Inferring the Source of Encrypted HTTP Connections

Marc Liberatore and Brian Neil Levine
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003-9264
liberato@cs.umass.edu, brian@cs.umass.edu

ABSTRACT

We examine the effectiveness of two traffic analysis techniques for identifying encrypted HTTP streams. The techniques are based upon classification algorithms, identifying encrypted traffic on the basis of similarities to features in a library of known profiles. We show that these profiles need not be collected immediately before the encrypted stream; these methods can be used to identify traffic observed both well before and well after the library is created. We give evidence that these techniques will exhibit the scalability necessary to be effective on the Internet. We examine several methods of actively countering the techniques, and we find that such countermeasures are effective, but at a significant increase in the size of the traffic stream. Our claims are substantiated by experiments and simulation on over 400,000 traffic streams we collected from 2,000 distinct web sites during a two month period.

Categories and Subject Descriptors

C.2.0 [**Computer-Communications Networks**]: General—*Security and protection (e.g., firewalls)*; C.2.3 [**Computer-Communications Networks**]: Network Operations—*Network monitoring*; I.5.4 [**Pattern Recognition**]: Applications; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Measurement, Security

Keywords

low-latency anonymity, network forensics, traffic analysis

This research was supported in part by National Science Foundation awards CNS-0133055 and ANI-0325868.

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.

1. INTRODUCTION

Encrypted connections provide confidentiality at many different network layers. In combination with a proxy, this setup forms the basis of private communication over the Internet. Many such systems prevent observers from determining the true destination of IP traffic: multi-proxy tunnels using the Tor anonymous communication system [6]; simple SSH tunnels to a single proxy; and IPSec ESP mode tunnels to a remote VPN concentrator. WPA link layer connections also hide the destination IP address and content of a connection from an observer.

In this paper, we evaluate traffic analysis techniques that infer the source of a web page retrieved under the cover of an encrypted tunnel. These techniques identify sources by comparing observed traffic to profiles of known sites created from packet lengths, and are referred to as *profiling attacks*. A previous study [1] has shown the attack is feasible, with a method achieving about 25% accuracy.

This type of attack on web traffic is an area of concern for advocates of privacy enhancing technologies. This includes the developers of Tor, who have recognized that this type of web page fingerprinting is a significant problem [5]. Currently deployed low-latency anonymity systems do not significantly adjust traffic to prevent comparison to profiles. The advances we detail in this paper increase the importance of addressing the attack in implementations — our study includes results with an accuracy of up to 90% for realistic scenarios.

Privacy is the dual problem of digital forensics — accordingly, the attack is also an important method of investigation that advances the field of digital forensics. This is because commonly used forensic techniques for gathering and analyzing network data are limited to traffic with overt IP headers and data [2, 3]. Encrypted tunnels thwart the legitimate gathering of evidence by authorized law enforcement. Advances in the field of forensics require moving beyond this limitation. This paper provides guidance for such advances.

Contributions. Our results are based on traces we gathered of encrypted communications to 2,000 web sites, which we collected four times a day for two months. We have made these traces publicly available for validation, collaboration, and future work. To our knowledge, this is the largest public collection of such traffic for the study of profiling attacks. We built two systems that identify traffic, one based on the naive Bayes classifier and one on Jaccard's coefficient, a straightforward similarity metric. Both systems rely on packet lengths but discard timing information.

Despite this simplification, we found that under reasonable assumptions, traces were identifiable between 66–90% of the time. On the basis of our experiments, we expect performance to scale well, and we assume that more sophisticated methods could do better. We also examine in simulation the effectiveness of per-packet padding (that is, increasing the length of packets) in an attempt to defeat our profiling system. We find that this approach is reasonably effective, lowering predictive accuracy to less than 8% while increasing traffic volume by 145%.

While profiling requires a set of candidate sites, we believe that it would not be difficult for an observer to profile all publicly accessible web sites on the Internet in time for the attack to succed. In part, this is due to our finding that classification accuracy degrades very slowly over time, giving the observer at least four weeks to collect the profile after the encrypted traffic is observed in many cases. Moreover, using the technique we evaluated, an investigator could store profiles of the front pages of all web sites on the Internet with about 13GB of storage.

The remainder of this paper is organized as follows. Section 2 describes related work. In Sections 3 and 4, we describe our attack model and data collection methodology, and in Section 5, we describe our experimental methodology and results. We discuss the implication of these results in Section 6, and conclude in Section 7.

2. RELATED WORK

Traffic analysis is a large field; in this section, we survey work in that field that is related to the analysis of anonymity systems. In particular, we discuss prior work on passive logging, profiling (or fingerprinting) attacks, and analysis of countermeasures to these attacks, as these are the most relevant to our work. We focus on work that examines HTTP and secure HTTP and the vulnerabilities and exposures inherent in those protocols. For a broader overview of the field, consult Raymond [11] or the Free Haven anonymity bibliography[1].

Wright, et al. [15, 16] and others have shown that low-latency anonymity systems are vulnerable to passive logging and an intersection attack. These results are complementary to our own: The attack allows identification of the endpoints in an anonymous communication system with path changes, whereas our technique allows an observer to infer the content (and thus, the endpoint) from the traffic itself.

Hintz [9], Sun, et al. [12], and Bissias, et al. [1] present profiling attacks of encrypted connections, though our study differs from each of these in important ways. Hintz's work is a preliminary proof-of-concept, examining total data sent over SSL connections. While the technique is reasonably effective, SSL and its successor, TLS, are not designed to hide traffic patterns [4, 7]. One of our classification techniques is drawn from the work presented by Sun, et al., though the problem we study differs from theirs in a nontrivial way. In their work, they compare the size of web objects, rather than packets. This comparison is possible due to their strong simplifying assumption that objects can be differentiated by examining the timing of TCP connections. This assumption is not valid for WEP/WPA links, VPN connections, and SSH tunnels, and in the presence of widely-supported pipelined HTTP connections. We make the weaker assumption that packets, not objects, can be distinguished, and we base our identification of pages on the individual packets that compose these pages. Bissias, et al. present work similar in its assumptions to ours. They examine packet lengths and timings as the basis of identification, though they use only the crude metric of cross-correlation to determine similarity. The study also is preliminary in nature, attempting to distinguish among only 100 web sites. While appropriate for an initial study, such a small sample is hard to generalize from; in this study, we show results for a larger, more robust data set.

[1] `http://freehaven.net/anonbib/topic.html`

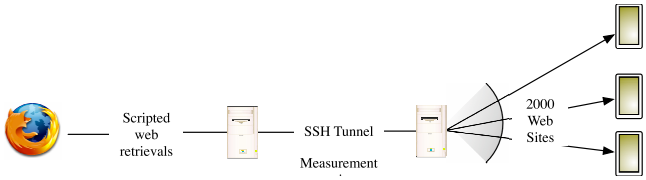

Figure 1: An illustration of the measurement setup. This figure also shows the position an observer may take to utilize profiling.

There are few careful studies of countermeasures to profiling attacks upon low-latency anonymity systems on the Internet. Fu, et al. [8], examine the technique of introducing dummy packets on links to defeat some types of traffic analysis. Their results are largely theoretical in nature, and designed to thwart general traffic analysis rather than specific profiling attacks. They find that variable intervals between dummy packets are more effective that constant intervals. Levine, et al. [10] examine dropping packets to foil statistical correlation. A generalization of partial-path cover traffic [13], this technique and their analysis is concerned mainly with foiling timing attacks. We believe that there is much work yet to be done in examining latency and bandwidth tradeoffs in cover traffic and delays for low-latency anonymity systems.

3. MODEL

In this section, we describe our model of an observer that can execute traffic analysis attacks to profile web sites and identify encrypted traffic on the basis of these profiles. We first describe our assumptions about how data can be collected by an observer. Then, we describe two specific methods to create these profiles, one based upon a similarity metric (Jaccard's coefficient) and one based upon a supervised learning technique (the naive Bayes classifier).

3.1 Observer Model

Figure 1 illustrates the network setup that we use throughout this paper. In it, the client connects to a remote proxy over an encrypted transport layer. The proxy makes requests on the client's behalf, and returns the results over the encrypted connection. The observer is limited to examining the encrypted traffic and creates a log of packet lengths (and interarrival times, if desired) corresponding to each distinct page load. Our observer has unlimited storage for these logs. We assume that the observer is not able to discriminate among individual objects as in Sun, et al. We assume the client uses a modern browser for retrieval, as described in Section 4.2, which prevents the observer from obtaining this information.

Because our techniques focus on packet lengths, it is not a requirement that the observer create profiles on the same link that she observes traffic. However, for simplicity, we assume this is the case.

We assume that the observer is able to determine where discrete communications begin and end (such as the loading of a web page and its associated objects). This is possible, for example, by observing sender think times that separate requests. In future, we will determine mechanisms for separating multiple requests and requests that appear with background traffic.

The proxy we evaluate in this paper is the OpenSSH implementation of a one-hop SOCKS proxy. However, we expect our results hold for VPN proxies and WPA base stations. Neither of these systems significantly alter packet lengths since they perform no buffering and packet aggregation or fragmentation. The Tor and JAP[2] low-latency anonymity systems provide only very limited aggregation and fragmentation. We believe that these systems will also be vulnerable to a form of this attack.

To launch the class of traffic analysis attacks that we evaluate in this paper, the observer requires a library of traffic traces between a client and a list of known web sites. We show in Section 5 that this library can be collected before or after the attack. Using the traces of connections to known destinations, the observer attempts to decide, in some fashion, which of the known traces most closely resembles the encrypted, unknown trace, as we explain below.

3.2 Profiling Methods

To determine similarity to known traces, the observer describes each trace in terms of *attributes* and lets each attribute range over many possible values. The problem then becomes an instance of supervised learning, as the observer has a set of labeled training instances (the traces gathered by the observer) and one or more unlabeled test instances (the observed, encrypted traces).

In the remainder of this paper, we denote each trace together with its attributes as an *instance*. Each instance has an attribute denoting the URL, or site, to which it corresponds. We also refer to this as the *class* of the instance. Each instance also has attributes that describe every packet in the trace. These attributes take the form of a tuple, *(direction, length)*. The *direction* denotes whether the packet went from the client to the server referenced in the URL, or vice versa; the length denotes the total length, in bytes, of the packet. The value assigned to each attribute is the number of packets observed in that trace with the corresponding *direction* and *length*.

Our first method for identifying unknown instances is to use a similarity metric; we use Jaccard's coefficient to measure similarity and thus determine the class of an instance. For two sets X and Y, Jaccard's coefficent S is defined as:

$$S(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{1}$$

We build a model for Jaccard's coefficient based classification as follows. Each site in the model represented by a set. If there is only one training instance per site, then the model is built as follows: For each packet length and direction in a training instance, a *(direction, length)* tuple is inserted into the set corresponding to that site. If there is more than one instance in the training set per site, then such tuples are inserted into the set iff they are present in the majority of the instances in the training set. To convert the similarity metric S to an estimate of class membership probability, we normalize $S(X,Y)$ for a given X by dividing by $\sum_{Y \in U} S(X,Y)$, where U is the set of all training sets.

Our second method for identifying unknown instances is the naive Bayes classifier. We do not give a full explanation of the classifier here: a recent textbook such as Witten and Frank [14] will provide details. In short, the naive Bayes classifier assumes independence between all attributes, and estimates the probability of a set of value $\overline{A} = A_1, \ldots, A_n$ belonging to a particular class C_i as:

$$p(C_i|\overline{A}) \propto p(C_i) \prod_{j=1}^{n} (p(A_j|C_i)) \tag{2}$$

In our experiments, which are described below, we use Witten and Frank's Weka toolkit implementation, `weka.classifiers.bayes.NaiveBayes`, with normal kernel density estimation enabled.

4. DATA COLLECTION

To investigate the effectiveness of a profiling attack, we required a data set consisting of logs of encrypted traffic between a client and many servers.[3] We imposed several requirements on this data set. First, it had to be of sufficiently large size to make the problem non-trivial, while not so large to prevent multiple collections each day with our limited resources. Second, it had to reflect a real-world group of users. Finally, it had to be collected in a fashion analogous to the manner in which an actual observer would attempt. In this section, we present the details of these processes so that others can validate or recreate our collection process.

4.1 Initial Data Collection

We used our department's Internet traffic as a basis for choosing sites to profile. This network is used by an estimated population of over three hundred faculty, staff, and students. By monitoring DNS requests within the department, we gathered a list of remote hosts to which users connected. We heuristically refined this list into a set of HTTP URLs which we believe are reasonably representative of the web browsing habits of users in our department.

The initial step was to log all requests to the department's DNS server from 01 December 2005 through 04 January 2006. This month of logs yielded 44,305,203 requests from 828 hosts. We removed all requests that were not for address (`A`) records, as HTTP traffic would not have generated them. We removed all requests that were from outside the department, as we were studying users within the department. We removed all requests that were for names within our domain, as these tend to correlate with intra-department service accesses (such as secure shell access) and not HTTP requests.

We then removed requests we judged to be the result of automatic processes. Specifically, we removed all requests for a given name that were made from the same host with

[2] `http://anon.inf.tu-dresden.de/`

[3] The anonymized logs we collected and used are available at `http://traces.cs.umass.edu/`

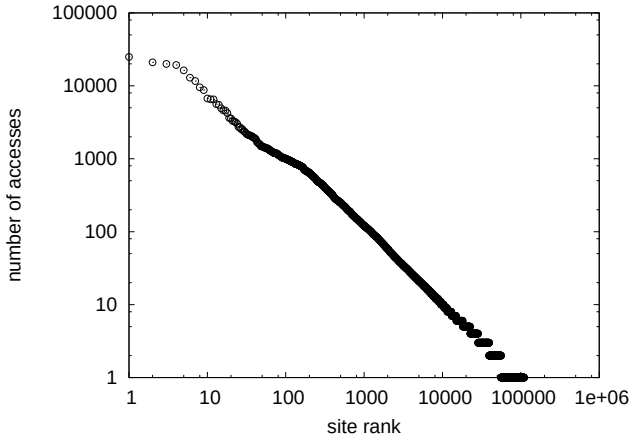

Figure 2: The relationship between site rank and number of accesses. Sites are ranked according to total number of accesses to each site observed.

an average frequency of greater than once per five minutes in any eight hour period. We then heuristically removed most requests for PlanetLab[4] machines.

From these requests, we constructed a list of URLs of the form `http://ipaddress/`, along with the associated count for each. On 01 February 2006, we attempted to contact each of these sites with an HTTP GET request on port 80. We removed from the list all sites that were unreachable, refused the connection, or returned an HTTP error that was not a redirection. We also replaced the `ipaddress` with the actual hostname the remote machine used, and we resolved all redirections. Finally, we summed the counts of all sites that redirected to the same URL. This final list of *(count, URL)* pairs was 109,479 pairs in length. For our experiments, presented in the next section, we focused on the 2,000 most accessed sites, which accounted for 64% of all web requests. The relationship between rank and number of accesses is show in Figure 2.

4.2 Page Retrieval

To gather realistic traces, we set up a client host with a recent GNU/Linux distribution. We used Mozilla Firefox 1.5[5] to retrieve each URL via a SOCKS proxy. OpenSSH 4.2p1[6] was set up to perform application level dynamic port forwarding (the `-D` option) and act as a proxy. This created an encrypted channel over which the HTTP requests and responses were forwarded, and it made distinguishing individual objects infeasible, as Firefox generally makes multiple simultaneous connections to load a web page that are multiplexed over the secure channel.

We configured Firefox to not cache data between retrievals, which allowed us to focus on the specific question of identifying encrypted streams by their profiles. We installed the latest Macromedia Flash plugin[7], as many of the URLs in our list contained content rendered by this plugin. We also configured Firefox to not attempt various extraneous connections, due to live bookmarks, automatic update checks, and the like. While disabling these features makes the resulting traffic somewhat less realistic, we believe it to be a reasonable trade-off to allow us to focus on the specific problem under investigation. As Firefox loaded each URL in our list, we used tcpdump 3.9.4[8], linked against libpcap 0.9.4, to log the first 68 bytes of each packet. This length is sufficient to capture IP and TCP headers of the packet and thus determine the total packet length. The information in these logs form the basis of the profiles that we detail in the next section.

[4] `http://www.planet-lab.org/`
[5] `http://www.mozilla.org/projects/firefox/`
[6] `http;//www.openssh.org/`
[7] `http://www.macromedia.com/`

5. EVALUATION

In this section, we describe the experimental methodology we used to evaluate our proposed classification methods as well as the results of that evaluation, based upon two months of data we collected. We give evidence that the two methods, one based upon Jaccard's coefficient and the other upon the naive Bayes classifier, have several properties of interest: We show that the Jaccard-based classifier's accuracy, under reasonable assumptions, is over 60%. We give evidence that it will scale reasonable well to large data sets. We show that many profiles, once constructed, remain valid for long periods of time. We show that training data can be gathered before or after test data, with a negligible effect upon accuracy. We show evidence that the identifiability of traces using these methods is a result of the distinctiveness in both the individual packet length as well as overall trace length. Finally, we examine the robustness of our method to several forms of static countermeasures, and find that users using these countermeasures must be willing to incur a bandwidth penalty of 145% in order to drive an observer's accuracy below 8%.

5.1 Experiment Setup

To evaluate the effectiveness of the profiling attack described above, we collected traces of encrypted traffic as described in Section 4. Specifically, each *sample* consists of the log of a retrieval of each of the 2,000 most-visited web sites. We created a new sample once every six hours for a period of two months, for a total of 480,000 samples.

From these samples, we performed individual experiments of several variables. Each experiment utilizes a set of training samples and a distinct set of testing samples. Each of the following three variables are relative to some sample i, the *initial sample*.

- t describes the number of sequential samples that form the training set.
- s describes the number of sequential samples that form the test set.
- Δ describes the number of sequential samples between the training and test sets. $\Delta = 0$ indicates the test set starts with the sample immediately following the last sample in the training set.
- N describes how many of the 2,000 sites we considered in a particular experiment. If it is less than 2,000, then we reduced the number of traces in each sample to this number by removing the traces corresponding to the same sites from all samples. We removed the least-popular sites first.

[8] `http://www.tcpdump.org/`

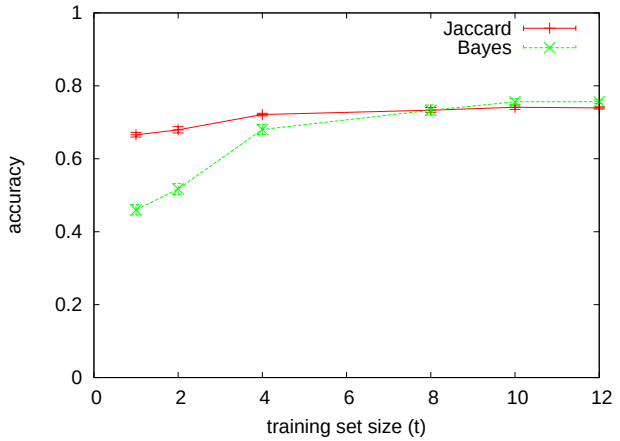

Figure 3: Effect upon accuracy of varying training set size.

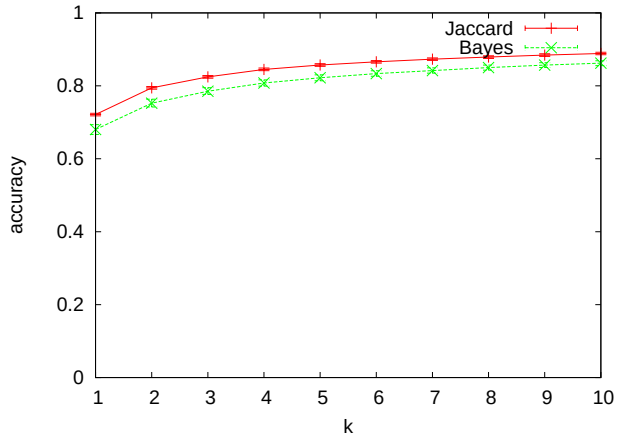

Figure 4: Effect upon accuracy of different values of k.

The result of each experiment is a table of probabilities of class membership for each of the instances in the test set. From this, we determine the *k-identifiability* of each instance. The k-identifiability for an instance is defined as 1 if the actual class of the instance is in the top k of the predicted class list, as ordered by probability estimate (predicted classes with the same estimate are ordered in an arbitrary but fixed manner) or 0 if the actual class is not in the top k. The *k-accuracy* for an experiment is the average of the k-identifiability value for each instance in the test set.

5.2 Classifier Performance

We evaluated the effect of changing each of the independent variables listed above. Unless the variable was the isolated and changing variable, each of the following graphs assumes $k = 1$, a training set size of $t = 4$ (one day of data), a test set size of $s = 4$, $\Delta = 3$ so that the training and test sets are one day apart, and $N = 1000$ sites. We chose random initial sample numbers such that 10 individual experiments were run with all other variables equal — these otherwise identical experiments are the source of the 95% confidence intervals in the graphs. Often the intervals are too small to be observed. Figures 3, 4, 5, and 7 show the results of these experiments.

In Figure 3, we show the effect of varying the training set size. While accuracy increases as more samples are added to the training sets, the rate of increase slows when $t = 4$. Thus, even small training sets give good accuracy, and additional training data give diminishing returns on accuracy. In Figure 5, we show the effect of varying Δ. Larger delays between the training and test sets result in lower accuracy, but the decrease appears to be linear in the delay and tolerably small (a drop from 73% to 63%), even after four weeks of delay. Thus, training data remains useful even after a significant amount of time has passed. This implies that it need not be collected too frequently, increasing the utility of large sets of such data. In Figure 4, the effect of a larger k is shown. Unsurprisingly, allowing the observer more chances to identify a trace allows for higher accuracy. At $k = 10$, the observer can expect to have correctly identified the trace 90% of the time.

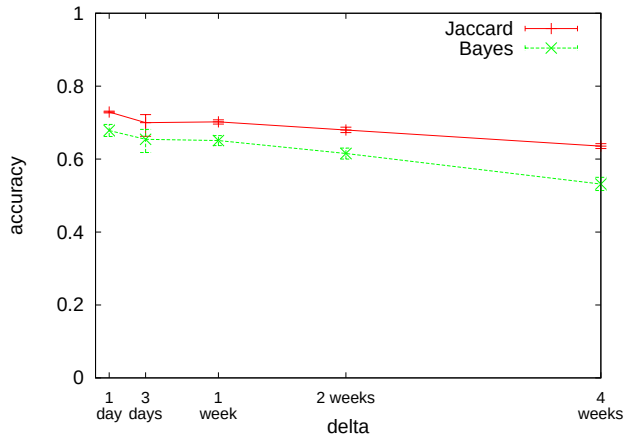

Figure 5: Effect upon accuracy of varying the delta (time between training and testing).

Figure 7 shows the effect of varying the number of sites in the training and test sets. The drop in accuracy appears to follow a relationship of the form:

$$acc = A \log_2 N + B \tag{3}$$

A linear regression analysis is presented in Table 1 which shows that a reasonably close log-linear relationship exists. This relationship implies that the profiling attack has good scalability properties, even as N grows to the size of the Internet.

5.3 Forensics Feasibility

This method of identifying encrypted traffic does not require gathering profile data prior to observing the traffic being analyzed. Such a situation occurs when the profiler is an investigator attempting to identify traces gathered in the past. Figure 6 shows the results of the experiments from Figure 5 with one important difference: the training set occurs after, rather than before, the test set. For otherwise identical parameters, the relative decrease in accuracy $(\frac{original - reversed}{original})$ is less than 3% ($p < 0.01$) across all of the experiments.

method	A	B	R-squared	squared error	F(1,18)	prob(F)
Jaccard	-0.03420	1.0679	0.9374	0.03398	269.6	0.0000
Bayes	-0.03901	1.0678	0.9358	0.03055	262.3	0.0000

Table 1: Regression analysis for $acc = A \log_2 N + B$

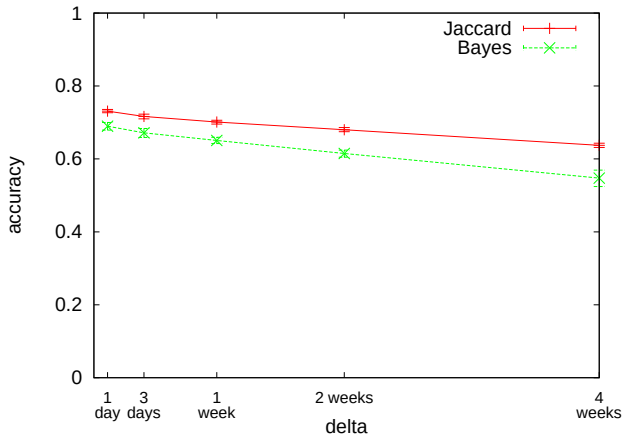

Figure 6: Results after swapping the test and training sets.

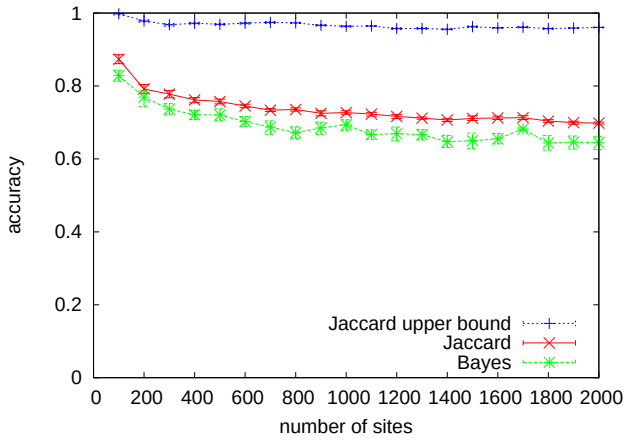

Figure 7: Effect upon accuracy of varying the number of total sites considered. Also shown in the theoretical maximum accuracy that the Jaccard-based classifier can achieve.

5.4 Explaining Performance

As the Jaccard-based classifier only sees occurrences of *(length, direction)* and not counts, some traces will be indistinguishable from others. If we assume no changes in the traces between the training and test sets, the accuracy of the Jaccard-based classifier is bounded by the number of unique traces within each sample. Since sites changes over time, and thus training and test sets can differ, this upper bound is well above the accuracy we observed in practice. Figure 7 shows the fraction of unique samples in the training set and the accuracy of the Jaccard-based classifier on these sets.

To model the source of this uniqueness, we examined the underlying distribution of (packet size, direction) tuples across all instances in the training sets for the 10 experiments corresponding to 2,000 sites. This distribution is shown in Figures 8 and 9. The distribution of per-trace occurrences has an entropy, as defined by:

$$H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{4}$$

of 7.53 bits. We observed that traces have an average of 36.87 unique tuples, drawn without replacement, from such a distribution. If we assume that the appearance of each tuple in each log is independent, then the expected information yielded to the Jaccard-based classifier by a trace is bounded by $7.53 \cdot 36.87 - \log_2 36! \approx 137$ bits. We estimated, via a Monte Carlo procedure, the actual number of bits to be slightly lower ($\approx$ 130), as each tuple can appear only once in a trace. This is far more information than is necessary to distinguish among the number of sites observed, yet as described above, not all sites are unique. We ascribe this discrepancy to a faulty assumption of independence between tuples. Future work could develop a better model to describe identifiability, but as we show below, privacy-conscious system designers will be likely to utilize padding in their designs and render this analysis unnecessary.

5.5 Countermeasure Effectiveness

If an initiator suspects the presence of an observer, he may attempt to obscure his traffic patterns through the use of padding. Here, we examine the effects of static per-packet padding (that is, the padding of each packet with dummy bytes to some predetermined size) on the identification method we propose. We examined four methods of padding in simulation:

- **linear:** Pad each packet to the nearest multiple of 50. This naive method adds a minimal number of bytes to each packet, and reduces the number of distinct sizes by up to a factor of 50.
- **exponential:** Pad each packet to the next largest power of two or the MTU, whichever is smaller. This method significantly reduces the number of distinct sites, to $\lceil \log_2 \text{MTU} \rceil$, while doubling the size of each packet, in the worst case. In practice, we found that this increased packet length by less than 9%, likely because most large packets are already of length equal to the MTU.
- **mice and elephants:** Pad each packet to either 100 or 1500. This method reduces the information available to the classifier even further than exponential padding. All small packets (mostly ACKs) are padded to one size, and all other packets to another. Use of this method comes at a cost of nearly 50% growth in data transmitted.
- **MTU:** Pad each packet to the MTU. This method dramatically increases overall data transmitted (by nearly

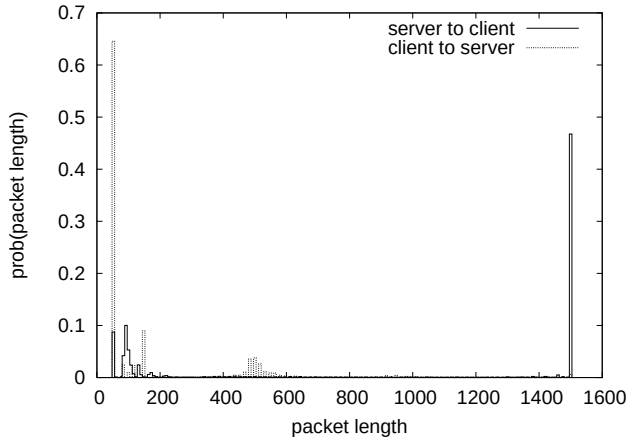

Figure 8: Overall packet length distribution. This distribution is based on every observed packet.

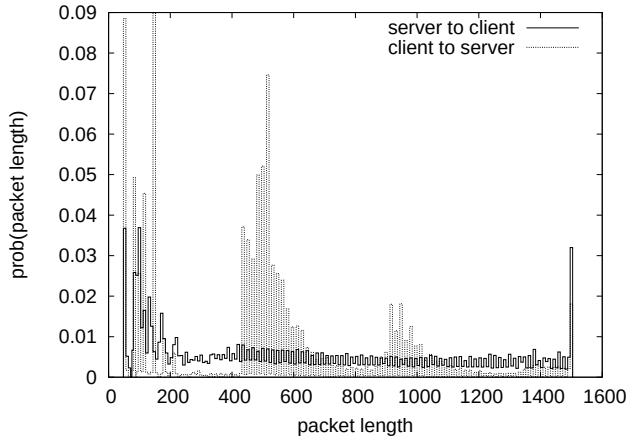

Figure 9: Per-trace occurrence distribution. Packet lengths are included in this distribution at most once per occurrence in each trace. This distribution more accurately reflects the Jaccard-based classifier's input than that of Figure 8.

150%) but renders all packets indistinguishable on the basis of packet length.

We assume the observer is able to determine the padding method being utilized, and can adjust his training sets by padding them in the same fashion. Thus, we evaluate accuracy on the basis of training and test sets that have been padded in identical fashions.

In Figure 10, we show each method's effect upon accuracy. (Recall that k-accuracy is defined in Section 5.1.) Table 2 lists each method's effect upon accuracy as well as the relative number of bytes transmitted. The Bayes-based classifier utilizes packet counts as well as packet size, and thus is better able to discriminate among instances with identical *(direction, length)* attributes but differing counts. The Bayes-based classifier retains enough accuracy, even under the MTU padding method, to be of concern to a privacy-conscious user. Based upon this result, we believe that designers of anonymous communications systems must utilize strong per-packet padding to preserve user privacy.

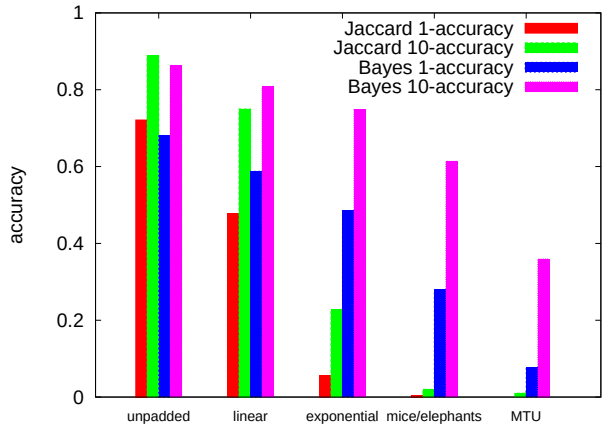

Figure 10: The effects of per-packet padding upon accuracy.

6. DISCUSSION

The main implication of this study is clear: encryption is not enough to protect user privacy. In most low-latency anonymity system designs, an encrypted connection to a proxy (or proxy network) is the basis for the privacy properties of the system. As our study shows, when an observer can utilize external knowledge, such as a library of trace profiles and knowledge of probable user behavior, the content of the data on the connection can be inferred.

Thus we offer the following advice to designers of low-latency systems: Pad packets entering the system to one of a small number of sizes. A system with this property will greatly reduce the amount of available information to an observer, and correspondingly reduce their ability to identify encrypted streams. For example, the current version of the Tor utilizes fixed size packets within its network of proxies. We conjecture this approach is insufficient to counter a traffic analysis attack. A preliminary examination of traces sent through the Tor system leads us to believe that we can discern underlying packet sizes at a finer level of granularity than this fixed size. We believe this discernment can be achieved by grouping packets based an interarrival time threshold, as Tor does not introduce deliberate delays to hide this information.

Building a library of profiles of encrypted HTTP traffic is a reasonable activity for both research and law enforcement purposes. Many forensic investigators will lack the time and tools to gather such data. We have shown the collecting such data is straightforward. Further, we have shown the such data can be used to build an identification system based solely upon packet size — such profiles can be collected from anywhere on the Internet and will not differ for most sites. It is certainly feasible to build software that will enable researchers or investigators to collaborate and create a profile of the entire Internet in a distributed fashion. This library would be similar to a distributed version of the National Software Reference Library[9] which maintains forensic hashes of many commercial applications and operating systems. Given the log-linear scaling of our clas-

[9] `http://www.nsrl.nist.gov/`

Padding	Jaccard 1-accuracy	Jaccard 10-accuracy	Bayes 1-accuracy	Bayes 10-accuracy	Data transmitted
none	0.721	0.889	0.680	0.862	1.000
linear	0.477	0.750	0.588	0.808	1.034
exponential	0.056	0.228	0.485	0.748	1.089
mice / elephants	0.003	0.020	0.279	0.614	1.478
MTU	0.001	0.010	0.077	0.359	2.453

Table 2: Per-packet padding and its effects upon accuracy and amount of data transmitted. The rightmost column shows the amount of data transmitted when using the specified padding method, relative to no padding.

sification method, we expect that such a library would have signification forensic value.

On average, our unoptimized measurement infrastructure retrieved one site every six seconds. Thus, we can collect traces of at least 600 sites an hour from just one computer, or over 100,000 sites in a week. Having more computers doing collection will increase the rate linearly, subject to bandwidth constraints.

The Netcraft survey[10] shows that are approximately 38 million active web sites on the Internet as of February 2006. If 400 volunteers profiled 600 sites an hour, then the entire Internet could be updated in a distributed library once a week. A greater number of volunteers would reduce the amount of work each has to do, or increase the accuracy of the profiles by updating the profiles more frequently. The mean size of an naive Bayes profile in our existing experiment is about 350 bytes; an archive of Internet top-level pages can thus be stored with less than 13GB. Of course, each Internet site consists of many pages. We conjecture that this problem can be addressed because many pages are based on common templates. For example, all Google web searches result in the same layout. The web site of the NY Times looks the same for our observer even though the content changes quite frequently. In general, we expect that sites with many, dynamically generate pages will follow templates out of a necessity for manageable administration; sites that with static content that is different on every page are likely to be updated only infrequently, and therefore require infrequent profiling.

7. CONCLUSION AND FUTURE WORK

We have shown that an observer can infer the contents of encrypted HTTP streams using a library of profiles collected before or after the encrypted stream. These profiles do not require much training data to construct, degrade slowly over time, and are compact. These properties make them useful to the forensic investigator and worrisome to privacy-conscious users. We believe that this study provides some guidance for protocol designers to prevent this attack in future systems, as well as a method for forensic investigators to gather evidence based upon current systems.

In the future, we plan to extend this work in several ways. First, we will examine the effect of introducing timing information into the profile. While this addition has the potential to significantly improve accuracy, it comes at a potentially high cost: the loss of location-neutrality in gathering the profiles. Second, we will examine the efficacy of more complex classification methods. Both of the methods we utilized in this study assume independence among the attributes in the traces. We have given evidence that this independence assumption is flawed, and we expect that classifiers which model inter-attribute dependence will have improved accuracy. Third, we will evaluate this identification method on larger data sets and with a more realistic network substrate. While we expect performance to remain consistent, it is possible that real-world systems may have unanticipated effects upon our method's effectiveness. Fourth, we will examine other padding schemes. We suspect that non-deterministic padding of packets will have a lower packet size overhead than an equivalently strong deterministic scheme. Finally, we will examine more carefully the effects of padding, fragmentation, and packet delays upon our classifier. We suspect these techniques will interact with TCP/IP implementations is non-obvious ways. The designers of low-latency anonymity systems are reluctant to unnecessarily manipulate the traffic stream, for fear of introducing excess latency into their systems. We would like to verify that reasonable security properties can still be attained in the face of more advanced attacks, and we will model the performance penalty that must be incurred to mitigate such attacks.

[10] `http://www.netcraft.com`

8. REFERENCES

[1] George Dean Bissias, Marc Liberatore, and Brian Neil Levine. Privacy vulnerabilities in encrypted HTTP streams. In *Proceedings of the Privacy Enhancing Technologies workshop (PET 2005)*, May 2005.

[2] Eoghan Casey. *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*. Elsevier, 2nd edition, 2004.

[3] Eoghan Casey. Network traffic as a source of evidence: tool strengths, weaknesses, and future needs. *Journal of Digital Investigation*, 1(1):28–43, 2004.

[4] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Challenges in deploying low-latency anonymity. `http://tor.eff.org/cvs/tor/doc/design-paper/challenges.pdf`.

[6] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[7] Alan O. Freier, Philip Karlton, and Paul C. Kocher. *Secure Socket Layer*. IETF Draft, November 1996. `http://home.netscape.com/eng/ssl3`.

[8] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of

countermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.

[9] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[10] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography (FC '04)*. Springer-Verlag, LNCS 3110, February 2004.

[11] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[12] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Berkeley, California, May 2002.

[13] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[14] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[15] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.

[16] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.