Message In A Bottle: Sailing Past Censorship

Luca Invernizzi
UC Santa Barbara
California, USA
invernizzi@cs.ucsb.edu

Christopher Kruegel
UC Santa Barbara
California, USA
chris@cs.ucsb.edu

Giovanni Vigna
UC Santa Barbara
California, USA
vigna@cs.ucsb.edu

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 of 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.

Keywords

Censorship Resistance, Deniable Communications, Steganography

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 [45], and even resorted to a complete isolation from the global network in critical moments [14]. For example, Pakistan recently blocked Wikipedia, YouTube, Flickr, and Facebook [2], and Syria blocked citizen-journalism media sites [53].

To sneak by the censorship, the dissident populations have resorted to technology. A report from Harvard's Center for Internet & Society [46] shows that the most popular censorship-avoidance vectors are web proxies, VPNs, and Tor [17]. These systems share a common characteristic: *They have a limited amount of entry points*. Blocking these entry points, and evading the block, has become an arms race: Since 2009, China is enumerating and banning the vast majority of Tor's bridges [59]. In 2012, Iran took a more radical approach and started blocking encrypted traffic [56], a move that Tor countered the same day by deploying a new kind of traffic camouflaging [55].

In this paper, we take a step back and explore whether it is possible to design a system with so many available entry points that it is impervious to blocking, without disconnecting from the global network.

Let's generalize the problem at hand with the help of Alice, a dissident who lives in the oppressed country of Tyria. Alice wants to establish, for the first time, a confidential communication with Bob, who lives outside the country. To use any censorship-resistant protocol, Alice must know *something* about Bob, and how to bootstrap the protocol. In the case of anonymizing proxies or onion-routing/mix networks (e.g., Tor), Alice needs the address of at least one of the entry points into the network, and Bob's address. Also, to achieve confidentiality, Alice needs Bob's public key, or something equivalent to that. In protocols that employ steganography to hide messages in files uploaded to media-hosting sites (such as Collage [12]) or in network traffic (such as Telex [64]), Alice must know the location of the first rendezvous point, and Bob's public key[1].

The fact that Alice needs to know this information inevitably means that the censor can learn it too (as he might pose as Alice). Bob cannot avoid this, without having some way to distinguish Alice from the censor (but this becomes a chicken-and-egg problem: How did Bob come to know that, since he never had any confidential communication with Alice before?). We believe that this initial *something* that Alice must 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 when trying to distribute bridge addresses to its users, without exposing these addresses to the censor [54]. It is because of this crack that China has been blocking the majority of Tor traffic since 2009 [59]: The number of entry points is finite, and a determined attacker can

ACSAC '13 Dec. 9-13, 2013, New Orleans, Louisiana USA
Copyright 2013 ACM 978-1-4503-2015-3/13/12 ...$15.00.

[1] To be precise, in Telex, Alice does not need to know the precise location of the proxy running the Telex protocol. However, she needs to be aware that a Telex proxy will occur somewhere in the path to the decoy destination. Here, we consider this destination to be the rendezvous point she must know.

enumerate them by pretending to be Alice.

With Message In A Bottle (MIAB), we propose a protocol in which Alice needs to know the *least possible* about Bob, and how to bootstrap the protocol. In fact, we will show that it is enough for Alice to know Bob's public key, and nothing else. Alice must know at least Bob's public key to authenticate him, so that she can be sure she is not talking to a disguised censor. However, contrary to systems like Collage and Telex, there is *no prearranged rendezvous point* where Alice and Bob must meet.

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? 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*. Using these pings as a broadcast system, Bob gets 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 potential meeting point for Alice and Bob. However, there are over 165 million blogs online [8], and since a blog can be opened trivially by anybody, for our practical purposes they constitute an infinite resource. As a result of one of our experiments (see Section 5.6), we will show that, to blacklist all the potential MIAB's drop points in a three-months period, the Censor should block at least 40 million fully qualified domain names, and four million IP addresses (as we will show, 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 will be 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.

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 delay-tolerant messages (e.g., emails, articles, tweets). Also, MIAB can be employed to exchange the necessary information to bootstrap more demanding censorship-resistant protocols. These might be more efficient than MIAB, at the cost of requiring more data upon bootstrap (such as Collage, or Telex). We will showcase a few applications of MIAB in Section 4.

Our main contributions are:

- A new primitive for censorship-resistant protocols that requires Alice to have minimal initial knowledge about Bob, and how to bootstrap the protocol;
- 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, Tyria's 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[2].

2.1 The Censor's Capabilities

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 available to him.

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 might 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.

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.

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 suspicious traffic, he will do so in a manner that the user would not easily notice.

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

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

does not need to know any information, but Bob's public key.

In particular, we aim to satisfy these properties for MIAB:

- **Confidentiality:** The Censor should not be able to read the messages exchanged between Alice and Bob.
- **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.
- **Unobtrusive deployment:** Deploying a MIAB instance should be easy and cheap, so that small organizations or private citizens with some disposable income (e.g., Bob) can do it.

We will give a detailed analysis about to which extent we achieved these properties in Section 6.

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 the notion of *blog pings*, a concept that will play a crucial role in our system. 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 [31], Google's PubSubHubbub [33], and rssCloud [50]).

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).

3.2 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 (also shown in Figure 1):

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 software to embed a message M into the photos. The message is hidden using public-key steganography [60] (PKS), with Bob's public key. We will discuss the available choices for the steganographic algorithm in Section 3.4.
4. Alice publishes the blog post, containing the processed photos. Alice can choose the blog arbitrarily, provided it supports blog pings. In Section 5.5, we will show that most of the popular blog engines provide blog pings.
5. The blog emits a ping to some ping servers.
6. 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.

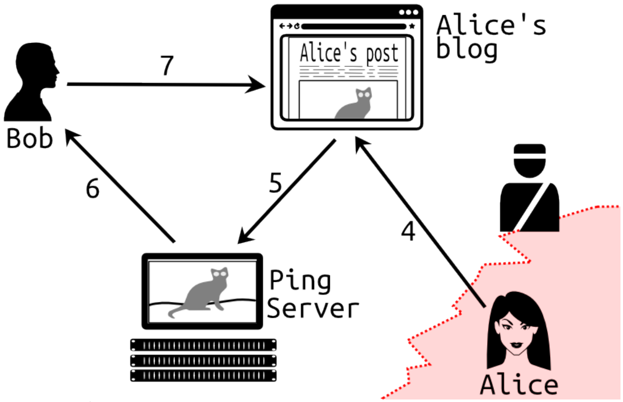


Figure 1: Alice sends a message to Bob, using the MIAB protocol in its basic form.

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

This concludes the MIAB protocol in its basic form; we will expand it further in Section 4.

3.3 Distributing The Software Bundle

To use MIAB, Alice needs a copy of the software, and Bob's public key. The key should rarely change, so it can be distributed with the software. This bundle can be downloaded in clear-text before the Censor becomes too controlling. Otherwise, it can be distributed via USB drives, or spam. Also, it could be published in multiple locations online, encoded in a variety of formats (e.g., DeCSS code diffusion [1]), so that the Censor cannot easily fingerprint and block them all. In this case, Alice should collect several of these bundles and compare them, to mitigate the possibility that she is using bundles that were disseminated by the Censor. However, this possibility cannot be ruled out with certainty. We expect that the Censor will also have a copy of the software.

Alice's need for the initial bundle is certainly a shortcoming of MIAB. However, distributing the software and establishing a root of trust is a fundamental problem, with no easy solution, that affects every related work (e.g., Collage, Tor – see Section 7). Moreover, MIAB represents a step toward solving this fundamental problem, as its bundle need updates only in exceptional cases (e.g., Bob's key has expired), and therefore can be used to bootstrap other protocols, distributing the rendezvous points. We note that also Telex has this property, but it needs the collaboration of ISPs to be deployed, whereas MIAB does not.

3.4 Choosing The Steganographic Scheme

The choice of an appropriate steganographic scheme is critical to achieve our confidentiality and deniability goals. Ideally, we would like to have a *perfectly secure* stegosystem, which is a system where the stego object (i.e., the object with an embedded hidden message) exactly matches the probability distribution of the cover source (i.e., the object before the embedding). Solutions to this problem exist [51,63], but they require an exact knowledge of the probability distribution of the cover source. Unfortunately, this knowledge is hard to obtain, if not impossible [9], from real digital media, such as photos. To overcome this setback, these solutions opt to artificially generate the cover object. More pragmatic approaches, instead, focus on hiding messages in real digital media by minimizing perturbations, in the hope that the image noise will make these alterations appear inconspicuous.

The positive dependence between the fraction of changes in the cover and their detectability is formalized under the "square root law of steganography" [27]. Fortunately, MIAB needs to embed a very limited payload, consisting only of a few hundred bytes, which are embedded in cover photos whose size is in the hundreds of Kbytes.

To minimize our detectability, then, we need to minimize the alterations made during the embedding, and to curtail our changes in difficult-to-model areas of the digital media. Among the current state-of-the-art proposals in this field, we selected, and used in our experiments, an adaptive LSB-matching steganographic approach that chooses the location of the pixels to alter with the help of Syndrome-Trellis Codes (STCs) [25]. LSB-matching, also called ± embedding, consists in randomly adding ± 1 to the least significant bit of pixels to match the hidden message. LSB-matching has been shown to be near optimal when only a single pixel can be utilized [23]. Pure LSB-matching schemes have been broken [32, 47], exploiting an invalid underlying assumption of LSB-matching, which considers natural image noise to be independent among pixels. Using STCs, LSB-matching can be augmented by choosing the pixels in which to embed the message wisely, minimizing detectable alterations. In particular, we chose a near-optimal algorithm [26] that minimizes distortions. This algorithm first associates a cost to each possible pixel alteration, and then it finds the least costly series of modification needed to embed the message. Both its space and time complexity are linear with respect to the size of the cover source.

LSB-matching with STCs is also the approach chosen by HUGO [48], which is a stegosystem that is optimized to embed large messages. Although MIAB does not benefit from HUGO's efficient embedding, as its messages are short, this stegosystem can give us a reference point about the state of the art in the steganalysis of a system close to ours. This is because HUGO has been targeted by steganalysis experts from all over the world, in the first international challenge on steganalysis called "Break Our Steganographic System" (BOSS [6]). During BOSS, each participating team competed in accurately classifying a set of photos from different cameras, where some photos carried a secret payload of approximately 0.4 bit per pixel (to put this into perspective, MIAB's embedding rate is less than 0.001 bpp). The winning team, Hugobreakers [30], achieved about 80% accuracy, with a false positive rate between 10% and 20%, depending on the digital camera that took the photos. Unfortunately, we cannot test these detectors against our implementation, because they are not publicly released. However, for our stegosystem, we consider it safe to expect a rate of false positives at least over 10-20%. This is because the messages that we embed are much shorter and the square root law of steganography applies. If we consider also that, as we will show in the next section, tens of thousands of photos are published in blog posts around the world every minute, we expect that, if our adversary tries to detect stego images in the blog posts stream, she will be overwhelmed by the sheer number of false positives.

Finally, we would like to note that the MIAB protocol is decoupled from the particular scheme in use, and swapping schemes is quite easy. Because of this, a MIAB implementation can support multiple PKS schemes, much like the SSL/TLS protocol does with cipher suites. In this scenario, Alice can pick the PKS scheme she feels safe to use, and encode her choice into a pre-determined place in the blog post (e.g., the fist letter of the title will decide the scheme). Other schemes that we have been considering are Gibbs constructions [24], and Greenstadt's work on an image-steganography scheme that can withstand re-encoding [34].

4. ACHIEVING TWO-WAY MESSAGING

The MIAB protocol can be used to achieve two-way communication in two ways: either by bootstrapping a more demanding censorship protocol (that requires more than just Bob's public key to be initiated), or with a channel-hopping protocol that establishes rendezvous points in Internet locations chosen at will by the users.

4.1 Bootstrapping Another Protocol

We can use MIAB to bootstrap Collage [12], which 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 [28] 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 be necessary.

We believe that MIAB is a good fit for bootstrapping Collage, because the only information that Alice must know in order to request a current copy of the task database is Bob's public key. MIAB could also be used to communicate with a censorship-resistant micro-blogging service, like `#h00t` [4], or a privacy-preserving one, like Hummingbird [16].

4.2 Covert Messaging

Another option to achieve two-way messaging is to extend the MIAB protocol to let Bob reply to Alice. To do so, Alice will need to inform Bob about the next rendezvous point, and about the steganographic scheme Bob should use in his reply.

If Bob is a human (i.e., he is not running an automated service), Alice will run the same algorithm presented in Section 3.2, with a small modification in Step 3. In particular,

- Alice will choose the next rendezvous point R_1 (e.g., `site_a.com` in Figure 2). As rendezvous point, Alice will choose the URL of a web page where Bob can post additional content: for example, a forum, a media-sharing service, or an online newspaper that lets its users comment the news. If possible, Alice should indicate a website that she routinely visits, so that the Censor will not be alarmed when she visits that URL. This website should be located outside Tyria, so that the Censor cannot see if Bob is visiting the site (the censor would need to know Bob's IP address, but MIAB does not prevent this from happening).
- Alice will decide which steganographic scheme S_1 Bob should use to hide his answer. Note that now Alice is

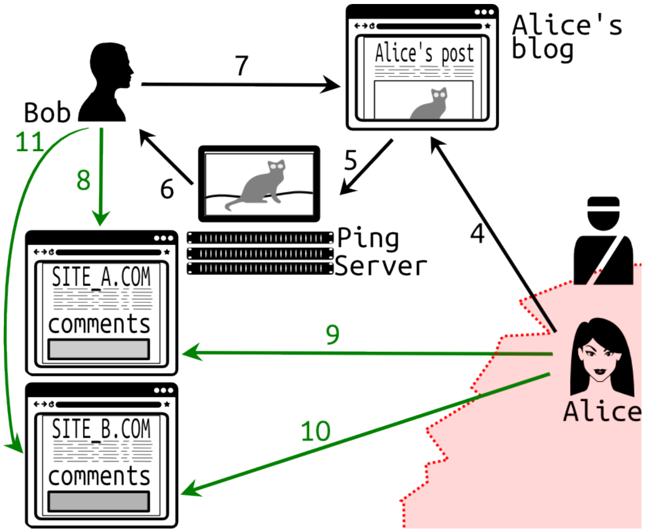


Figure 2: Covert messaging with MIAB.

not restricted to PKS schemes: She can also indicate a shared-key scheme, along with a secret password to initialize the scheme.

- If Alice intends to send additional messages to Bob, she should also specify an additional rendezvous point, R_2 (`site_b.com` in the Figure), and steganographic scheme S_2.

Alice will then append this information to the message M, which she will then embed in the photos. Following the usual protocol, Bob will then recover Alice's message, and obtain R_1, R_2, S_1, and S_2. Bob will then prepare his reply to Alice, and post it (Point 8 in Figure 2), encoded with S_1, on the rendezvous point R_1.

Alice will periodically check R_1, looking for Bob's reply (Point 9 in the same Figure – we will empirically show in Section 5 that this wait is less than ten minutes). Alice could check R_1 manually or, if the rendevouz point supports it, by email or RSS feed. When she finds Bob's message, she will decode it with S_1, completing the two-way messaging cycle.

If Alice requires any further communication with Bob, she will post her reply, encoded with S_2, on the rendezvous point R_2, which Bob will periodically check. Following the guidelines for the previous message from Alice, this reply will also contain R_3 and S_3, if Alice expects an answer from Bob (and, optionally, R_4 and S_4, if she plans to send additional messages).

If Bob is running an automated service, Alice will also need to instruct the service on how to reply. To do so, much like a Collage task, she will add to M the sequence of actions that the automated service must follow to post a reply (e.g., perform a `POST` request to `http://site_a.com`, with payload `comment=ANSWER`). Alice has the option to provide the cover channel for the automated service's reply (i.e., she could write a comment relevant to R_1, or an image, and embed it in M). Otherwise, the automated service will embed the reply into a spam message, which will then post. Spam messages can be generated (e.g., see [41]), or real spam messages can be collected from a set of email accounts.

Note that, for the sake of simplicity, in describing this covert messaging protocol we assumed that Alice choses the rendezvous points and encryption schemes manually. However, this can be automated, so that Alice can use the protocol without hassle. In particular

- MIAB's client software can detect the available steganographic schemes on Alice's computer, and pick one at random whenever required
- Alice, with the help of a browser extension, can collect potential rendezvous points while surfing the web in her usual routine. The extension would observe when Alice makes `POST` requests containing files or long strings, without being authenticated to the website she is interacting with (e.g., over clear-text HTTP, or without a `Cookie` HTTP header) , and record them as possible rendezvous points. Essentially, the extension would observe Alice's browsing, looking for commenting systems and media-sharing web services. Once one of these rendezvous point is used in the covert messaging, the browser extension can collect the answer automatically, after waiting for the appropriate time for Bob to answer (see Section 5 for details).

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`), 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.

5.1 Blog-Fetching Cluster

We use four servers with Intel Xeon 2.40GHz processors, 4 Gigabytes of RAM, and high-speed Internet connection. The need for a cluster of machines comes from the additional overhead of our detailed data collection for this experiment. Without it, MIAB can be run on a single machine. We have successfully done so, and the code published online refers to this second, high performance, version.

5.2 Ping Server

We use `webblogs.com`. This server is the oldest ping server of the web, and receives million of pings every day. For example, blogs hosted on Google's `blogger.com` ping this server by default, unless the blog owner disables this. The server does not allow real-time monitoring of pings, but releases a tarball with the latest pings every five minutes.

5.3 Steganography

For our experiments, we have implemented the stegosystem described in Section 3.4. To embed a message, first we encrypt it with RSA using Bob's public key, then we apply our tool. As we mentioned in Section 3.4, we can estimate our resilience to steganography by looking at how effectively the HUGO stegosystem was attacked during the BOSS competition. For the reader's convenience, a summary of the results is given in Figure 3. Since MIAB's embedding rate is at least two orders of magnitude smaller than the one used during BOSS, and that the square root law of steganography applies, we expect a dramatic decrease of detection rates, and increase of false positive rates. In particular, a direct application of this law indicates that we should expect single digit detection ratios. Unfortunately, we cannot verify this, as the detection systems used during BOSS have not been release

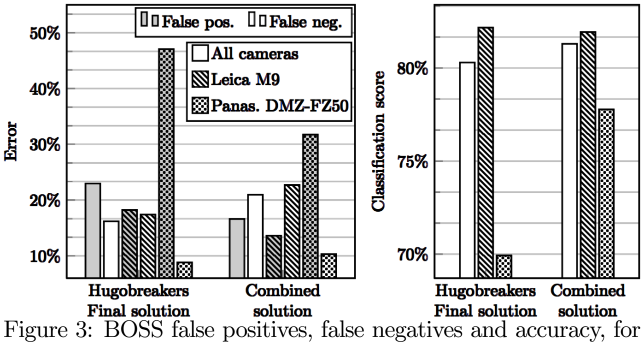


Figure 3: BOSS false positives, false negatives and accuracy, for the winning team and all teams combined (by simulating collusion among teams)

publicly. Even if we consider the detection rates of Figure 3, the number of false positives detected is likely to overwhelm the Censor, considering the vast prevalence of clean photos in the blogs' stream. Also, Alice can throw the Censor off track by publishing a few clean images for each stego photo published.

Since our stegosystem does not use a key, the Censor will be able to extract the hidden encrypted message, if there is any. If there is none, the Censor will just extract gibberish. If it is possible to tell the difference between the two kinds of extracted data, the Censor will break our stegosystem. Since our messages are encrypted with RSA, they will have high entropy, which is a feature that can be easily quantified to be used for machine-learning classification. However, our stegosystem is designed to select the pixels in which to embed the message preferring noisy, hard to modify areas of natural photos, which are also highly entropic. To check if it is likely that the Censor will be able to use the extracted data to tell apart clean images from the ones with stego content, we have trained a few classifiers that aim to do so.

First, we have prepared two datasets. In the first dataset, we have applied (the decoder part of) our stegosystem to extract gibberish from 5,000 clean photos downloaded from Flickr [28]. To reach a broad number of photographers, we selected the photos using Flickr popular tags. In the second dataset, we have encrypted with RSA 5,000 random 450-bytes messages (this length is an upper bound for what MIAB needs). From these datasets, we extracted features that quantify the entropy of the strings. In particular, we have calculated the Hamming's entropy, the Chi-Square test's p value, the arithmetic mean of the byte, the Monte Carlo approximation for π, and the correlation coefficient between consecutive bytes.

Using these features, we trained and tested, with a 10-fold cross validation, K-Nearest Neighbor classifiers (with $1 \leq k \leq 50$), a Naive Bayes classifier, two neural networks (with one and two hidden layers), and a Support Vector Machine classifier. Also, we trained a vote meta-classifier, combining these results. The precision and recall of the best in precision for each class of classifiers is shown in Table 1. We attribute the poor performance of the classifiers to the combination of the shortness of the messages and the natural and digital noise in the photos, that is empirically entropic enough to be hardly distinguishable from RSA-encrypted messages.

5.3.1 Public-key Cryptography

We ship a self-signed X.509 certificate, containing the RSA public key, with a 2048 bits modulus.

Classifier	Precision		Recall	
	RSA	Flickr	RSA	Flickr
K-NN ($k = 14$)	52.05%	51.37%	41.76%	61.52%
Naive Bayes	50.11%	50.98%	90.48%	9.90%
ANN (2 hidden layers)	50.14%	50.09%	40.36%	59.86%
SVM	48.15%	48.77%	58.52%	38.52%
Vote	82.56%	17.08%	49.89%	49.48%

Table 1: Best classifiers for each class.

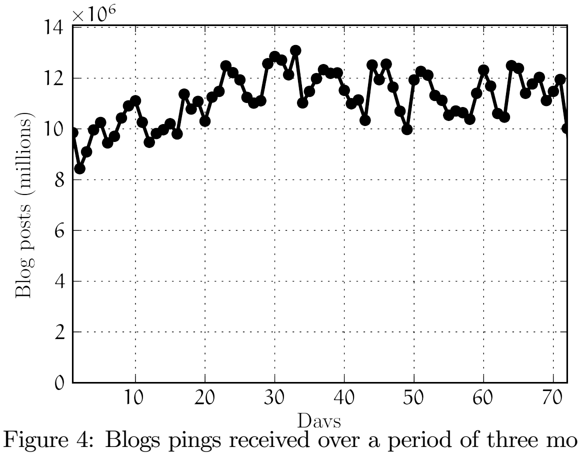


Figure 4: Blogs pings received over a period of three months

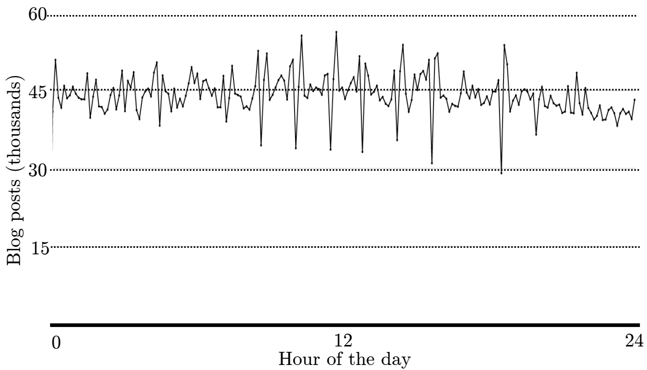


Figure 5: Blog ping received during day one (5 minutes slices)

5.4 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, as well as our single-machine version of MIAB.

Over the period of three months (72 days), we have seen $814,667,299$ blog posts, as shown in Figure 4. 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.

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 5 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 6.

5.4.1 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

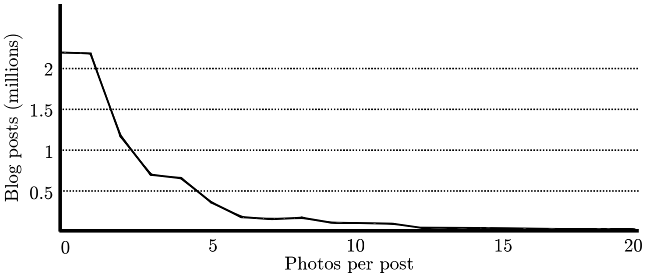


Figure 6: Number of photos per blog during day one

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).

An advantage of the stego algorithm we have chosen is that it is much faster in extracting potential hidden messages than embedding them. On average, it takes 0.56 seconds per image to embed a message, and 0.02 seconds to recover it. In particular, performing exclusively our steganographic extraction on a 5-minute pings chunk takes, on average, 2m:51s on a Intel-i7 eight-core laptop. Performing exclusively the RSA decryption takes 2m:35s. Fetching all the images from the blogs takes an average of 4m:17s. Since the steganographic test is CPU bound, and the blogs fetching is I/O bound, the two operations can execute simultaneously with an acceptable loss in performance: processing a five-minutes chunk of pings with the full MIAB, with either scheme, completes under five minutes on average, with a processor load under 90%. This is close to the requirements of Tor exit nodes, which several individuals are running, and it is well in reach for companies countering censorships, like Twitter.

Depending on the running time and memory requirements of the PKS scheme Alice and Bob have chosen, a single machine may not be able to cope with the workload. To solve this issue, Bob can opt to scale horizontally, adding more machines. Otherwise, Alice and Bob may agree beforehand that Bob will only fetch and analyze blog posts whose URL has some particular properties. For example, they might agree that blog posts with stego messages will have a URL whose length is even (or odd). In this example, this agreement would reduce the workload approximately by a factor of two, and Alice can make sure that Bob will keep examining her posts by choosing the blog post title accordingly, since the title is usually part of the URL. Of course, the Censor will also be aware of this, so this device will reduce the workload equally for both the Censor and Bob.

The maximum length of the embedded message depends on the channel and the PKS scheme in use.

In our scheme, we enforce a size limit of 450 bytes, since as the covert message becomes larger, the chance that the image might be flagged as suspicious by the Censor also increases.

5.4.2 *Scale*

MIAB should cope well with a high volume of hidden messages. A single machine can accommodate 15-20 million posts per day. To put this in perspective, if the entire population of Iran would start blogging daily, Bob would need five more machines. Also, a large number of useless posts made by MIAB users should not affect search engine results, because search engines already deal with a high volume of spam pings [38].

5.5 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.

In Table 2, 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 3, 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`).

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

Table 2: Blogging platform popularity: Alexa's top million (source: `builtwith.com` [11])

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 3: Blogging platforms: 100 most popular blogs (source: `pingdom.com` [49])

5.6 Estimating The Blocking Set

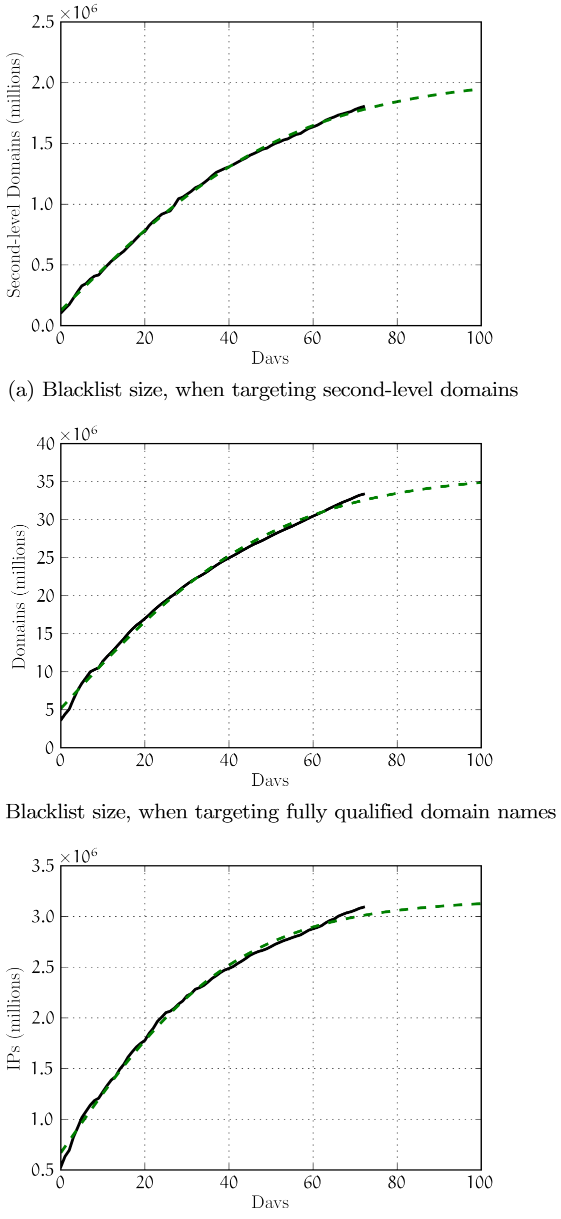
The Censor might attempt to block all blogs by blacklisting their domain names, or IP addresses. It is essential to MIAB's availability that the blocking set (that is, the set of domains or IP addresses to block) is large, as this makes the maintenance of a blacklist impractical (this will be further 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 7b), second-level domains (e.g., `blogspot.com.`, see Figure 7a), and IP addresses[3] (see Figure 7c).

5.7 Bypassing The Blacklist

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 [43]. 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 [18]). Note that our blacklists only cover blogs that sent pings during our experiment. The total numbers will be higher,

[3] Note that some blogs advertise only their IP address in their pings: this is why Figure 7c shows a faster trend than Figure 7a.



(a) Blacklist size, when targeting second-level domains

(b) Blacklist size, when targeting fully qualified domain names

(c) Blacklist size, when targeting IP addresses.

Figure 7: Estimating the blacklist size.

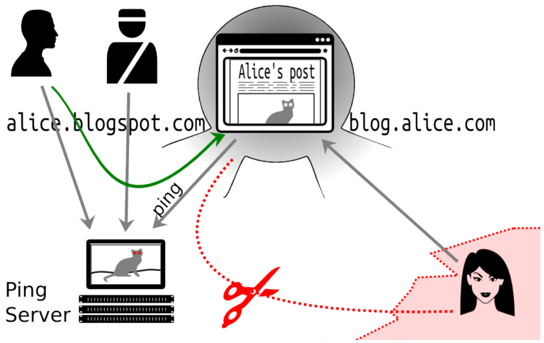


Figure 8: Bypassing the blacklist.

as the trend shown in Figures 7b, 7a, and 7c suggests. To better quantify this, we checked the (hypothetical) blacklists' 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 through MIAB, thanks to a weakness in this blacklisting approach.

Bloggers usually associate multiple domain names/IP addresses to their blog: on `blogger.com`, this is configurable in the "Basic Settings" admin page. These domain names constitute the *entry points* to the blog. Blog pings contain only one of these entry points. Bob only needs one of these entry points to visit the blog. Instead, the Censor needs each one of the entry points, because he wants to prevent Alice from accessing her blog. So, Alice can side-step the blacklist using an entry point not included in her blog's pings, which the Censor cannot easily discover.

For example, in Figure 8, Alice can edit her blog at `blog.alice.com`, a domain name that never sends pings and, therefore, is not in the blacklist. Her blog emits 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 [37]. In 2011, a court established they contributed to the spread of religious terrorism [29]. Despite the block, Kazakh bloggers have moved to self-hosted or less popular blogging platforms, and continue to grow in numbers [37]. There are over 165 million blogs on the Internet [8], and a good part of them are self-hosted, making the maintenance of a blacklist impractical (see Section 5.6). 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 *block any domain that emits a ping.*

We discussed how to evade suck a blacklist in Section 5.7. Also, this approach is 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 5.6).

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 *re-encode images* to remove any steganographic content. Since the censor cannot find all the entry points to the blogs, he must re-encode every image crossing his country's borders (not only blogs). 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 (see also Section 3.4). For example, a technique to do so is used in the shared-key steganographic scheme YASS [52], which can survive a second compression, additive noise, and light filtering. The steganographic primitive and channel (images, videos, or text) can be chosen at will, and should be selected when deploying MIAB, considering the particular Censor faced.

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. Also, the Censor will need to detect steganographic content with a very low fraction of false positives, because of the vast prevalence of clean image in blog posts. For more details, see in Section 3.4.

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 (e.g., a photoblog, like Instagram). 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).

The Censor might try to *perform a replay attack.* Let's suppose that the Censor suspects that Alice used MIAB to post a message M to Twitter. To confirm his suspects, the Censor would collect blog posts from Alice, and publish the same posts on a blog platform under his control. If the same message M is tweeted when the posts are re-published, the Censor's suspicion is probably correct. The Censor can repeat the process to rule out any remaining doubt. Replay attacks can be thwarted by including a hash of the URL of the blog post in the message. After receiving the message, Bob will check that the hashed URL in the message matches with the one of the post, and will discard the message if the check fails. Also, Bob could keep a hash of the photo in a Bloom filter, to avoid reuse.

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.

6.4 Unobtrusive Deployment

With our proof-of-concept application, we have demonstrated that it is possible to run a MIAB instance on a single modern machine with a fast domestic Internet connection. We recommend a 100Mbps connection, such as the one provided by Comcast in the US, also considering the fluctuations in the speed achievable at rush hours. Therefore, any private citizen with some disposable income, and living in a country with a modern offering for Internet connectivity, can run a MIAB instance. Moreover, this burden is sustained only by Bob; Alice has minimal requirements to participate in the protocol.

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 [3,39]. In 2010, a report from Harvard's Center for Internet & Society [46] 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 have a low latency, which makes web surfing possible (which MIAB does not). In doing so, they do not satisfy any of the goals of MIAB: They can be blocked, since it is possible 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 [10,35].

One of the first systems that address deniability is Infranet [20]. 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/Onion Networks

Mix networks (e.g., Mixminion [15]), and Onion networks (e.g., Tor [17]), focus on anonymity and unlinkability, offering a network of machines through which users can establish a multi-hop encrypted tunnel. They also typically have a low latency, which makes web surfing possible (in contrast with MIAB). In some cases, depending on the location of the proxies with respect to the Censor, it is possible to link sender and receiver [7,42]. These systems do not typically provide deniability, although Tor tries to mask its traffic as an SSL connection [57]. The availability of these systems depends on the difficulty to enumerate their entry points, and their resistance to flooding [19]. 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 [58]. In response, Tor has implemented bridges [54], which is a variation on Feamster's key-space hopping [21]. However, these bridges have some limitations that can expose the address of a bridge operator when he is visiting websites through Tor [40]. 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 [59].

An interesting variation on proxy-based censorship circumvention is described in Fifield et al. [22]. Here, a group of Internet users run an Adobe Flash application in their web browser. This application creates a short-lived proxy (in their implementation, the proxy is for the Tor network), so that these users can contribute a part of their bandwidth to the network. These ephemeral web proxies lower the barrier to set up a Tor proxy, and their short lives make their enumeration more cumbersome. However, a central part of this system is a special service, called the Facilitator, whose job is to keep track and distribute proxy addresses. By sniffing the Facilitator's traffic, the Censor can learn the proxy addresses, as they get requested from the clients. The authors point out that, once the system gets popular, keeping a blacklist might be cumbersome, because of the sheer number of available proxies. Also, as the author point out, the Censor might flood proxy registrations, and exhaust the list of proxies that the Facilitator keeps. This, also, might be mitigated by sheer numbers, as the authors point out.

7.3 Anonymous Publishing

Here, the focus is on publishing and browsing content anonymously. Freenet [13] and Publius [62] 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 [61]). 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 [5], Collage [12]), and the ones that do (e.g., Telex [64]). 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 4.1, 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 the tasks database, which contains the rendezvous 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 rendezvous 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.

Finally, we would like to stress that the Censor we are facing in MIAB is capable and willing to man-in-the-middle or block HTTPS connections (e.g., like China [44], and Iran [56] are doing). If this is not true, there is a simpler solution to our problem: the usage of a third-party service. For example, Alice could send a message though one of the many webmail services hosted outside the censored country (e.g., Gmail) to an address where an anti-censorship service run by Bob is listening. In fact, this is one of the channels used to distribute Tor bridges (by sending an email to `bridges@torproject.org`). This method is highly available, because of the great number of free webmail offerings. However, even if Alice and Bob have a deniable method of embedding and retrieving messages from ordinarily-looking emails, the webmail service will have an indisputable proof that Alice and Bob exchanged something, and the Censor might, at any point in the future, pressure the webmail service to release that information to her (e.g., like China did by wire-tapping Skype [36]). Instead, a steganographic message published in a broadcast channel, like MIAB's, will be out of reach from the Censor, at least as long as Bob's private key is kept safe.

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.

9. ACKNOWLEDGEMENTS

This work was supported in part by the Office of Naval Research (ONR) under grant N000140911042, the Army Research Office (ARO) under grant W911NF0910553, the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537, and Secure Business Austria.

10. REFERENCES

[1] Gallery of css descramblers. `http://www.cs.cmu.edu/~dst/DeCSS/Gallery/`.

[2] Agence France-Presse. Pakistan blocks Facebook over Mohammed cartoon. `http://www.google.com/hostednews/afp/article/ALeqM5iqKZNUdJFQ6c8ctdkUWOC-vktIEA`.

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

[4] Bachrach, D., Nunu, C., Wallach, D., and Wright, M. #h00t: Censorship resistant microblogging. *arXiv:1109.6874* (2011).

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

[6] Bas, P., Filler, T., and Pevnỳ, T. "break our steganographic system": The ins and outs of organizing boss. In *Information Hiding* (2011), Springer, pp. 59–70.

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

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

[9] Böhme, R. *Advanced statistical steganalysis.* Springer-Verlag, 2010.

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

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

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

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

[14] 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`.

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

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

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

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

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

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

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

[22] Fifield, D., Hardison, N., Stark, J., Porras, R., Boneh, D., and Tor, S. Evading censorship with browser-based proxies. *Proceedings of the 12th international conference on Privacy Enhancing Technologies*.

[23] Filler, T., and Fridrich, J. Fisher information determines capacity of ε-secure steganography. In *Information Hiding* (2009), Springer, pp. 31–47.

[24] Filler, T., and Fridrich, J. Gibbs Construction in Steganography. *IEEE Transactions on Information Forensics and Security 5*, 4 (Dec. 2010), 705–720.

[25] Filler, T., and Fridrich, J. Design of Adaptive Steganographic Schemes for Digital Images. *Proceedings of SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics XIII* (Feb. 2011).

[26] Filler, T., Judas, J., and Fridrich, J. Minimizing embedding impact in steganography using trellis-coded quantization. *IEEE Transactions on Information Forensics and Security* (Feb. 2011).

[27] Filler, T., Ker, A. D., and Fridrich, J. The Square Root Law of Steganographic Capacity for Markov Covers. In *Proceedings of SPIE, Electronic Imaging, Security and Forensics of Multimedia Contents XI* (2009).

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

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

[30] Fridrich, J., Kodovskỳ, J., Holub, V., and Goljan, M. Breaking hugo–the process discovery. In *Information Hiding* (2011), Springer, pp. 85–101.

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

[32] Goljan, M., Fridrich, J., and Holotyak, T. New blind steganalysis and its implications. In *Proceedings of SPIE, Electronic Imaging, Security and Forensics of Multimedia Contents* (2006).

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

[34] Greenstadt, R. Zebrafish: A steganographic system. *Massachussets Institute of Technology* (2002).

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

[36] Human Rights Watch. Letter from human rights watch to skype and skype's response. `http://www.hrw.org/node/11259/section/19`.

[37] Kelly, S., and Cook, S. Freedom on the net. *Freedom House* (2011).

[38] Kolari, P. Pings, spings, splogs and the splogosphere: 2007 updates.

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

[40] McLachlan, J., and Hopper, N. 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* (2009), ACM, pp. 31–40.

[41] Mimic, S. Homepage. `https://spammimic.com`.

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

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

[44] Netresec. Forensics of chinese mitm on github. `http://www.netresec.com/?page=Blog&month=2013-02&post=Forensics-of-Chinese-MITM-on-GitHub`.

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

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

[47] Pevny, T., Bas, P., and Fridrich, J. Steganalysis by subtractive pixel adjacency matrix. *information Forensics and Security, IEEE Transactions on 5*, 2 (2010), 215–224.

[48] Pevný, T., Filler, T., and Bas, P. Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. 161–177.

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

[50] rssCloud. Homepage. `http://rsscloud.org`.

[51] Ryabko, B. Y., and Ryabko, D. B. Asymptotically optimal perfect steganographic systems. *Problems of Information Transmission 45*, 2 (2009), 184–190.

[52] Sarkar, A., Solanki, K., and Manjunath, B. Further study on yass: Steganography based on randomized embedding to resist blind steganalysis. *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents* (2008).

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

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

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

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

[57] Tor. Iran blocks Tor; Tor releases same-day fix.

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

[59] 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`.

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

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

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

[63] Wang, Y., and Moulin, P. Perfectly secure steganography: Capacity, error exponents, and code constructions. *Information Theory, IEEE Transactions on 54*, 6 (2008), 2706–2722.

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