Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier

Dominik Herrmann
University of Regensburg
93040 Regensburg, Germany
dominik.herrmann
@wiwi.uni-r.de

Rolf Wendolsky
JonDos GmbH
Bruderwöhrdstr. 15b
93055 Regensburg, Germany
rw@jondos.de

Hannes Federrath
University of Regensburg
93040 Regensburg, Germany
hannes.federrath
@wiwi.uni-r.de

ABSTRACT

Privacy enhancing technologies like OpenSSL, OpenVPN or Tor establish an encrypted tunnel that enables users to hide content and addresses of requested websites from external observers This protection is endangered by local traffic analysis attacks that allow an external, passive attacker between the PET system and the user to uncover the identity of the requested sites. However, existing proposals for such attacks are not practicable yet.

We present a novel method that applies common text mining techniques to the normalised frequency distribution of observable IP packet sizes. Our classifier correctly identifies up to 97 % of requests on a sample of 775 sites and over 300,000 real-world traffic dumps recorded over a two-month period. It outperforms previously known methods like Jaccard's classifier and Naïve Bayes that neglect packet frequencies altogether or rely on absolute frequency values, respectively. Our method is system-agnostic: it can be used against any PET without alteration. Closed-world results indicate that many popular single-hop and even multi-hop systems like Tor and JonDonym are vulnerable against this general fingerprinting attack. Furthermore, we discuss important real-world issues, namely false alarms and the influence of the browser cache on accuracy.

Categories and Subject Descriptors

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

General Terms

Measurement, Security

Keywords

traffic analysis, low-latency anonymity, forensics, text mining

CCSW'09, November 13, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

The continuously increasing number of Internet access points allows users to go online in public. While this development is generally appreciated, it comes at a price: Users face potentially malicious service providers who can observe their traffic at low cost. Moreover, wireless networks make it particularly easy for other users to eavesdrop on foreign connections.

A number of privacy enhancing technologies (PET) have been devised to address those risks. Apart from single-hop systems like virtual private networks (VPN), simple SSL proxies or OpenSSH tunnels, there are sophisticated multi-hop communication systems like Tor and JonDonym (also referred to as JAP or AN.ON in the past). They enable users to encrypt their traffic in order to protect the contents and destination addresses from observers. In general, this is achieved by establishing an encrypted tunnel from the user's machine to a trusted endpoint.

The protection offered by PETs is endangered by *traffic analysis techniques* (cf. [29] for an overview), though. Attackers may gain information regarding the transmitted contents or the identities of the communication partners by inspecting the encrypted traffic. In this paper, we concentrate on a special kind of traffic analysis techniques that is known as "website fingerprinting". An attacker can learn the identity, i. e. the URLs, of websites that are downloaded over an encrypted tunnel by comparing the observed traffic to a library of previously recorded fingerprints. Website fingerprinting attacks exploit the distinct structure and size of HTML pages and the included scripts, style sheets, images and other media objects.

Recent work [22] has demonstrated that such an attack is feasible against OpenSSH tunnels due to characteristic patterns of encrypted IP packets. However, their results do not seem to be ready for practical usage: even under ideal conditions, their classifiers achieve an accuracy of only 73 %, that means only 73 % of the requested web sites were identified correctly. Furthermore, all previous studies have focused on the protection offered by simple encrypted tunnels only. They have not yet looked at more sophisticated technologies like web mixes or onion routing, emerging techniques that help to improve privacy on the Internet.

There is already a number of traffic analysis attacks specifically tailored for web mixes and onion routing (e. g. [26, 1, 2]). They differ from our approach, though: they exploit certain design and implementation characteristics. Our attack is more general as it is agnostic of internal matters of the attacked PETs, but solely relies on the IP packet size frequency distribution. Thus, it may be applied against any deployed PET with practically no alterations. Of course, its accuracy may still be improved by tuning it for specific PETs.

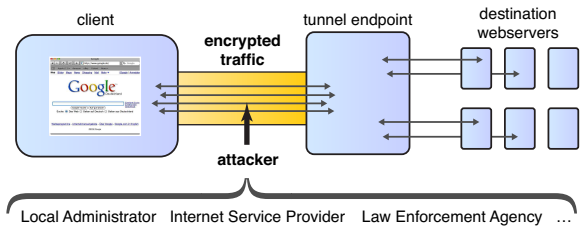

Figure 1: Website fingerprinting scenario and conceivable attackers

Contributions. In this paper, we present a novel website fingerprinting technique based on a Multinomial Naïve-Bayes classifier. According to our empirical evaluation, in a controlled environment our suggested approach is superior to comparable techniques in terms of effectivity and efficiency. We also discuss issues in real-world environments. Furthermore, we present the first comparative survey of the protection offered by a selection of popular PETs against our fingerprinting attack.

The structure of this paper is as follows: In Section 2 we explain our scenario for website fingerprinting and discuss the properties of the attacker. In Section 3 we review related work and contrast our contributions to existing publications. Section 4 describes our research methodology and the construction of our novel website fingerprinting method. We provide the results of various experiments in Section 5. After discussing our findings in Section 6 we conclude in Section 7.

2. SCENARIO

Website fingerprinting methods assume the following scenario: a user wants to protect the information which websites he requests from the Internet against third parties. This can be achieved with systems that utilise privacy enhancing technologies to hide the traffic. Usually, the user will have to install a *dedicated client software* that establishes an encrypted link (tunnel) to a trusted server located on the Internet. This *tunnel endpoint* relays the HTTP requests of one or multiple clients to the various destination web servers. This scenario is illustrated in Figure 1.

We further assume that the attacker employing website fingerprinting

- is able to record the traffic of the victim,
- is located between the victim and the PET system, so that he can retrieve the victim's true IP address from the traffic, and
- is able to identify the victim based on his IP address.

According to the attacker classification in [28] the following attackers have this sort of power: local administrators, the user's ISP, governments and secret services. In case the victim is on a wireless network, an attacker does not need any hardwired infrastructure, a mobile computer is sufficient. Thus, arbitrary users on the wireless network may carry out the attack. We exclude attackers on the user's workstation, who can trivially attack his privacy, and attackers having access to the servers of the PET system, who can launch far more powerful attacks.

The attack consists of two phases: in the *training phase* the attacker creates traffic fingerprints for a large number of sites (or for a small set of interesting sites) and stores them together with the site URLs in a database. In the *testing phase* the attacker records the encrypted traffic of the user, creates fingerprints of small traffic chunks and tries to match them with records in the database.

Summing up, website fingerprinting methods enable a *passive*, *local*, *external* attacker (definitions according to the classification in [9]) to circumvent the security offered by privacy enhancing technologies. This is especially intriguing because most systems have been designed with exactly this minimum attacker model in mind and should provide sufficient protection against this sort of attacker.

3. RELATED WORK

There is a huge number of publications on traffic analysis techniques. In this section we concentrate on attacks on HTTP traffic that aim to uncover the URLs of websites that are transmitted over an encrypted channel.

Mistry [24] and Cheng et al. [7] were among the first to demonstrate how attackers can determine the URLs of websites requested via encrypted SSL connections. They showed that the transmitted data volumes were characteristic for specific websites. Their attacks were constrained to single web servers only, though. Furthermore, the attacker would need access to the server in order to determine the sizes of all HTML files and all objects referenced therein. This attack is not feasible any more with the prevalence of connection pipelining (cf. RFC 2616, section 8.1 [13]) and multiple simultaneous connections as advocated by HTTP 1.1. These optimisations prevent the attacker from observing the size of individual objects during transmission.

Hintz and Sun et al. analyse the encrypted traffic of HTTP proxy servers that use SSL to protect the contents. They show how an attacker can identify websites if he is in possession of a library of their previously recorded "website fingerprints". A website fingerprint is essentially the histogram of the sizes of transferred files that can be observed on the wire when a website is requested. While Hintz [14] demonstrates his attack only for a small number of websites, Sun et al. [30] present evaluation results for a sample of 100,000 websites. With their classifier that is based on Jaccard's coefficient, they are able to correctly identify 75 % of the sites in their sample, at the cost of a false positive rate of 1.5 %. The attacks are reasonably effective, but they cannot be applied to tunnel-based PETs that hide the individual connections – and therefore the file sizes – from outsiders. A common drawback of file-based attacks is that they cannot be applied to VPNs, OpenSSH tunnels or contemporary anonymisation services like Tor.

We are aware of only two publications which are comparable to the work presented in this paper: Bissias et al. [4] and Liberatore and Levine [22] study an improved form of the fingerprinting attack, using 100 and 2,000 websites, respectively. In contrast to earlier approaches, their attack does not rely on the actual file sizes of the transmitted objects. Instead, it is based on patterns to be found in the observed encrypted *IP packets*. Both publications analyse the protection offered by an OpenSSH tunnel. Bissias et al. apply the rather crude metric of the correlation coefficient to the packet size and packet inter-arrival time traces observed during the download of a website. The effectivity of their approach is rather limited: They can only identify 20 % of the web sites and have to guess up to three times for the accuracy to approach 100 %.

In [22] Liberatore and Levine neglect timing information and the order of packets. They compare packet size histograms with the Jaccard coefficient and a Naïve Bayes classifier with kernel density estimation, which will be reviewed in Sections 4.4.1 and 4.4.2 of this paper. Given a sample of 1,000 sites and 4 training instances per site, they are able to identify 73 % of the instances. They also publish the effect of various padding techniques on the classification accuracy in their publication and demonstrate that the attack can be foiled by padding IP packets. In contrast to previous pub-

lications, their packet-based fingerprinting may be applied to all sorts of encrypted tunneling services deployed today.

Apart from Liberatore's and Levine's padding simulations, there has been little research regarding protective measures against packet-based fingerprinting so far. We are aware of the efforts of Kiraly et al., though. They describe *Traffic Flow Confidentiality* [17], an extension to the IPsec code of the Linux kernel offering sophisticated padding and packet clocking schemes. However, they have not yet provided a thorough evaluation of their schemes against website fingerprinting attacks. Using a data set of 50 pages, Wright et al. demonstrate that their *Traffic Morphing* technique [36] is able to thwart statistical traffic analysis algorithms by efficiently modifying traffic of a website in a way so that it looks like another one. As of this writing, these countermeasures have only been evaluated in controlled environments, though. They have still to be integrated into popular PET systems. We haven't included them in our study, because in this paper we focus on an evaluation of technologies that are already widely deployed today.

Another related area of research deals with privacy issues in anonymized network traces: an attacker may be able to determine which websites were accessed in publicly available NetFlows by searching them for characteristic website fingerprints. Recent work [8, 20] suggests that such attacks are difficult to carry out in reality due to several issues, e. g. noise intruced by the browser cache, interleaved connections, the inability to separate individual page downloads and problems with closed-world evaluations of accuracy. As of now, these issues are only understood for website fingerprinting based on anonymized NetFlow traces, though, which is different from website fingerprinting for encrypted tunnels based on IP packet size distributions. As some of their findings are certainly of relevance for our attack, we discuss our results in the light of their research in Section 6.

Although our approach is based on the work of Liberatore and Levine, it differs in several ways. First, we apply a novel fingerprinting technique and evaluate it in terms of accuracy and performance. Secondly, we compare the effectivity of two existing fingerprinting methods with our technique using a common data set to guarantee comparability. Finally, while related publications concentrate on a singular technique, i. e. OpenSSH tunnels, we provide evaluation results for a number of popular PETs.

4. METHODOLOGY

In this section we will describe the evaluated systems and fingerprinting techniques as well as the construction of our novel technique based on the Multinomial Naïve Bayes classifier.

4.1 Analysed Systems

In contrast to previous works, each of which focused on a single system for encrypted tunnels, we have evaluated the protection offered by six systems utilising various privacy enhancing technologies against website fingerprinting. All of the systems under consideration have distinct properties, which motivates an analysis of differences and similarities.

The analysed systems fall into two categories: single-hop and multi-hop systems. The first group constitutes systems that consist of a single proxy server that relays traffic of a single client or multiple clients to the destination web servers. The connection between the web browser and the proxy server is encrypted. The following single-hop systems are analysed in our evaluation:

OpenSSH can be configured with *dynamic forwarding* to offer a local SOCKS proxy (cf. RFC 1928 [21]) used by the browser.[1] The SSH Connection Layer (cf. RFC 4254 [37]) supports multiplexing of connections and flow control, which facilitates fast web surfing.

OpenVPN provides a virtual private network which transparently relays all traffic. We configured OpenVPN[2] in *routing mode* to encapsulate raw IP packets via OpenSSL within UDP packets.

CiscoVPN establishes a virtual private network with a remote VPN concentrator. In contrast to OpenVPN, the evaluated Cisco VPN establishes an IPsec tunnel via ESP. The client has been configured for NAT traversal via NAT-T (cf. RFC 2948 [15]), which wraps ESP traffic in UDP packets.

Stunnel can be used to establish an encrypted connection to a remote proxy server (in our case *tinyproxy*[3]). In contrast to the other systems, Stunnel[4] does not establish a long-lasting tunnel. Instead, each TCP connection of the browser to the proxy server is relayed individually via OpenSSL, causing TCP and TLS handshakes for each connection to the proxy server.

In future work we also plan to analyse WiFi connections encrypted on the link layer (WPA), which belong to this category as well.

Single-hop systems have a common drawback: users must trust the service provider because he sees the unencrypted data that is sent to the destination servers. This is different for multi-hop systems, which relay users' traffic over several hosts before sending it to the web server. Clients encrypt their traffic multiple times to ensure that only the last node, which does not know the identity of its users, has access to the decrypted data. We have analysed two popular systems, which belong to this category, Tor[5] and JonDonym[6]:

Tor is based on the idea of Onion Routing [10], i. e. the Tor client wraps the data packets in multiple layers of encryption, which are "peeled off" as packets are relayed over multiple *onion routers*.

JonDonym (formerly known as JAP and AN.ON) is an implementation of *web mixes* [3] and adapts the idea of Chaum's mix networks [6] for web traffic. An important difference between JonDonym and Tor is their network topology: while the Tor client constructs short-lived circuits by choosing (usually) three Tor routers, a user of JonDonym selects a static cascade of (usually) three mixes over which he intends to relay his web traffic.

After a preliminary study we decided to not include the multi-hop system I2P[7] in our study because its performance and stability was at this time not reliable enough to generate enough samples. Moreover, I2P is not primarily designed for relaying web traffic, but rather for communication within the I2P network.

Note that while the analysed single-hop systems relay all traffic with only minor interference, the multi-hop systems operate on fixed-length messages, which may considerably affect the shape of the network traffic.

[1] Homepage: http://www.openssh.org/

[2] Homepage: http://www.openvpn.net/

[3] Homepage: https://www.banu.com/tinyproxy/

[4] Homepage: http://www.stunnel.org/

[5] Homepage: http://www.torproject.org/

[6] Homepage: http://www.jondonym.de/

[7] Homepage: http://www.i2p2.de/

4.2 Research Assumptions

Comparable website fingerprinting studies in [4, 22, 30] have modeled the real world using a set of simplifying assumptions. Although it has been argued that a determined attacker may very well be able so satisfy those assumptions, recent work on NetFlows indicates that some of them are challenging [8, 20]. For the sake of comparability of results, we based our experimental setup for the main part of this paper on the same assumptions, though:

1. The attacker knows the PET the victim is using and may use it himself to create a database of website fingerprints. To this end, the attacker may e. g. inspect the format of the encrypted payload or the TCP ports used for communication.

2. The attacker knows all the pages the victim is going to retrieve and creates fingerprints for them (i. e. the classifier will never encounter a website for which it has not been trained). Recent research has shown that the relative high accuracies achieved in such a closed-world scenario cannot be reproduced in real environments. While we stick to this assumption in the main part of the paper, we will discuss its impact on our technique in Section 6.

3. The attacker is able to build fingerprints using a similar Internet access like the victim (ruling out network-level differences).

4. The attacker knows the victim's browser and its configuration and is able to use it to create his fingerprints database (ruling out any browser-specific transmission differences).

5. The victim's browser is configured appropriately (no caching, no prefetching, not querying for software updates). While this configuration deviates from the default, it is (a) necessary for comparability with previous work, and (b) follows the privacy guidelines of the developers of the PETs Tor and JonDonym (cf. the JonDoFox[8] browser).[9] In Section 6 we will show that the browser cache has only a moderate impact on the accuracy in our sample.

6. The attacker can extract all packets belonging to an individual website from the victim's traffic, i. e. the victim is requesting pages sequentially due to think times between requests and there are no background downloads or overlapping transmissions. While previous work has shown that it may be difficult to extract individual web sessions from network traces in reality, this assumption improves reproducibility and comparability of our experiments.

We plan to relax the aforementioned assumptions and analyse their influence on the effectivity of the attack in future work. In Section 6 we will revisit some of them and provide first results retrieved from appropriate modifications of our setup.

4.3 Modelling the Classification Problem

The website fingerprinting scenario outlined in Section 2 can be expressed as a data mining classification problem. A classification problem is a supervised induction task, in which the *training instances* are labeled with class information. The goal is to accurately predict the class of *test instances*. Given an unlabeled traffic dump, i. e. a test instance $\vec{x}$ of a victim, the attacker wants to find the corresponding URL, i. e. its class c_i, from the set of all candidate classes C. To this end, the attacker trains the classifier using n labeled training instances $I_{\text{train}} = \{(\vec{x}_1, c_1), \ldots, (\vec{x}_n, c_n)\} \forall c_i \in C$. The next paragraph describes the attribute vectors we use for instance representation and motivates the use of text mining techniques.

[8] https://www.jondos.de/en/jondofox

[9] Actually, Torbutton users may choose between disabling caching completely and enabling memory-only caching.

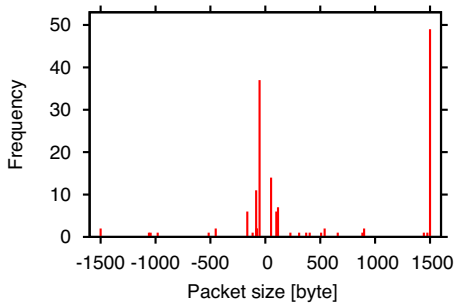

Figure 2: Instances are represented by the frequency distribution of IP packet sizes as observed during the download of a website (packets *sent from client* to server denoted by negative sign). Figure shows traffic observed during download of a website using OpenSSH.

Attributes. A crucial step in data mining is feature extraction and representation (cf. [35]). Based on the results of previous research and our preliminary studies, we decided to operate on the frequency distribution of the IP packet sizes that an attacker can observe during the transmission of a website. While – in contrast to [4] – we neglect information regarding packet order and timing, we do take into account the flow direction (upload/download) of the packets. We denote the flow direction by assigning a positive sign to all packets that the client received and a negative sign to packets that the client sent. Thus, each instance is represented by a multiset $x = x_1^{f_{x_1}}, x_2^{f_{x_2}}, \ldots, x_m^{f_{x_m}}$ containing packet sizes x_j and their respective occurrence frequencies f_{x_j}. From the multisets, attribute vectors can be easily obtained: $\vec{x} = \mathbf{f} = (f_{x_1}, f_{x_2}, \ldots, f_{x_m})$, $f_{x_j} \in \mathbb{N}_0$ for all existing packet sizes m in the sample. Figure 2 visualises an arbitrary instance from our sample as a histogram. Note that our instances closely resemble the typical document representation in the domain of text mining, where instances are represented by term frequency vectors.

4.4 Known Website Fingerprinting Techniques

In this section we will shortly describe the previously published website fingerprinting techniques we applied to our sample.

4.4.1 Jaccard's Classifier

Jaccard's coefficient is a similarity metric for sets [31], which is often used for unsupervised learning tasks. It can be applied to classification problems as well, though, and it has already been successfully utilised for the purpose of website fingerprinting [22, 30]. In order to obtain sets from instances represented as multisets, packet frequencies are neglected and only the binary information regarding the occurrence of the various packet sizes is considered. Jaccard's coefficient of two instances represented as sets A and B is then given by $s_{AB} = \frac{|A \cap B|}{|A \cup B|}$, $s_{AB} \in [0;1]$.

Work by Liberatore and Levine shows how to obtain a classifier based on this similarity metric (which we call *Jaccard's classifier*) that provides an estimate of class membership probability [22]. In their study of OpenSSH tunnels, Jaccard's classifier achieves over 60 % accuracy under certain research assumptions (cf. Section 4.2). Apparently, OpenSSH does not hide the characteristic packet sizes

created during the download of many sites. Their results suggest that actual packet frequencies are of minor importance for the classification.

4.4.2 Naïve Bayes Classifier with Kernel Density Estimation

The Naïve Bayes (NB) Classifier is a widely used supervised learning method whose characteristics have been researched extensively. It naïvely assumes independence of attributes, which is often not the case for real-world problems. Nevertheless, it has been applied to many classsification problems with great success. Of particular interest is its application to traffic analysis problems (cf. [12, 38, 25, 34]) and to website fingerprinting [22] in previous works. A detailed description of this classifier can be found in the literature, e. g. in [35, 16].

In contrast to Jaccard's classifier, NB operates directly on multiset instances, i. e. it takes into account the frequencies of the various packet sizes. Each packet size is treated as a totally independent attribute with the occurrence frequency as attribute value. With simulations Liberatore and Levine find that NB will be more robust than Jaccard's classifier, if IP packets are padded [22]. They show that it is less effective in the absence of padding, though.

By naïvely assuming that all attribute values are the results of independent random variables X_j, the NB classifier estimates the probability that an unlabeled test instance represented as attribute vector $\mathbf{f}$ belongs to some class c_i. This is repeated for all classes in order to find the class with the highest probability. The probabilities are calculated by estimating for each attribute x_j the probability that it occurs f_{x_j} times in an instance belonging to class c_i and multiplying the probabilities of all attributes:

$$p(c_i|\mathbf{f}) = \frac{p(c_i)\,p(\mathbf{f}|c_i)}{p(\mathbf{f})} \sim p(c_i)\prod_{j=1}^{n} p(X_{x_j} = f_{x_j}|c_i)$$

$p(X_j = f_{x_j}|c_i)$ is usually estimated by assuming that X_j is normally distributed and estimating a Gaussian distribution from the training data. Kernel density estimation improves on this concept by estimating the probabilities $p(X_{x_j} = f_{x_j}|c_i)$ via an aggregation of a large set of Gaussian kernels.

While this method has shown very effective in practice, it introduces additional computational complexity. John and Langley deduce that with k attributes and n training instances, time complexity of training the classifier is $O(nk)$, while for testing of m instances it is $O(mnk)$ [16] . Another drawback of the NB classifier is that it attaches particular importance to the exact number of occurrences of packet sizes. Given a very small number of (homogenous) training instances, the variance of the estimated Gaussian kernel may become very small. In this case NB may already fail to classify a test instance correctly when occurrence frequencies of its packet sizes deviate from the distribution in the training instances only slightly.

4.5 Our Novel Website Fingerprinting Method

We will now describe how to build a Multinomial Naïve Bayes classifier applying classical text mining techniques to traffic fingerprints of websites.

4.5.1 Multinomial Naïve Bayes (MNB) Classifier

The Multinomial Naïve Bayes (MNB) classifier is a classical technique used in the domain of text mining. It has been used for semi-automated document classification tasks such as the detection of spam mails. Documents are represented as term frequency vectors which are similar to our packet size frequency vectors. Although the construction of the MNB classifier relies on naïve assumptions[10], this method has proven very accurate in practice. Due to space restrictions we cannot describe all details here, a detailed explanation of MNB can be found in a recent text book, e. g. [23, p. 258].

There is an important difference between NB and MNB. The NB classifier presented in Section 4.4.2 estimates probability of class membership using Gaussian kernels, thus choosing the class, whose occurrence frequencies of the various packet sizes match best with the observed values in the test instance. In contrast, MNB operates on the frequency distribution of all packet sizes at once, i. e. it compares the silhouette of the histogram of the test instance (cf. Figure 2) with the aggregated histogram of all training instances per class. Therefore, the calculation of the conditionals $P(\mathbf{f}|c_i)$ is different from the NB classifier:

$$P(\mathbf{f}|c_i) \sim \prod_{j=1}^{m} P(X = x_j|c_i)^{f_{x_j}}$$

Given m unique packet sizes present within all training instances of all classes, the resulting probability is proportional[11] to the product of $P(X = x_j|c_i)$, which is the probability that a certain packet size x_j is drawn from the aggregated multiset of all packet size frequencies of the training instances of class c_i. The individual conditional probabilities will contribute f_{x_j} times to the result, where f_{x_j} is the number of occurrences of packet size x_j in the unlabeled test instance.

4.5.2 Application of Text Mining Transformations

Researchers have come up with a set of optimisations that address unwanted properties that impair the accuracy of the MNB classifier (cf. [35]). Due to their importance for practical text mining problems it is interesting to investigate their utility for website fingerprinting.

TF Transformation. Using the raw occurrence frequencies the decisions of the MNB classifier are biased towards classes which contain many packets and/or packets with high frequencies. This is very relevant to traffic fingerprints of typical websites, which are dominated by TCP acknowledgements and packets with the size of the MTU. In text mining this problem is addressed by a sublinear transformation of the frequencies: $\mathrm{f}^*_{x_j} = \log(1 + \mathrm{f}_{x_j})$. This is referred to as *term frequency (TF) transformation.*

IDF Transformation. The MNB classifier treats all attributes (packet sizes) equally, neglecting their relevance. In fact, some packet sizes (e. g. with the size of the MTU) are part of every instance and do not confer much information about the class. This is similar to the classification of text documents, where this problem is alleviated using the *inverse document frequency (IDF) transformation*. Given n training instances the occurrence frequencies f_{x_j} are transformed using the *document frequency* df_x, i. e. the number of instances that contain term x: $f^*_{x_j} = f_{x_j} \cdot \log \frac{n}{\mathrm{df}_{x_j}}$.

The application of both of the aforementioned transformations is referred to as *TF-IDF transformation*.

Cosine Normalisation. Although the length of a text seems to be a promising attribute for classification of documents, results from empirical research have shown that the accuracy of the MNB classifier can be improved by normalising the lengths of all doc-

[10] Positional independence assumption (the order of terms is neglected) and conditional independence assumption (terms occur independently from each other).

[11] It is only proportional and not equivalent to the product because for clarity reasons we have left out the *multinomial factor*, which is constant for all classes and therefore does not influence the decision of the classifier.

uments. This is achieved by applying cosine normalisation to the attribute vectors, i. e. the transformed frequencies are divided by the Euclidean length of the raw vectors: $f_{x_j}^{\text{norm}} = \frac{f_{x_j}^*}{\left\|(f_{x_1}^*,\ldots,f_{x_m}^*)\right\|}$.

5. EVALUATION

In the following sections, we show that the application of selected text mining transformations can improve the accuracy of the MNB classifier so much that it outperforms the other classifiers against OpenSSH tunnels. Furthermore, we show the high robustness of the classifier and find that it is able to identify more than 89 % of instances even when only a single training instance is available. With increasing age of the recorded fingerprints the results drop, but only rather slowly. Finally, evaluating the protection offered by various privacy enhancing technologies, we find that users of typical single-hop systems fall prey to the website fingerprinting attack, while the multi-hop systems offer some protection.

The accuracy of the classifiers were evaluated against real-world traffic dumps. All evaluations were carried out with Weka, an open-source data mining toolkit by the authors of [35]. We integrated our own Java implementation of Jaccard's classifier which conforms to the description in [22] into Weka, used Weka's *NaïveBayes* classifier with Gaussian kernel density estimation and its *MultinomialNaïveBayes* classifier in conjunction with the *StringToWordVector* filter to implement our MNB classifier. All experiments were carried out using Weka's *Experimenter* module using the *FilteredClassifier* that takes care that the classifiers do not learn anything about the test instances during training.

5.1 Data Collection and Sampling

The evaluation took place on real-world traffic dumps written by a proxy server with a typical configuration. As we were the first to analyse multiple privacy enhancing technologies, we could not make use of existing traffic dumps available from previous studies. On the other hand, we could not get data from real anonymisation systems, as most of them do not log any data. Instead, we accessed the log files of a medium-range proxy server used by approximately 50 schools serving about 130,000 HTTP requests per day. Only existing data was used and no data was created just for the purpose of this analysis, as this would normally need the permission of all users and is therefore very complex [12]. We retrieved the 2,000 most accessed DNS domain names from the log files between January 2007 and September 2007 and inspected them manually for approval. Most notably, we removed all URLs with domains whose servers did not return a website for humans, e. g. update notification pages and ad banner servers. After data cleansing 775 domain names remained in the sample.

In order to get realistic traffic dumps, we set up a Linux client machine, installed all the client programs for the analysed systems mentioned in Section 4.1 and connected it directly to the Internet. We created a Ruby script *AutoFox.rb*, which is based on *firewatir*[13] and the *JavaScript Shell*[14], to automatically retrieve the sites with *Firefox 2.0* [15]. We configured the browser according to the research assumptions (cf. Section 4.2) and adjusted the proxy settings to relay all web traffic over the system under consideration.

Before each request AutoFox started an instance of *tcpdump* on the client machine with an appropriate capture filter that gathered only the IP headers of the encrypted packets. Once the download of the site was finished, the traffic dump was collected for later analysis and, after a small pause, the script proceeded with the next site. If the download was not finished within 90 seconds it was aborted. After all 775 URLs had been visited, the script restarted the browser and started over from the beginning. We left *AutoFox* running several days in a row for each of the analysed systems. While we created over 73 GB of HTTP traffic between 2008-01-08 and 2008-03-06 with this setup, the (uncompressed) 300,733 tcpdump log files containing just the IP headers amounted to 14 GB only. We extracted packet size and direction of each packet from the tcpdump logs, flagged packets sent from the client to the server with a negative sign and stored them in a MySQL database. The anonymised traces used for our experiments are available at the authors' website.[16]

System	N_{total}	N_{total}/hour	$\overline{p}$
CiscoVPN	29,770	562	527
OpenSSH	131,960	498	497
Stunnel	21,154	480	1,491
OpenVPN	75,724	476	535
JonDonym	33,615	320	504
Tor	8,510	85	613

Table 1: Total number of instances per system, download speed and average number of packets per instance.

Table 1 provides an overview of the samples. The systems are very different in terms of *download speed* N_{total}/hour and *average number of packets per instance* $\overline{p}$. Given the results of an empirical study regarding their performance [33], the slow download speeds offered by the multi-hop systems Tor and JonDonym are not surprising. While download speeds are of little importance for website fingerprinting based on IP packet sizes, differences in $\overline{p}$ could influence the accuracy of the classifier a lot.

Due to large variations in Tor's performance we decided not to clock the downloads statically, but to download sites as fast as possible. Therefore, the number of instances per site and day varies between 2 (Tor) and 17 (CiscoVPN). We avoided *resubstitution errors* during classification by ensuring that training and test instances were drawn from different parts of the sample separated by Δ_t days. For each experiment we sampled a certain number of training and test instances from the dataset so that the following conditions were met:

- using a pseudo-random number generator with seed s_i, for each site n_{train} training instances were drawn randomly from a contiguous 48-hour time window starting at a random time $t_{\text{trainstart}}$ and ending at $t_{\text{trainstart}} + 48\text{h}$,
- for each site n_{test} test instances were drawn from another 48-hour window starting at $t_{\text{trainstart}} + 48\text{h} + \Delta_t$.

This methodology ensures that training and test sets are *stratified*, which is an important precondition for consistency and relevance of the results (cf. [35]). For each experiment we created 25 samples using 25 different seeds s_i to ensure that the whole dataset was utilised. All experiments were repeated for all samples.

Statistical Significance of Results. We used the *corrected resampled paired t-test* [35, p. 157] to compare various settings using a significance level of $\alpha = 0.05$. We will indicate p-values smaller than α, i. e. the null hypothesis that two results are not different is rejected with probability of error below α, with the term *significant* in italics.

[12] For privacy reasons, we moreover suggested to the proxy operator to stop his logging activities as soon as possible.

[13] Homepage: http://code.google.com/p/firewatir/

[14] Homepage: http://www.croczilla.com/jssh/

[15] Homepage: http://www.mozilla.com

[16] http://www-sec.uni-regensburg.de/website-fingerprinting/.

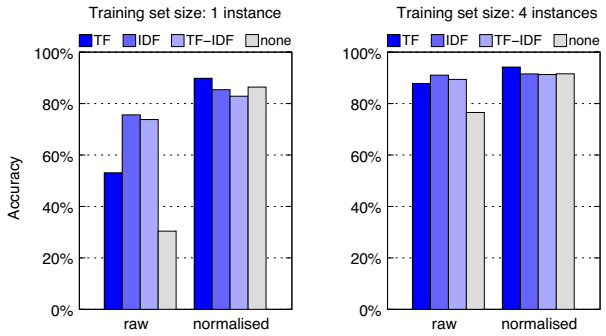

Figure 3: Accuracy of MNB classifier for various transformations

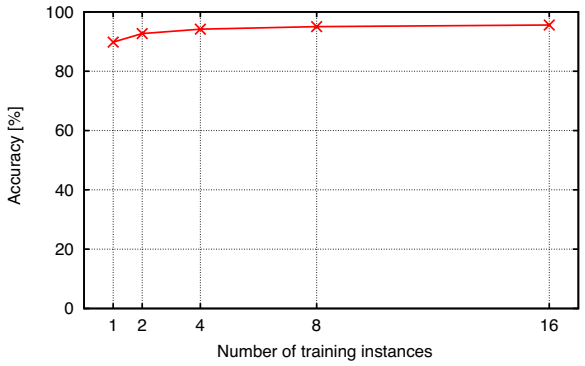

Figure 4: Accuracy for various training set sizes

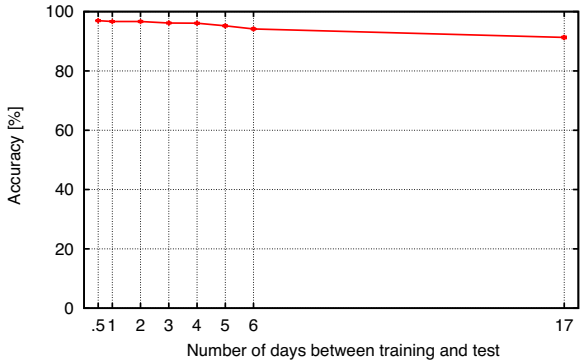

Figure 5: Accuracy for various time spans between training and test

5.2 Performance of the MNB Classifier

The protection of OpenSSH against website fingerprinting has been extensively studied in related work. We will now evaluate the MNB classifier against this system. As will be shown later, the accuracy a classifier achieves against OpenSSH is of relevance for a number of other systems as well.

Accuracy. We use the *accuracy* of a classifier as a measure of its effectivity. This metric indicates the proportion of instances (recorded traffic dumps of individual website downloads) that a classifier attributed to the correct class (URL).

5.2.1 Influence of Transformations

First, we will analyse the influence of the text mining transformations presented in Section 4.5.2 on the accuracy of the classifier. The results of an experiment with $n_{\text{train}} = \{1,4\}$, $n_{\text{test}} = 10$ and $\Delta_t = 6$ days using the OpenSSH dataset are shown in Figure 3. Without normalisation IDF and TF-IDF transformations surpass the TF transformation in terms of accuracy. Accuracies rise *significantly* (cf. Section 5.1 for our definition of *significance*), once attribute vectors are normalised. Normalisation of frequencies that have been processed by the IDF transformation is counterproductive, though. On the OpenSSH dataset the combination of the TF transformation with normalisation achieves best results (94.18 % for 4 training instances).

5.2.2 Size of Training Set

Smaller numbers of required training instances mean less effort for carrying out the attack. Figure 4 shows the results for different values of n_{train} (keeping $n_{\text{test}} = 10$ and $\Delta_t = 6$ days). With *only one* training instance, already 90 % of test instances are classified correctly. Accuracy slowly approaches 96 % for 16 training instances. According to the t-tests, the difference in accuracy is *not significant* between 1 and 2 instances, between 2 and 4 instances and between 4 and 8 instances. Accuracies differ *significantly* for 1 and 4 training instances, though. Therefore, in the following we will use 4 training instances, which seems to be a good compromise between necessary resources and achievable accuracy.

5.2.3 Robustness

Our website fingerprinting attack exploits characteristics in the frequency distribution of the size of IP packets observed during transmission. This distribution is determined by the contents and structure of the HTML page and any embedded elements. Consequently, changes of the content may affect the fingerprint of the site, thus diminishing the accuracy of the classifier. There is extensive research regarding the type and frequency of changes to web sites (cf. for example [32, 18, 19, 27]). A key finding of those research efforts is the distinction between changes in terms of *site structure*, which occurs rather seldom, and *content*, which occurs frequently on many popular websites.

Fortunately, changes to the content affect the frequency distribution of IP packets only moderately, i. e. our website fingerprints remain accurate for a rather long time. Changes to the content are more challenging, though, for fingerprinting attacks that rely on the file sizes of transferred objects (cf. [24, 7, 14, 30] and for attacks on NetFlows [8, 20].

To test this hypothesis, we ran multiple experiments with the MNB classifier on the OpenSSH dataset with $n_{\text{train}} = 4$, $n_{\text{test}} = 10$ for varying values of Δ_t. The results are shown in Figure 5. Even for delays as long as $\Delta_t = 17$ days, 91.3 % of the test instances were classified correctly. The accuracies for delays up to 4 days do not differ *significantly*. We estimated the relationship between Δ_t and accuracy with a regression analysis and found accuracy $\approx 97.11 \cdot e^{-3.735 \cdot 10^{-3} \Delta_t}$.

This robustness property makes the attack more convenient. The attacker will achieve good results, even if he updates his database of fingerprints only once in a while – or a couple of days *after* he recorded the traffic dump of the victim. The MNB classifier is known for its ability to adjust itself to changing characteristics by continuous training with new instances. This property, which is referred to as *concept drift* in the text mining area [23, p. 269], makes the MNB classifier an ideal technique to implement long-term website fingerprinting attacks.

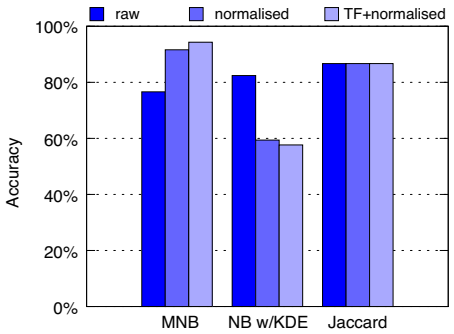

Figure 6: Accuracy of website fingerprinting techniques against OpenSSH using various transformations.

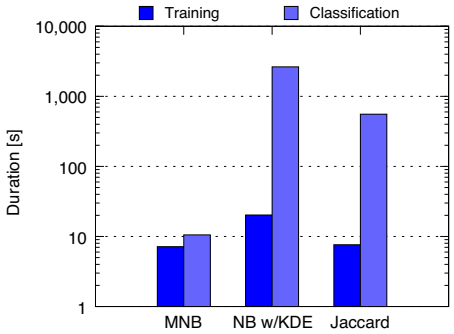

Figure 7: Time needed for training and classification using 4 training and 4 test instances per site.

5.3 Comparison of Website Fingerprinting Methods Against OpenSSH

In this section we will compare existing website fingerprinting techniques to our MNB classifier using the OpenSSH dataset as a common testbed. Therefore, we compare accuracies and time needed for training and testing with parameters $n_{\text{train}} = 4$, $n_{\text{test}} = 4$ and $\Delta_t = 6$ days.

Figure 6 shows the accuracies of the three classifiers. The performance of the Naïve Bayes classifier (82.42 %) is *significantly* worse than Jaccard's classifier (86.68 %), which is in line with the findings in [22]. Note that the accuracy of the former degrades when text mining transformations are applied to packet size frequencies, while the accuracy of the latter is not affected at all as Jaccard's classifier is unaware of the packet frequencies anyway.

While our MNB classifier is outperformed by the other methods when the frequencies are not transformed (76.60 %), it achieves a *significantly* higher accuracy once cosine normalisation is applied to the attribute vectors (91.63 %). Adding the TF transformations adds another *significant* increase to the values (94.31 %).

Having established the significance of this ranking for OpenSSH, one cannot deduce that the MNB classifier outperforms the other classifiers against *any* privacy enhancing technology, of course. Due to space restrictions in this paper, we are unable to present evaluation results of all the fingerprinting techniques against all systems, though.

Figure 7 shows the time needed for training and testing of $4 \cdot 775 = 3,100$ instances. According to our results, the MNB classifier is faster than the other classifiers for both, training and testing. It is able to classify about 294 instances per second. The poor result of NB can be attributed to the high cost of kernel density estimation. As we haven't tuned our implementation of Jaccard's classifier for performance, its performance might still be improved by some degree.

Conclusions. According to the results in this section the MNB classifier outperforms the previously proposed website fingerprinting techniques in terms of effectivity and efficiency against traffic protected by the well-researched OpenSSH tunnels. Its accuracy especially benefits from normalising attribute vectors, which replaces actual occurrence frequencies with relative ones. Furthermore, we have shown that it is qualified for practical purposes: the instance representation remains stable over several days, high accuracies can be achieved with a single training instance only, and it would be even fast enough for real-time traffic analysis for small user groups.

Apparently, MNB is a good compromise between Jaccard's classifier and NB with kernel density estimation. While the former neglects occurrence frequencies altogether and is thus unable to discriminate sites with similar packet sizes, the latter treats occurrence frequencies very seriously – apparently *too* seriously. As NB matches instances based on absolute frequencies, it is misled once packet size frequencies change. In contrast, using cosine normalisation the MNB classifier operates on relative frequencies only, thus focusing on the proportions of the frequencies of the packet sizes. According to our results, this feature representation scheme is the most accurate one against OpenSSH and – as will be shown in the next section – is effective against other privacy enhancing technologies as well.

5.4 Attacking Popular PETs Using the MNB Classifier

Finally, we present a comparison of the accuracy of MNB against the analysed systems presented in Section 4.1. Table 2 summarises the results. For all tested single-hop systems, we achieved an accuracy of more than 94 %, the best against Stunnel, where over 97 % of instances were classified correctly. Stunnel offers *significantly* less protection than all other systems, which is probably due to the additional information gained from the many TCP and TLS handshakes in each traffic dump. At first sight, Tor *significantly* offers the best protection against the attack (accuracy below 3 %).

With accuracies of more than 90 % none of the tested single-hop systems offers sufficient protection against the MNB classifier. From an information-theoretic viewpoint, even the multi-hop systems do not protect perfectly, though: the accuracies found for them are well above the accuracy achievable by randomly guessing the class without any context knowledge ($\frac{1}{775} \approx 0.13\,\%$).

Allowing the classifier to make multiple predictions for each test instance increases its accuracy (cf. [22]). We repeated the experiments with a modified MNB classifier: instead of just predicting the class with the highest probability, it retrieved the top k classes from the list of predicted classes (sorted in descending order by class membership probability). If the actual class was among the list of predicted classes, the test instance was counted as correctly classified, otherwise as incorrectly classified. For $k = 3$ and $k = 10$ the accuracy values for Tor increase to 16.69 % and 22.13 %, respectively, for JonDonym they increase to 31.70 % and 47.53 %.

The different encapsulating mechanisms of the systems are reflected by $n_{\text{sizes\_total}}$, the total number of distinct packet sizes of the aggregated packet size frequency distribution of all instances. According to our results, low values of $n_{\text{sizes\_total}}$ do not correlate with poor accuracy (cf. CiscoVPN), though, and higher values of $n_{\text{sizes\_total}}$ do not automatically lead to higher accuracies (cf. JonDonym vs. Tor).

	Stunnel	OpenSSH	CiscoVPN	OpenVPN	JonDonym	Tor
$n_{\text{sizes\_total}}$	1605	420	108	2898	205	869
Best classifier	TF-N	TF-N	TF-N	TF-N	N	N
Avg. accuracy	97.64 %	96.65 %	96.17 %	94.94 %	19.97 %	2.96 %
Std. deviation	0.16 %	0.22 %	0.24 %	0.30 %	0.47 %	0.22 %

Table 2: Aggregated results of all instances grouped by evaluated system using the MNB classifier. Table shows total number of unique packet sizes in the datasets, the combination of transformations that achieved best results on average (TF-N: TF transformation and cosine normalisation; N: cosine normalisation only) and the average accuracy for 20 samples.

While the classifier benefits from cosine normalisation of attribute vectors against all systems, the TF transformation is supportive for the single-hop systems only. Apparently, the different occurrence frequencies are needed for classification of sites transferred by multi-hop systems. This finding is in line with the results from simulations in [22]: it is due to the repackaging of HTTP requests and padding, which is carried out by both, JonDonym (mix packets: 998 bytes) and Tor (cell size: 512 byte).

But anyway, why do the dumps of Tor and JonDonym contain so many different packet sizes, despite a fixed size of their packet/cell content? The most frequent packet sizes in the Tor traffic dumps are, in descending order, 1500, −52, −638, 52, 638 and 1150 bytes, accounting for 87.6 % of all Tor packets. The remainder is likely caused by packet fragmentation carried out by the operating system and retransmissions, which we frequently observed in Tor's traffic dumps. The application of the TF transformation is counterproductive for Tor's and JonDonym's traffic as it distorts the frequency proportions so that the *noise* becomes more relevant. This is illustrated in Figure 8, which contains the aggregated histograms before and after application of the TF transformation to Tor's aggregated histogram.

Our results also indicate that the protection offered by JonDonym is primarily lower in comparison to Tor, because JonDonym's traffic contains less noise, as indicated by the smaller number of packet sizes and the fact that the 93.6 % of packets have sizes of 1500, −52, −1050, 1050 and 52 bytes. We attribute this difference to the fact that our Tor client established a multitude of connections to various onion routers running differing versions of Tor and operating systems. For JonDonym we preselected a static mix cascade in the JAP client to collect the traffic dumps. This hypothesis can be tested in future work by repeating our experiments on a private Tor network running in a controlled environment.

Of course, the poor accuracy against multi-hop systems does not mean that these systems are not vulnerable. Our system-agnostic method operates naïvely on all observable IP packets neglecting their content or relevance. Specifically tailored feature extraction methods that neglect irrelevant packets and reconstruct the fixed-length messages of JonDonym and Tor will likely be more effective.

6. DISCUSSION

It is not really surprising that the evaluated single-hop systems offer practically no protection against the website fingerprinting attack. These systems encrypt the transferred HTTP messages without structural modifications, thus exposing the deterministic fragmentation of the data sent over TCP streams to the observer. This fragmentation is likely dependent to some degree on the operating system, the type of the Internet connection and the browser and its configuration. We therefore expect that the accuracy of website fingerprinting attacks is degraded in case training and testing instances are not recorded in the same environment. This issue is not relevant to our classifier only, but applies to many other traffic analysis techniques as well. Consequently, results generated from empirical studies based on the assumptions mentioned in Section 4.2, have to be regarded as best case estimations. Future research has to analyse the impact on accuracy that is incurred by relaxing those research assumptions.

Some of the assumptions have only little impact on the accuracy of our classifier. In a preliminary study we tried to classify OpenSSH traffic caused by a browser with *caching enabled*. Using a comparable setup like [8] we configured the browser with a cache size of 2 GB so that no objects would be expunged from the cache due to space restrictions. We then repeated the experiment as outlined in Section 5.1. Results dropped *significantly*, but only slightly from 96.65 % (with caching disabled) to 91.70 %. Although this finding suggests that our technique is only moderately affected by caching, one has to be aware of the fact that the deterministic nature of the page retrieval process used in our setup may not accurately reflect the diversity of possible cache states in reality. This restriction also applies to related studies like [8], though. More sophisticated simulations may provide a more realistic evaluation of browser caches in the future.

Relaxing other assumptions is more problematic, e. g. due to the cost of *false alarms*[17], when not all of the URLs in the pool are known to the attacker. A *false alarm* is a classification for a URL that is in the set of *uninteresting instances*. An ideal classifier would refrain from choosing a class for any of the uninteresting sites. As with spam filters, too many false classifications are prohibitive for a practical system. In order to get an impression about the consequences, we repeated our experiments with an OpenSSH dataset where only 78 of all 775 sites were considered *interesting* for the attacker. A tuned version of the MNB classifier which was optimized for a minimum of *false alarms* caused a false alarm for 1.4 % of the uninteresting instances (correctly not classifying 98.6 % of them). While the performance in terms of false alarms may be acceptable, this comes at the cost of a considerable drop in the accuracy for the interesting sites: only 40 % of the instances of the *interesting sites* were classified correctly, which is far below the accuracies we achieved when false alarms were neglected (94.18 %, cf. Section 5.2.1). Such relatively low accuracy levels have also been observed in related studies like [30, 8]. Accordingly, the efficacy of our website fingerprinting technique in real environments is still limited.

Although website fingerprinting is still not feasible under all circumstances, we advise developers of privacy enhancing technologies to consider the impact and risk of traffic analysis attacks against their systems. Accuracies will likely be further improved in the future by applying more sophisticated pattern recognition and data mining techniques. While already deployed multi-hop systems demonstrate ways of protection, there is no protective single-hop system around that is widely deployed yet.

[17]Please note that the term *false positives* is intentionally not used here, as it is used to convey another meaning in classical data mining.

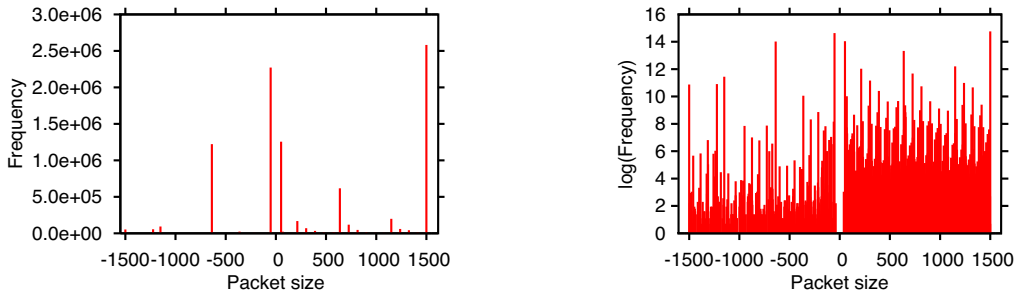

Figure 8: Aggregated packet size histograms for Tor before and after application of TF transformation

7. CONCLUSION

We have shown that a classical text mining classifier, *Multinomial Naïve Bayes*, can be used by an attacker to determine which websites were downloaded over an encrypted connection. Our technique simply operates on the frequency distribution of IP packet sizes, which is robust against small modifications of websites. Furthermore, the packet size distribution can be easily recorded with common network monitoring tools by a passive, external observer. Within several experiments we have demonstrated that our method is robust and succeeds to identify almost all websites from our sample, if one of the popular single-hop systems is employed. Without system-specific tuning its effectivity is still limited against the contemporary multi-hop systems Tor and JonDonym, though, but they still fail to offer perfect protection against this general attack. Attacks specifically tailored for these systems, e. g. by reproducing Tor's cells from the encrypted traffic, will likely offer higher accuracy.

Our classifier outperforms two similar methods, Jaccard's classifier and a Naïve Bayes with kernel density estimation. Our results indicate that its increased performance is mainly due to the normalisation of the packet size frequency vectors, which can be considered a compromise between the properties of the aforementioned methods, i. e. neglecting occurrence frequencies altogether or relying on absolute numbers of packets.

We have also discussed the efficacy of the attack in real world environments, contributing two more findings: reproducing the results of related work, the achievable accuracy drops considerably once the closed-world setup with its restricted set of websites is left behind. In contrast to the findings for website fingerprinting in anonymized NetFlows, our results indicate that enabling the browser cache – which has been turned off in all previous website fingerprinting studies for encrypted tunnels – affects accuracy only moderately.

We haven't seen a practical website fingerprinting attack for privacy enhancing techniques yet. Within controlled experimental environments, the accuracy of such attacks has steadily increased in the last years, though. As future research efforts will likely overcome the remaining limitations, the development *and* implementation of efficient countermeasures becomes an important task for the PET community.

On a minor note, we showed how to apply classical text mining techniques to a contemporary challenge in the privacy enhancing technologies area. There are probably many more possible applications of interest for the security and privacy research community.

8. REFERENCES

[1] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price. Browser-Based Attacks on Tor. In Borisov and Golle [5], pages 184–199.

[2] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Tor. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20, New York, NY, USA, 2007. ACM.

[3] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: a system for anonymous and unobservable Internet access. In *International workshop on Designing privacy enhancing technologies*, pages 115–129, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[4] G. Bissias, M. Liberatore, D. Jensen, and B. N. Levine. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Proc. Privacy Enhancing Technologies Workshop (PET)*, pages 1–11, May 2005.

[5] N. Borisov and P. Golle, editors. *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, volume 4776 of *Lecture Notes in Computer Science*. Springer, 2007.

[6] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[7] H. Cheng and R. Avnur. Traffic Analysis of SSL Encrypted Web Browsing. http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps.

[8] S. Coull, M. Collins, C. Wright, F. Monrose, and M. Reiter. On Web Browsing Privacy in Anonymized NetFlows. In *Proceedings of the 16th USENIX Security Symposium*, Boston, MA, August 2007.

[9] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In Dingledine and Syverson [11], pages 54–68.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[11] R. Dingledine and P. F. Syverson, editors. *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*. Springer, 2003.

[12] J. Erman, A. Mahanti, and M. Arlitt. Internet Traffic Identification using Machine Learning. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, San Francisco, CA, USA, November 2006.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616 Hypertext Transfer Protocol – HTTP/1.1, June 1999.

[14] A. Hintz. Fingerprinting Websites Using Traffic Analysis. In Dingledine and Syverson [11], pages 171–178.

[15] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg. RFC 3948 UDP Encapsulation of IPsec ESP Packets, January 2005.

[16] G. H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In P. Besnard and S. Hanks, editors, *UAI*, pages 338–345. Morgan Kaufmann, 1995.

[17] C. Kiraly, S. Teofili, G. Bianchi, R. L. Cigno, M. Nardelli, and E. Delzeri. Traffic Flow Confidentiality in IPsec: Protocol and Implementation. In *Preproceedings Third IFIP/FIDIS Summer School "The Future of Identity in the Information Society"*, August 2007.

[18] W. Koehler. An analysis of Web page and Web site constancy and permanence. *Journal of the American Society for Information Science*, 50(2):162–180, 1999.

[19] W. Koehler. Web Page Change and Persistence – A Four-Year Longitudinal Study. *Journal of the American Society for Information Science and Technology*, 53(2):162–171, 2002.

[20] D. Koukis, S. Antonatos, and K. G. Anagnostakis. On the privacy risks of publishing anonymized ip network traces. In H. Leitold and E. P. Markatos, editors, *Communications and Multimedia Security*, volume 4237 of *Lecture Notes in Computer Science*, pages 22–32. Springer, 2006.

[21] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928 SOCKS Protocol Version 5, March 1996.

[22] M. Liberatore and B. N. Levine. Inferring the Source of Encrypted HTTP Connections. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263, New York, NY, USA, 2006. ACM Press.

[23] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] S. Mistry and B. Raman. Quantifying Traffic Analysis of Encrypted Web-Browsing. http://bmrc.berkeley.edu/people/shailen/Classes/Security Fall98/paper.ps, December 1998.

[25] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press.

[26] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[27] A. Ntoulas, J. Cho, and C. Olston. What's New on the web? The Evolution of the Web from a Search Engine Perspective. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 1–12, New York, NY, USA, 2004. ACM.

[28] A. Panchenko and L. Pimenidis. Towards Practical Attacker Classification for Risk Analysis in Anonymous Communication. In *Proceedings of Communications and Multimedia Security, 10th IFIP TC-6 TC-11 International Conference, CMS 2006, Heraklion, Crete, Greece, October 19-21, 2006, Proceedings*, volume 4237 of *Lecture Notes in Computer Science*, pages 240–251, 2006.

[29] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *International workshop on Designing privacy enhancing technologies*, pages 10–29, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[30] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical Identification of Encrypted Web Browsing Traffic. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[31] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[32] P. Warren, C. Boldyreff, and M. Munro. The Evolution of Websites. In *IWPC '99: Proceedings of the 7th International Workshop on Program Comprehension*, page 178, Washington, DC, USA, 1999. IEEE Computer Society.

[33] R. Wendolsky, D. Herrmann, and H. Federrath. Performance Comparison of Low-Latency Anonymisation Services from a User Perspective. In Borisov and Golle [5], pages 233–253.

[34] N. Williams, S. Zander, and G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.

[35] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.

[36] C. Wright, S. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250. IEEE, February 2009.

[37] T. Ylonen and C. Lonvick. RFC 4254 The Secure Shell (SSH) Connection Protocol, January 2006.

[38] D. Zuev and A. W. Moore. Traffic Classification Using a Statistical Approach. In C. Dovrolis, editor, *PAM*, volume 3431 of *Lecture Notes in Computer Science*, pages 321–324. Springer, 2005.