— Technical Report —

Torben: Deanonymizing Tor Communication using Web Page Markers

Daniel Arp, Fabian Yamaguchi, and Konrad Rieck
University of Göttingen, Germany

Report No. IFI-TB-2014-01

Technical Reports of the
Institute of Computer Science
University of Göttingen
ISSN 1611-1044

December 2014

University of Göttingen
Institute of Computer Science
Goldschmidtstr. 7
37077 Göttingen
Germany

Phone: +49 551 39–172000
Fax: +49 551 39–14403
E-Mail: office@cs.uni-goettingen.de
WWW: www.ifi.informatik.uni-goettingen.de

Torben: Deanonymizing Tor Communication using Web Page Markers

Daniel Arp, Fabian Yamaguchi, and Konrad Rieck
University of Göttingen
Göttingen, Germany

Abstract

The Tor network has established itself as de-facto standard for anonymous communication on the Internet, providing an increased level of privacy to over a million users worldwide. As a result, interest in the security of Tor is steadily growing, attracting researchers from academia as well as industry and even nation-state actors. While various attacks based on traffic analysis have been proposed, low accuracy and high false-positive rates in real-world settings still prohibit their application on a large scale. Instead, the few known cases of deanonymization have been reported to rely on vulnerabilities in browser implementations and cannot be considered weaknesses in Tor itself.

In this paper, we present Torben, a novel deanonymization attack against Tor. Our approach is considerably more reliable than existing traffic analysis attacks, simultaneously far less intrusive than browser exploits. The attack is based on an unfortunate interplay of technologies: (a) web pages can be easily manipulated to load content from untrusted origins and (b) despite encryption low-latency anonymization networks cannot effectively hide the size of request-response pairs. We demonstrate that an attacker can abuse this interplay to design a side channel in the communication of Tor, allowing short web page markers to be transmitted and exposing the web page a user visits over Tor. In an empirical evaluation with 60,000 web pages, our attack enables detecting these markers with an accuracy of over 91% and no false positives.

1 Introduction

The Tor network is one of the largest efforts to provide anonymity and privacy on the Internet. The network implements a low-latency anonymity service based on the concept of Onion Routing [7, 33]. The network consists of over 6,000 relay nodes worldwide that enable its users to relay communication through a circuit of these nodes, for example, to anonymously express their opinion or circumvent digital censorship. As of today, Tor is used by over a million users and, in comparison to related projects such as JAP [2, 17] and I2P [15, 19], can be considered the de-facto standard for anonymous communication on the Internet.

With the increasing use of Tor in practice, research on attacks against this service have gained considerable attention. A large body of work has studied *passive attacks* based on traffic analysis, most notably website fingerprinting and traffic confirmation attacks. The former enable an attacker to detect patterns indicative for web pages in Tor traffic [e.g., 3, 11, 25, 34]. Although these approaches provide good results in closed-world settings, in practice they suffer from high false-positive rates and unstable patterns due to changing web content. Traffic confirmation attacks on the other hand are not limited to web traffic but require an attacker who is able to eavesdrop both ends of a communication over a long period of time [e.g., 6, 21, 31, 38]. A second strain of research has thus considered *active attacks* against the Tor network, such as path-selection attacks based on network congestion [e.g., 9, 16, 24] and traffic watermarking using packet delays [e.g., 12, 13, 35, 37]. While these techniques provide a more accurate and faster deanonymization, they involve a significant effort for the adversary, requiring attacker control at different locations of the network. As a result, the few known cases of deanonymization of Tor have been reported to instead make use of advertisement networks or rely on vulnerabilities in browser implementations [29, 30] and are thus unrelated to insecurities of Tor in general.

In this paper, we present *Torben*, a novel deanonymization attack against Tor that is significantly more reliable than traffic analysis attacks but far less intrusive than browser exploits. In contrast to other active attacks, Torben operates entirely on the application layer and does not require Tor nodes or routers to be controlled by the adversary. The attack is based on an unfortunate interplay of technologies: First, web pages can often be manipulated to load dynamic content from untrusted origins, for example, using advertisements or user-provided content. Second, despite encryption, low-latency anonymization networks cannot effectively hide the size of request-response pairs in web traffic. We show that an attacker can abuse this interplay to design a side channel in the communication of Tor. This side channel enables the transmission of short *web page markers* that expose the web page a user visits to an observer between the Tor client and the entry node. Although it is well-known that active web content allows to track the visitors of web pages, we are the first to show that it can be used to deanonymize Tor users in a short period of time.

We demonstrate the efficacy of our attack in different experiments with real-world Tor traffic. We conduct a comparative closed-world evaluation with the top 100 web pages from the Alexa ranking, where Torben significantly outperforms website fingerprinting attacks [3, 11, 25]. In an open-world evaluation with 60.000 web pages, our attack enables identifying marked web pages with over 91% accuracy and no false positives, demonstrating the reliability of the proposed side channel. Finally, we conduct a live experiment with 4 users, each visiting web pages over Tor for roughly two hours. In this experiment, the Torben attack allows identifying 91% of the marked web pages under real-world conditions.

Our experimental results demonstrate the severe effect of the proposed deanonymization attack against Tor. While previous work on website fingerprinting already indicates a risk of exposing web pages through traffic patterns, our work finally shows that using an active attack, these patterns can be identified with high accuracy and very few false alarms. As a consequence, there is an urgent need for defenses in anonymization services protecting users from active attacks at the application layer.

In summary, our contributions are the following:

- We present a novel side channel in Tor communication. By issuing HTTP requests from the user's browser, an attacker is able to induce distinct patterns observable in encrypted traffic.
- We demonstrate that this side channel can be used to perform a novel deanonymization attack against Tor, allowing us to transmit *web page markers* exposing the page visited by a user.
- Finally, we show that these web page markers can be accurately detected in real Tor traffic with high accuracy by combining techniques from signal processing and machine learning.

The remainder of the paper is organized as follows: In Section 2 we provide background information on the Tor network and outline the attack scenario. We continue to describe the Torben attack in Section 3 and evaluate its efficacy in Section 4. Related work is discussed in Section 6 and Section 7 concludes the paper.

2 Background

Before presenting our deanonymization attack and discussing details of how to transmit data through the underlying side channel, we need to briefly review the basics of the Tor network (Section 2.1) and define the attack scenario we are considering (Section 2.2).

2.1 The Tor Network

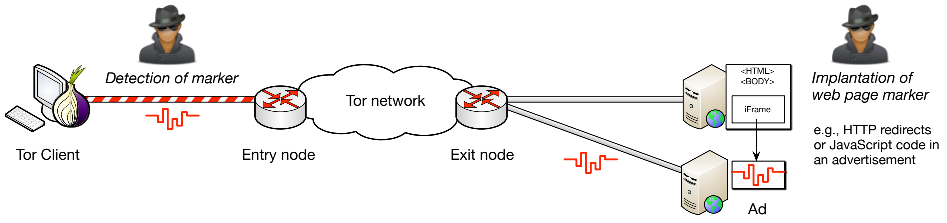


Figure 1: Attack scenario: A web pager marker is implanted using embedded or user-provided content, such as an advertisement or an image link. The marker induces a traffic pattern visible at the entry node, for example, using a chain of HTTP redirects or JavaScript code generating HTTP requests.

The Tor network [7] is a low-latency anonymization network whose purpose is to protect the privacy of its users by obfuscating their network traffic. This is done by tunneling user traffic through arbitrary paths in the Tor network which consist of multiple hops (Tor relays) that run the Tor software and are operated by volunteers. The security of Tor is based on the use of strong encryption and the large number of relays that can be used to establish a path, thus significantly lowering the ability of an attacker to easily eavesdrop a communication or link sender and receiver.

A user who wants to establish a connection to a server through Tor has to run a Tor client on his computer which will first select a path through the Tor network. A path usually consists of three Tor nodes: An entry node, a middle node and an exit node. The client creates a circuit by incrementally negotiating symmetric session keys with each of the selected Tor nodes on the path. After establishing a circuit, the user can send data over the Tor network using fixed size Tor cells which are multi-layer encrypted with the previously negotiated session keys. Each relay node on the path then removes one layer of encryption while the cell is forwarded to its destination. A circuit can transport multiple TCP streams, while multiple circuits can be multiplexed over a connection between two Tor relays.

The Tor nodes communicate with each other using TLS which ensures that the cells always arrive in the correct order they were sent. Furthermore, each relay node only knows its successor and predecessor but not the complete path. Therefore, Tor still provides anonymity even if a local attacker controls one node in the circuit, since she is still unable to link both parties of a communication. Moreover, each circuit is only used for 10 minutes until a new circuit is created, thereby limiting the ability of an adversary to discover and monitor a circuit over a longer time frame.

2.2 Attack Scenario

For our attack, we consider a scenario that involves an active attacker. We assume that, first, this attacker is able to monitor the encrypted communication between a Tor client and the entry node, and, second, she is able to embed markers in a web page of interest. An overview of this attack scenario is depicted in Figure 1.

To illustrate this scenario further, let us consider a totalitarian regime or law enforcement agency that wants to determine whether a particular user visits a certain web page, despite the fact that Tor is being used to anonymize the communication. Clearly, this attacker can be expected to be capable of observing the encrypted network communication between the user's browser and the Tor entry node. However, this alone is a vast underestimation of her capabilities, as it considers a passive attacker. It is reasonable to assume that the user's browser may be exposed to attacker-provided web content at some point throughout the browsing session. This content may be delivered through a multitude of vectors. Based on the chosen vector, we consider the following two variants of the attack scenario:

- **Remote markers.** In this scenario, the attacker exploits the fact that web pages often embed content from different origins, some of which might be controlled by the attacker. For example, the attacker may be able to host an advertisement on the web page via an advertisement network or reference an image located at a third-party server. Furthermore, message boards and social networks provide many opportunities to remotely embed markers into a web page.

- **Local markers**. In this scenario, the attacker is able to locally inject content into a web page. For example, an intelligence agency may manipulate the content of a web page directly at the server or modify outgoing data to implant the marker. Similarly, a more subtle attacker may exploit vulnerabilities in the code of a web page to add a local marker, for instance, using stored cross-site scripting.

Regardless of the type of the web page marker, attacker-provided content is loaded in the user's browser and can be used to induce a traffic pattern in the Tor communication, for example, through JavaScript code generating HTTP requests or a chain of HTTP redirects. This pattern can be detected in the encrypted traffic between the Tor client and the entry node, ultimately enabling an adversary to deanonymize the visitors of marked web pages.

3 A Side-Channel Attack on Tor

A fundamental limitation of low-latency anonymization networks is that they cannot effectively hide the sizes and order of relayed packets. For users browsing web pages via Tor, this means that HTTP request and response sizes directly influence the stream of TLS records observed between the Tor client and the entry node. Unfortunately, this setting can be exploited by an attacker. If the user accesses attacker-controlled content, such as JavaScript code or an embedded iFrame, the attacker gains partial control over the stream of request and response sizes.

The overall idea of our attack is to leverage this control to carry out a side-channel attack by creating distinct communication patterns in the encrypted data stream that can be effectively detected using machine learning techniques. While this idea is simple at core, applying it to construct a successful attack requires careful engineering of a number of different components. In particular, the following four challenges need to be addressed:

- **Preprocessing of network traces.** Network traces need to be preprocessed and transformed into a robust representation suitable for analysis of request and response sizes (Section 3.1).

- **Side channel design.** A reliable side channel needs to be designed that allows short messages to be transmitted to an attacker observing the encrypted data stream of Tor (Section 3.2).

- **Transmission of web page markers.** Markers need to be transmitted using the side channel, such that the visit of a marked web page induces a distinct pattern in the encrypted traffic (Section 3.3).

- **Detection of web page markers.** Finally, a method for automatic detection of these web page markers is required that enables identifying individual markers in real network traces (Section 3.4).

In the following sections, we discuss in detail how we addressed each of these challenges and provide background information on involved protocols as well as empirical evidence that support our design decisions where required.

3.1 Preprocessing of Network Traces

The success of traffic analysis attacks critically depends on the choice of a suitable representation of observed network communication. With this goal in mind, we preprocess network traces by leveraging inherent properties of the Tor protocol and the protocols it depends on, thus allowing us to remove noise sources and highlight those properties of network traffic that are controllable by the attacker. The key insight our preprocessing scheme is based on, is that by controlling the size of HTTP requests and responses, we do not gain control over the size of IP packets or TLS records but only over the *amount* of data transferred from one change of direction to the next. We devise a representation emphasizing this aspect of the record stream in a two-step procedure outlined in the following.

3.1.1 TCP Stream Reassembly

All Tor communication takes place via Tor cells encapsulated in TLS records (see Section 2). Regardless of this, the vast majority of known traffic analysis attacks on Tor operate directly on raw sequences of IP packets. While HTTP communication relayed by the Tor network may influence these sequences, they are distorted by several noise sources that needlessly complicate analysis. In particular, delayed or dropped IP packets cause the transport layer to issue re-transmissions of packets. Moreover, packets may be re-ordered making it hard to associate requests with responses. Finally, several Tor connections are usually operated in parallel which appear interleaved at the IP layer.

Fortunately, we can easily address all of these problems by reassembling TCP streams using readily available tools such as *tshark*. This allows all subsequent analysis to be carried out on streams of TLS records as opposed to raw IP packets. In effect, independent Tor connections are indeed processed independently, the order of Tor cells is preserved, and finally, artifacts of TCP/IP such as re-transmissions and acknowledgments carrying no data are removed. As a result of this step, we obtain a trace of incoming and outgoing TLS records for each Tor connection. To simplify all further processing, we map traces to sequences of record sizes where a positive and negative sign are used for incoming and outgoing traffic respectively.

3.1.2 Filtering and Merging TLS Records

We proceed to apply the following chain of transformations to account for various properties of Tor and TLS that impact the analysis of network traffic.

a) *Filtering empty records.* The observed sequence of TLS records depends on the version of the OpenSSL library used by the entry node and the proxy node. In particular, recent versions of OpenSSL introduce a mitigation strategy for the BEAST attack that transmits empty records at regular intervals in all TLS streams. As a first step, we therefore filter sequences such that they only contain entries representing records of 100 bytes or more. This preserves Tor cells as the minimum cell size is 512 bytes.

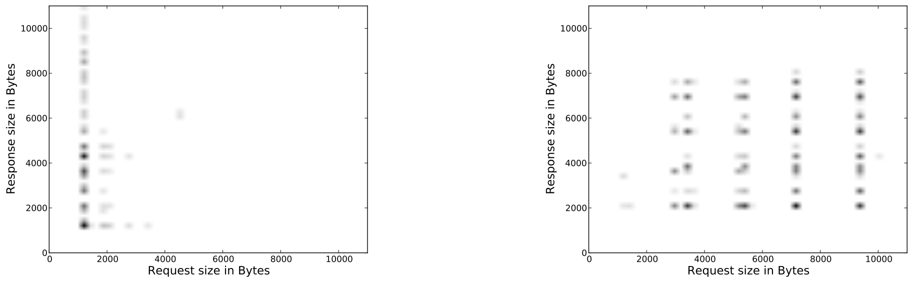
b) *Merging of records.* Unfortunately, we found that the same HTTP request can produce several different sequences of TLS record sizes, and thus analyzing sequences of record sizes is bound to fail. The reason for this behavior is that Tor cells are not mapped to TLS records one by one but instead, several Tor cells may be merged into a single record. This process depends entirely on the interplay of buffer flushing inside nodes on the Tor communication path and is hard to control by attackers. Instead of striving to model this chaotic behavior, we accept that we simply do not have control over the sizes of TLS records or IP packets but only over the amount of data transferred before each direction change. We therefore merge adjacent TLS records going into the same direction to obtain a sequence representing the amount of data rather than individual records.

c) *Filtering control cells.* While the vast majority of Tor cells relay user data, several cells exist that are unrelated to user data. In particular, Tor implements a flow-control mechanism that leads to an exchange of single Tor cells at regular intervals. To filter these cells, we discard any sizes smaller than twice the cell size after merging. The rational behind this step is that neither HTTP requests nor responses typically fit into a single cell. Upon removal of these single cells, we merge TLS records again to connect data relay cells previously separated by control cells.

d) *Normalization of sizes.* The concrete sizes of transfered data vary slightly depending on the version of Tor and its libraries. As a simple normalization, we express all sizes in multiples of 2000 bytes.

Upon completion of the preprocessing step, each Tor connection is represented by a sequence of integers that encodes the amount, direction and order of data transfers. We refer to these sequences as *data transfer sequences* throughout the rest of this paper.

3.2 Side Channel Design



(a) Request-response pairs of 10,000 web pages.

(b) Request-response pairs of our side channel.

Figure 2: Distribution of request and response sizes in 10,000 web pages and our side channel.

The previous section presents a preprocessing scheme that can be applied to network traces to highlight data transfers in Tor communication. However, it is still unclear whether attackers can reliably encode messages in the order, amount and direction of transfers and hence, whether we can carry out a corresponding side-channel attack. In particular, the side channel should have the following two properties: First, normal web traffic should be clearly distinguishable from any side-channel communication to make false positives very unlikely. Second, the transmission speed needs to be high enough to allow short byte sequences to be transmitted before the user leaves a web page.

Side-channel communication can only be distinguished from normal web traffic, if data transfer patterns exist that are atypical for normal web pages. To determine whether such patterns exist, we record traces of typical web page loading and analyze request-response pairs. Figure 2a shows the distribution of request-response pair sizes for 10,000 web pages from the Alexa ranking. The plot indicates that the vast majority of requests are rather short, requiring only about a thousand bytes. Hence, the range of request sizes between $2,000$ and $8,000$ bytes seems suitable to establish a side channel, as these sizes are large enough to be atypical but still enable a fast transmission. We thus propose to encode and transmit messages as follows:

A message is first padded to a multiple of four bits and then split into quadbits. Each quadbit (q_1, q_2, q_3, q_4) is then mapped to a request-response pair (r_1, r_2), where r_1 is the request size and r_2 the response size. To this end, we define an auxiliary function f that maps two bits q_i and q_j to a size value as follows

$$f(q_i, q_j) = (q_i + 2q_j) \cdot s + c,$$

where s controls the spacing between different sizes and c is an offset. To discriminate requests and responses, we mark the size of requests with a negative sign. Consequently, we obtain eight different sizes corresponding to the alphabet $\mathcal{A} = \{-f(1,1), \dots, -f(0,0), f(0,0), \dots, f(1,1)\}$. In accordance with Figure 2a, we set $s = 2000$ and $c = 2000$ for our side channel. Using this alphabet, we can then map a quadbit to a request-response pair with the following function

$$\begin{aligned} m : \{0,1\}^4 &\longrightarrow \mathcal{A} \times \mathcal{A}, \\ m(q_1, q_2, q_3, q_4) &\longmapsto -f(q_1, q_2), f(q_3, q_4). \end{aligned}$$

By computing the function m for all quadbits of a message, concatenating the resulting sizes and marking requests with a negative sign, we finally obtain a data transfer sequence suitable for transmission over the side channel.

Figure 2b shows the distribution of request-response pairs for 50 random messages encoded using our scheme and transmitted $1,000$ times each. The 4×4 grid induced by our alphabet is clearly visible in the plot and placed at the desired offset in the size space.

3.3 Transmission of Web Page Markers

Equipped with a side channel, we can now expose visited web pages by transmitting suitable messages from the user's browser. This, however, creates two additional challenges. First, a suitable browser-based mechanism for transmission of correctly ordered sequences of HTTP requests needs to be found, and second, *web page markers* that encode the names or URLs of visited web pages need to be constructed.

3.3.1 Issuing HTTP Requests

As detailed in Section 2.2, we assume that an attacker is able to execute JavaScript code within the browser of a Tor user. This code can be embedded in a displayed advertisement, injected via cross-site scripting or contained in any other included JavaScript code. For establishing the side channel, the code does not need to operate in the context of the marked web page and thus our attack is not effected by the same-origin policy.

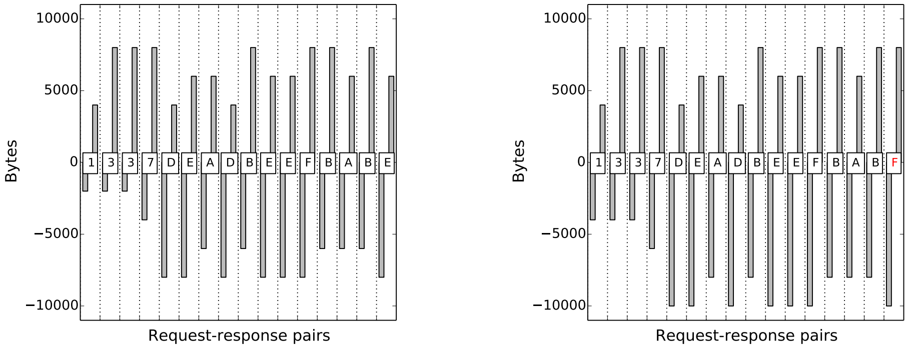
The standard JavaScript object `XMLHttpRequest` offers a mechanism for request transmission that fits our needs perfectly. The object allows requests to be issued from the user's browser synchronously while offering fine-grained control over request content and headers. Moreover, it is available in all modern browsers and can be considered a core browser feature. We can thus employ `XMLHttpRequest` to transmit a request-response pair (r_1, r_2) using the following URL

```
http://server.com/res?str
```

where `res` is simply a resource of size r_1, such as an image or a document, and `str` a random string of length r_2 attached to the URL. We choose a random string here to reduce caching effects induced by some browsers. Note that the contacted server can be any server offering resources with the required sizes. Similar in spirit to *gadgets* of binary code used in return-oriented programming, it is sufficient to find a server that offers resources with four different sizes to successfully carry out the attack. Consequently, the server does not need to be controlled by the attacker and hence provides no information about the attacker's origin.

As an illustrative example, we encode and transmit the 16 symbols long message "`1337 dead beef babe`" over the presented side channel. Figure 3a shows the message encoded as a data transfer sequence before transmission, while Figure 3b shows the corresponding sequence observed in encrypted Tor traffic. Apart from a constant scaling factor introduced by protocol overhead, the two sequences are almost identical. In particular, the order of request-response pairs is preserved and all but the last request-response pair can be perfectly reconstructed from the observed traffic.

Finally, we need to note that our attack is not limited to JavaScript code. In fact, in any other mechanism for automatically issuing requests from the user's browser can be used to establish the side channel. For example, it is possible with only little adaption to transfer request-response pairs using a chain of HTML redirects.



(a) Original message.

(b) Observed data transfer sequence.

Figure 3: Example of the side channel: (a) original 8-byte message encoded as a data transfer sequence; (b) observed data transfer sequence in Tor traffic (Negative numbers indicate request sizes, positive number response sizes).

3.3.2 Web Page Markers

We are now able to transmit short messages over the side channel to expose web pages visited using Tor. If the attacker only wants to monitor a single web page, choosing a constant message is sufficient. However, to monitor multiple web pages or track how a user browses from one page to another, the web page markers need to be designed such that they have a large pairwise Hamming distance, thereby minimizing the probability for confusions.

Ideally, the attacker employs an error-correcting code to generate messages with a high Hamming distance. For example, the Walsh-Hadamard code can be used to encode messages of length $k \leq 7$ as code words of length 2^k with maximized minimum Hamming distances [1]. A practical drawback of this approach is that no natural mapping between web pages and markers exists and thus, the web pages to be monitored and their corresponding markers need to be fixed in advance.

As a compromise, we use the 20 byte SHA-1 hash value of the monitored URL as a web page marker. This ensures that there is a natural mapping between URLs and their web page markers. Moreover, it is a design criterion for cryptographic hashes to generate statistically independent and equally distributed bits, and we can expect web page markers to be equally distributed in the message space, thereby having a rather low probability of confusions.

3.4 Detection of Web Page Markers

The presented side channel allows web page markers to be transmitted and recognized by human observers, yet due to the vast amount of network traffic to analyze, our attack only becomes practical if the markers can be detected automatically. In this section, we present an automated approach for the detection of web page markers in monitored traffic that combines techniques from information retrieval and machine learning.

3.4.1 Positional N-grams

To identify markers in a monitored data transfer sequence, we use a sliding window and classify the sequence under each window independently. This highlights an advantage of our attack over related traffic analysis attacks. Since the marker is transmitted in a small time frame, it is perfectly valid to limit the analysis to small sequences of the network trace, in our experiments 100 symbols, thus significantly reducing the computational resources required for training a classifier and detecting web page markers.

Several learning algorithms seem suitable for detection of markers, yet most of these algorithms are defined for vectors and cannot be directly applied to sequential data. To address this issue, we construct a map from sequences to a vector space that encodes information about the contained symbols and their positions. In particular, we introduce *positional n-grams*—an extensions of classic n-grams [5, 28]—specifically adapted for detecting web page markers in Tor traffic.

Let $S = \mathcal{A}^n$ denote the set of all n-grams, that is, all possible sequences of n symbols from our alphabet $\mathcal{A}$. We then define a positional n-gram p to be a pair (s, i) where s is an n-gram and $i \in \mathbb{N}$ indicates the position of s in a sequence. We denote the set of all positional n-grams by $P = S \times \mathbb{N}$. To compensate noise in the network traffic, we do not consider a single position i, but instead focus on a range of positions between i and $i + \tau$, where τ is a tolerance parameter. Based on these definitions, we can define a map ϕ from a sequence x to a vector space $\mathbb{R}^{|P|}$ by

$$\begin{aligned} \phi : \mathcal{A}^* &\longrightarrow \mathbb{R}^{|P|}, \\ \phi(x) &\longmapsto (\#(x, p))_{p \in P} \end{aligned}$$

where $\#(x, p)$ returns the number of occurrences of the positional n-gram $p = (s, i)$ between the positions i and $i + \tau$ in the sequence x. The vector $\phi(x)$ encodes information about the symbols, their order and their position in x—thus reflecting the basic properties of a web page marker.

3.4.2 Probabilistic Classification

Using the function ϕ we are finally ready to apply a machine learning classifier for detection and discrimination of different web page markers. In particular, we employ a multi-class Support Vector Machine (SVM) with probabilistic outputs [14, 26] for this task. The SVM is trained on the sequences of the individual markers—independently from the particular target web pages. Given an unknown sequence, the SVM can then be used to identify the most likely web page marker present in this sequence. Due to the probabilistic output, we are able to quantify the confidence of this decision, which enables us to determine sequences that do not correspond to any of the known markers.

At first sight, the high dimensionality of the vector space induced by positional n-grams may seem prohibitive for an efficient classification using an SVM. Fortunately, this is not problematic in practice for two reasons. First, vectors are only sparsely populated, allowing storage in space efficient data structures. Second, if $\hat{S}$ is the set of n-grams actually observed, the cardinality of the corresponding set of positional n-grams $\hat{P}$ is $|\hat{S}| \cdot (l + \tau)$, where l is length of the windows. As a result, the training as well as the application of the SVM can be carried out within a couple of minutes on several thousands of data transfer sequences.

4 Evaluation

We evaluate the Torben attack in a series of experiments that allows us to assess its effectiveness in different settings. After presenting our experimental setup (Section 4.1 & 4.2), we first evaluate the Torben attack in a closed-world setting (Section 4.3), where users can only visit web pages from a fixed set. While this setting is rather unrealistic, it has been extensively studied in the website fingerprinting literature and thus enables us to compare our approach to prior work. We proceed to consider an open-world setting, where users can freely choose web pages from a potentially infinite number of web pages—60,000 in our experiment—(Section 4.4). The open-world setting allows us to obtain a good approximation for the false-positive rate of our attack. Finally, we perform a live experiment where we evaluate the ability of Torben to identify web page markers in real-world traffic generated by multiple users (Section 4.5).

4.1 Data Collection

To automatically visit a large number of web pages over the Tor network in an acceptable time, we use the *Selenium WebDriver* (version 2.38.3), a browsing automation plug-in for Mozilla Firefox. The plug-in is installed along side the Tor browser bundle (version 3.5), the standard distribution of Tor. This ensures that the browser configuration used in our experiments almost exactly matches the configuration employed by most Tor users.

For our experiments we automatically visit different sets of the top one million web pages from the Alexa ranking (retrieved in February 2014). For the vast majority of web pages, the *Selenium WebDriver* can automatically determine when the page is fully loaded based on loading of the page icon. In rare cases, successful page load is not detected within 3 minutes. In these cases we discard this web page and use the next page in the Alexa ranking instead. Moreover, we remove variants of very similar pages. For example, we consider only `google.com` and not `google.de` or `google.fr`. This is necessary to obtain a fair comparison with website fingerprinting approaches as these approaches are known to fail if web pages are too similar.

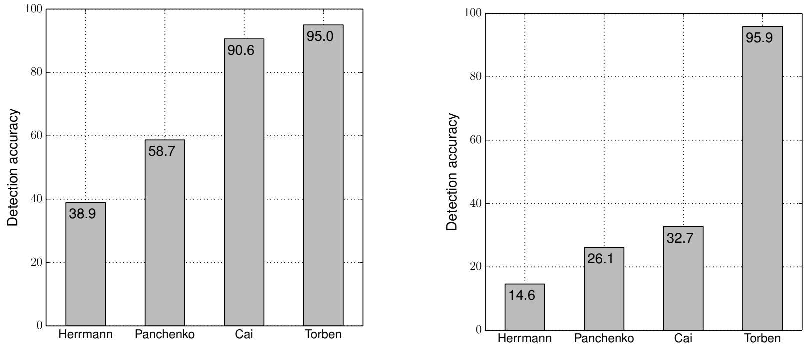
4.2 Detection Setup

To perform the Torben attack in practice, we implement the preprocessing, transmission and detection steps outlined in Section 3. For extracting positional *n*-grams from network traces, we use the tool *Sally* [27] (version 0.8.3) and for learning the probabilistic classification the library *LIBSVM* [4]. All experiments are conducted on an Intel i7 2,66 GHz CPU with 8 GB of RAM, except for the website fingerprinting attack by Cai et al. [3]. The latter attack requires the computation of a large kernel matrix and is conducted on four AMD Opteron 6378 CPUs with 64 cores and 256 GB RAM each.

For training the detection method described in Section 3.4, we generate 100 web page markers using the SHA-1 hashes of the top Alexa web pages and record 50 transmissions for each of these markers over Tor. Note that only the web page markers and not the actual web pages are recorded for training our detection method. The resulting network traces are then used to train the multi-class SVM, where we perform a model selection via 10-fold cross validation and fix the n-gram length to $n = 3$, the tolerance to $\tau = 9$ and the SVM regularization to $C = 0.1$. The resulting detector is used unmodified in all of the following experiments.

4.3 Closed-World Evaluation

In the first experiment, we consider a closed-world setting, where the user is only able to visit a limited set of web pages. In this scenario, the attacker is able to use a classifier that only discriminates between the web pages of this set. Although it is unlikely that this scenario applies to many situations in practice, it already shows whether the web page markers are distinct enough from each other to allow for a reliable detection. Moreover, the experiment enables us to compare the effectiveness of our approach with that of related website fingerprinting attacks.

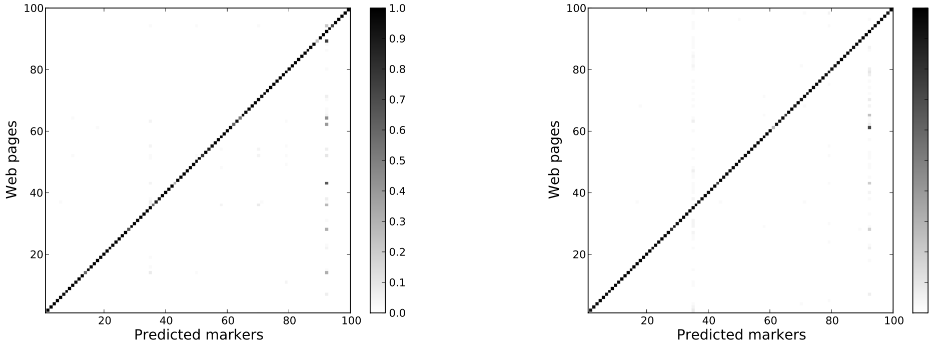


(a) February 2014

(b) April 2014

Figure 4: Deanonymization performance of Torben and website fingerprinting attacks. The performance is evaluated on the top 100 Alexa web pages visited in February and April 2014.

We visit each of the top 100 Alexa web pages 50 times, first in February 2014 and a second time in April 2014 resulting in two data sets. To simulate the attack scenario of a remote marker, we use a reverse proxy and inject a small JavaScript snippet into each page that opens a separate browser window containing the marker, similar to an advertisement. The marker is transmitted after a delay of 30 seconds. As 10 of the 100 web pages load very slowly over Tor and largely overlap with the marker, we increase the delay to 120 seconds in these cases.



(a) February 2014

(b) April 2014

Figure 5: Confusion matrices for the Torben attack. The confusion is determined on the top 100 Alexa web pages visited in February and April 2014.

The transmission time of the web page markers ranges from 12 to 20 seconds with a mean of 18 seconds. On average, 300 packets with a total of 390,000 bytes are necessary for sending the complete marker to the user. As the transmission happens in the background, these slight differences to the original web page traffic are hardly noticeable, in particular because communication over Tor often suffers from rather long loading times.

To enable a comparison with related website fingerprinting attacks, we implement the approaches by Herrmann et al. [11], Panchenko et al. [25] and Cai et al. [3], where the first is slightly modified to use an SVM instead of an MNB classifier. For all approaches, we conduct the same experimental procedure as for the Torben attack, except that we do not implant markers into the web pages. Furthermore, our detection method is trained solely on web page markers, whereas the fingerprinting attacks are trained on a sample of the web pages visited in February 2014.

Figure 4a shows the performance of Torben and the website fingerprinting attacks on the web pages visited in February 2014. Our approach is able to correctly deanonymize 95% of the web page visits, whereas the performance of the other attacks ranges from 38.9% to 90.6%. However, all website fingerprinting attacks suffer from changes in web content. When applied to the same web pages visited in April 2014, none of the passive approaches is able correctly identify more than a third of the web pages, as shown in Figure 4b. By contrast, the Torben attacks attains a similar performance as in February and significantly outperforms the website fingerprinting attacks, as the learned markers do not change over time.

We need to note here that the attack models underlying website fingerprinting and Torben differ: While website fingerprinting generally assumes a passive adversary that only monitors communication, Torben builds on an active attacker that is capable of embedding markers. As detailed in Section 2.2, however, this embedding often requires only little effort and thus is a realistic threat.

Moreover, we observe in Figure 5 that confusions only happen between a handful of web pages. This is likely an artifact of the probabilistic classification employed in our attack. If a web page marker is incorrectly identified, the classification tends to favor particular markers, which in our setting correspond to the numbers 92 and 37. Nonetheless, these confusions are rare and do not severely impact the deanonymization performance of our attack.

Overall, this experiment demonstrates that the proposed side-channel attack can deanonymize web page visits with high accuracy in a closed-world setting. Due to the active injection of markers, our approach outperforms passive attacks that are unable to compensate changes in the content and resulting traffic patterns of web pages.

4.4 Open-World Evaluation

In this experiment, we consider an open-world setting, where the user can freely visit web pages from a large unknown set and only few of these pages are tagged with a web page marker. The adversary is thus interested in determining whether marked web pages have been visited by a particular user.

We extend the previous experiment by additionally choosing 60,000 web pages at random from the Alexa top one million ranking, none of which are part of the top 100 web pages. Each of these web pages is then visited once over Tor without sending a marker. This large set of web pages enables us to estimate the false-positive rate of the Torben attack. Due to the usage of an SVM with probabilistic output, we can simply identify data transfer sequences that do not contain a web page marker by setting a threshold on the determined probabilities.

Figure 6 shows a ROC curve for the detection performance of Torben in this experiment. Note that the false-positive rate on the x-axis is given in the range 0 to 0.001, while the detection rate is shown on the y-axis between 0.6 and 1.0. The attack enables detecting 91% of the 100 web pages with no false positives in a set of 60,000 additional web pages. The detection rate then increases only slowly which indicates that the markers got corrupted. However, in most cases the classifier is able to compensate the noise that is probably introduced by delays in the network or additional network traffic.

The outcome of the open-world evaluation demonstrates the reliability of the web page markers, which are unlikely to be confused with regular web page traffic and enable detecting marked pages with a very low false-positive rate. This reliability rests on the design of the side channel that makes use of atypical request-response pairs for transmitting information (Section 3.2).

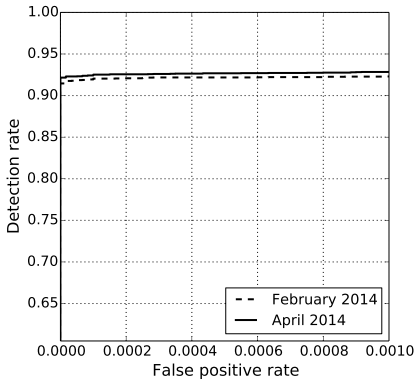


Figure 6: Detection performance of the Torben attack on 100 marked and 60,000 unmarked Alexa web pages in February and April 2014.

4.5 Live Evaluation

The previous experiments already show the effectiveness of our approach on a large amount of data that has been recorded automatically. However, in order to fortify the obtained results, we conduct a further experiment in which we analyze web traffic of real users.

In this experiment, four different users surf through the web for roughly two hours using Tor. Each user visits arbitrary web pages and from time to time, randomly chosen marked web pages. We simulate the attack scenario of a remote marker as described in Section 4.3 and set the transmission delay for markers to 120 seconds.

Afterwards, the recorded traffic of each user is split into chunks of three minutes that are separately analyzed using a sliding window as described in Section 3.4. Hence, the classifier outputs a label and a probability score for each chunk. If the probability score of a chunk is below a particular threshold, we consider that the user did not visit any marked web pages during this time frame. We select this threshold to be $t = 0.1$ based on several test runs which we conducted in advance.

From a total of 34 visited marked web pages, we are able to detect and classify 31 of them correctly. Furthermore, the classifier does not output any false positives. As an example, Figure 7 shows the probability scores which we obtained for a particular user. The tick marks on the x-axis denote time frames in which the user has visited a marked web page. Note, that although he classifier missed one marked web page, the probability score for this page is still significantly higher than for non-marked web pages.

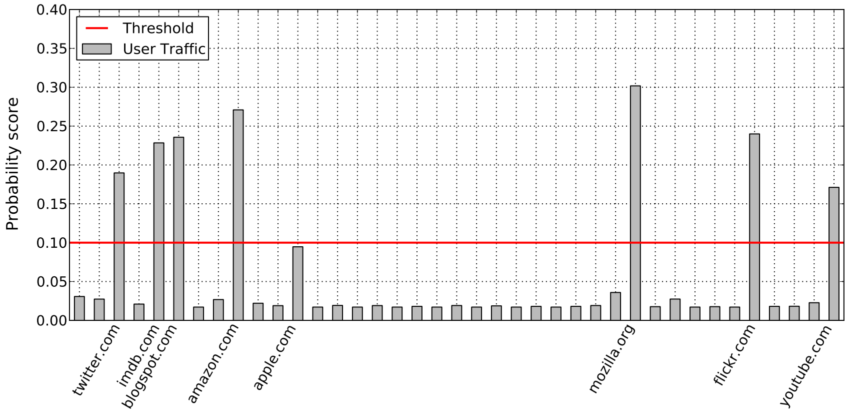


Figure 7: Deanonymization results of a particular user recorded within a time frame of 1 hour and 59 minutes.

5 Limitations and Defenses

Our experiments show that Torben is a very reliable deanonymization technique against Tor. The attack is based on a fundamental property of low-latency anonymization networks—the inability to conceal the size of data transfers—which makes crafting defenses against Torben a difficult task. In this section, we explore the limitations of our approach and provide starting points for the development of effective defenses.

First of all, our attack makes no attempts of hiding web page markers. In consequence, developing a detection mechanism that identifies and suppresses communication on our side channel may be an effective method to protect users. Such mechanisms could be implemented either on the client side directly in the form of a browser plug-in or even by Tor nodes to protect all clients at once. This, however, may lead to an arms race between the attackers and defenders in which the attackers may frequently change parameters of the attack, such as request/response sizes, requiring a regular update of the detection mechanism.

Second, our evaluation shows that web page markers may be corrupted if they largely overlap with the loading of a web page. This suggests that, although our classification model is able to compensate some traffic interference using positional features, it fails if the interference becomes too severe. A second option would thus be to introduce chaff traffic to disturb the side-channel communication. Yet, this option may not be satisfactory in practice, as it introduces additional traffic and might lower Tor's overall performance. Whether such chaff traffic can be selectively injected to only destroy indicative traffic patterns is an interesting question for further research.

Finally, our specific attack variant is based on JavaScript and obviously disabling JavaScript is a possible defense. However, we would like to stress that sequences of HTTP requests can be generated using many other mechanisms. In particular, it is possible to create a chain of HTML redirects to perform side-channel communication, thus allowing the attack to be performed without any active web content. Moreover, it should be noted that JavaScript is enabled by default in all modern web browsers as well as the Tor browser bundle, simply because many popular web pages will not work without JavaScript.

6 Related Work

Attacks against anonymization techniques have been a vivid area of research in the last years. In this section we thus discuss approaches related to our side-channel attack, starting from early deanonymization attacks on encrypted communication (Section 6.1) and reaching over to passive and active attacks against low-latency anonymization networks (Section 6.2 & 6.3).

6.1 Attacks on Encrypted Communication

Several researchers have recognized that while it may not be possible to decipher encrypted messages, when transmitted over a communication link, important characteristics of the data such as its size and partitioning into data units is often disclosed. For example, multiple approaches exist to identify web pages in encrypted HTTP traffic by observing characteristic packet lengths [22, 32]. To prevent these attacks, it is often possible to introduce artificial changes to packet sizes and timing intervals, thereby obfuscating message characteristics. This is for instance done by the traffic transformation tool HTTPOS [23].

Unfortunately, these countermeasures not only introduce traffic overhead but are also unable to hide the overall size of data transfers [8] over the network, a problem that we exploit in the Torben attack.

6.2 Passive Attacks against Tor

Herrmann et al. [11] present one of the first passive attacks on Tor. Similar to Liberatore et al. [22] they consider IP packet lengths and apply an MNB classifier to identify web pages by their traffic patterns. Although their attack works well on encrypted web traffic, it only achieves a low accuracy on Tor traffic. A higher accuracy is achieved by Panchenko et al. [25] by considering additional features, such as the amount of data sent before each direction change, thus allowing characteristic traffic bursts to be identified. Panchenko et al. obtain a detection performance of up to 73% in an open-world setting on 5,000 randomly chosen web pages from the Alexa top million. Continuing this line of research, Cai et al. [3] consider the ordering of packets while taking displacements into account using the Damerau-Levensthein distance. The authors evaluate their approach in a closed-world setting and achieve an accuracy of 83% on 800 different web pages. Wang et al. [34] are able to reach an even higher detection result using similar features over TLS records.

All of these passive approaches, however, suffer from false-positive rates that are too high to enable a practical application [20]. Moreover, our experiment demonstrate that website fingerprinting suffers from changes in web content and degrades in deanonymization performance over time.

Several methods have been proposed to counter website fingerprinting attacks [8, 10, 36]. Unfortunately, all of these methods introduce traffic overhead or timing delays which can hardly be tolerated in practice. For example, Wright et al. [36] propose a traffic morphing technique which morphs the packet size distribution of a given web page to the distribution of another web page. To this end, they solve a convex optimization problem in order to derive a morphing matrix that minimizes the number of bytes of overhead. Regardless of this, the bandwidth overhead would still significantly lower Tors bandwidth which could hinder the further adoption of Tor by users.

While website fingerprinting attacks consider an attacker only between the client and its entry node, traffic confirmation attacks require an attacker who is able to correlate the traffic on both ends of a communication path. Several authors demonstrate the effectiveness of such attacks [e.g., 6, 21, 31, 38]. However, this scenario requires the attacker to have access to both ends of the connection over a long period of time.

6.3 Active attacks against Tor

Passive traffic analysis attacks require traffic to be observed for a longer period of time before users can be effectively deanonymized [6, 18]. Approaches where attackers actively attack the communication path to lower the required amount of time have therefore been proposed.

For example, several authors propose attacks on the Tor network itself that attempt to reveal the complete communication path and thus link communication parties [9, 16, 24].

Similar to our approach, Murdoch et al. [24] correlate a specific traffic scheme sent from a corrupted server with probe traffic observed on a Tor node in order to determine whether this particular node is part of an observed connection. However, since the number of nodes in the Tor network has significantly increased in recent years, this attack has become unpractical.

Instead of trying to reveal the complete communication path, it is also possible for an attacker to inject specific patterns at one end of a communication that can be observed at the other end to link both communication parties. Several of these watermarking schemes have been proposed for mixed networks which use inter-packet delays to encode a watermarking sequence [12, 13, 35, 37]. A slightly different approach is presented by Yu et al. [37] who inject watermarks into the traffic using spread spectrum techniques. In this case, a pseudo-noise code is used to spread the signal over a large spectrum at the server-side. The same code is then used at the client-side to de-spread the signal and thus identify the traffic.

In contrast to Torben, these attacks do not take place at the application layer and are thus much more difficult to realize in practice. Furthermore, they consider a stronger attacker who is able to control the exit node or the link between the exit node and the server.

7 Conclusion

Tor is among the largest and best understood anonymization networks operated to date, protecting the privacy of over a million users worldwide. This paper presents a novel deanonymization attack on Tor that exploits a fundamental weaknesses of low-latency anonymization networks. In particular, we show that an attacker capable of providing web content to users, e.g., through banner advertisements or cross-site scripting, is able to deanonymize users via a side-channel attack. By transmitting web page markers through this side channel, the attacker can expose the web pages a user visits within a couple of seconds. This attack is considerably more effective than known website fingerprinting attacks and far less intrusive than browser exploits used in the wild. Fortunately, the side-channel communication is clearly visible in network traces and hence it may be possible to implement detection approaches as a first countermeasure against this attack.

Acknowledgments

The authors would like to thank Xiang Cai for making the implementation of his website fingerprinting attack publicly available.

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge Press, 2009.

[2] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: a system for anonymous and unobservable internet access. In *Proc. of International Workshop on Design Issues in Anonymity and Unobservability*, 2001.

[3] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2012.

[4] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. 2011.

[5] M. Damashek. Gauging similarity with *n*-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[6] G. Danezis. The traffic analysis of continuous-time mixes. In *Proc. of Privacy Enhancing Technologies Symposium (PETS)*, 2005.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. of USENIX Security Symposium*, 2004.

[8] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proc. of IEEE Symposium on Security and Privacy*, 2012.

[9] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on Tor using long paths. In *Proc. of USENIX Security Symposium*, 2009.

[10] X. Fu, B. Graham, D. Xuan, R. Bettati, and W. Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proc. of International Conference on Parallel Processing (ICPP)*, 2003.

[11] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier. In *Proc. of ACM Workshop on Cloud Computing Security*, 2009.

[12] A. Houmansadr and N. Borisov. Swirl: A scalable watermark to detect correlated network flows. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2011.

[13] A. Houmansadr, N. Kiyavash, and N. Borisov. Rainbow: A robust and invisible non-blind watermark for network flows. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.

[14] T.-K. Huang, R. C. Weng, and C.-J. Lin. Generalized Bradley-Terry models and multi-class probability estimates. *The Journal of Machine Learning Research*, 7:85–115, 2006.

[15] The Invisible Internet Project. `http://geti2p.net/`, visited May 2014.

[16] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann. The sniper attack: Anonymously deanonymizing and disabling the Tor network. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2014.

[17] The JonDonym Anonymization Service. `http://anonymous-proxy-servers.net`, visited May 2014.

[18] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2013.

[19] jrandom (Pseudonym). Invisible Internet Project (I2P) Project Overview. Design document, August 2003.

[20] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2014.

[21] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright. Timing attacks in low-latency mix systems. In *Financial Cryptography*, 2004.

[22] M. Liberatore and B. N. Levine. Inferring the source of encrypted http connections. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, CCS '06, 2006.

[23] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci. Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2011.

[24] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proc. of IEEE Symposium on Security and Privacy*, 2005.

[25] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Proc. of ACM Workshop on Privacy in the Electronic Society*, 2011.

[26] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[27] K. Rieck, C. Wressnegger, and A. Bikadorov. Sally: A tool for embedding strings in vector spaces. *Journal of Machine Learning Research (JMLR)*, 13(Nov):3247–3251, Nov. 2012.

[28] A. Robertson and P. Willett. Applications of n-grams in textual information systems. *Journal of Documentation*, 58(1):48–69, 1998.

[29] S. Rosenblatt. Nsa tracks google ads to find tor users. `http://www.cnet.com/news/nsa-tracks-google-ads-to-find-tor-users/`, visited October 2014.

[30] B. Schneier. How the nsa attacks Tor/Firefox users with Quantum and FoxAcid. `https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html`, 2013.

[31] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: attacks and defenses. In *Proc. of European Symposium on Research in Computer Security (ESORICS)*, 2006.

[32] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proc. of IEEE Symposium on Security and Privacy*, 2002.

[33] The Tor project. `http://www.torproject.org`, visited May 2014.

[34] T. Wang and I. Goldberg. Improved website fingerprinting on Tor. In *Proc. of ACM Workshop on Privacy in the Electronic Society*, 2013.

[35] X. Wang, S. Chen, and S. Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *Proc. of IEEE Symposium on Security and Privacy*, 2007.

[36] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.

[37] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao. Dsss-based flow marking technique for invisible traceback. In *Proc. of IEEE Symposium on Security and Privacy*, 2007.

[38] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proc. of Privacy Enhancing Technologies Symposium (PETS)*, 2004.