censorship

Luca Invernizzi, Christopher Kruegel, Giovanni Vigna

UC Santa Barbara

Abstract

Exploiting recent advances in monitoring technology and the drop of its costs, authoritarian and oppressive regimes are tightening the grip around the virtual lives of their citizens. Meanwhile, the dissidents, oppressed by these regimes, are organizing online, cloaking their activity with anti-censorship systems that typically consist of a network of anonymizing proxies. The censors have become well aware of this, and they are systematically finding and blocking all the entry points to these networks. So far, they have been quite successful. We believe that, to achieve resilience to blocking, anti-censorship systems must abandon the idea of having a limited number of entry points. Instead, they should establish first contact in an online location arbitrarily chosen by each of their users. To explore this idea, we have developed Message In A Bottle, a protocol where any blog post becomes a potential "drop point" for hidden messages. We have developed and released a proof-of-concept application using our system, and demonstrated its feasibility. To block this system, censors are left with a needle-in-a-haystack problem: Unable to identify what bears hidden messages, they must block everything, effectively disconnecting their own network from a large part of the Internet. This, hopefully, is a cost too high to bear.

1 Introduction

The revolutionary wave of protests and demonstrations known as the *Arab Spring* rose in December 2010 to shake the foundations of a number of countries (e.g., Tunisia, Libya, and Egypt), and showed the Internet's immense power to catalyze social awareness through the free exchange of ideas. This power is so threatening to repressive regimes that censorship has become a central point in their agendas: Regimes have been investing in advanced censoring technologies [37], and even resorted to a complete isolation from the global network in critical moments [10]. For example, Pakistan recently blocked Wikipedia, YouTube, Flickr, and Facebook [23], and Syria blocked citizen-journalism media sites [44].

To sneak by the censorship, the dissident populations have resorted to technology. A report from Harvard's Center for Internet & Society [38] shows that the most popular censorship-avoidance vectors are web proxies, VPNs, and Tor [13]. These systems share a common characteristic: *They have a limited amount of entry points*. Blocking these entry points, and evading the blocking effort, has become an arms race: China is enumerating and banning the vast majority of Tor's bridges since 2009 [50], while in 2012, Iran took a more radical approach and started blocking encrypted traffic [47], which Tor countered the same day by deploying a new kind of traffic camouflaging [46].

In this paper, we take a step back and explore whether it is possible to design a system that is so pervasive that it is impossible to block without disconnecting from the global network.

Let's generalize the problem at hand with the help of Alice, a dissident who lives on the oppressed country of Tyria and wants to communicate with Bob, who lives outside the country. To establish a communication channel with Bob in any censorship-resistant protocol, Alice must know *something* about Bob. In the case of anonymizing proxies or mix-networks (e.g., Tor), this datum is the address of one of the entry points into the network. In protocols that employ steganography to hide messages in files uploaded to media-hosting sites (such as Collage [8]) or in network traffic (such as Telex [54]), Alice must know the location of the first rendezvous point.

The fact that Alice has to know something about Bob inevitably means that the censor can learn that too (as he might *be* Alice). Bob cannot avoid this without having some information to distinguish Alice from the censor (but this becomes a chicken-and-egg-problem: How did Bob get to know that?). We believe that this initial *something* that Alice has to know is a fundamental weakness of existing censorship-resistant protocols, which forms a crack in their resilience to blocking. For example, this is the root cause of the issues that Tor is facing with distributing bridge addresses to its users without exposing them to the censor [45]. It is because of this crack that China has been blocking the majority of Tor traffic since 2009 [50]: the number of entry points is finite, and a determined attacker can enumerate them by claiming to be Alice.

In Message In A Bottle (MIAB), we have designed a protocol where Alice knows the *least possible* about Bob. In fact, we will show that Alice must know Bob's public key, and nothing else. Alice must know at least Bob's public key to authenticate him and be sure she is not talking to a disguised censor. However, contrary to systems like Collage and Telex, there is *no rendezvous point* between Alice and Bob.

This may now sound like a needle-in-a-haystack problem: If neither Alice nor Bob know how to contact the other one, how can they ever meet on the Internet? In order to make this possible and reasonably fast, MIAB exploits one of the mechanisms that search engines employ to generate real-time results from blog posts and news articles: *blog pings*. Through these pings, Bob is covertly notified that a new message from Alice is available, and where to fetch it from. We will show that, just like a search engine, Bob can monitor the majority of the blogs published on the entire Internet with limited resources, and in quasi real-time. In some sense, every blog becomes a possible meeting point for Alice and Bob. However, there are over 165 million blogs online [5], and since a blog can be opened trivially by anybody, for our practical purposes they constitute an infinite resource. We have estimated that, to blacklist all the possible MIAB's drop points, the Censor should block 40 million fully qualified domain names, and four million IP addresses (as we will show in Section 5.3, these are conservative estimates). For comparison, blacklisting a single IP address would block Collage's support for Flickr (the only one implemented), and supporting additional media-hosting sites requires manual intervention for each one.

In MIAB, Alice will prepare a message for Bob and encrypt it with his public key. This ciphertext will be steganographically embedded in some digital photos. Then, Alice will prepare a blog post about a topic that fits those photos, and publish it, along with the photos, on a blog of her choosing. Meanwhile, Bob is monitoring the stream of posts as they get published. He will recognize Alice's post (effectively, the bottle that contains the proverbial message), and recover the message.

We envision that MIAB might be used to bootstrap more efficient anonymity protocols that require Alice and Bob to know each other a little better (such as Collage, or Telex).

Our main contributions are:

- A new primitive for censorship-resistant protocols that requires Alice to have minimal knowledge about Bob;
- A study of the feasibility of this approach;
- An open-source implementation of a proof-of-concept application of MIAB that lets user tweet anonymously and deniably on Twitter.

2 Threat model

The adversary we are facing, the Censor, is a country-wide authority that monitors and interacts with online communications. His intent is to discover and suppress the spreading of dissident ideas.

Determining the current and potential capabilities of modern censors of this kind (e.g., Iran and China) is a difficult task, as the inner workings of the censoring infrastructure are often kept secret. However, we can create a reasonable model for our adversary by observing that, ultimately, the Censor will be constrained by economic factors. Therefore, we postulate that the censorship we are facing is influenced by these two factors:

- The censoring infrastructure will be constrained by its cost and technological effort
- Over-censoring will have a negative impact on the economy of the country

From these factors, we can now devise the capabilities of our Censor. We choose to do so in a conservative fashion, preferring to overstate the Censor's powers than to understate them. We make this choice also because we understand that censorship is an arms race, in which the Censor is likely to become more powerful as technology advances and becomes more pervasive.

We assume that it is in the Censor's interest to let the general population benefit from Internet access, because of the social and economic advantages of connectivity. This is a fundamental assumption for any censorship-avoidance system: If the country runs an entirely separate network, there is no way out[1].

Because the Censor bears some cost from over-blocking, he will favor blacklisting over whitelisting, banning traffic only when it deems it suspicious. When possible, the Censor will prefer traffic disruption over modification, as the cost of deep-packet inspection and modification is higher than just blocking the stream. For example, if the Censor suspects that the dissidents are communicating through messages hidden in videos on YouTube, he is more likely to block access to the site rather than re–encoding every downloaded video, as this would impose a high toll on his computational capabilities. Also, we assume that if the Censor chooses to alter uncensored traffic, he will do so in a manner that the user would not easily notice.

The Censor might also issue false TLS certificates with a Certificate Authority under its control. With them, he might man-in-the-middle connections with at least one endpoint within his country.

As part of his infrastructure, the Censor might deploy multiple monitoring stations anywhere within his jurisdiction. Through these stations, he can capture, analyze, modify, disrupt, and store indefinitely network traffic. In the rest of the world, he will only be able to harness what is commercially

The Censor can analyze traffic aggregates to discover outliers in traffic patterns, and profile encrypted traffic with statistical attacks on the packets content or timing. Also, he might drill down to observe the traffic of individual users.

The Censor will have knowledge of any publicly available censorship-avoidance technology, including MIAB. In particular, he might join the system as a user, or run a MIAB instance to lure dissidents into communicating with him. Also, he might inject additional traffic into an existing MIAB instance to try to confuse or overwhelm it.

3 Design

In its essence, MIAB is a system devised to allow Alice, who lives in a country ruled by an oppressive regime, to communicate confidentially with Bob, who resides outside the country. Alice does not need to know any information about Bob except his public key.

In particular, MIAB should satisfy these properties:

- **Confidentiality:** The Censor should not be able to read the messages sent by Alice.
- **Availability:** The Censor should not be able to block MIAB without incurring unacceptable costs (by indiscriminately blocking large portions of the Internet).
- **Deniability:** When confidentiality holds, the Censor should not be able to distinguish whether Alice is using the system, or behaving normally.
- **Non-intrusive deployment:** Deploying and operating a MIAB instance should be easy and cheap.

To achieve these properties, the MIAB protocol imposes substantial overhead. We do not strive for MIAB's performance to be acceptable for low latency (interactive) communication over the network (such as web surfing). Instead, we want our users to communicate past the Censor by sending small messages (e.g., emails, articles, tweets). Also, MIAB can be employed to exchange the necessary information to bootstrap other anonymity protocols that require some pre-shared secret, as we will discuss in Section 4.

The only requirement that Alice must satisfy to use this protocol is to be able to make a blog post. She can create this post on any blog hosted (or self-hosted) outside the Censor's jurisdiction.

3.1 Components

Before explaining the details of the MIAB protocol, we must introduce two concepts that will play a crucial role in our system. The first is the notion of a *blog ping*. The second is the metadata that is associated with digital photos.

[1]We are strictly speaking about traditional ways to leak messages to the Internet through network communication.

3.1.1 Blog pings.

A blog ping is a message sent from a blog to a centralized network service (a *ping server*) to notify the server of new or updated content. Blog pings were introduced in October 2001 by Dave Winer, the author of the popular ping server `weblogs.com`, and are now a well-established reality. Over the last ten years, the rising popularity of pings pushed for the development of a wealth of protocols that compete for performance and scalability (e.g., FriendFeed's SUP [26], Google's PubSubHubbub [27], and rssCloud [43]).

Search engines use blog pings to efficiently index new content in real time. Since search engines drive a good part of the traffic on the Internet, blog writers adopt pings to increase their exposure and for Search Engine Optimization. Consequently, the vast majority of blogging platforms support pings, and have them enabled by default (e.g., Wordpress, Blogger, Tumblr, Drupal).

3.1.2 Digital photo metadata.

Digital cameras store metadata embedded within each photo. This metadata keeps records of a variety of information, including the condition of the shot, GPS coordinates, user comments, copyright, post-processing, and thumbnails. Three metadata-encoding formats dominate in popularity, and are supported by the vast majority of cameras and photo editing software: IIM [31], XMP [55], and EXIF [16].

IIM (Information Interchange Model) is an older standard developed in the nineties by the International Press Telecommunications Council. It was initially conceived as a standard for exchanging text, photos and videos between news agencies. In 1995, Adobe's Photoshop adopted IPTC-IIM to store metadata in digital photos, and it is now ubiquitous in the digital camera world.

The second format, XMP (Extensible Metadata Platform) was developed by Adobe in 2001. XMP is XML/RDF-based, and its standard is public and open-source.

The last format is EXIF (Exchangeable Image File Format). It was developed by the Japanese Electronics Industry Development Association (JEIDA) in 1998, and evolved over the years. It is based upon the TIFF format.

A particular EXIF tag, `MakerNote`, stores manufacturer-specific binary content.

To observe the popularity of each of these formats, we downloaded 15,000 photos from Flickr. To reach a broad number of photographers, we selected the photos using Flickr popular tags [20]. The distribution of the metadata formats in our dataset is shown in Table 1. EXIF is by far the most popular format, as it was present in 96.63% of the photos with metadata (note that a photo might contain more than one format). We also observed that the `MakerNote` tag usually accounts for the majority of metadata space.

3.2 The MIAB Protocol

Our scene opens with Alice, who lives in a country controlled by the Censor. Alice wants to communicate with Bob, who is residing outside the country, without the Censor ever knowing that this communication took place. To do so, Alice sends a message with the MIAB protocol following these steps:

1. Alice authors a blog post of arbitrary content. The content should be selected to be as innocuous as possible.
2. Alice snaps one or more photos to include in the post.
3. Alice uses the MIAB client software to embed a message M into the photos. The message is hidden using shared-key steganography [42]. We will identify the shared key as K_s. Alice chooses the shared key arbitrarily.
4. K_s is encrypted with the public key of Bob K_P. This cipher-text is hidden in the photos' metadata, using tags that appear inconspicuous to the Censor (e.g., `Exif.Photo.ImageUniqueID`, which contains an arbitrary identifier assigned uniquely to each image). To better understand the location of the various data embedded in photos, please refer to Figure 1.
5. Alice publishes the blog post, containing the processed photos. Alice can choose the blog arbitrarily, provided it supports blog pings. We will discuss this in more detail in Section 5.2.
6. The blog emits a ping to some ping servers.
7. Meanwhile, Bob is monitoring some of the ping servers, looking for steganographic content encrypted with his public key. Within minutes, he discovers Alice's post, and decrypts the message.

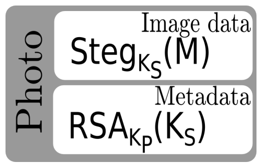

Figure 1: Alice's message

	Number of photos	% of dataset	% of photos with metadata
No metadata	2093	13,95%	
EXIF present	12,472	83.15%	96.63%
XMP present	2972	19.81%	23.03%
IPTC-IIM present	666	4.44%	5.16%

Table 1: Metadata formats in photos on Flickr

8. Bob reads the message, and acts upon its content (more on this later).

Looking at Figure 1, the reader might wonder why we did not choose to store the message M, encrypted with K_P, directly into the metadata. The reason is that available space is limited (e.g., in our implementation, we found about 13.5 bytes of space). Instead, the image data is of the order of megabytes in size, and we can embed in it a message of a few kilobytes (this size depends on the size of the image, and the steganographic scheme used).

Also, notice that Steps 3 and 4 are essentially a simple public-key image steganography (PKIS) [51] scheme: A message is hidden in an image using a public key, and can only be retrieved with the corresponding private key. Note that any PKIS scheme could be used in place of Steps 3 and 4, and might improve the deniability of MIAB (for example, because it would not be necessary to alter the images' metadata). In our proof-of-concept implementation of MIAB, we have chosen the presented scheme because we can use shared-key image-steganography tools, which are widely available (e.g., Outguess [40], StegHide [29]). We are not aware of any open-source implementation of any other PKIS scheme. MIAB's deniability is a direct consequence of the PKIS scheme used, but our main contribution is, then, to create an highly-available protocol that is difficult and expensive for the Censor to block.

To be able to use MIAB, Alice needs a copy of the software, which will also contain Bob's public key. This bundle can be downloaded before the Censor becomes too controlling, or can be published around the web in different formats, so that the Censor cannot easily fingerprint and block it. Optionally, it can be distributed through spam, or offline via USB drives. We expect that the Censor will also have a copy of the software.

4 Applications

The MIAB protocol can be a useful component for a variety of applications. We will outline the ones we believe are most interesting, one of which we implemented.

4.1 Covert messaging

In this setting, Alice wants to send a message (e.g., email or tweet) out of the country in a deniable fashion. She does so simply by using the MIAB protocol to publish her message embedded in a blog post, leaving instructions to Bob on how to handle it. To showcase this application, we have implemented a Twitter proxy that relays the messages found through MIAB to Twitter, publishing them under the `@corked_bottle` handle. MIAB could also be used to communicate with a censorship-resistant microblogging service, like `#h00t` [2], or a privacy-preserving one, like Hummingbird [12].

To achieve two-way messaging, Alice composes a comment to the blog post she is about to publish. She sends this comment to Bob with MIAB, appended to her message. When Bob wants to reply to Alice's message, he steganographically encodes his answer inside the comment, and posts the comment on Alice's blog (or any blog Alice has chosen for posting her messages). Alice will either be notified of Bob's action by email or RSS feed, or she will check her blog every now and then. Alice's behavior should not be considered suspicious by the Censor, since blog writers have a habit of checking comments on their posts frequently.

To appear even more inconspicuous to the Censor, Alice might arbitrarily pick any other blog available on the Internet, and tell Bob to post the comment containing the reply there. A good meeting point for this is one of the blogs and online newspapers (provided they allow comments) that Alice reads on a regular basis.

How Bob hides his reply will depend on Alice's directives. An option is that Alice includes an image (or provides a private link to one) in her comment, and Bob embeds the message into this image. Since Alice and Bob have already established a shared secret, there is no need for metadata. Another option is that Bob encodes his reply in the comment's text. Note that our scheme authenticates only Bob, as not being identified is in Alice's interest. However, after the first message is sent, Bob and Alice share a session key, so that the Censor cannot pretend to be Alice and take over her role in the current message exchange.

This method of covert messaging suffers a noticeable overhead with respect to standard messaging. To achieve better efficiency, we envision that MIAB could be used to bootstrap some faster, shared-key protocol.

4.2 Bootstrapping Collage

Collage [8] is a protocol that uses user-generated content (e.g., photos, tweets) as drop sites for hidden messages. Differently from the MIAB approach, in Collage, the drop sites must be decided upon in advance: These rendezvous points are the secret that Alice and Bob share. To put and retrieve the messages from these rendezvous points, the users of Collage have to perform a series of *tasks*. For example, if the drop site is a photo posted on Flickr [19] under the keyword "flowers," the *sender task* will be the series of HTTP requests required to post a photo with that keyword, and the *receiver task* will be composed of requests designed to retrieve the latest photos with that tag.

To bootstrap a Collage installation, the database of tasks that Collage employs must be distributed offline. This database needs to be constantly updated as the Censor reacts and the drop sites change (both in location and structure). It is, therefore, crucial that this bootstrap database is up-to-date: Otherwise, the agreement on the drop points between sender and receiver will be lost, breaking the communication channel.

Once Collage has been bootstrapped, further updates can be received through the Collage network. To receive these updates, however, the Collage client must be connected to the Internet. When the connectivity is sporadic, the client's database might become obsolete, and a new bootstrap round will become necessary.

We believe that MIAB is a good fit for bootstrapping Collage, because the only information that Alice must know about Bob in order to request a current copy of the task database is Bob's public key. This key should rarely change, so it can be shipped with the software. If the Censor is already too controlling to allow this, the key and the software (which is just a few kilobytes in size) can be published online in a variety of formats, so that it will be difficult for the Censor to find and block them all.

5 Implementation

To demonstrate the feasibility of MIAB, we have implemented a proof-of-concept application that can be used to post anonymous messages on Twitter, circumventing the block on social networks imposed by Tyria's Censor.

To provide a more open evaluation, we have published the code online: The application can be downloaded at `http://www.message-in-a-bottle.us`, together with our public key.

We have set up a small cluster of machines that monitors one of the most popular blog ping servers (`webblogs.com`[2]), looking for MIAB messages (our cluster plays Bob's part in the algorithm). When a message is found, its content is posted on Twitter under the handle `@corked_bottle`.

Also, the code can be used with a new public/private key-pair to verify how messages are recovered.

5.0.1 Public-key cryptography

Our application can use either GnuPG or X.509 certificates. We ship a self-signed X.509 certificate, containing the RSA public key.

5.0.2 Steganography

We use `outguess` [41] as the steganographic primitive. This tool works on Linux, *BSD, and Windows, so it will most probably run on Alice's computer. We are aware that an attack on `outguess` is known [25]. We choose this particular software out of convenience: Any steganographic tool can be used, and the change required

[2]For example, blogs hosted on Google's `blogger.com` ping this server, unless the blog owner disables this.

in the code is trivial. See also Section 3.2 for a discussion on an alternative approach with public-key image steganography.

5.0.3 Blog-fetching cluster

We use four servers with Intel Xeon 2.40GHz processors, 4 Gigabytes of RAM, and high-speed Internet connection.

5.0.4 Photo metadata

We hide the output of $\text{RSA}_{K_P}(K_s)$ in EXIF and XMP metadata, and in the filename. The ciphertext is stored in base 10, 16, or 60, depending on the format stored in the tags. We employ two photos, hiding 13.5 bytes of key per photo. The list of tags employed, and some sample content, is provided in Table 2. The Table also reports the number of bits, calculated with Shannon's entropy, that are *undetectable*. That is, the bits whose values have a high entropy throughout our dataset, and for which it is difficult to tell if they have been altered. In the example data in the Table, the undetectable bits are highlighted in bold. Note that the sample content has been extracted from a photo downloaded from Flickr [19]. In the case of tags holding a `DateTime` object, we hide the ciphertext only in the minutes and seconds.

To show that this ciphertext could be hidden in a variety of locations, we store a part of it in the digits contained in the filename (e.g., `DSC01234.JPG`). All these decision are arbitrary, and every MIAB instance can make different choices.

Metadata tag	Example	Format	Description	Undetectable bits
`Exif.Photo.SubSecTimeOriginal`	**22**	Ascii	Centiseconds when the photo has been taken (the date and time up to the seconds precision are kept in the DateTimeOriginal tag)	6.64
`Exif.Photo.SubSecTimeDigitized`	**90**	Ascii	Centiseconds for the DateTimeDigitalized tag	6.64
`Exif.Photo.SubSecTime`	**75**	Ascii	Centiseconds for the DateTime tag	6.64
`Exif.Photo.ImageUniqueID`	**713E4D7FC20E42C1 A1617CE642C28017**	Ascii	Arbitrary, unique identifier assigned to the photo	16
`Xmp.xmpMM.OriginalDocument`	**3856450D6BAB4CE5 AFD64DAF89C3C198**	Ascii	Arbitrary, unique identifier assigned original document, when editing a photo	16
`Xmp.aux.ImageNumber`	**49,202**	Ascii	Photoshop's auxiliary image number	16
`Exif.Image.ImageNumber`	**233**	Long	Unique number assigned to a photo	16
`Exif.Image.DateTime`	2011-09-25 15:**52:04**	Ascii	The date and time of image creation (we only alter the minutes and seconds).	11.81
`Exif.Photo.DateTimeOriginal`	2011-08-23 11:**27:27**	Ascii	The date and time when the original image data was generated (we only alter the minutes and seconds)	11.81

Table 2: EXIF and XMP tags: example photo taken from Flickr

5.1 Evaluation

MIAB relies on Bob's ability to fetch and process all the blog posts created on the Internet in real time. To prove that this is feasible, we have done so with our cluster.

Over the period of three months (72 days), we have seen $814,667,299$ blog posts, as shown in Figure 2. The average number of posts seen per day is $11,314,823$, and the highest traffic we have experienced is $13,083,878$ posts in a day. Being a proof-of-concept, our implementation delivers suboptimal performance in terms of blog throughput (it is written in Python). So, we had to use four machines to keep up with the post rate. To estimate the performance limits of a single machine fetching and analyzing web pages, we tested Apache Nutch [22], a high-performance web crawler written in Java and built upon Apache Hadoop [21]. In our experiment, we were able to fetch more than 15 million pages per day. Since we have observed that our MIAB cluster is limited by its

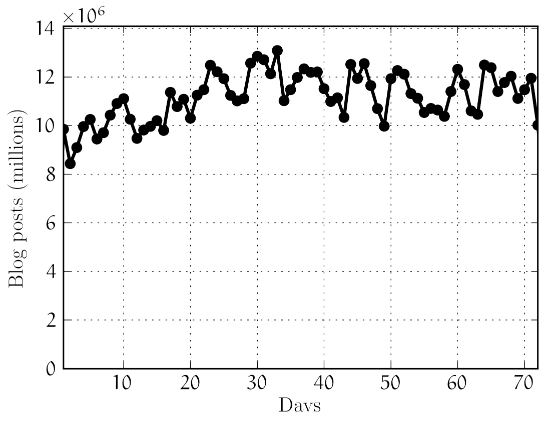

Figure 2: Blogs pings received over a period of three months

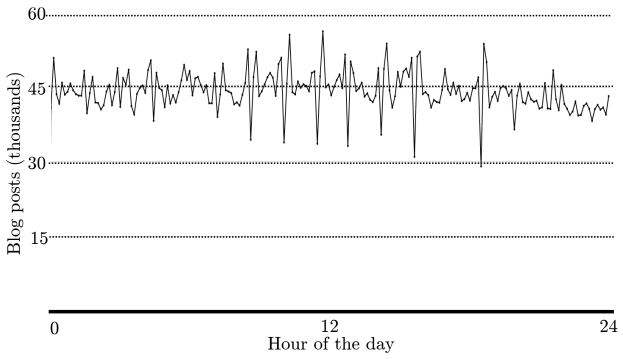

Figure 3: Blog ping received during day one (5 minutes slices)

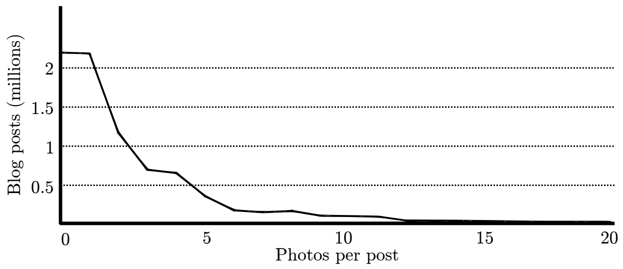

Figure 4: Number of photos per blog during day one

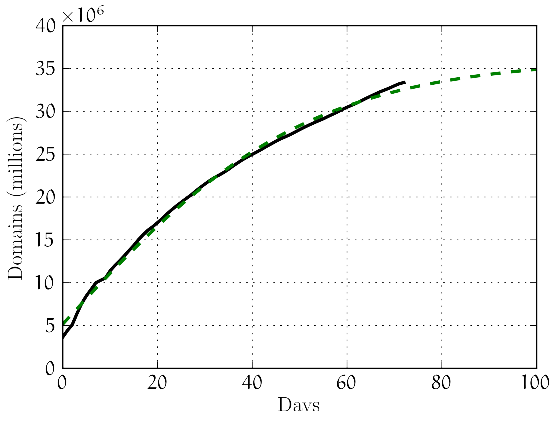

Figure 5: Blacklist size, when targeting fully qualified domain names

capacity of fetching and parsing the blog posts, and not by the image processing, we expect that an optimized implementation of MIAB should be able to achieve a similar performance.

The ping server we chose releases every five minutes the pings that it received during that period. To keep up with the ping rate, we process all the blog posts of one chunk of pings before the next one is released. Therefore, the maximum latency between the publishing of a post with MIAB content and our tweeting is ten minutes (five minutes in the ping server's queue, and another five before we process the post), with an average of five minutes.

During the course of a day, we have found that the rate of blog posts is quite stable, with an average of $44,146$ posts every five minutes, and a low standard deviation of $3,963$. See Figure 3 for an example.

We observed that approximately half of the blog posts contain photos. The distribution of the number of photos per post is shown in Figure 4.

5.2 Choosing the Right Blog

To send a message through MIAB, Alice has to publish a post on a blog that supports pings. Alice might already have posting privileges on such a blog: In this case, she can just use it. Otherwise, Alice needs to open one, or be added as an author to an existing one. It is therefore interesting to see how difficult it is for Alice to open a blog supporting pings.

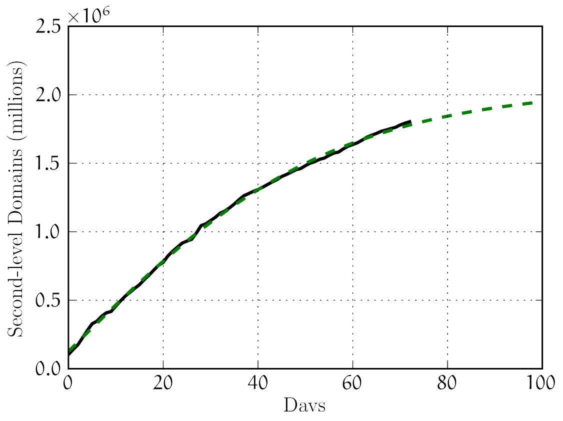

Figure 6: Blacklist size, when targeting second-level domains

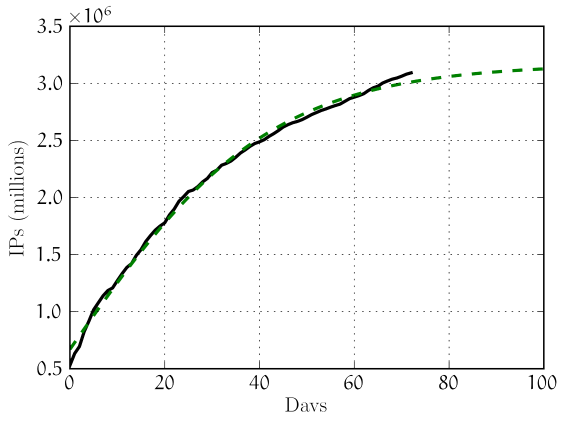

Figure 7: Blacklist size, when targeting IP addresses.

Platform	Number of sites (%)	Ping support
WordPress	63.49	yes
Joomla!	11.17	yes
Drupal	8.57	yes
Blogger	3.14	yes

Table 3: Blogging platform popularity: Alexa's top million (source: `builtwith.com` [7])

Platform	Number of sites	Ping support
WordPress	32	yes
TypePad	16	yes
Moveable Type	20	yes
BlogSmith	14	n/a, proprietary
custom made	8	n/a
Drupal	4	yes
Blogger	3	yes
Expression Engine	1	n/a, proprietary
Scoop	1	no
Bricolage	1	no, CMS

Table 4: Blogging platforms: 100 most popular blogs (source: `pingdom.com` [39])

In Table 3, we show the four most popular blogging platforms for the Alexa's top million websites in February 2012. These platforms account for more than 85% of all blogs, and they all support pings.

In Table 4, we show the platform that the 100 most popular blog chose to use in 2009. Of these, at least 75 support pings.

If the blog Alice has chosen does not support blog pings, Alice can perform the ping manually, as most ping servers offer this feature (e.g., `http://blogsearch.google.com/ping`).

5.3 Estimating the Blocking Set

The Censor might attempt to block all blogs by blacklisting their domain names, or IP addresses. It is essential to MIAB's availability that the blocking set (that is, the set of domains or IP addresses to block) is large, as this makes the maintenance of a blacklist impractical (this will be further discussed in Section 6.1). To estimate the size of the blocking set, we generated three blacklists from three months of recorded pings. It can be seen that these blacklists target Fully Qualified Domain Names (e.g., `myblog.blogspot.com.`, see Figure 5), second-level domains (e.g., `blogspot.com.`, see Figure 6), and IP addresses (see Figure 7).

These blacklists are cumbersome in size. For example, according to Netcraft, during the time of our experiment, there were $662,959,946$ FQDNs running web servers on the Internet [36]. In our experiment, we would have blacklisted $33,361,754$ FQDNs, which is 5% of the ones found by Netcraft. Blocking second level domains is also not practical. In our experiment, we blacklisted $1,803,345$ of them, which account for 1.4% of the global domain registrations (which are $140,035,323$, according to DomainTools [14]). Note that our blacklists only cover blogs that sent pings during our experiment. The total numbers will be higher, as the trend shown in

Figures 5, 6, and 7 suggests. To better quantify this, we checked the (hypothetical) blacklists's coverage on the pings that were emitted the day after we stopped updating the blacklists. We found that the blacklist targeting FQNDs had 12% miss ratio, the one targeting second level domains had 11% miss ratio, and the one targeting IP addresses had 11% miss ratio.

Even if the Censor chooses to pay the price of over-blocking and enforces these blacklists, Bob and Alice can still communicate though MIAB, thanks to a weakness in this blacklisting approach. That is, a blog might have multiple domain names (or IP addresses) assigned to it, but its pings are usually sent by only one of them. So, Alice can edit her blog at `blog.alice.com`, a domain name that never sends pings and, therefore, is not in the blacklist, and configure her blog to emit pings from the secondary address `alice.blogspot.com`. Bob will receive the ping, and visit the latter address to obtain the MIAB messages. In turn, the Censor also sees the ping, and will blacklist the secondary address in Alice's country. Alice will be still able to modify her blog through `blog.alice.com`, sneaking by the blacklist, and Bob will be able to visit `alice.blogspot.com`, because the Censor's block does not affect him.

6 Security Analysis

In this section, we discuss MIAB's resilience against the attacks that the Censor might mount to break the security properties that MIAB guarantees: availability, deniability, and confidentiality.

Since we lack a formal verification of the security of the steganographic primitives, and the powers of a real Censor are unknown, these goals are necessarily best effort.

6.1 Availability

The Censor might prevent Alice from sending messages through MIAB. Our main argument against this is that the Censor will be forced to ban a large part of the web, and he is likely unwilling to do so because of the pushback from its population and economic drawbacks. We will now describe in more detail the attacks that the Censor might devise to limit availability. For this discussion, we assume that deniability holds: In particular, the Censor cannot detect content generated with MIAB.

The Censor could *block blogging platforms*. In fact, this has already been attempted: Kazakhstan has been banning blogging platforms and social networks since 2008 [32]. In 2011, a court established they contributed to the spread of religious terrorism [24]. Despite the block, Kazakh bloggers have moved to self-hosted or less popular blogging platforms, and continue to grow in numbers [32]. There are over 165 million blogs on the Internet [5], and a good part of them are self-hosted, making the maintenance of a blacklist impractical (see Section 5.3). Even if the Censor is able to fingerprint blogs and successfully block each of them, there are other services that generate blog pings, for example, commenting services (e.g., Disqus, which is installed on 750,000 websites). Also, for her MIAB messaging, Alice might open a blog in favor of the ruling party: The Censor will be less likely to block that, because it shows that the population is supporting the regime. The Censor might also use pings to identify blogs, blocking any domain that emits a ping. This approach, which we already discussed in Section 5.3, is also vulnerable to a denial of service attack: Bob could forge bogus pings claiming to be any arbitrary domain (blog pings, just like emails, do not authenticate the sender), forcing the Censor to add it to the blacklist. Iterating this attack, Bob can de facto render the blacklist useless. To counter this, the Censor will have to maintain a whitelist of allowed domains. This is harmful to the regime, because of the social and economic costs of over-blocking, and can be easily circumvented (as we discussed in Section 5.3).

The Censor might want to *block pings*, or *shut down ping servers*. This attack is irrelevant when the blog (or ping server) is hosted outside the Censor's jurisdiction.

The Censor might try to *prevent Alice to post* on any blog. This requires both the ability to fingerprint a blog, and the technological power to deeply inspect the network traffic to detect and block posting. The cost of this attack increases as Alice will most likely be using a TLS connection (i.e., `https://`) to make the post, requiring the Censor to either man-in-the-middle the connection or perform statistical timing attacks. Also, Alice can create a post in a variety of ways, for example using client applications or by email (as many blogging platforms support that). The Censor therefore has to block all these vectors.

The Censor might try to *coerce content hosts to drop* MIAB content. However, to do so, he needs to identify such content (when deniability should prevent him from doing so). Also, the Censor has to drop the content fast: Bob will fetch the blog after just a few minutes, since blog pings are sent in real time.

The Censor might try to *overwhelm* MIAB *by creating large quantities of bogus messages*. Bob should be able to provide a fair service to its users by rate limiting blogs, and ban them when they exhibit abusive behavior. Since Bob will first inspect the list of pings, he will avoid fetching blogs that are over their quota or blacklisted. Also, search engines will fetch the same blog pings and will analyze and index their content. Since the Censor, to create a large number of blog posts, will have to either generate the content automatically or replay content, search engines should detect his behavior and mark it as spam. Blog ping spam (or *sping*) is already a problem that search engines are facing, since pings are used in Search Engine Optimization campaigns. Because of this, much research has been invested into detecting this type of behavior. Bob could leverage that research by querying the search engine for the reputation of blog or blog posts. Using a combination of blacklists, rate limitation, and search engines reputation systems, Bob is likely to thwart this attack.

The Censor might *perform traffic analysis* to distinguish regular blog posts from MIAB ones. However, since the blog post is created by a human, this is unlikely to be successful.

The Censor might decide to *strip metadata* from the digital pictures uploaded to blogs, or *re-encode the images* to remove any steganographic content. First, this is computationally expensive for the Censor. Moreover, MIAB can easily evolve and hide the ciphertext in other parts of the post (e.g., in the text), or use a steganographic system that can withstand re-encoding [28].

6.2 Deniability

The Censor might try to identify who is using MIAB, thus compromising deniability.

The Censor might try to *detect steganography*. MIAB uses steganography as a primitive, so its deniability relies on a good steganographic algorithm, and will benefit from any advancement in steganography. We discussed the use of public-key image steganography in Section 3.2. Moreover, Bob might support several steganographic algorithms, and let Alice pick the one she feels safe to use. Her choice can be encoded anywhere in the post (e.g, the fist letter of the title).

The Censor might try to *detect anomalies in digital photo metadata*. This is not relevant if the public-key image steganographic algorithm chosen does not alter them. Also, the content stored in the metadata is ciphertext: Assuming that the cipher is sound, the attacker cannot distinguish ciphertext from a random permutation of the input (otherwise, the cipher would fail an *indistinguishability under chosen ciphertext attack*). This attack is irrelevant if we choose appropriately the metadata tags where the ciphertext is hidden. Several tags are well-suited to carry arbitrary data (e.g., `Exif.Image.OriginalRawImageDigest` `Exif.Photo.ImageUniqueID`). Since a blog post can contain several photos (in Section 5.1, we have shown that it is quite common that a post contains up to five photos), the size of the ciphertext is not a factor that forces us to use detectable tags. Also, in our implementation, we chose not to use the EXIF's `MakerNote` tag, which stores proprietary binary data. This tag, however, usually accounts for the majority of the size of the metadata, and has several vendor-specific tags where additional ciphertext could be stored. We decided against using it in our proof-of-concept implementation, because we found enough bits of available entropy within the standard tags (see Table 2).

The Censor might try to *detect anomalous blogging behavior*. Since Alice manually publishes the post, it will be difficult to detect anomalous behavior in a single post (provided that the message that Alice composes is innocuous). However, the Censor might detect an anomaly if the posting frequency is higher than the other blogs hosted on the same platform. Alice can mitigate this risk by keeping under control her post rate. Also, Alice can use a blogging platform that encourages frequent posts with photos (i.e., a photoblog, or a *moblog* - a mobile blogging platform). Alice might also have multiple online personas, each of which posts on a different blog.

The Censor might try to *run an instance of* MIAB *to trick Alice into communicating with him*. This is a limitation common to any system that relies on a public key to identify the receiving party. This is mitigated by publishing Bob's public key in a variety of formats on the Internet, so that the Censor cannot find them and block or alter them.

The Censor might try to *perform a timing attack, correlating the effect of the action specified in the* MIAB *message* (in our implementation, the posting of a new tweet) *with the publishing of a new post*. We can mitigate this by letting the MIAB user specify when he wants the aforementioned action to be taken, inserting random delays and generating decoys (in our implementation, tweeting a message that was not received).

6.3 Confidentiality

To break confidentiality, the Censor must get access to the plaintext of the message that Alice sent. The message is encrypted with a public key, of which only Bob knows the private counterpart. Assuming that the cryptographic primitive is secure, the Censor will not be able to read the message. Also, the Censor might run an instance of MIAB to trick Alice, as we discussed in the previous section.

7 Related Work

There has been an extensive and ongoing discussion on anonymous and censorship-resistant communication. In this section, we review the main directions of research, highlighting how they measure against the goals that we have set for MIAB.

7.1 Anonymization proxies

The most established way to achieve online anonymity is through proxies, possibly with encrypted traffic [1, 33]. In 2010, a report from Harvard's Center for Internet & Society [38] shows that 7 of the 11 tools with at least 250,000 unique monthly users are simple web proxies.

These systems focus on hiding the user identities from the websites they are surfing to. They deliver a high performance but, in doing so, they do not satisfy any of the goals of MIAB: They can be easily be blocked, since it is easy to discover the addresses of the proxies and block them (low availability). Also, even if the traffic is encrypted, the users cannot deny that they were using the system (no deniability), and it is possible to mount fingerprinting and timing attacks to peek into the users' activity [6, 30].

One of the first systems that address deniability is Infranet [17]. Uncensored websites deploying Infranet would discreetly offer censored content upon receiving HTTP traffic with steganographic content. Availability is still an issue, since the Censor might discover and block these websites, so the collaboration of a large number of uncensored websites becomes essential.

7.2 Mix networks

Mix networks (e.g., Tor [13], Mixminion [11]) focus on anonymity and unlinkability, offering a network of machines through which users can establish a multi-hop encrypted tunnel. In some cases, depending on the location of the proxies with respect to the Censor, it is possible to link sender and receiver [4, 35]. These systems do not typically provide deniability, although Tor tries to mask its traffic as an SSL connection [48].

The availability of these systems depends on the difficulty to enumerate their entry points, and their resistance to flooding [15]. Protecting the entry nodes has proven to be an arms race: For example, the original implementation of Tor provided a publicly-accessible list of addresses. These entry points were blocked in China in 2009 [49]. In response, Tor has implemented bridges [45], which is a variation on Feamster's key-space hopping [18]. However, these bridges have some limitations that can expose the address of a bridge operator when he is visiting websites through Tor [34]. Also, distributing the bridges' addresses without exposing them to the Censor is theoretically impossible (since the Censor could play Alice's role, and Tor cannot distinguish one from the other). However, it might be possible to increase the cost of discovering a bridge so much that the Censor finds it impractical to enumerate them. This problem remains very complex and, so far, none of the approaches attempted has succeeded:

Proof of this is the fact that China has been blocking the vast majority of bridges since 2010, as the metrics published by Tor show [50].

7.3 Anonymous publishing

Here, the focus is on publishing and browsing content anonymously. Freenet [9] and Publius [53] explore the use of peer-to-peer networks for anonymous publishing. These systems deliver anonymity and unlinkability, but do not provide deniability or availability, as the Censor can see where the user is connecting, and block him.

Another direction that is being explored is to make it difficult for the Censor to remove content without affecting legitimate content (Tangler [52]). This system suffers similar pitfalls as Freenet, although availability of the documents is improved by the *document entanglement*.

7.4 Deniable messaging

Systems for deniable messaging can be split into two categories: The ones that require no trusted peer outside the country (e.g., CoverFS [3], Collage [8]), and the ones that do (e.g., Telex [54]). CoverFS is a FUSE file system that facilitates deniable file sharing amongst a group of people. The file system synchronization happens though messages hidden in photos uploaded to a web media service (Flickr, in their implementation). As mentioned by the authors, while CoverFS focuses on practicality, the traffic patterns could be detected by a powerful Censor. This would affect its availability and deniability.

Collage, which we already described in Section 4.2, improves CoverFS' concept of using user-generated content hosted on media services to establish covert channels. Collage has higher availability, as the sites where the hidden messages are left change over time, and it provides a protocol to synchronize these updates. As the authors point out, Collage deniability and availability are challenged when the Censor performs statistical and timing attacks on traffic. Also, if the Censor records the traffic, he can join Collage, download the files where the messages are embedded, and find out from his logs who uploaded them. To join the Collage network, a user needs to have an up-to-date version of tasks database, which contains the rendez-vous points. Since its distribution has to be timely, this is challenging (and can be solved with MIAB). Also, without regular updates, the tasks database can go stale, which requires a new bootstrap.

In Collage, the tasks defining the rendez-vous points are generated manually. Although every media-sharing site can potentially be used with Collage, generating all these tasks requires substantial work. MIAB, instead, can use millions of domains out of the box without human intervention, which makes monitoring more difficult.

A Censor might also join Collage and disseminate large quantities of bogus messages. Since the burden of finding the right message is on Collage's users, they would be forced into fetching many of the bogus messages, thus exposing themselves to statistical attacks. This could be mitigated by rate limiting the message fetching, but this decision would result in unbounded delays between the posting of the message and its retrieval, challenging the availability of the system. An effective solution for this denial-of-service is to generate a separate task list for any pair of people that need to communicate with each other. This can be done using MIAB as a primitive to disseminate these databases to the parties, so that they can bootstrap a Collage session. Once the two parties are exchanging messages, they can arbitrate the Collage channel they want to employ, depending on the performance that their communication requires.

Telex is a system that should be deployed by Internet Service Providers (ISPs) that sympathize with the dissidents. A user using the Telex system opens a TLS connection to an unblocked site (that does not participate in the protocol), steganographically embedding a request in the TLS handshake. When the ISP discovers the request, it diverts the SSL connection to a blocked site. While Telex deniability is sound, it can be subject to a denial of service by the Censor. Also, the Censor can discover ISPs implement Telex by probing the paths to various points on the Internet, and prevent the traffic originating from the country from going through those ISPs. Overall, Telex comes close to satisfying the goals that we set for MIAB. However, it requires the ISP collaboration, which is hard to set up, as the authors mention. MIAB, on the other hand, requires minimal resources (a good

Internet connection and a single machine should suffice, when implemented efficiently), and hence, it can be setup today, as we demonstrated by hosting our public implementation of this system.

A simpler solution to our problem is to use an webmail service running outside the censored country (e.g., Gmail) to send a message to an address where a anti-censorship service is listening. In fact, this is one of the channels used to distribute Tor briges (by sending an email to `bridges@torproject.org`). Since the Censor can downgrade the dissident's connection to the webmail provider and view the traffic, this method is not deniable. Also, the Censor can block emails sent to that address, challenging availability.

8 Conclusions

We have introduced MIAB, a deniable protocol for censorship resistance that works with minimal requirements. MIAB can be used as a standalone communication system, or to bootstrap protocols that achieve higher performance, at the cost of more demanding requirements. We have demonstrated MIAB feasibility with the implementation of a proof-of-concept prototype, and released its code open-source. The deployment of a MIAB instance requires minimal configuration and resources, since it relies on well-established and popular technologies (blog pings and blogging platforms). Also, we have shown that MIAB is resilient to attacks to its availability, deniability and confidentiality. Although a powerful Censor might be able to disrupt MIAB, in doing so he will suffer a high cost, effectively cutting the population of his jurisdiction out of a major part of the Internet.

References

[1] Anonymizer. Home page. `http://plone.anonymizer.com/`.

[2] D. Bachrach, C. Nunu, D. Wallach, and M. Wright. #h00t: Censorship resistant microblogging. *arXiv:1109.6874*.

[3] A. Baliga, J. Kilian, and L. Iftode. A web based covert file system. In *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, page 12. USENIX Association, 2007.

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

[5] BlogPulse. Report on indexed blogs. `http://goo.gl/SEpDH`, 2011.

[6] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 2005.

[7] BuiltWith. Content management system distribution. `http://trends.builtwith.com/cms`, 2012.

[8] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *USENIX Security Symposium*, 2010.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[10] CNN. Egyptians brace for friday protests as internet, messaging disrupted. `http://articles.cnn.com/2011-01-27/world/egypt.protests_1_egyptian-authorities-muslim-brotherhood-opposition-leader`.

[11] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[12] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. *IEEE Symposium on Security and Privacy*, 2012.

[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 21–21. USENIX Association, 2004.

[14] DomainTools. Internet statistic. `http://www.domaintools.com/internet-statistics/`.

[15] N. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the 18th conference on USENIX security symposium*, pages 33–50. USENIX Association, 2009.

[16] EXIF. Homepage. `http://www.exif.org`.

[17] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium, August*, 2002.

[18] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In *Privacy Enhancing Technologies*, pages 125–140. Springer, 2003.

[19] Flickr. Homepage. `http://flickr.com`.

[20] Flickr. Popular tags. `http://www.flickr.com/photos/tags/`.

[21] T. A. Foundation. Apache hadoop. http://hadoop.apache.org/.

[22] T. A. Foundation. Apache nutch. http://nutch.apache.org/.

[23] A. France-Presse. Pakistan blocks facebook over mohammed cartoon. http://www.google.com/hostednews/afp/article/ALeqM5iqKZNUdJFQ6c8ctdkUW0C-vktIEA.

[24] A. France-Presse. Kazakhstan blocks popular blogging platforms, 2011.

[25] J. Fridrich, M. Goljan, and D. Hogea. Attacking the outguess. In *ACM Workshop on Multimedia and Security*, 2002.

[26] FriendFeed. Simple update protocol. http://code.google.com/p/simpleupdateprotocol/.

[27] Google. Pubsubhubhub. http://code.google.com/p/pubsubhubbub/.

[28] R. Greenstadt. Zebrafish: A steganographic system. *Massachusset Institute of Technology*, 2002.

[29] S. Hetzl. Steghide. http://steghide.sourceforge.net/.

[30] A. Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies*, 2002.

[31] IPTC-IIM. Homepage. http://www.iptc.org.

[32] S. Kelly and S. Cook. Freedom on the net. *Freedom House*, 2011.

[33] B. Labs. The lucent personalized web assistant. http://www.bell-labs.com/project/lpwa/, 1997.

[34] J. McLachlan and N. Hopper. On the risks of serving whenever you surf: vulnerabilities in tor's blocking resistance design. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 31–40. ACM, 2009.

[35] S. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.

[36] Netcraft. May 2012 web server survey. http://news.netcraft.com/archives/2012/05/02/may-2012-web-server-survey.html.

[37] H. Noman and J. York. West censoring east: The use of western technologies by middle east censors, 2010-2011. 2011.

[38] J. Palfrey, H. Roberts, J. York, R. Faris, and E. Zuckerman. 2010 circumvention tool usage report. 2011.

[39] Pingdom. The blog platforms of choice among the top 100 blogs. http://royal.pingdom.com/2009/01/15/the-blog-platforms-of-choice-among-the-top-100-blogs/, 2009.

[40] N. Provos. Outguess - universal steganography. http://outguess.org, 1999.

[41] N. Provos. Defending against statistical steganalysis. In *10th USENIX security symposium*, 2001.

[42] N. Provos and P. Honeyman. Hide and seek: An introduction to steganography. *Security & Privacy, IEEE*, 2003.

[43] rssCloud. Homepage. http://rsscloud.org.

[44] TechCrunch. Syrian government blocks bambuserâĂŹs live video of crisis. http://eu.techcrunch.com/2012/02/17/syrian-government-blocks-bambusers-live-video-of-crisis/.

[45] Tor. Bridges. https://www.torproject.org/docs/bridges.

[46] Tor. Help users in iran reach the internet. https://lists.torproject.org/pipermail/tor-talk/2012-February/023070.html.

[47] Tor. Iran blocks encrypted traffic. https://blog.torproject.org/blog/iran-partially-blocks-encrypted-network-traffic.

[48] Tor. Iran blocks tor; tor releases same-day fix. https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix.

[49] Tor. Tor partially blocked in china. https://blog.torproject.org/blog/tor-partially-blocked-china.

[50] Tor. Tor users via bridges from china. https://metrics.torproject.org/users.html?graph=bridge-users&start=2010-01-01&end=2012-02-20&country=cn&dpi=72#bridge-users.

[51] L. Von Ahn and N. Hopper. Public-key steganography. In *Advances in Cryptology-EUROCRYPT 2004*, 2004.

[52] M. Waldman and D. Mazieres. Tangler: a censorship-resistant publishing system based on document entanglements. In *ACM conference on Computer and Communications Security*. ACM, 2001.

[53] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *USENIX Security Symposium*, 2000.

[54] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *USENIX Security Symposium*, 2011.

[55] XMP. Homepage. http://www.adobe.com/products/xmp/.