Privacy Vulnerabilities in Encrypted HTTP Streams

George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine

University of Massachusetts, Amherst, MA 01003, USA
{gbiss,liberato,jensen,brian}@cs.umass.edu

Abstract. Encrypting traffic does not prevent an attacker from performing some types of traffic analysis. We present a straightforward traffic analysis attack against encrypted HTTP streams that is surprisingly effective in identifying the source of the traffic. An attacker starts by creating a profile of the statistical characteristics of web requests from interesting sites, including distributions of packet sizes and inter-arrival times. Later, candidate encrypted streams are compared against these profiles. In our evaluations using real traffic, we find that many web sites are subject to this attack. With a training period of 24 hours and a 1 hour delay afterwards, the attack achieves only 23% accuracy. However, an attacker can easily pre-determine which of trained sites are easily identifiable. Accordingly, against 25 such sites, the attack achieves 40% accuracy; with three guesses, the attack achieves 100% accuracy for our data. Longer delays after training decrease accuracy, but not substantially. We also propose some countermeasures and improvements to our current method. Previous work analyzed SSL traffic to a proxy, taking advantage of a known flaw in SSL that reveals the length of each web object. In contrast, we exploit the statistical characteristics of web streams that are encrypted as a single flow, which is the case with WEP/WPA, IPsec, and SSH tunnels.

1 Introduction

The problem of keeping Internet communication private is remarkably hard. One method of protecting the privacy of a network connection is to use an encrypted link to a proxy or server. Encrypted links are possible at the link layer using WEP/WPA to a wireless base station, at the network layer using IPSec ESP mode to a VPN concentrator, or at the transport layer using an SSH tunnel to an anonymizing proxy. In all cases, the identity of the final destination is kept confidential from an eavesdropper by encrypting IP packet headers.

Maintaining user privacy is not such a simple matter. In this paper, we show that an encrypted connection is not sufficient for removing traffic patterns that

This paper was supported in part by National Science Foundation awards CNS-0133055, ANI-0325868, and EIA-0080199.

often reveal the web site that a user is visiting. Specifically, we examine the success rate of traffic analysis attacks used by an eavesdropper that test against learned profiles of web site traffic inter-arrival times and packet sizes.

We examine real traces of encrypted, proxied HTTP traffic, and our attacks attempt to discern the responder to each web request. The method of our attack is straightforward. In advance, the attacker gathers a *profile* of specific websites according to some criteria, which may be their popularity or level of interest to the attacker. The profile is composed of two features from the encrypted HTTP response stream: the packet size and inter-arrival time distributions. The attacker then monitors the traffic of a wireless link or a wired link to which he has access. When a burst of traffic occurs, the attacker tests the trace against a library of profiled web sites looking for a good match.

We tested our method by taking traces for three months of hourly retrievals of 100 popular web sites. Our evaluations show that many web sites are subject to this attack. With a training period of 24 hours and a 1 hour delay afterwards, the attack achieves only 23% accuracy. However, an attacker can easily pre-determine which of trained sites are easily identifiable. Accordingly, against 25 such sites, the attack achieves 40% accuracy; with three guesses, the attack achieves 100% accuracy for our data. Longer delays after training decrease accuracy, but not substantially. Note that with random guessing, this attack can expect to be correct only $1/n$th of the time among n profiles, and k/nth of the time with k guesses. While previous work exists on similar attacks, ours is the first to consider an encrypted web connection that does not reveal individual web objects, which is the realistic case for WEP/WPA, VPNs, and SSH tunnels.

The remainder of this paper is organized as follows. Section 2 describes recent related work. In Section 3 we present our data collection methodology, and in Section 4 we describe how we identify encrypted traffic. Section 5 is a summary of our future research goals, and Section 6 concludes.

2 Related Work

There is a large body of work on the topic of general traffic analysis and information hiding. We do not provide an extensive overview here; consult Raymond [1] for an informal overview. Instead, we present an overview of recent developments in theoretical and experimental traffic analysis and countermeasures. We include work that examines HTTP and secure HTTP and the vulnerabilities and exposures inherent in those protocols.

Hintz [2] describes a fingerprinting attack similar to ours. It is limited in several ways. First, it considers only the total amount of data sent over each SSL-encrypted connection. When the client is using more sophisticated tunneling software that re-uses connections (as we assume in this paper), this attack would degrade. Additionally, Hintz's work is a very preliminary proof-of-concept and does not provide a significant evaluation.

Sun, et al. [3] investigate the use of statistical techniques to identify encrypted web traffic. They assume an SSL encrypted link between a browser

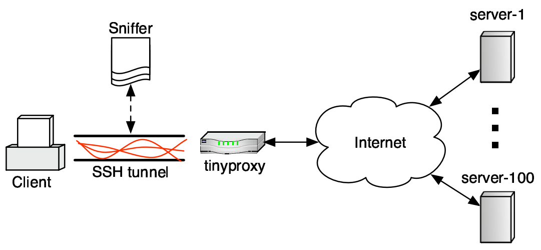

Fig. 1. Measurement setup.

and an anonymizing proxy, and focus on attacking this first and weakest link. They make the strong simplifying assumption that web objects can be differentiated by examining the TCP connections between the browser and the proxy. This assumption is not valid for WEP/WPA links, VPN connections, and SSH tunnels, and in the presence of widely-available, unencrypted, pipelined HTTP connections. They use the patterns of object sizes to classify web sites and their technique cannot function without these object sizes. Privacy-aware users are likely to know that SSL/TLS alone is not sufficient to hide traffic patterns, as is mentioned in the protocol specifications [4, 5].

Fu, et al. have produced at least two related papers. In the first [6], they describe the use of active probes to determine traffic payload rate and the use of statistical pattern recognition to evaluate traffic analysis attacks and corresponding countermeasures. This differs from our work in at least two key areas. First, they determine payload rate, a much simpler problem than the more exact classification we are attempting. Second, they require the ability to inject packets into the network, in the form of active pings. This active measurement is required during both the "off-line" data collection and training and "on-line" classification. In contrast, our technique only performs active measurements during the training phase, and is entirely passive during the classification phase.

In their second paper [7], they examine link padding as a means of defeating traffic analysis attacks. In particular, the authors establish a formal theoretical framework for link padding systems and derive closed-form formulas for estimation of detection rates. Additionally, they compare two methods of introducing padding, and conclude that variable intervals between padding are superior to constant. This result may prove useful in defending against our technique; we intend to investigate this technique in the future.

3 Data Collection

To begin our study, we collected data from January 01, 2004 until March 31, 2004. In this section, we describe the procedure we used to collect our data. We used the results of a prior user study by Wright, et al. [8] to determine the sites most visited by users of the computers in our department. The study tracked

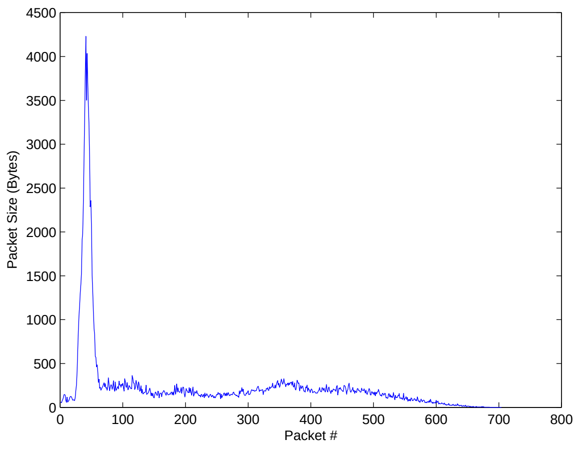

Fig. 2. A sample size profile for `www.amazon.com`.

the web traffic of 24 volunteers for 214 days. We examined the proxy logs from Wright's study and used the 100 most-visited sites for experiments in our study.

To retrieve a baseline version of each site, we scripted an instance of Mozilla Firefox 1.0 to retrieve a site's main page and all graphics or other objects. Fig. 1 illustrates our measurement setup. We configured Firefox to connect through an instance of tinyproxy 1.6.3 bound on a local port via an SSH tunnel (OpenSSH 3.5p1 with compression and encryption enabled). All processes involved in the collection were running on the same machine. Our script retrieved the main page of each of the 100 sites every hour. We used tcpdump 3.7.2 to sniff and record the encrypted traffic over the SSH tunnel.

For each HTTP trace, we recorded two features: the inter-arrival time of each packet and the size of each packet. Each is a chronological data sequence that we call a *time trace* and *size trace*, respectively. No other features are available to an attacker; we did not perform any cryptanalysis.

For each day over a three month period, we collected inter-arrival and size traces once per hour, for a total of over 200,000 distinct data points.

4 Identifying Encrypted Traffic

The main goal of our study is to answer the following question: does a trace of web traffic, sent through an encrypted tunnel to a proxy, leak sufficient information to allow us to determine the site being accessed? As this section details, we have found that many popular web sites are reasonably identifiable even with relatively old training data (see Figure 4), and that some are extremely distinctive.

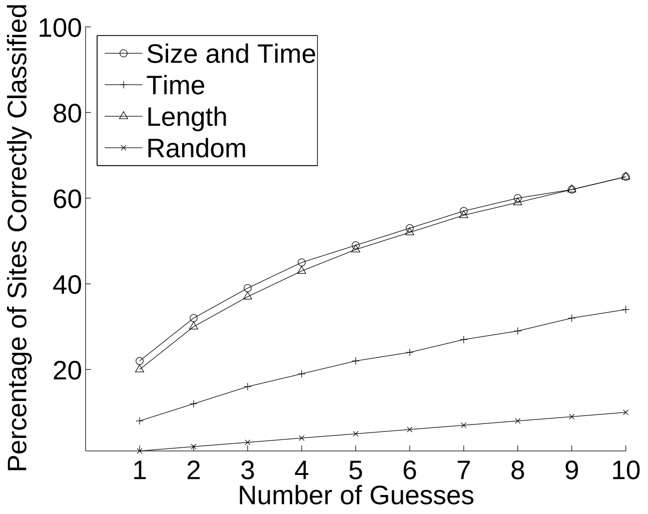

Fig. 3. Accuracy per number of guesses with a one hour gap and different classification methods. In all figures, the gap refers to the time between a 24 hour training period and a single test.

4.1 Performing the Attack

For our study, our attacker model is as follows. A client desiring privacy is connected to a server over an encrypted link. This link could be a VPN, an SSH tunnel, or a WEP-enabled access point. Before the attack, the attacker sets up a link with similar network characteristics and gathers packet traces. We believe this to be a reasonable assumption — the attacker could gather traces from the same ISP, or at the site of the victim's accesses, such as an Internet café. From these sets of packet traces, the attacker constructs a set of *profiles*, as described below. Then, the attacker monitors the encrypted link and attempts to match the profile of activity detected on the link with the set of known profiles, returning a ranked list of matching candidates. We assume that think times dominate network delay and that the attacker can easily distinguish separate sets of requests to a server.

Since we contacted each site many times over the course of months during our data collection, our data set is comprised of numerous traces from every site. For each site, there is an associated set of packet traces. We restrict our attention to two particular characteristics of each such packet trace: the inter-arrival time and the packet size. We organize our data as a set of tuples: (N, D, I, S), where N is a unique identifier for each site, D is the timestamp of the particular collection, and I and S represent inter-arrival time and packet size traces, respectively.

We define an *inter-arrival time trace*, I, as the sequence of n inter-arrival times, $\{t_1, t_2, \ldots, t_n\}$, between packets for a given trace. To construct a *time profile*, we coalesce a set of inter-arrival times, each corresponding to the same

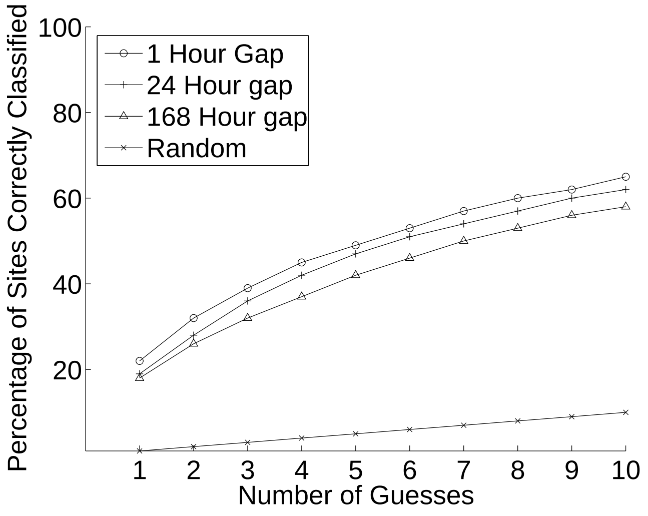

Fig. 4. Accuracy per number of guesses. The random line is the result of choosing site labels uniformly at random. In this and all following figures, we are using the combination of both the size and time profiles.

site, N, but a different time, D, into a single, new inter-arrival time trace. We take the arithmetic mean of each t_i in the set for each $1 < i < n$, and use this as the new t_i in the time profile. A corresponding trace and profile exist for packet sizes, which we denote as the *size trace* and *size profile*, respectively. Figure 2 shows, as an example, a profile of `www.amazon.com`.

To compare an individual trace to a profile, we use the *cross correlation* of the two sequences of values. In general, the cross correlation, r, between two sequences of real numbers $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_n\}$ is defined as:

$$r = \frac{\sum_{i=1}^{n} [(x_i - \bar{x})(y_i - \bar{y})]}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})}} \qquad (1)$$

where $\bar{x}$ and $\bar{y}$ are the means of the corresponding series. Intuitively, the cross correlation estimates the degree to which two series are correlated. It is a summary statistic indicating whether the individual numbers in two sequences have similar magnitude. We call the cross correlation of a trace and a profile, the *similarity* of the trace to the profile. When attempting to classify an unknown data set, we compute its similarity to each of the profiles we have previously created. A sequence of guesses is returned, each corresponding to a particular site and ordered by the magnitude of the similarity.

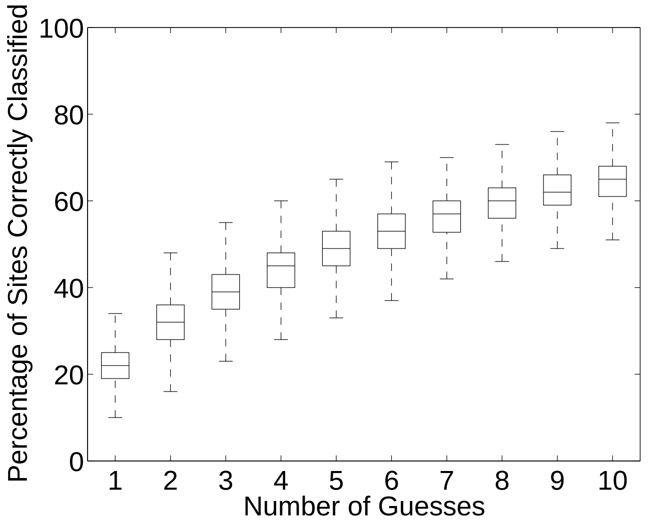

Fig. 5. Accuracy per number of guesses with a one hour gap.

To evaluate our classifier, we built profiles from each contiguous 24 hour span of traces of each site. Some sites were missing data, but no 24 hour training period contained more than one missing trace. When a trace was missing from the training data, we omitted it from the profile. We then tested the performance of the classifier (as described above) on single traces from some time in the future. We call the amount of time between the last trace in the profile until the tested trace the *gap*. We evaluated the classifer with gaps of one hour, 24 hours, and 168 hours (one week). We constructed the two profile types for each training set, and we analyzed three methods of classifying data for the attacker:

– Size profile only;
– Time profile only;
– Size and time profile: the product of the size and time profile similarities.

We found the third method, shown in Figure 3, to be most effective overall, and utilized that method in all further results presented here. As shown in Figure 4, accuracy decreases as the gap between training and testing grows, but the trend remains consistent. The implications for the attacker are clear: training immediately before the attack is best, but even old data allows for some identification.

4.2 Predicting Identifiability

Some sites may be more identifiable than others. There is an obvious way to evaluate identifiability based on our classification methodology. By examining the *rank* of the correct site in the list our classifier returns, we have a metric

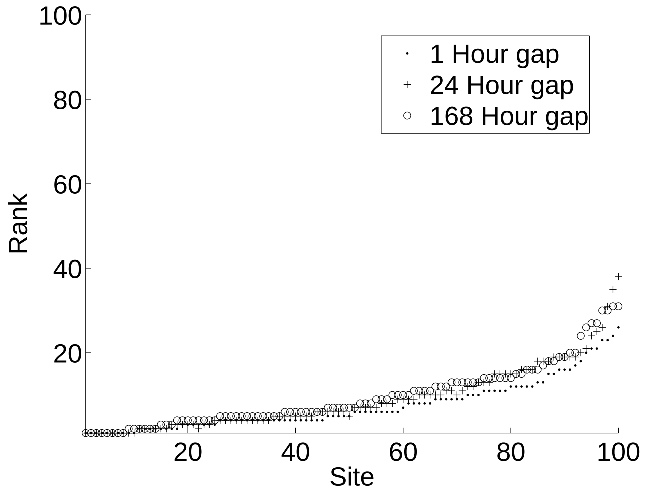

Fig. 6. Per-site rank in the ordered list of guesses.

describing how identifiable a site is. We show in Figure 6 this metric for each of the 100 sites and various gap sizes. As one would expect, smaller gap sizes result in high identifiability. Some sites are surprisingly and consistently identifiable, while others are quite difficult to differentiate. This trend is shown more explicitly in Figure 7. The accuracy of the most identifiable sites is much higher. In Figure 8, we show the accuracy for the top 25 sites. Accuracy with one guess is 40%, and increases to above 70% with two guesses.

This is of profound importance to the attacker, as she is able to tell *a priori* which sites will have such identifiability by examining the ranking data for that site. In general, an attacker would like to know which sites are most recognizable, and a defender would like to know how to make a site less distinguishable. We believe that a metric such as this ranking metric can guide both attackers and defenders in determining the relative identifiability of sites. However, further study is needed to discover the specific sources of identifiability.

5 Future Work

We intend to extend this work in a variety of ways. What follows is a short list of the improvements and extensions we are currently considering.

We expect the time profile to be highly dependent on the specific path between the server and client. However, we believe that packet sizes are not strongly dependent on that path, as the Internet has a fairly standard MTU size of 1500 bytes. Thus, it may be possible to train a size profile from one location on the Internet, and test against that profile elsewhere. Further experiments are needed to confirm this conjecture.

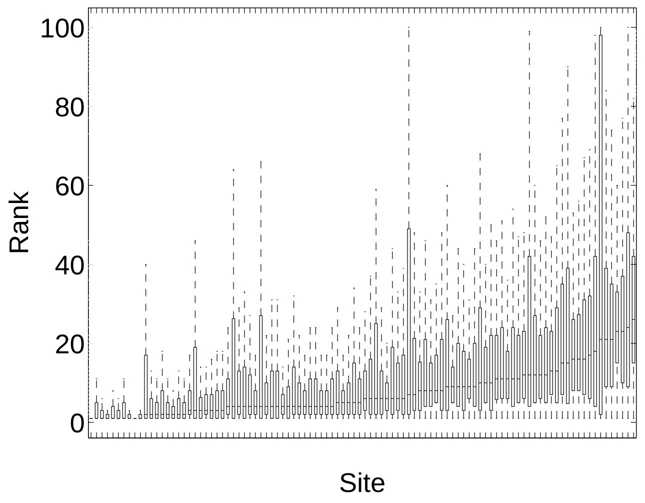

Fig. 7. Per-site rank in the ordered list of guesses with a one hour gap.

It is unclear what affect the specific web browser has on the profile. There are only a handful of web browsers currently in wide use, and they tend to use one of several rendering engines: Microsoft Corporation's Internet Explorer, the Mozilla Foundation's Gecko (used in all recent versions of Netscape, AOL, and Firefox browsers), and the KDE project's khtml (used in Apple's Safari browser). A separate profile may be needed for each of these engines, or it may be the case that they perform server requests in the same order.

Examining more sites would help show the robustness of this attack. An argument can be made that there are billions of web pages and it may be impossible to distinguish among all of them. But for a targeted attack, an attacker might only need to be able to identify a few hundred or thousand with a low false positive rate. Similarly, it would be of great value for a site operator to know what characteristics of their site or network account for identifiability, and if it is possible to obfuscate these characteristics.

The technique we used for identification is not particularly insightful, and yet we are able to achieve an 20% accuracy rate with a single guess, and over 40% if we limit ourselves to a small set of possible sites and a single guess. Multiple guesses on this small set quickly drive our accuracy toward 100%. A more sophisticated approach, such as density estimation of a profile, would likely yield better results that are more robust to small fluctuations in network conditions or site changes. A more careful characterization of the identifiable characteristics of the traffic would likely also lead to a higher identification rate. Similarly, examining actual user behavior may yield further insights: users typically navigate from page to page, and we may be able to leverage some form of Bayesian inference to improve our results. It would also be enlightening to attempt to discern how

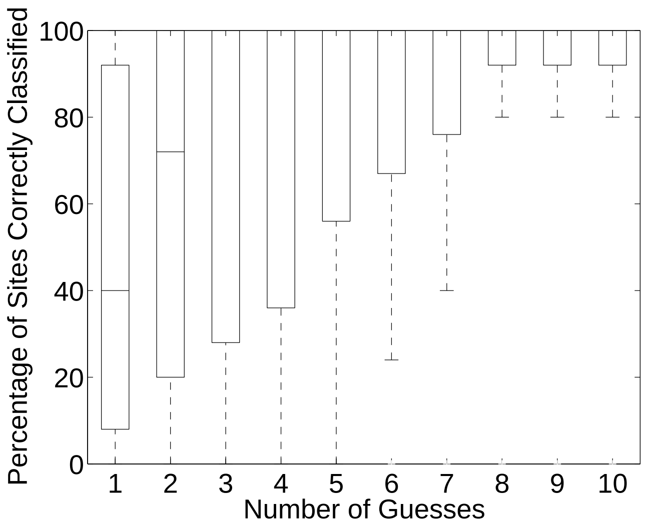

Fig. 8. Accuracy per number of guesses for the 25 most identifiable sites with a one hour gap.

much of each portion of delay, packet fragmentation, or packet size is due to server or browser configuration and network path effects.

Also interesting would be an examination of how multiple proxies, each on separate networks on the Internet, affect our attack. Some of our previous results suggest the attack will work. Wright, et al. [9] examined a traffic analysis attack to link a single network flow routed through multiple anonymizing proxies, where the measurements are taken simultaneously at different points in the stream. In contrast, our attack creates a profile that remains effective over a long period of time (i.e., on the order of days) but has no intermediate proxies to add new statistical characteristics to the stream. We believe it would be possible, under some circumstances, to compose the two attacks if mixing is not performed by the intermediate proxies. Experiments are needed to confirm this conjecture.

Finally, an exploration of defenses is needed. The effects of delay or link padding and other traffic shaping must be quantified to help future designers of privacy-preserving protocols in avoiding this type of attack.

6 Conclusion

We have presented a straightforward, real-world, successful attack against supposedly private HTTP transactions. This attack is based upon forming profiles of possible sites being visited, and matching traffic against these profiles. It requires some preliminary work on the part of the attacker, but thereafter yields surprisingly effective and results. We have also shown a simple way of determining in advance the efficacy of the attack. Finally, we have pointed out interesting ways in which this attack could be extended, and possible methods of defense.

References

1. Raymond, J.F.: Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In: Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability. Volume 2009. (2001) 10–29
2. Hintz, A.: Fingerprinting websites using traffic analysis. In: Proceedings of Privacy Enhancing Technologies workshop (PET 2002), Springer-Verlag, LNCS 2482 (2002)
3. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: Proceedings of the IEEE Security and Privacy Conference. (2003)
4. Dierks, T., Allen, C.: RFC 2246: The TLS protocol version 1 (1999)
5. Freier, A.O., Karlton, P., Kocher, P.C.: Secure Socket Layer. IETF Draft. (1996) http://home.netscape.com/eng/ssl3.
6. Fu, X., Graham, B., Bettati, R., Zhao, W.: Active Traffic Analysis Attacks and Countermeasures. In: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing. (2003) 31–39
7. Fu, X., Graham, B., Bettati, R., Zhao, W.: Analytical and Empirical Analysis of Countermeasures to Traffic Analysis Attacks. In: Proceedings of the 2003 International Conference on Parallel Processing. (2003) 483–492
8. Wright, M., Adler, M., Levine, B.N., Shields, C.: Defending Anonymous Communication Against Passive Logging Attacks. In: Proceedings of the IEEE Symposium on Security and Privacy (Oakland). (2003) 28–41
9. Levine, B.N., Reiter, M., Wang, C., Wright, M.: Stopping Timing Attacks in Low-Latency Mix-Based Systems. In: Proceedings of Financial Cryptography (FC). (2004)