Information Leakage Measurement and Prevention in Anonymous Traffic

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Shuai Li

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Nicholas Hopper, Adviser

June 2019

Acknowledgements

I would like to extend much gratitude and appreciation to my advisor Nicholas Hopper, not only for his support, understanding, and advising during my Ph.D research, but also for his being a excellent role model for research, advising, teaching, and family.

I thank Andrew Odlyzko, Stephen McCamant, and Kangjie Lu for being in my committee, as well as their valuable comments and supports for my thesis.

I thank George Karypis, Michael Carl Tschantz, Vern Paxson, Dan Middleton for their generous suggestions and helps, which mean a lot to me.

I thank my lab colleagues with which I had valuable discussions, including Huajun Guo, Mike Schliep, John Geddes, Se Eun Oh, and Max Schuchard. I also thank Stephen's and Kangjie's groups for bringing me many perspectives in my research.

Last but not least, I extremely thank my lifelong friend and partner Wen Xing for her enormous supports, encouragements, and companionship in my pursuit of Ph.D degree, which I am grateful and cherish wholeheartedly. I am also grateful for my cutest Bichon Juan Juan's companion.

for my grandparents Baoyu He and Jinsuo Guo who raised me up

Abstract

The pervasive Internet surveillance and the wide-deployment of Internet censors lead to the need for making traffic anonymous. However, recent studies demonstrate the information leakage in anonymous traffic that can be used to de-anonymize Internet users.

This thesis focuses on how to measure and prevent such information leakage in anonymous traffic. Choosing Tor anonymous networks as the target, the first part of this thesis conducts the first large-scale information leakage measurement in anonymous traffic and discovers that the popular practice of validating WF defenses by accuracy alone is flawed. We make this measurement possible by designing and implementing our website fingerprint density estimation (WeFDE) framework. The second part of this thesis focuses on preventing such information leakage. Specifically, we design two anti-censorship systems which are able to survive traffic analysis and provide unblocked online video watching and social networking.

Contents

List of Tables

List of Figures

1 Introduction

Past decade witnesses the pervasive Internet surveillance and the wide-spread deployment of Internet censors. The Snowden leaks reveal the NSAs massive spying programs targeting individuals, private tech companies, and foreign countries; more than one-third of the worlds population lives under Internet censorship [39]. These issues lead Internet users to the call for being anonymous, so as to protect their online privacy and freedom against surveillance and censorship.

Then comes the popularity of anonymous networks such as Tor, which has 2 million daily active users. The Tor network not only applies encryption upon its traffic, but also it instructs a user to connect through three Tor relays until hitting the destination, so as to keep the user anonymous. The success of the Tor project even goes beyond anonymity: Tor is also able to help users to circumvent Internet censorship! Meanwhile, the blossom of anti-censorship systems came after China blocking Tor by differentiating Tors cipher suite. These systems share one core idea: proxy steganography, which intends to make proxy connections resemble innocent cover protocols. Blocking these anti-censorship systems costs the censor the shutdown of the innocent cover applications, which the censor is considered unwilling to do due to the collateral damage. The success of Tor and the blossom of stenographic proxies seem to resonate the optimism held by Google former CEO Eric Schmidt, who predicted in 2013 that, censorship around the world could end in a decade, and better use of encryption will help people overcome government surveillance [70].

Traffic analysis is the bad news. In face of traffic analysis, encryption is not enough. An example is the success of website fingerprinting (WF) attacks upon Tor networks. By analyzing traffic features such as the packet count and order, WF attackers are able to de-anonymize Tor users with over 90% accuracy in the lab setting. Another example is the vulnerability of stenographic proxies to traffic analysis. Censors are able to detect imitation flaws to recognize the usage of the stenographic proxies; the content mismatch between the stenographic proxy and the cover protocol leads to distinct traffic features, rendering traffic analysis effective in detecting the usage of stenographic proxies. As Susan Landau and Whitfield Diffie put, Traffic analysis, not cryptanalysis, is the backbone of communications intelligence [28].

Thesis Overview. It's the information leakage in traffic that makes traffic analysis possible. However, how to measure and prevent information leakage in anonymous traffic is still unclear. This thesis aims at answering the question. Specifically, it includes:

- Information Leakage Measurement. Choosing Tor networks as the target, we conduct the first large-scale information leakage measurement in anonymous traffic, and

we discover that the popular practice of validating WF defenses by accuracy alone is flawed. We make this measurement possible by designing and implementing our website fingerprint density estimation (WeFDE) framework [52].

- Information Leakage Prevention. To defend the stenographic proxy against traffic analysis, we design and implement two anti-censorship circumvention systems Mailet [53] and Facet [54], which are able to survive traffic analysis and provide unblocked online video watching and social networking.

1.1 Measuring Information Leakage in WF Attacks and Defenses

Website fingerprinting attacks exploit the features of the traffic to de-anonymize Tor networks. By looking at the packet count or order, the attacker can learn which website a Tor user is visiting with over 90% accuracy. In response, website fingerprinting defenses are proposed to reshape the traffic so as to eliminate the information leakage from the traffic features.

We conducts the first large-scale information leakage measurement in anonymous traffic. Such large-scale measurement was lacking in the literature due to two barriers: (a) features are high dimensional, suffering from the curse of the dimensionality; and (b) the complex properties of the features. We design and implement website fingerprint density estimation (WeFDE) framework to overcome above challenges: we use the adaptive kernel method to model the probability distribution of the features, which is able to adaptively deal with discrete, continuous, or even partly discrete and partly continuous features; we measure the pairwise mutual information of features to reduce dimension with the help of DBSCAN clustering algorithm. We apply WeFDE to a comprehensive list of 3043 features (including, to the best of our knowledge, all features in the Tor WF literature) extracted from a 211219 Tor web browsing visits for about 2200 websites.

We discovers that the popular practice of validating WF defenses by accuracy alone is flawed. Most of works on WF defenses evaluate the effect of the defense by classification accuracy: if the accuracy of the classifier is low enough, the defense is believed to be secure with minimal information leakage; one defense is believed to be better than another if it results in lower accuracy. But our information leakage measurement results demonstrate that, low information leakage implies low classification accuracy, but the converse is not necessarily true, thus we argue that validating WF defenses by accuracy alone is flawed! The reasons include: (a) accuracy is classifier-dependent, and (b) accuracy is all-or-nothing.

We also provides the new information-theoretic insights upon WF features. Our study finds that: (a) 45.36% of 183 most informative features are redundant; (b) an individual feature leaks no more than 3.45 bits information in the closed-world setting with 100 websites, which is the maximum leakage we observe in our experiment from the feature of rounded outgoing packet count; (c) download stream, though having more packets than upload stream, leaks less information; (d) a larger world size has little impact on a WF features individual information leakage.

1.2 Preventing Information Leakage in Anti-censorship System Design

Existing stenographic proxies are vulnerable to traffic analysis. The first reason is that some proxies try to imitate unblocked cover protocols. As convincingly mimicking a sophisticated distributed system is an insurmountable challenge, the censor is able to find discrepancies for detecting the proxy. The second reason is content inconsistencies, which can arise when the behavior of a cover protocol depends on the characteristics of the traffic it carries and proxies do not match content to these characteristics. For example, since the FreeWave server tunnels modem traffic instead of voice signal over a VoIP channel, its communication session can be identified by traffic analysis.

In light of these potential problems, finding a single cover protocol to carry arbitrary Internet content seems difficult. However, a recent survey of Chinese users of circumvention tools found most users circumvent the Chinese Great Firewall to use three services: unfiltered search engines such as Google, uncensored social networks such as Facebook and Twitter, and video sharing sites like YouTube and Vine. My research methodology is to serve most circumvention needs through a small set of unobservable transports, so that the information leakage in the traffic of anti-censorship systems is minimized. We design and implement two unobservable transports: Mailet provides safe and unfiltered social website access through email channels with the help of secure computation; Facet enables the users in censored regime to watch YouTube and Vimeo videos in real-time.

Mailet: Instant Social Networking under Censorship. Mailet is an unobservable transport which provides unfiltered social website access by using email applications. Mailet servers and clients exchange the text content of a social media website via emails. Specifically, the client sends an email to an inbox accessed by the server with the specified service details included; and on behalf of the client, the Mailet server communicates with the social website and emails back the response text, if any. This design guarantees channel consistency and has no imitation flaws. This makes Mailet immune to existing attacks.

Mailet is secure against untrustworthy proxies. One big challenge for the Mailet design is how the proxy can prove to the social website that it has been granted by the user to manage its online account. The straightforward approach is to let the user share the credential with the proxy, which is apparently risky and should not be done anytime; another approach is to let the user authorize the proxy through OAuth standard, which would not work for two reasons: (a) the authorization would fail because the user cannot access the social website to complete OAuth protocol, and (b) even the proxy is authorized by the user, but if it is malicious, the user cannot prevent any malicious attempt within its account. Our design enables Mailet proxies to provide privileged services without learning the social media login credentials of the client, using a threshold trust approach. In Mailet, the client is allowed to split and distribute the credential to a set of Mailet servers, and each server holds a share of the secret. Without learning the other shares, a single server cannot recover the credential alone.

A challenge in decentralized credential mechanism is how to privately combine and represent the credential to social websites. In other words, the Mailet servers should collaboratively recover the original credential, and include it in a TLS connection to the social

website, while still preventing each other from learning the other copy. This task is usually regarded as a secure two-party computation problem: two parties (Mailet proxies) holding separate secret inputs (the credential copies) evaluate a common function (a TLS record message) without disclosing their inputs to each other. However, since a TLS record message in Mailet is usually large (several hundred bytes), a standard two-party computation is too costly. We implemented a credential recovery by using an optimized 2PC algorithm. The results show that 2PC has to take nearly 6 seconds to finish and consumes about 6 MB bandwidth between Mailet proxies. In addition, it uses about 90% CPU and 9

To overcome this challenge, we propose a novel GCM based Credential Recovery (GCM-CR) approach to secretly combine the decentralized credential without using the high overhead secure two-party computation. This design uses Galois/Counter Mode (GCM) cipher suite in the TLS connection, and takes the advantage of Encrypt-then-MAC (EtM) of GCM mode to compute a valid TLS record message. Since this scheme involves no 2PC, a valid TLS record can be computed efficiently. Comparing with the conventional 2PC computation, our approach can achieve a speedup of 120.

Facet: Streaming over Videoconferencing for Censorship Circumvention. Facet is a system that enables the clients in a censored regime to watch YouTube, Vine and Vimeo videos in real-time. The basic idea of Facet is to send videos from these sites as the video content of a videoconferencing call in the case of our prototype, a Skype call between a Facet server and a client. Like all proxy steganography systems, it relies on the assumption that the censor is unwilling to indiscriminately block all or most sessions of the cover protocol (Skype) to avoid collateral damage. Under this assumption, Facet provides the following features:

- No emulation flaws. because the video is transmitted over an actual two-way Skype call, there is no difference between implementations to allow identification;
- content consistent. Arbitrary videos may have different characteristics from videoconferencing calls, leading to detectable differences in packet sizes. We implement one of the most popular binary classifiers for VoIP traffic and show that unaltered YouTube videos sent over Skype are distinguishable from Skype calls. To defeat this, we introduce video morphing, in which the Facet server frames the requested video within a randomly selected videoconference call. This increases the false positive rate of a classifier that can recognize 90% of Facet calls to nearly 40%.
- real-time delivery. Most stenographic anti-censorship tools are designed for regular web browsing, and often have limited bandwidth for clients. In contrast, Facet is aimed at delivering real-time video service for clients achieving the same throughput as the videoconferencing service.
- our approach is provider independent. Since the emulator devices in Facet are built independently from the videoconferencing systems, Facet can be adopted widely on any conferencing platform.

- no deployment at client side. For Facet clients, there is no need to install any client software (which is often blocked), or to pre-share secrets with the server.

1.3 Roadmap of the Thesis

Section 2 introduces the related works in anonymous traffic; Section 3 gives the attack model we consider in this thesis. Section 4 focuses on how to measure information leakage in anonymous traffic and introduces WeFDE framework; Section 5 and Section 6 focus on how to prevent information leakage in anonymous traffic by proposing Mailet and Facet systems; Section 7 concludes this thesis.

2 Related Works

This section introduces the related works in Internet Surveillance. Specifically, we first introduce the Internet surveillance which is aimed at recognizing unwanted traffic and block it, and then we describe the surveillance for de-anomymization.

2.1 Censorship as a Result of Internet Surveillance

Internet surveillance can be applied with the goal of blocking the unwanted traffic. The surveillance entity can be company network administrator, Internet Service Provider (ISP), or even national censors. To circumvent this surveillance for blockage, a set of parrot circumvention systems are proposed. The goal of the systems is to prevent surveillance entity from detection by traffic analysis. The following introduces these parrot circumvention systems and the discovered flaws in these systems.

2.1.1 Parrot Circumvention Systems and Their Imitation Flaws

Parrot Circumvention Systems. The parrot circumvention systems disguise their communications with the circumvention system user by emulating the well-known non-blocked protocols. SkypeMorph [60] disguises the communication between a Tor client and bridge as Skype Voice over Internet Protocol (VoIP). SkypeMorph starts the video call between the bridge and client as camouflage, then it drops the genuine connection and transmit Tor's TCP traffic over non-Skype UDP. A packetizer module is used to emulate the Skype UDP traffic. StegoTorus [87] obfuscates the Tor protocol. It can choose HTTP request as a cover protocol and embeds hiddentexts in the URL and cookie fields. For HTTP responses, StegoTorus utilizes the attached files such as PDF and Flash to carry the hiddentexts. CensorSpoofer [82] is designed to provide an unblocked web browsing service by IP spoofing. Consider the upstream traffic of the web browsing is lightweight, it uses low capacity channels like emails for transmission. For downstream traffic, it directly sends the traffic to the user but with the faked IP source address to fool the censor. As a consequence, the IP address of the proxy is concealed in both upstream and downstream traffic.

Imitation Flaws. [33] shows the parrot systems fail to achieve the unobservability. Firstly, SkypeMorph and StegoTorus fail to imitate side channels, which are created by the genuine systems for traffic control, user login, etc. Secondly, StegoTorus and CensorSpoofer fail to imitate reactions. StegoTorus returns different error messages when the censor sends HTTP requests to it, and CensorSpoofer fails to properly select the spoofed IP address/port, making the address/port behave differently in responding to a censor's probe. Thirdly,

SkypeMorph and StegoTorus are incorrect in imitation. Both SkypeMorph and StegoTorus wrongly imitate Skype UDP packets for lack of SoM field. Also StegoTorus generates incorrect PDF lacking the *xref table*.

2.1.2 Circumvention Systems without Imitation Flaw

FreeWave [36] is designed to circumvent the Internet censorship by hijacking the Skype protocol. The web browsing traffic between a FreeWave client and server is modulated into acoustic signals, so that it can be carried via Skype acoustic channels. Since FreeWave directly uses the genuine Skype VoIP service, it is claimed to avoid flaws in camouflage.

SWEET [97] is proposed to provide the unblockable web browsing service via email channels. In the infrastructure, the user in the censored regime encapsulates its web browsing traffic in emails, and sends these emails to the SWEET server. At the server side, it extracts the traffic from the emails, and forward the traffic to the expected destination. After receiving the replies from websites, the SWEET server embeds the responses in the emails and sends them back to the user. Since the email service provider does not collude with the censor, the SWEET is claimed unobservable due to the censor's disability in recognizing the SWEET session.

CloudTransport [12] uses the cloud storage service to circumvent the Internet censorship. It proposes a passive-rendezvous protocol, which enables the CloudTransport client and bridge to communicate through oblivious cloud systems.

2.1.3 Inevitable Inconsistency.

Recent work [26] reveals even perfect emulation can not guarantee the proxy unobservability and unblockability. The failure roots in the inevitable content and channel inconsistencies between genuine and proxy protocols.

Content Inconsistency. In FreeWave, a modulated acoustic signal rather than human speech is transmitted over VoIP. This content inconsistency is proved sufficient for a censor to identify FreeWave connection by traffic analysis. For SWEET, it utilizes the email channels to transmit the network layer traffic, making the connections distinguishable from the genuine email users. For instance, the SWEET client and server have to exchange 8 emails on average in order to complete one web browsing request. Since this burst of emails is rare for genuine email users, it gives the censor the opportunity to identify the SWEET connections. Besides, this deficiency limits the SWEET user to at most 10 to 15 web browsing requests, far from satisfying the user's needs.

Channel Inconsistency. Both SkypeMorph and FreeWave require reliable channels to transmit Tor TCP packets or synchronization frames. But VoIP usually adopt unreliable UDP transmission. This inconsistency enables a censor to stall SkypeMorph and FreeWave by packet dropping, while having negligible impacts on the genuine VoIP service. For SWEET, such channel inconsistency also exists. The email channel is delay tolerant, but the SWEET infrastructure can hardly handle the delayed email delivery, which makes the HTTP/HTTPS connections timeout or stalls the handshake protocol. That means the

sensor could delay the email delivery to interrupt the SWEET connection without severely affecting normal email users.

2.2 Anonymity under Internet Surveillance

Internet users may turn to anonymous networks such as Tor to hide their Internet activity. Unfortunately, these anonymous users are often the target for surveillance entity to monitor and de-anonymize. One popular approach that the surveillance entity can deploy is website fingerprinting attacks. This section introduces the existing website fingerprinting methods, defense designs against such attacks, and how WF attacks and defenses are measured.

2.2.1 Website Fingerprinting Attacks

The first category of WF attacks targeted encrypted protocols with no packet length hiding [56]. Liberatore and Levine [56] exploited the unique packet length of HTTPS to fingerprint a website. They adopted Jaccard coefficient and Naïve Bayes classifier as two methods, being able to identify between 66% and 90% of the time for 2000 websites. More recent website fingerprinting attacks focus on Tor anonymous service, in which the unique packet length is hidden by fixed-size Tor cells. Cai *et al.* [16] used edit distance to compare Tor packet sequences, and achieved 86% accuracy in the closed-world scenario with 100 websites. Wang and Goldberg [85] further improve the accuracy to 91% by using Tor cells instead of TCP/IP packets, deleting SENDMEs, and applying new metrics such as fast Levenshtein. Their later work [84] increases the accuracy by using a KNN classifier. Panchenko *et al.* [65] introduces a new method to extract the packet number information, which increases the accuracy by 2%. Recently, Hayes and Danezis [32] use random forests to construct the current state-of-art website fingerprinting attack.

2.2.2 Website Fingerprinting Defenses

Several defenses have been proposed to defeat WF attacks. One category of defenses try to randomize the traffic fingerprints by traffic morphing [90], loading a background page [66], or randomized pipelining [68]. These are demonstrated ineffective by several works [16, 84].

Another category of defenses try to hide traffic features by deterministic approaches. By holding the packets or creating dummy packets, BuFLO [21] requires the packets sent in fixed size and fixed time interval. The packets are padded until reaching a transmission time threshold τ if their original transmission time is shorter. Otherwise, BuFLO lets the traffic finish. CS-BuFLO [14] is proposed to extend BuFLO to include congestion sensitivity and some rate adaptation. Tamaraw [15] improves the efficiency of BuFLO by two methods. First, it allows different transmission rate for outbound and inbound traffic. Second, it pads to make the packet count a multiple of parameter L: if the packet number in one direction is more than nL and less than $(n+1)L$, it sends padding packets until the count is $(n+1)L$. Supersequence [84] utilizes clustering algorithms to group websites. For each group of websites, Supersequence computes a super trace to be the manner of transmitting

the instances of the websites under this group. WTF-PAD [44] uses adaptive padding to be efficient. Our paper includes these defenses for information leakage evaluation. We leave recently proposed defenses [18, 86] in our feature work.

2.2.3 Website Fingerprinting Evaluation

Juarez *et al.* [43] evaluates the effectiveness of WF attacks in practical scenarios, enumerating several assumptions about user settings, adversary capabilities, and the nature of the web that do not always hold. Without these assumptions, the accuracy of the attacks are significantly decreased. Cai *et al.* [15] use a comparative method to analyze defenses. They apply generators to transform a website class C into C', making C and C' differ only by one (category of) feature. Then they evaluate whether a specific defense is successful in hiding the feature. Though they claim this method can shed light on which features convey more information, the information leakage comparison between features is unclear and not quantified.

Cherubin [29] provides the lower bound estimate for the error of a WF attacker. Given a set of features, the error of the Nearest Neighbor classifier is used to estimate the lower bound of the Bayes error, which is further used to be the lower bound for the error of any classifier. Based on such lower bound, a new privacy metric called (ξ, Φ)–privacy is proposed. Though this privacy metric is not dependent on any specific classifier, it is still a variant of the accuracy/error metric, and therefore the flaw of accuracy also applies to (ξ, Φ)–privacy.

3 Attack Model

This section introduces the attack models of Internet Surveillance. In this proposal, we consider two types of Internet Surveillance, which are surveillance for blocking unwanted Internet Traffic, and surveillance with the purpose of deanonymizing users' Internet activity.

3.1 Surveillance for Censorship.

We assume a state-level censor seeking to block unwanted Internet connections and the usage of circumvention tools. Specifically, the censor is considered to have the following capabilities:

In-depth Traffic Analysis. Censor can sniff the suspicious traffic and block it if signs of unwanted connections are found. The techniques can be keyword filtering, static IP address filtering, and protocol fingerprinting. Also, the censor is considered capable of analyzing covert traffic by statistical techniques. Recent studies [89, 91, 92] show the packet length in covert protocols leaks information about the transmitted traffic, and it can be used by the censor to infer the usage of circumvention tools.

Proactive Detection. The censor can be proactive. It can act as a user of circumvention tools for blockage. This attack can be proactive probing or enumeration attacks. For probing, the censor sends probes to potential circumvention proxies, which will be blocked if responding to provide the circumvention service. Enumeration attacks refer to the censor enumerating a proxy pool list, manually or automatically, for blocking these proxies. Both of these two attacks have been shown in real-world censorship.

Active Interference. The censor can interrupt circumvention tools by actively interfering. Particularly, the censor can delay, drop, or even inject packets into the session of potential circumvention systems. Such active interference, if crafted properly, does not necessarily increases the false alarms. For example, Geddes *et al.* [27] show the censor can drop Acks in Skype sessions to disrupt SkypeMorph with little interference to the genuine Skype sessions.

The censor is considered to permit widely-used encrypted protocols such as TLS and IPsec. In addition, the censor is assumed unwilling to block begign Internet applications, such as videoconferencing (Skype, Google Hangout, and QQ), or Internet Email service. The reason could be business related or the political costs of doing so.

3.2 Surveillance for De-anonymization

This kind of Internet surveillance is to de-anonymize Internet users' activity. We suppose users come to anonymous networks for protecting their online anonymity. Then website fingerprinting attacks are what the attackers use to de-anonymize the users in anonymous networks. The attacker can be an Internet Service Provider (ISP) or a malicious Tor entry guard. It is supposed to be passive (no packet manipulation), but it can eavesdrop the traffic originated from or destinated to the user. Without turning to traffic contents or its IP addresses (both can be encrypted or obfuscated), the attacker inspects the traffic fingerprints for detection. These fingerprints can be packet length or the transmission time. Neither Cryptographic algorithms nor the anonymous services such as Tor can cover such fingerprints. State of art attacks [32, 65, 84] demonstrate that the fingerprints carry sufficient information that the attacker can pinpoint the visited website by more than 90% accuracy (with assumptions). In the following, we introduce two attack models of the website fingerprinting attack.

Closed-World Attack Model. An attacker in the closed-world knows a set of websites $C = \{c_1, c_2, \cdots, c_n\}$ the user may visit. We adopt an equal-prior model, in which the user visits a website with probability $1/n$. The attacker's goal is to decide which one has been visited.

Open-World Attack Model. The attacker in this attack model has a set of websites for monitoring; its goal is to decide whether the user visited a monitored website or not, and if yes, which monitored website. Though the user may visit any website, a non-monitored set of websites are introduced to approximate the user visiting the non-monitored websites. We consider a popularity-prior model, in which we give prior probabilities to websites by their popularity, without considering whether the websites are monitored or not.

4 Measuring Information Leakage in Website Fingerprinting

4.1 Overview

The Tor anonymity network uses layered encryption and traffic relays to provide private, uncensored network access to millions of users per day. This use of encryption hides the exact contents of messages sent over Tor, and the use of sequences of three relays prevents any single relay from knowing the network identity of both the client and the server. In combination, these mechanisms provide effective resistance to basic traffic analysis.

However, because Tor provides low-latency, low-overhead communication, it does not hide traffic features such as the volume, timing, and direction of communications. Recent works [32, 65, 84] have shown that these features leak information about which website has been visited to the extent that a passive adversary that records this information is able to train a classifier to recognize the website with more than 90% accuracy in a closed-world scenario with 100 websites. This attack is often referred to as a Website Fingerprinting (WF) attack. In response, many works [14, 15, 18, 21, 62, 66, 68, 84, 86, 90] have proposed defenses that attempt to hide this information, by padding connections with extra traffic or rearranging the sequence in which files are requested.

Defense Evaluation. To evaluate a defense, the popular practice is to train classifiers based on altered traffic characteristics and evaluate the effect of the defense by classification accuracy. If the accuracy of the classifier is low enough, the defense is believed to be secure with minimal information leakage; one defense is believed to be better than another if it results in lower accuracy.

Accuracy vs. Information Leakage. We raise a question: does low accuracy always mean low information leakage from WF defenses? Our answer is no. The first reason is that accuracy is classifier-dependent. It is possible that the information leakage of a WF defense is high, but the classifier is ineffective, so that its accuracy is low. More importantly, accuracy is all-or-nothing: classifiers output a single guess and if it is wrong, this is judged to mean the defense has been successful. But it ignores cases where a classifier may confuse some pages with a small set of others. In such situations, an attacker may well be able to significantly reduce the set of likely pages represented by a fingerprint, even if they cannot reliably choose the correct page from among this set. We can see that the fingerprint can contain a great deal of *information* about the web page even if the classifier cannot accurately identify the correct page. Accuracy is prone to *underestimate* the information

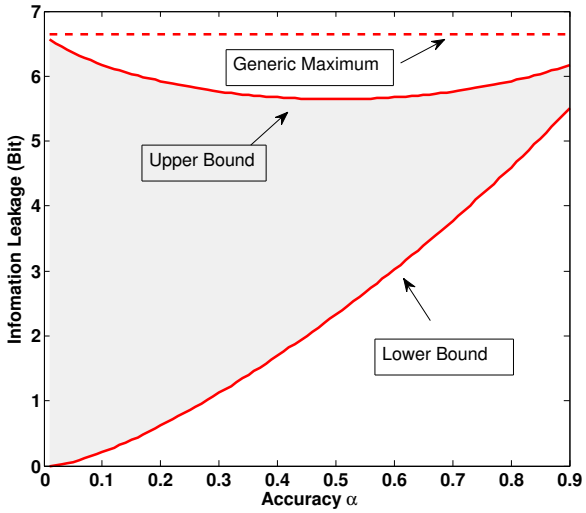


Figure 1: Accuracy vs. Information Leakage: from a classifier's perspective (closed-world setting with 100 websites)

leakage in WF defenses, or in other words, low accuracy doesn't necessarily mean low information leakage.

We further prove the above observation by the information-theoretic quantification upon a given accuracy. We find that in a closed-world setting with n websites, a feature set yielding a classifier with accuracy α could leak information through the classifier with the uncertain range $(1-\alpha)\log_2(n-1)$ (the difference between the maximum and minimum). The proof also shows that such uncertainty increases with lower accuracy. Figure 1 shows that when $n = 100$ and $\alpha = 0.95$, the uncertain range is only 0.33 bit; but when $\alpha = 0.05$, the possible leakage could be as high as 6.36 bits and as low as 0.06 bits! This uncertainty reveals the potential discrepancy between information leakage and accuracy in evaluating WF defenses, though its impact on WF attacks is limited. Low information leakage implies low classification accuracy, but the converse is not necessarily true, thus we argue that **validating WF defenses by accuracy alone is flawed**.

Feature Evaluation. Different features may carry different amounts of information. WF defense designers can evaluate features to find more informative ones to hide [15]; attackers can do so to discover highly informative features and optimize their feature set [32]. Existing works [15, 32] designed comparative methods to rank the features by their information leakage, but these methodologies do not give a straightforward way to quantify the *relationships* between features. How much information do features A and B share? If feature A is more informative than features B or C alone, are features B and C together more informative than A? These methodologies are unable to answer these questions.

We argue that these coarse-grained evaluations of features and defenses are overly simplistic. The analysis of new WF attack features and defenses should start with the ques-

tion: how much information is leaked? To answer this question, two challenges should be addressed. The first challenge is finding a way to model the behavior of WF features and the interaction between them; these features can have highly complex relationships and behavior, exhibiting distributions that could be discrete, continuous, or even partly discrete and partly continuous. The second challenge is the curse of dimensionality when estimating the total information leakage, as the state-of-art feature sets are usually high-dimensional. Unfortunately, existing works [19, 59] limited their experimental measurement to features' *individual* information leakage, and they cannot overcome these challenges.

Information Leakage Measurement Framework. In this paper, we develop WeFDE (for **We**bsite **F**ingerprint **D**ensity **E**stimation), a methodology for modelling the likelihood functions of website fingerprints, and a set of tools for measuring the information leaked by these fingerprints. To address the first challenge, WeFDE uses adaptive kernel density estimation [79] to model the probability density function of a feature or a category of features. By allowing kernels to determine their bandwidth separately, we can adaptively model both continuous and discrete density functions; by estimating multi-dimensional kernels over sets of features, we can model the interactions between features. We address the second challenge by introducing a set of dimension reduction approaches. Firstly, we measure features' pairwise mutual information to exclude redundant features. Secondly, we use Kononenko's Algorithm [17, 49] and DBSCAN [23] to separate features into sub-groups, which have pairwise mutual information higher than a threshold ϵ within each group, and lower than ϵ among different groups. Then we apply adaptive kernels for each sub-group with reduced dimensionality. Finally, our experiment shows that by including enough highly informative features we are able to approximate the overall information leakage. This enables us to further reduce the dimensionality of our measurement.

Measurement Results. We apply WeFDE to a comprehensive list of 3043 features (including, to the best of our knowledge, all features in the Tor WF literature [16, 21, 32, 65, 67, 75, 84, 85]) extracted from a 211219 Tor web browsing visits for about 2200 websites. Among the features of WF attacks, we find that: (a) 45.36% of 183 most informative features are redundant; (b) an individual feature leaks no more than 3.45 bits information in the closed-world setting with 100 websites, which is the maximum leakage we observe in our experiment from the feature of rounded outgoing packet count; (c) download stream, though having more packets than upload stream, leaks less information; (d) a larger world size has little impact on a WF feature's individual information leakage. We also include WF defenses such as Tamaraw [15], BuFLO [21], Supersequence [84], WTF-PAD [44], and CS-BuFLO [14] to study the discrepancy between accuracy and information leakage. Our experimental results confirm this discrepancy and demonstrate that accuracy alone is not reliable to validate a WF defense or compare multiple ones. We also find that the information leakage of WTF-PAD [44] is unusually high. Interestingly, recent work [**?**] confirms our result by achieving 90% classification accuracy against WTF-PAD.

Contributions. We provide our contributions as follows. First, this paper identifies that validating WF defenses by accuracy alone is flawed. By information-theoretic quantification,

		Source	Instances	Batches
Open-World	Closed-World	Alexa 1-100	55779	20
	Monitor	[84]	17985	8
	Non-Monitor	Alexa 1-2000	137455	10

Table 1: DATASET. We adopt Crawler [10] to collect the network traffic in batches. This crawler uses Selenium to automate the Tor Browser Bundle and applies Stem library to control the Tor process. It extended the circuit renewal period to 600,000 minutes and disabled `UseEntryGuard` to avoid using a fixed set of entry guards. We apply the method in [65] to extract the cell packets from the traffic and measure the information leakage based on the cell packets.[2]

we find that when classification accuracy is low, its corresponding information leakage is far from certain. Second, we propose WeFDE which makes it possible to measure the joint information leakage from a large set of features. In contrast, existing works only limited their experimental measurement to features' individual information leakage, and they cannot cope with features of complex property. WeFDE overcomes these two limitations. Third, we use WeFDE to perform information leakage measurement for all 3043 features proposed in the Tor website fingerprinting literature, based on a large dataset having 211219 Tor web browsing visits to about 2200 websites. As far as we know, our work is the first large-scale information leakage measurement in the literature. Fourth, our measurement results provide the new information-theoretic insights upon WF features, and these results give the empirical confirmation that accuracy is not reliable to validate a WF defense or compare multiple ones.

4.2 Traffic and its features

A user's traffic is a sequence of packets with timestamps which are originated from or destinated to it. We use $T(C)$ to denote the traffic when the user visited the website C. Then

$$T(C) = \langle (t_0, l_0), (t_1, l_1), \cdots, (t_m, l_m) \rangle \ , \tag{1}$$

where (t_i, l_i) corresponds to a packet of length $|l_i|$ in bytes with a timestamp t_i in seconds. The sign of l_i indicates the direction of the packet: a positive value denotes that it is originated from the server, otherwise the user sent the packet. Table 1 describes our collected traffic for information leakage measurement.

In the state-of-art website fingerprinting attacks [32, 65, 84], it is the features of the traffic rather than the traffic itself that an attacker uses for deanonymization. One of the contribution of this paper is that it measures a *complete* set of existing traffic features in literatures of website fingerprinting attacks in Tor [21, 32, 65, 67, 75, 84]. Table 2 summarizes these features by category. More details about the feature set can be found in the following.

[2]Our dataset allows the websites to have different number of instances. This uneven distribution is mostly caused by the failed visits in the crawling process. Note that it doesn't impact our information leakage measurement.

1. Packet count. Counting the number of packets is found helpful for an attacker. Specifically, we include the following features based on packet count: (a) the total packet count, (b) the count of outgoing packets, (c) the count of incoming packets, (d) the ratio between the incoming packet count and that of the total, and (e) the ratio between the outgoing packet count and that of the total.

2. Time Statistics. Firstly, we look at the packet inter-arrival time for the total, incoming, and outgoing streams, individually. We extract the following statistics and add them into our feature set: (a) maximum, (b) mean, (c) standard deviation, and (d) the third quartile. Secondly, we embrace the features based on transmission time. We add the first, second, third quartile and total transmission time into our feature set.

3–4. Packet Ordering. we explore the n-gram features which are widely adopted features extracting packet ordering. A n-gram is a contiguous sequence of n packet lengths from a traffic sequence. Let's take 2-gram as an example. Suppose the traffic sequence is $\langle(l_1, t_1)$, (l_2, t_2), (l_3, t_3), $(l_4, t_4)\rangle$, then the 2-grams are (l_1, l_2), (l_2, l_3) and (l_3, l_4). We consider the frequencies of each grams as features and we measure bigram, trigram, 4-gram, 5-gram, and 6-gram for comparison.

In addition, the number of packets transmitted before each successive incoming or outgoing packets also captures the ordering of the packets. We record such features by scanning the first 300 packets of the incoming and those of the outgoing respectively.

5–7 and 9. Intervals and Bursts. We firstly adopt interval-based features to capture the traffic bursts. An interval is defined as a traffic window between a packet and the previous packet with the same direction.

We use two approaches for interval-based features: Interval-I [84] records the first 300 intervals of incoming packets and those of the outgoing, Interval-II [75] uses a vector $\mathbf{V}$ in which $\mathbf{V}(i)$ records the number of intervals with the packet number i. We use two vectors to count the incoming and outgoing intervals separately, and we fix the vectors' dimension to be 300 (An interval having more than 300 packets is counted as a interval with 300 packets). We also apply grouping [67] on $\mathbf{V}$ to obtain extra features: $\sum_{i=3}^{5}\mathbf{V}(i)$, $\sum_{i=6}^{8}\mathbf{V}(i)$, and $\sum_{i=9}^{13}\mathbf{V}(i)$. We name this approach to be Interval-III.

We also adopt [84]'s approach of counting the bursts for outgoing packets. A burst of outgoing packets is defined as a sequence of outgoing packets, in which there are no two adjacent incoming packets. We extract the packet number in each burst and use the maximum and the average as features. We also add the total burst number, as well as the number of bursts with more than 5 packets, 10 packets, and 20 packets, respectively.

8. Packet Distribution. We divide the packet sequence into non-overlapping chunks of 30 packets and count the number of outgoing packets in first 200 chunks as features. We ignore the chunks after the 200 chunks if any, and pad 0s to have 200 features in case of having less than 200 chunks [84].

We also apply the approaches in [32] to have additional features: (a) calculate the standard deviation, mean, median, and maximum of the 200 features, and (b) split them into 20 evenly sized subsets and sum each subset to be new features.

Index	Category Name [Adopted by]	No.
1	Packet Count [21, 32, 65, 67, 84]	13
2	Time Statistics [21, 32, 84]	24
3	Ngram [this paper]	124
4	Transposition [32, 84]	604
5	Interval-I [32, 84]	600
6	Interval-II [75]	602
7	Interval-III [67]	586
8	Packet Distribution [32]	225
9	Bursts [84]	11
10	First 20 Packets [84]	20
11	First 30 Packets [32]	2
12	Last 30 Packets [32]	2
13	Packet Count per Second [32]	126
14	CUMUL Features [65]	104

Table 2: Feature Set: 14 categories with 3043 features

10–12. First 30 and Last 30 Packets. We explore the information leakage from the first and last 30 packets. Particularly, we include first 20 packets as features, and we extract the packet count features (incoming packet count and outgoing packet count) from the first and last 30 packets, respectively.

13. Packet count Per Second. We count the packet number in every second. To make the feature number fixed, we count the first 100 seconds and pad 0s if the transmission time is less than 100 seconds. The standard deviation, mean, median, minimum, and maximum of these features are also included.

We also include the alternative count of packets per second features [32]. We split the packet count per second features into 20 evenly sized subsets and sum each subset to obtain the alternative features.

14. CUMUL Features. Panchenko *et al.* [65] introduce the CUMUL features. A cumulative representation is extracted from the packet trace, and n features are derived by sampling the piecewise linear interpolant of the representation at n equidistant points. We adopt such features with $n = 100$.

It's worth noting that "packet" here refers to a Tor cell packet. We extract our features based on the cell packet traces. In addition, in 2011 [67] includes a feature named HTML marker, which counts the total size of incoming packets from the first outgoing packet and the next outgoing packet. Such summation was considered to be the size of the HTML document and therefore is informative. We find such claim is not accurate anymore, and we find no updated details of how to reproduce such a feature. As a result, we do not include this feature in our measurement.

4.3 System Design

4.3.1 Methodology

The features leak information about which website is visited. Total packet count is a good example. Figure 2 shows that visiting `www.google.de` creates 700 to 1000 packets, while browsing `www.facebook.com` results in 1100 to 1600 packets. Suppose an attacker passively monitors a Tor user's traffic, and it knows that the user has visited one of these two websites (closed-world assumption). By inspecting the total packet count of the traffic, the attacker can tell which website is visited.

Different features may carry different amounts of information. Figure 2 displays the download time in visiting `www.google.de` and `www.facebook.com`. The former loads in about 3 to 20 seconds, and the latter takes 5 to 20 seconds; Their distributions of download time are not easily separable. As a result, the attacker learns much less information from the download time than from total packet count in the same closed-world scenario.

This raises question of how to quantify the information leakage for different features. We adopt mutual information [57], which evaluates the amount of information about a random variable obtained through another variable, for this measurement, defined as:

DEFINITION. Let F be a random variable denoting the traffic's fingerprint, and suppose W to be the website information, then $I(F;W)$ is the amount of information that an attacker can learn from F about W, and $I(F;W)$ equals to:

$$I(F;W) = H(W) - H(W|F) \tag{2}$$

In the following, we describe our system to measure this information leakage.

4.3.2 System Overview

Aimed at quantifying the information leakage of a feature or a set of features, we design and develop our **We**bsite **F**ingerprint **D**ensity **E**stimation, or WeFDE. Compared with existing systems such as leakiEst [19], WeFDE is able to measure joint information leakage for more than one feature, and it is particularly designed for measuring the leakage from WF defenses, in which a feature could be partly continuous and partly discrete.

Figure 17 shows the architecture of WeFDE. The information leakage quantification begins with the Website Fingerprint Modeler, which estimates the probability density functions of features. In case of measuring joint information of a set of features, Mutual Information Analyzer is activated to help the Modeler to refine its models to mitigate the curse of dimensionality. During the information leakage quantification, the Website Fingerprint Modeler is used to generate samples. By Monte Carlo approach [31], the Information

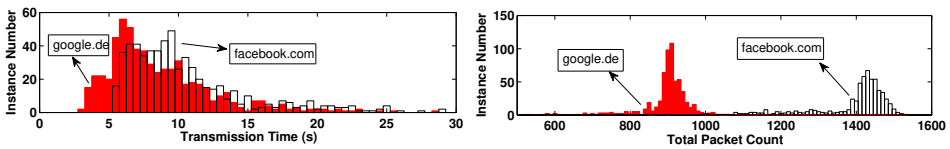


Figure 2: Different features may carry different amount of information: time vs. total packet count

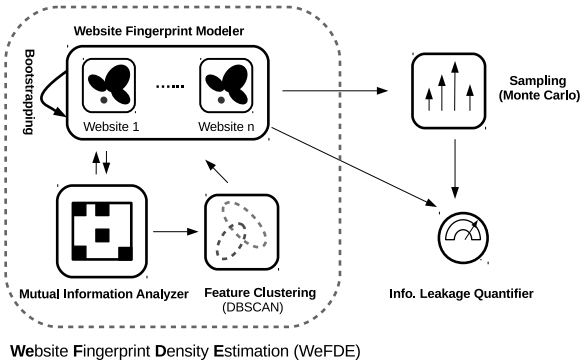


Figure 3: WeFDE's Architecture

Leakage Quantifier derives the final information leakage by evaluating and averaging the samples' leakage. In the following, we describe our modules in detail.

4.3.3 Website Fingerprint Modeler

The task of Website Fingerprint Modeler is to model the probability density function (PDF) of features. A popular approach is to use a histogram. However, as the traffic features exhibit a great range of variety, it's hard to decide on the number of bins and width. WeFDE adopts Adaptive Kernel Density Estimate (AKDE) [71], which outperforms histogram in smoothness and continuity. AKDE is a non-parametric method to estimate a random variable's the probability density function. It uses kernel functions—a non-negative function that integrates to one and has mean zero—to approximate the shape of the distribution.

Adaptive Kernel Density Estimate in WeFDE. This part gives details about Adaptive Kernel Density Estimate (AKDE) and the bandwidth selection approaches in WeFDE.

We start by how WeFDE applies AKDE to estimate a single feature's probability distribution:

$$\hat{p}_(\bar{f}|c_j) = \frac{1}{n}\sum_{c=1}^{m}\frac{1}{h_c}K(\frac{\bar{f}-p_c}{h_c}) \tag{3}$$

where

h_c is the bandwidth in AKDE,
$K(\cdot)$ is the kernel function of a Gaussian, and
$p_1, p_2, \cdots, p_m$ are the observations for $\bar{f}$ in visiting c_j

Choosing proper bandwidths is important for AKDE to make an accurate estimate. If a feature is continuous, WeFDE adopts plug-in estimator [78]. In case of failure, we use the rule-of-thumb approach [78] as the alternative. If the feature is discrete, we let the bandwidth be a very small constant (0.001 in this paper). The choice of the small constant has no impact on the measurement, as long as each website uses the same constant as the bandwidth in its AKDE.

Our AKDE may model a discrete feature as if it's continuous. The reason is that the domain of the feature $\bar{f}$ could be very large and requires much more samples than we can

collect in order to accurately estimate in discrete. Take total packet count of Facebook.com as an example. We observe 238 different values from our 600 samples in the range of 576 and 1865. If the feature is processed as discrete, the estimated probability for observed values would be inaccurate, and the missed values would be considered impossible in the inappropriate way. Our solution is to consider such a discrete feature to be continuous, so that the kernels would smooth the probability distribution estimation and assign an appropriate probability to the missed values, making our measurement more accurate.

Our AKDE is able to distinguish a continuous-like discrete feature. Take the feature of transmission time as an example. This feature is used to be continuous, but when defenses such as Tamaraw [15] are applied, the feature would become discrete. Our AKDE is able to recognize by two approaches. The first approach is about using a threshold β. If the same traffic instances are observed more than β times in our dataset, these instances are distinguished as discrete cases, and our AKDE would consider their features to be discrete. The second approach is template matching used in BuFLO case. We precompute a pattern of traffic believed to be discrete, and we consider the instances matching the pattern as discrete as well. In case of BuFLO, the pattern is the resulted traffic instance with transmission time τ.

Moreover, our AKDE can handle a feature which is partly continuous and partly discrete (or in other words, a mixture of continuous and discrete random variables). Such features exist in a WF defense such as BuFLO [21] which always sends at least T seconds. These features would be discrete if the genuine traffic can be completed within time T, otherwise, the features would be continuous. Thanks to AKDE which allows different observations to have their separate bandwidths, we compute the bandwidths separately for discrete and continuous feature values. According to [24, 25], AKDE is able to model a feature with mixed nature by selecting adaptive bandwidths for its observations.

We further extend WeFDE to model a set of features by adopting the multivariate form of AKDE. However, when applying multivariate AKDE to estimate a high dimensional PDF, we find AKDE inaccurate. The cause is the curse of dimensionality: as the dimension of the PDF increases, AKDE requires exponentially more observations for accurate estimate. Considering that the set of features to be measured jointly could be large (3043 features in case of total information measurement), we need dimension reduction techniques. In the following, we introduce our Mutual Information Analyzer to mitigate the curse of dimensionality.

4.3.4 Mutual Information Analyzer

The introduction of Mutual Information Analyzer is for mitigating the curse of dimensionality in multivariate AKDE. It helps the Website Fingerprint Modeler to prune the features which share redundant information with other features, and to cluster features by dependency for separate modelling.

This Analyzer is based on the features' pairwise mutual information. To make the mutual information of any two features have the same range, WeFDE normalizes it by Kvalseth's method [50] (other normalization approaches [81] may also work). Let $\mathrm{NMI}_{\max}(c, r)$

denote the normalized mutual information between feature c and r, then it equals to:

$$\mathrm{NMI}_{\max}(c, r) = \frac{I(c; r)}{\max\{H(c), H(r)\}}$$

Since $I(r; c)$ is less than or equal to $H(c)$ and $H(r)$, $\mathrm{NMI}_{\max}(c, r)$ is in $[0, 1]$. A higher value of $\mathrm{NMI}_{\max}(c, r)$ indicates higher dependence between r and c or in other words, they share more information with each other.

Grouping By Dependency. A workaround from curse of dimensionality in higher dimension is to adopt Naive Bayes method, which assumes the set of features to be measured is conditionally independent. Naive Bayes requires many fewer observations, thanks to the features' probability distribution separately estimated. However, we find dependence between some features of the website fingerprint, violating the assumption of Naive Bayes.

We adopt Kononenko's algorithm (KA) [17, 49], which clusters the highly-dependent features into disjoint groups. In each group, we model the joint PDF of its features by applying AKDE. Among different groups, conditional independence is assumed. KA takes the advantage of how Naive Bayes mitigates the curse of dimensionality, while keeping realistic assumptions about conditional independence between groups.

We use clustering algorithms to partition the features into disjoint groups. An ideal clustering algorithm should guarantee that any two features in the same group have dependence larger than a threshold, and the dependence of the features in different groups is smaller than the same threshold. This threshold allows us to adjust the independence degree between two groups. We find that DBSCAN [23] is the right choice.

DBSCAN is a density-based clustering algorithm. It assigns a feature to a cluster if this feature's distance from any feature of the cluster is smaller than a threshold ϵ, otherwise the feature starts a new cluster. Such a design enables DBSCAN to meet our goal above. To measure features' dependence, we calculate their normalized pairwise mutual information matrix M; then to fit in with DBSCAN, we convert M into a distance matrix D by $D = \mathbf{1} - M$, where 1 is a matrix of ones. A feature would have distance 0 with itself, and distance 1 to an independent feature. We can tune ϵ in DBSCAN to adjust the degree of independence between groups. We choose $\epsilon = 0.4$ in the experiments based on its empirical performance in the trade-off between its impact on information measurement accuracy and KA's effectiveness in dimension reduction.

We model the PDF of the fingerprint by assuming independence between groups. Suppose KA partitions the fingerprint $\vec{\mathbf{f}}$ into k groups, $\vec{\mathbf{g}}_1, \vec{\mathbf{g}}_2, \cdots, \vec{\mathbf{g}}_k$, with each feature belonging to one and only one group. To evaluate the probability $p(\vec{\mathbf{f}}|c_j)$, we instead calculate $\hat{p}(\vec{\mathbf{g}}_1|c_j)\hat{p}(\vec{\mathbf{g}}_2|c_j)\cdots\hat{p}(\vec{\mathbf{g}}_k|c_j)$, where $\hat{p}(\cdot)$ is the PDF estimated by AKDE.

As a hybrid of the AKDE and Naive Bayes, Kononenko's algorithm avoids the disadvantages of each. First, Kononenko's algorithm does not have the incorrect assumption that the fingerprint features are independent. It only assumes independence between groups, as any two of them have mutual information below ϵ. Second, Kononenko's algorithm mitigates the curse of dimensionality. The groups in Kononenko's algorithm have much less features than the total number of features.

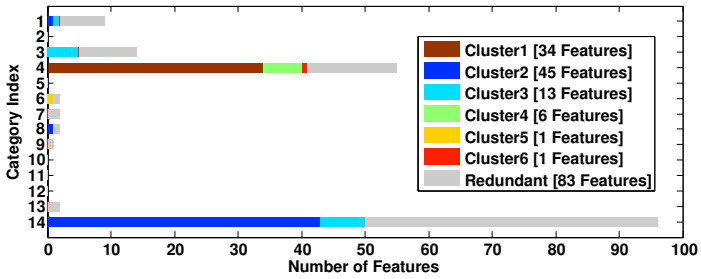


Figure 5: The Outcome of Mutual Information Analyzer

Dimension Reduction. Besides the KA method to mitigate the curse of dimensionality, we employ two other approaches to further reduce the dimension.

The first approach is to exclude features being represented by other features. We use the pairwise mutual information to find pairs of features that have higher mutual information than a threshold (0.9 in this paper). Then we prune the feature set by eliminating one of the features and keeping the other.

Our second approach is to pick out a number of the most informative features to approximate all features' information leakage. Given a set of features to measure, we sort the features by its individual information leakage. Instead of measuring all features' information leakage, we pick out top n features that leak the most information about the visited websites. The measurement results by varying n are shown in Figure 8 and Figure 12. It shows that with n increasing, the top n features' information leakage would increase at first but finally reach a plateau. This phenomenon shows that the information leakage of sufficient top informative features is able to approximate that of the overall features. Such observation is also backed by [32], which discovered that including more top informative features beyond 100 has little gain for classification.

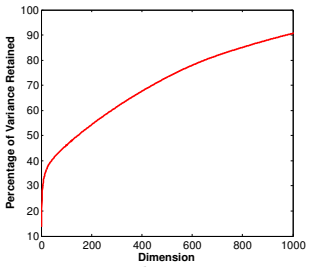


Figure 4: the Percentage of Variance Retained in PCA

We didn't choose other dimension reduction methods such as Principal Component Analysis (PCA) [42]. Our goal is to mitigate the curse of dimensionality in modelling *website fingerprints* by AKDE; but methods like PCA transform website fingerprints into *opaque components* which are much less understandable. More importantly, our experimental results demonstrate the poor performance of PCA. Figure 4 shows that the percentage of variance retained when PCA reduces to a specific dimension. Note that the percentage of variance is the popular approach to estimate the information loss in PCA. It displays that if our goal is to reduce the dimension from 3043 to 100, the percentage of variance retained after PCA is under 50%, indicating the high information loss. Thus, PCA doesn't fit in our case.

The Results. Figure 5 displays the outcome of our Mutual Information Analyzer. We picked out 100 most informative features (excluding the redundant ones), and we apply Mutual Information Analyzer to obtain 6 clusters. Figure 5 shows how many features each

category contributes, and which cluster the feature belongs to.

We find that redundant features are pervasive among the highly informative features. We look at 183 most informative features, and 45.36% of them are redundant. This phenomenon suggests future feature set engineering may be able to find many redundant features to prune without hurting its performance for website fingerprints.

Figure 5 shows a cluster may consist of features from different categories. For example, Cluster2 has features from category 1, 8, and 14, and Cluster3 has features from category 1, 3, and 14. This phenomenon shows features from different categories may share much information (that's why they are clustered together). Figure 5 also shows features from same category are not necessarily in the same cluster. For instance, the category 4 features are clustered into three different clusters.

Figure 5 also shows that categories do not necessarily have features to be included in clusters. We find that some categories lack top informative features, ending up with absence of their features in clusters. Here, we clarify that we don't claim WeFDE to be free of information loss. In fact, just like other dimension reduction approaches such as PCA, there is information loss in WeFDE, but it is minimal [32]. It's also worth noting that though some categories or features are not chosen by WeFDE, this doesn't necessarily mean all of their information is lost, as their information may be shared and represented by other included categories or features.

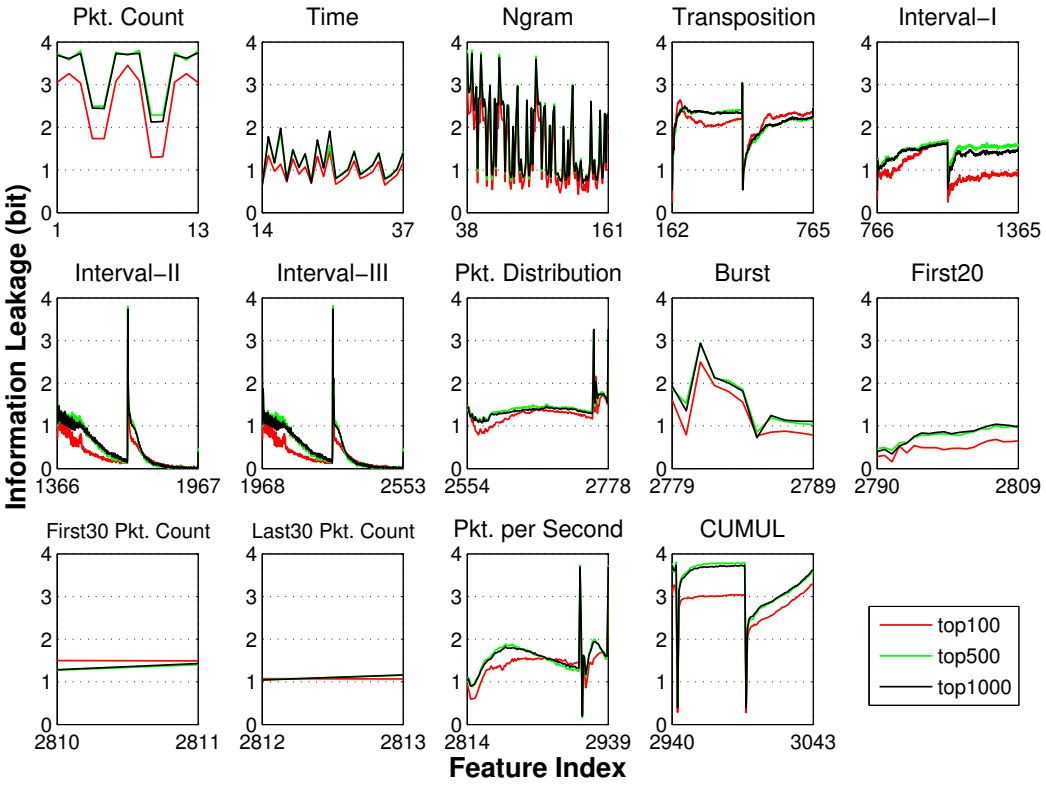


Figure 6: Closed-World Setting: Information Leakage for Individual Features (bit)

Looking at individual features, we find 33 out of 83 highly informative features are redundant with total packet count. These features include incoming packet count and 2-gram (-1,-1), but exclude outgoing packet count (NMI between total and outgoing packet count is 0.4414). The reason is that the number of incoming packets are much more than the number of outgoing packets in website browsing, so that total packet count is highly dependent on incoming packet count.

Due to page limit, we release all our measurement results in our anonymous Github repository[3].

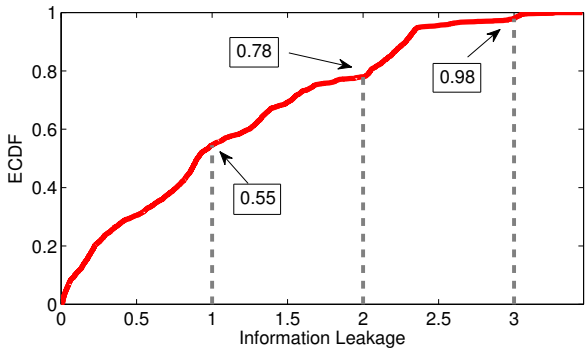


Figure 7: Empirical Cumulative Distribution Function (ECDF) for Information Leakage of Individual Features

4.4 Closed-World Information Leakage

In closed-world setting, an attacker is assumed to know the possible websites that a user may visit. The information leakage under this setting comes to *which website is visited*. The following part gives more details about how to calculate this information.

Closed-world Setting. Suppose C is a random variable denoting possible websites that a user may visit. Then the information leakage $I(F;C)$ in the closed-world scenario is:

$$\begin{aligned} I(C;F) &= H(C) - H(C|F) \\ H(C) &= -\sum_{c_i \in C} \Pr(c_i) \log_2 \Pr(c_i) \\ H(C|F) &= \int_{\Phi} p(x) H(C|x) dx \end{aligned} \tag{4}$$

where

$Pr(c_i)$ is the probability that the visited website is c_i,

Φ is the domain for the feature F, and

$p(x)$ is the probability density function for variable x.

In the measurement, we adopt Alexa top 100 websites with 55779 visits in our closed-world setting, as is shown in Table 1. We assume equal prior probability for websites,

[3]https://github.com/s0irrlor7m/InfoLeakWebsiteFingerprint

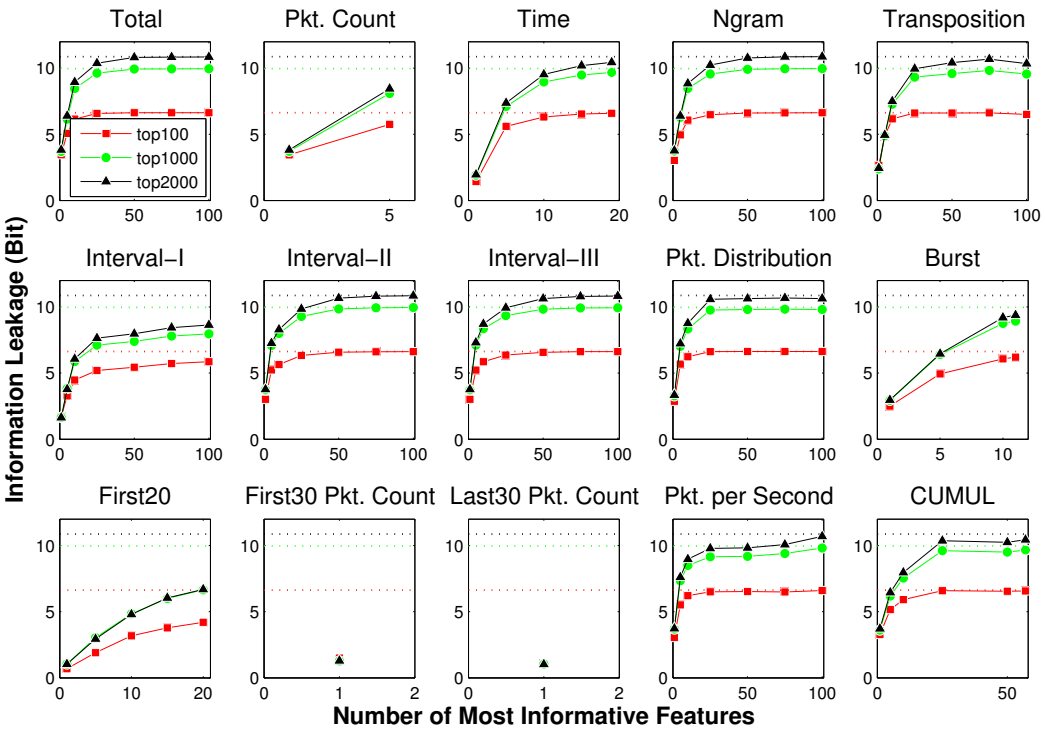


Figure 8: Closed-World Setting: Information Leakage by Categories (bit)

and we set Monte Carlo sample number to 5000. We measure 3043 features' individual information leakage and their joint leakage by categories. We run this measurement and the following ones on a workstation with 32 cores (Intel Xeon CPU E5-2630 v3 @ 2.40GHz). The measurement time differs depending on the settings, but a typical measurement like the following can be finished within 10 hours. The following introduces part of our results. Full measurement results can be found at our anonymous Github repository.

Individual Information Leakage. Our measurement results upon individual features are shown in Figure 7. Among these 3043 features, we find: (a) 2.1% features leak more than 3 bits information, meaning that an attacker is able to narrow down the possibilities to one eighth by any of these features; (b) 19.91% features leak less than 3 bit but more than 2 bits information; (c) 23.43% features leak 1 bit to 2 bits information; and (d) 54.55% features leak less than 1 bit information. It is clear that nearly half of the features are able to help an attacker to reduce the size of the anonymity set by half. Yet our experiment shows that a single feature leaks no more than 3.45 bits information, which is the maximum leakage we observe in our experiment from the feature of rounded outgoing packet count. We also observe that outgoing packet count without rounding leaks 3.26 bits information, 0.19 bit less than the rounded one. We observe similar information leakage increase by rounding for total packet count and incoming packet count. Our results confirm the observation in [67] that rounding packet count can help website fingerprinting attacks.

Web-browsing is characterized by asymmetric traffic. The incoming packets, which con-

tain the requested contents, usually outnumber the outgoing packets carrying the request. A natural question is, does download stream having more packets leak more information than upload stream? The answer is no: download stream leaks 3.04 bits information, 0.22 bit less than the incoming stream. Our measurement suggests that defense design should give no less (if not more) attention to upload stream in hiding both streams' packet count.

The most informative timing feature is the average of inter-packet timing for download stream with 1.43 bit leakage. Among inter-packet timing features, the maximum leaks the least information with around 0.7 bit leakage. Another category of timing features is transmission time. We observe that the information leakage increases from 25% percentile to 100% percentile transmission time. The most information leakage for transmission time comes to the total transmission time, which leaks 1.21 bit information. Furthermore, our measurement shows that information leakage of timing features has little difference for upload and download stream.

We also experiment the impact of world size on individual feature's information leakage. We try to answer: with a larger world size, whether the information leakage of indiviual features increases or decreases. We further adopt Alexa top 500 and top 1000 separately for closed-world setting, and we conduct the same information leakage measurement as above. Note that the information leakage upper bound under the world size 100, 500, and 1000 is 6.64, 8.97, and 9.97 bits, respectively. Our finding is that the impact of world size on information leakage is minimal, as is shown in Figure 6. Particularly, when the world size increases from 500 to 1000, the features' individual information leakage is almost the same. Further analysis will be given in subsection 4.4.1.

Joint Information Measurement. Among the 100 most informative features, many of the features share redudant information with other features. We set a threshold to 0.9, and if two features have mutual information larger than 0.9, we would consider a feature sharing most of its information with another one. Our results show that 62 of the 100 most informative features can be represented by the other 38 features, demonstrating the prevalence of redundant features in website fingerprint. This finding shows the necessity and effectiveness of our Mutual Information Analyzer in recognizing features sharing redundant information. Figure 8 also shows that after including sufficient non-redundant features, the category information leakage tends to reach plateau. This phenomenon shows that we can approximate the information of a category by including sufficient non-redundant most informative features in this category.

Categories such as Time, Ngram, Transposition, Interval-II, Interval-III, Packet Distribution, Packet per Second, and CUMUL leak most of the information about the visited websites; other categories such as Packet Count, Interval-I, Burst, First20, First30 Packet Count, Last30 Packet Count leak 5.75, 5.86, 6.2, 4.20, 1.29, and 1.03 bits information, respectively. Our measurement shows that Interval-II and Interval-III leak more information than Inerval-I, with 6.63 bits for both Interval-II and Interval-III. In addition, we find that Interval-II and Interval-III are faster than Interval-I in reaching the plateau, indicating the former twos not only leak more information but also with less features. It is clear that recording intervals by their frequency of packet count (adopted in Interval-II and Interval-

III) is more preferable than recording them in sequence (Interval-I).

We also experiment the impact of world-size on information leakage upon categories in closed-world setting. We find that with the increase of world size, most categories exhibit more information leakage, except First30 and Last30 Packet Count. Note that categories such as First20, Burst, Packet Count show little increase when the world size increases from 1000 to 2000. We leave the discussion to subsection 4.4.1.

4.4.1 World Size and Information Leakage

In this section, we discuss the impact of the world size on our information leakage measurement.

We start with the closed-world setting. We observe that with the increase of the world size, the information leakage for most categories and the total increases as well, while the individual information leakage of features is little impacted (particularly when the world size increases from 1000 to 2000). To explain the conflicting observations, we highlight the notion of maximum possible information leakage of a setting. A feature (or a set of features) leaks no more information than the information that the setting has. For example, in our closed-world setting with 100 websites, the total information leakage is 6.63 bits. But if we let the world size be 2, the total leakage is no more than 1 bit, no matter how distinguishable the fingerprint is. Therefore we argue that the increased information leakage with larger world size for most categories and the total is because the website fingerprint has the ability to leak more information than the information that our closed-world settings have. This phenomenon leads to an interesting question: what is the maximum information leakage the website fingerprint is able to leak in a sufficiently larger world size, which we include in our future work.

For the features' individual information leakage, we observe that the leakage in each setting is much less than the information that these setting have, and that the world size has little impact on the measurement. We explain the reason for the little impact of the world size by the following theorem:

Theorem 2. Let's consider x closed-world settings with equal world size n. Suppose a feature $F = \bar{f}$ has *valid* information leakage of $I_1, I_2, \cdots, I_x$ in each closed-world setting. In the combined closed-world setting with nx world size, the information leakage of $F = \bar{f}$ would be $\frac{I_1+I_2+\cdots+I_x}{x}$.

Proof: let's denote the information leakage in each closed-world setting to be:

$$I_l = log_2(n) + \sum_{l \in \{1,\cdots,n\}} q_l(i) log_2(q_l(i)) \tag{5}$$

, where $q_l(i)$ is the probability of visiting the ith website in the lth closed-world setting conditioned on $F = \bar{f}$.

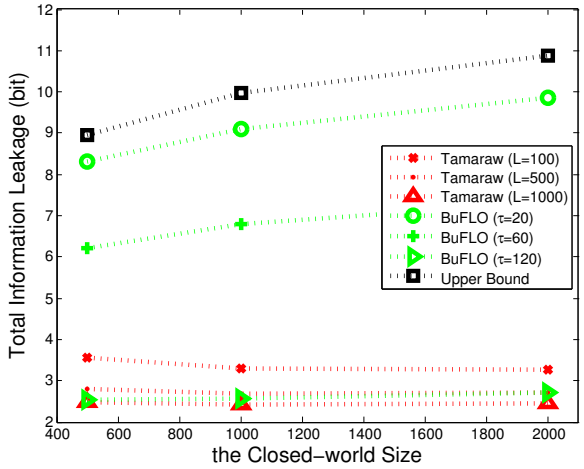


Figure 9: Defenses with Different World Size.

In the combined closed-world setting, the information leakage of $F = \bar{f}$ is

$$\begin{aligned} & log_2(nx) + \sum_{l\in\{1,\cdots,x\}} \{ \sum_{i\in\{1,\cdots,n\}} \frac{q_l(i)}{x} log_2(\frac{q_l(i)}{x})\} \\ &= log_2(n) + \frac{1}{x} \sum_{l\in\{1,\cdots,x\}} \sum_{i\in\{1,\cdots,n\}} q_l(i) log_2(q_l(i)) \\ &= \frac{I_1 + I_2 + \cdots + I_x}{x} \end{aligned} \tag{6}$$

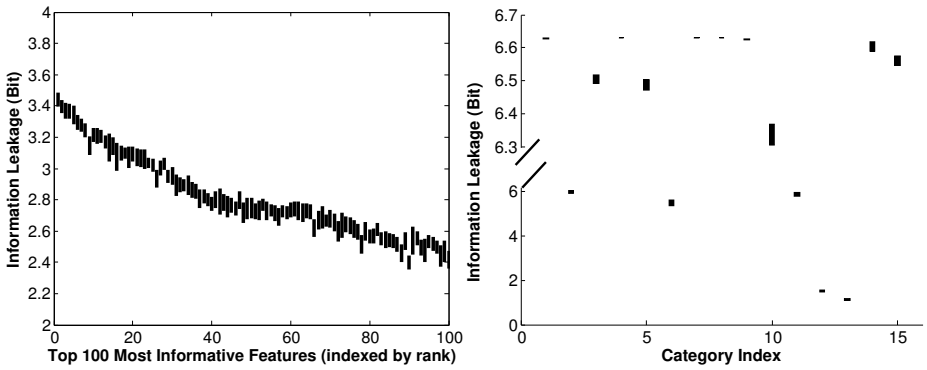
This theorem reveals the relation between world size and information leakage. With each closed-world setting including sufficient websites, the combined larger world size would have little impact on the information leakage.

We also evaluate world size impact on defenses in closed-world setting. Figure 9 shows that in Tamaraw, world size has little impact on information leakage. No matter how large the world size is, the information leakage for Tamaraw is around $3.3, 2.72, 2.45$ bits for $L = 100, 500, 1000$. BuFLO with $\tau = 120$ is not impacted by world size, but BuFLO with $\tau = 20, 60$ see the increase of information leakage. The different impact from world size roots in BuFLO's mixed nature.

We discuss the world size impact on the open-world setting. Here the world size refers to the size of the non-monitored websites. We find that with a larger world size, the maximum information leakage decreases. In addition, as is shown in Section 4.6, world size also has little impact on the measure.

4.5 Validation

Information Leakage Measurement Validation. This section shows how accurate our measurement is. We adopt bootstrapping with 20 trials to give the 90% confidence interval for the information leakage measurement. Figure 10 (a) shows the confidence intervals for



(a) (b)

Figure 10: Information Leakage Measurement Validation: 90% Confidence Interval for the Measurement

top 100 most informative features. We find that the width of the intervals is less than 0.178 bit, and the median is around 0.1 bit. Figure 10 (b) gives the 90% confidence interval for 15 categories. The width of these intervals is less than 0.245 bit, with the median 0.03 bit. We find that the interval having the largest width is the category of Interval-I. The bootstrapping results validate our information leakage measurement.

bootstrapping: accuracy estimation for information leakage quantification

We use bootstrapping [22] to estimate the accuracy of our information-theoretic measurement. bootstrapping is a statistical technique which uses random sampling with replacement to measure the properties of an estimator.

We implement bootstrapping to estimate the confidence interval of the information leakage. We describe our bootstrapping in the following:

- Step 1: for the observations of the each website, we apply random sampling with replacement, in which every observation is equally likely to be drawed and is allowed to be drawed more than once (with replacement). We let the sampling size be equal to observation size.
- Step 2: we apply our measurement on the newly constructed dataset of resamples and obtain the information leakage.
- Step 3: Step 1 and Step 2 are repeated K times to obtain K values for the information leakage; We therefore find the `CI` confidence interval based on these K values.

Subsampling [69] is a special bootstrapping technique. It uses sampling without replacement, and its sampling size is usually much smaller than the observation size.

Dataset Validation. We use the top 100 Alexa websites in the closed-world setting, as do previous works. But what if the top 100 Alexa websites are not representative for Tor networks? Do our information leakage results still hold? While the representative websites are still unknown, we validate our results by subsampling [69].

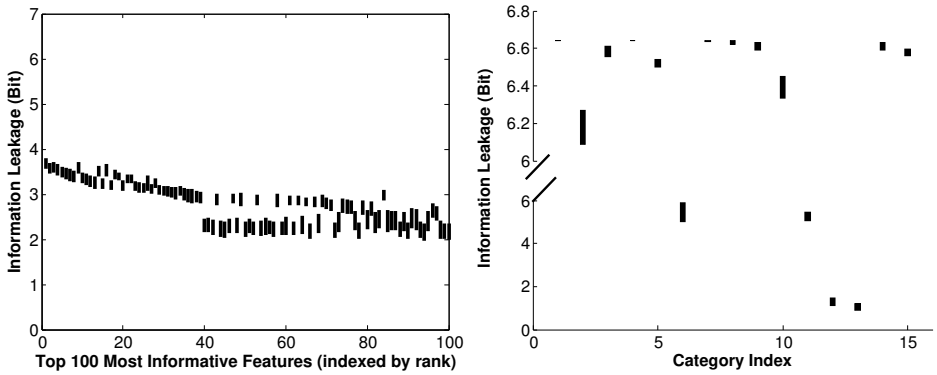


Figure 11: Dataset and Generalization: the 90% confidence interval by website subsampling

Subsampling is a bootstrapping technique without replacement (more details are given in Section 4.5). It provides the bootstrapped confidence interval for our information leakage results when the representative websites are unknown. In the experiment, we have 2200 websites for subsampling. In each round, we randomly sample 100 websites without replacement to construct the bootstrapped dataset. Repeating the same procedure n times ($n = 20$ in our experiment), we have n such datasets to obtain n bootstrapped measurements. Finally, we get the bootstrapped confidence interval for validation.

Figure 11 displays the 90% confidence interval for the top 100 most informative features and 15 categories of features. Not surprisingly, including different websites in the closed-world setting does make a difference in the measurement, but Figure 11 shows such impact is very limited. Among top 100 informative features, most of them have confidence interval with less than 0.5 bit width, so do most of categories (even less for some categories). The exception only comes to category Interval-I. By subsampling, we validate our information leakage results even when the true representative websites are still unknown.

4.6 Open-world Information Leakage

In the closed-world scenario, the attacker knows all possible websites that a user may visit, and the goal is to decide *which* website is visited; In the open-world setting, the attacker has a set of monitored websites and tries to decide whether the monitored websites are visited and which one. The difference in information leakage is that the open-world has $n + 1$ possible outcomes, whereas the closed-world has n outcomes where n is the number of (monitored) websites. We include the details about how to quantify this information in the following.

Open-world Setting. The information leakage $I(F; O)$ in the open-world scenario is:

$$I(F; O) = H(O) - H(O|F) \tag{7}$$

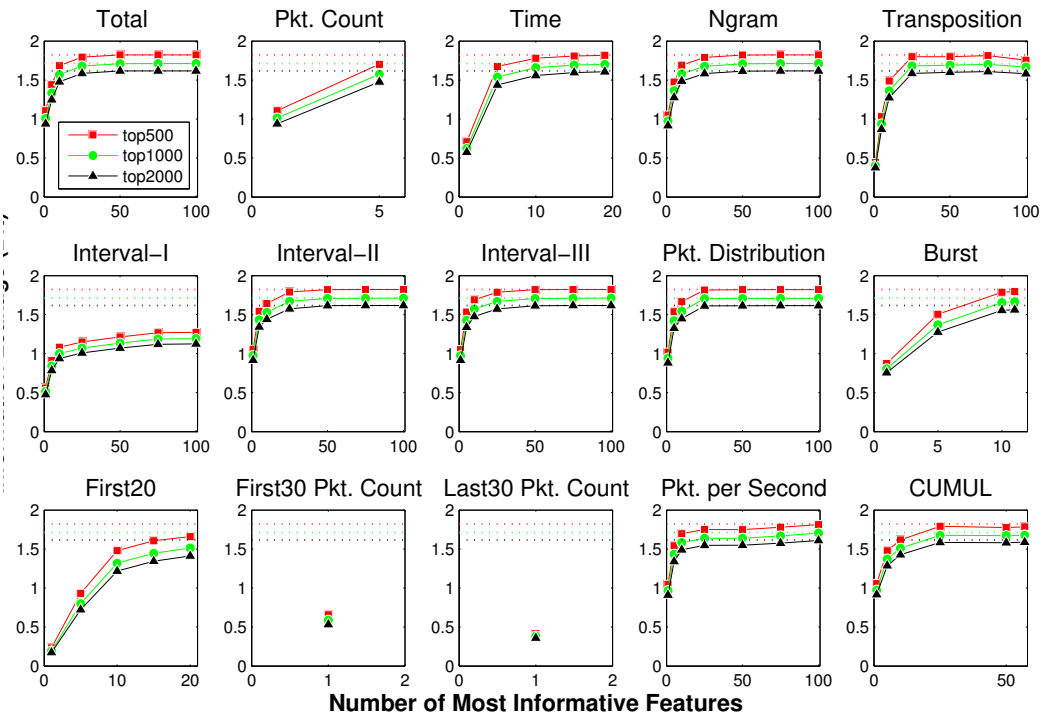


Figure 12: Open-World Setting: Information Leakage by Categories

$$H(O) = -\sum_{c_i \in M} \Pr(c_i) \log_2 \Pr(c_i) - \left\{ \sum_{c_j \in N} \Pr(c_j) \right\} \log_2 \{ \sum_{c_j \in N} \Pr(c_j) \} \tag{8}$$

$$H(O|F) = \int_{\mathbb{F}} p(f) H(O|f) df \tag{9}$$

$$H(O|f) = -\sum_{c_i \in M} \Pr(c_i|f) \log_2(\Pr(c_i|f)) - \left\{ \sum_{c_j \in N} \Pr(c_j|f) \right\} \log_2 \{ \sum_{c_j \in N} \Pr(c_j|f) \} \tag{10}$$

where O is a random variable denoting the visited website belongs to the monitored or the non-monitored, and if it is monitored, which one. M denotes the monitored set of websites, and N denotes the non-monitored set of websites. $\mathbb{F}$ denotes the domain for feature F.

Experimental Results This section describes part of our results for the open-world information leakage. For more information, please visit our Github repository.

Experiment Setup. We adopt the list of monitored websites from [84] and collected 17984 traffic instances in total. Our non-monitored websites come from Alexa's top 2000 with 137455 instances in total. We approximate the websites' prior probability by Zipf law [11, 30], which enables us to estimate a website's prior probability by its rank. We conduct experiments with top $500, 1000, 2000$ non-monitored websites separately, and we show the experimental results in Figure 12.

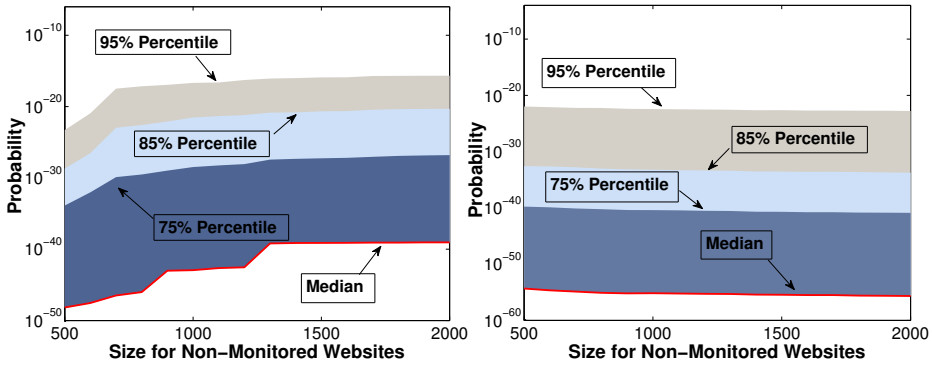
Figure 12 shows that the open-world information leakage is decreased when including more non-monitored websites, with $1.82, 1.71, 1.62$ bit for top500, top1000, top2000, respectively. Including more non-monitored websites decreases the entropy of the open-world setting rather than increasing it. The reduced information is in part because of the prior on monitored websites. Compared with closed-world setting with similar world size, open-world scenario carries much less information.

Similar with the closed-world setting, Figure 12 shows that most categories except First20, First30 and Last30 Packet count, and Interval-I leak most of the information. This shows that the difference in world setting has little impact on categories' capability in leaking information.

We also investigate how the size of the non-monitored websites influences our measurement. We focus on the total leakage and build the AKDE models for the non-monitored websites with the varying size of the non-monitored, respectively. We evaluate how the difference of these AKDE models influences measurement. Specifically, we evaluate (a) how monitored samples are evaluated at these AKDE models, and (b) how samples generated by these AKDE models are evaluated at the monitored AKDE. Figure 13 shows the results. Figure 13 (a) shows that these AKDE models of the non-monitored, though differing in size, assign low probability (below 10^{-10} with 95% percentile) to monitored samples. Figure 13 (b) shows that though these AKDE models for the non-monitored generate different samples, the difference on how these samples are evaluated by the AKDE model of the monitored is little: they are all assigned low probability below 10^{-20} with 95% percentile. The results lead to the estimation that introducing extra lower rank websites into the non-monitored set would not significantly change the low probability that the non-monitored AKDE assigns to monitored samples, and the low probability that the monitored AKDE assigns to samples generated by the non-monitored AKDE, thanks to the low prior probability of these websites. The information leakage is therefore little impacted.

4.7 Measuring WF Defenses with Information Leakage

This section firstly gives the theoretical analysis on why accuracy is not a reliable metric to validate a WF defense. Then we measure the WF defenses' information leakage to confirm the analysis. Note that we choose the closed-world setting in the evaluation, as the setting is most advantageous for attackers, and we can get an upper bound for the defense's security.



(a) (b)

Figure 13: Size of the Non-Monitored Websites and Open-World Information Leakage Measurement: (a) Monitored Samples at Non-Monitored AKDE, and (b) Non-Monitored Samples at Monitored AKDE

4.7.1 Accuracy and Information Leakage

Using accuracy to prove a defense to be secure is flawed. This section would investigate the information leakage corresponding to a specific accuracy. We find that given a specific accuracy, the actual information leakage is far from certain.

Theorem 1. Let $\{c_1, c_2, \cdots, c_n\}$ denote a set of websites with prior probabilities $p_1, p_2, \cdots, p_n$, and v_i denote a visit to website c_i. Suppose a website fingerprinting classifier D which recognizes a visit v_i to be $D(v_i)$. The classifier would succeed if $D(v_i) = c_i$, otherwise it fails. Assume a defense has been applied, and this classifier has α accuracy in classifying each website's visits. Then the information leakage obtained by the classifier is uncertain: the range of the possible information leakage is

$$(1-\alpha)log_2(n-1) \tag{11}$$

Proof: Let $I(D;V)$ denote the information leakage that the classifier attains. we have

$$\begin{aligned} I(D;V) =& H(D) - H(D|V) \\ =& H(D) - \sum_{v_i \in V} p(v_i) H(D|v_i) \end{aligned} \tag{12}$$

We then evaluate $H(D|v_i), v_i \in V$. With the accuracy α, we have $Pr(D = v_i|v_i) = \alpha$. However, it is uncertain about the probability $Pr(D = v_j|v_i)$ where $j \neq i$, from the knowledge of the accuracy. We put the possibility in two extremes to obtain the range of possible evaluation. In one case, suppose that the classifier determines v_i to be from the website $C_{\hat{j}}$ with probability $1-\alpha$, so that the maximum of $I(D;V)$ is obtained as

$$\max\{I(D;V)\} = H(D) + \alpha \log_2 \alpha + (1-\alpha) \log_2 (1-\alpha) \tag{13}$$

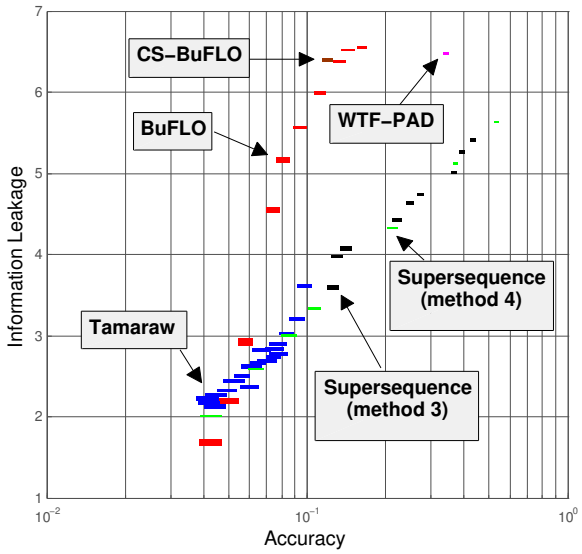


Figure 14: Website Fingerprinting Defenses: Accuracy vs. Information Leakage

In the other case, suppose that the probabilities $Pr(D = v_j|v_i) = (1-\alpha)/(n-1)$ where $j \neq i$, which means that except the correct decision, the classifier determines the visit v_i to belong to any website other than C_i with equal probability. Such a case yields the minimum possible information leakage, which is:

$$\min\{I(D;V)\} = H(D) + \alpha \log_2 \alpha + (1-\alpha) log_2 \frac{1-\alpha}{n-1} \tag{14}$$

As a result, the range of potential information leakage $I(D;V)$ conveyed by the accuracy α is

$$(1-\alpha) log_2 (n-1) \tag{15}$$

An Example. Figure 1 shows an example for the theorem. Note that the range is invariable no matter what we assume for websites' prior probability. We can see a wide range of possible information leakage when a low accuracy is given, showing that low accuracy doesn't necessarily guarantee low information leakage.

4.7.2 Information Leakage Measurement upon WF Defenses

We include Tamaraw [15], BuFLO [21], Supersequence [84], WTF-PAD [44], and CS-BuFLO [14] to quantify the information leakage upon defensed traffic.

We adopt the implementation of BuFLO, Tamaraw, and Supersequence [8] to generate the defensed traffic, with τ =5, 10, 20, 30, 40, 50, 60, 80, 100, or 120 for BuFLO, L ranging from 10 to 100 with step 10 and from 200 to 1000 with step 100 for Tamaraw. We include the method 3 of Supersequence, with 2, 5, or 10 super clusters, and 4, 8, or 12 stopping points. We also include the method 4 of Supersequence, with 2 super clusters, 4 stopping points, and 2, 4, 6, 8, 10, 20, 35, or 50 clusters. We use the implementation [7] to create the

WTF-PAD traffic. We were recommended to use the default `normal_rcv` distributions on our dataset, as finding an optimal set of distributions for a new dataset is currently a work in progress [7]. We apply the KNN classifier [84] on our WTF-PAD traces, and we can get similar accuracy (18.03% in our case). This classification result validates our WTF-PAD traces. We use the implementation [29] to generate simulated CS-BuFLO traces.

Upon each type of defensed traces, we evaluate the overall information leakage and the classification accuracy at the same time. The measurement is conducted in closed-world setting with 94 websites. To evaluate the total information leakage, we assume equal prior probability for websites and adopt $k = 5000$ for Monte Carlo Evaluation. We use bootstrap [22] with 50 trials to estimate the 96% confidence interval for the information leakage and accuracy. For details about bootstrap, please see Section 4.5. Note that we redo the dimension reductions for each defense, as a WF defense changes the information leakage of a feature and the mutual information between two features. The classifier we adopt is a variant of the KNN classifier [84]. The only change we make is the feature set: we use our own feature set instead of its original one. The purpose is to have equivalent feature sets for classifications and information leakage measurements. The reason for choosing this KNN classifier is that it is one of the most popular website fingerprinting classifiers to launch attacks and evaluate defense mechanisms. It's also worth noting that the original feature set of the KNN classifier is a subset of our feature set. The experimental results are shown in Figure 14.

4.7.3 Accuracy is inaccurate

Accuracy is widely used to compare the security of different defenses. A defense mechanism is designed and tuned to satisfy a lower accuracy as an evidence of superiority over existing defenses [21]. With defense overhead being considered, new defense mechanisms [15, 44] are sought and configured to lower the overhead without sacrificing accuracy too much. But if accuracy fails to be a reliable metric for security, it would become a pitfall and mislead the design and deployment of defense mechanisms. This section describes the flaws of accuracy and proves such a possibility.

Accuracy may fail because of its dependence on specific classifiers. If a defense achieves low classification accuracy, it's not safe to conclude that this defense is secure, since the used classifiers may not be optimal. More powerful classifiers may exist and output higher classification accuracy. We prove this possibility in our experiment. To validate WTF-PAD, four classifiers were used including the original KNN classifier, and the reported highest accuracy was 26%. But using the KNN classifier with our feature set, we observe 33.99% accuracy. Thus, accuracy is not reliable to validate a defense because of its dependence on specific classifiers.

Defenses having equivalent accuracy may leak varying amount of information. Figure 14 demonstrates such a phenomenon when taking BuFLO ($\tau = 40$) and Tamaraw ($L = 10$) into consideration. Accuracy of both defenses is nearly equivalent, with 9.39% for BuFLO and 9.68% for Tamaraw. In sense of accuracy, BuFLO ($\tau = 40$) was considered to be as secure as Tamaraw($L = 10$). However, our experimental results disap-

prove such a conclusion, showing BuFLO ($\tau = 40$) leaks 2.31 bits more information than Tamaraw (which leaks 3.26 bits information). We observe the similar phenomenon between WTF-PAD and Supersequence.

More importantly, a defense believed to be more secure by accuracy may leak more information. Take BuFLO ($\tau = 60$) as an example. Its accuracy is 7.39%, while accuracy of Tamaraw with $L = 10, 20, 30$ is 9.68%, 9.15%, and 8.35% respectively. Accuracy supports BuFLO ($\tau = 60$) is more secure than Tamaraw with $L = 10, 20, 30$. However, our measurement shows that BuFLO ($\tau = 60$) leaks 4.56 bit information, 1.3 bit, 1.61 bit, and 1.75 bit more than Tamaraw with $L = 10, 20, 30$! Take WTF-PAD as another example. The accuracy for WTF-PAD is 33.99%, much lower than the 53.19% accuracy of Supersequence method 4 with 2 super clusters, 50 clusters, and 4 stopping points. But the information leakage of WTF-PAD is around 6.4 bits, much higher than the leakage of the latter which is about 5.6 bits. Our experimental results prove the unreliability of accuracy in comparing defenses by security.

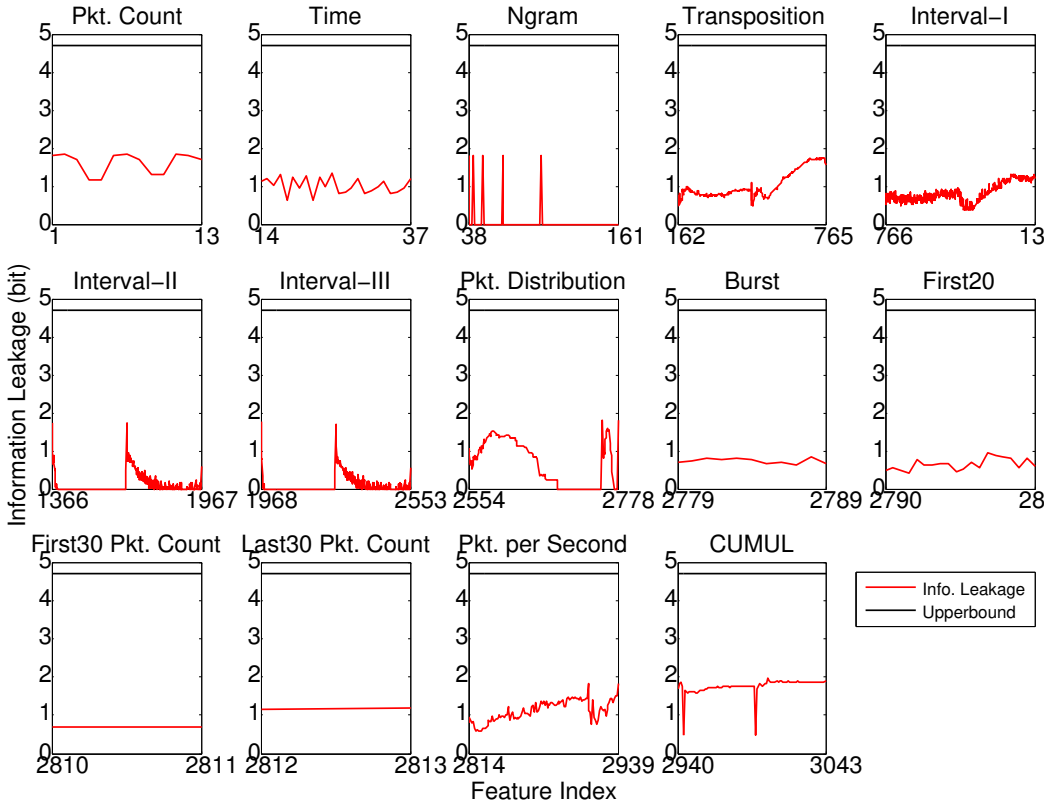


Figure 15: Information (bits) about the number of images in a page given by each of the 3043 individual traffic features found in our preliminary work

4.8 An Example of Applying WeFDE

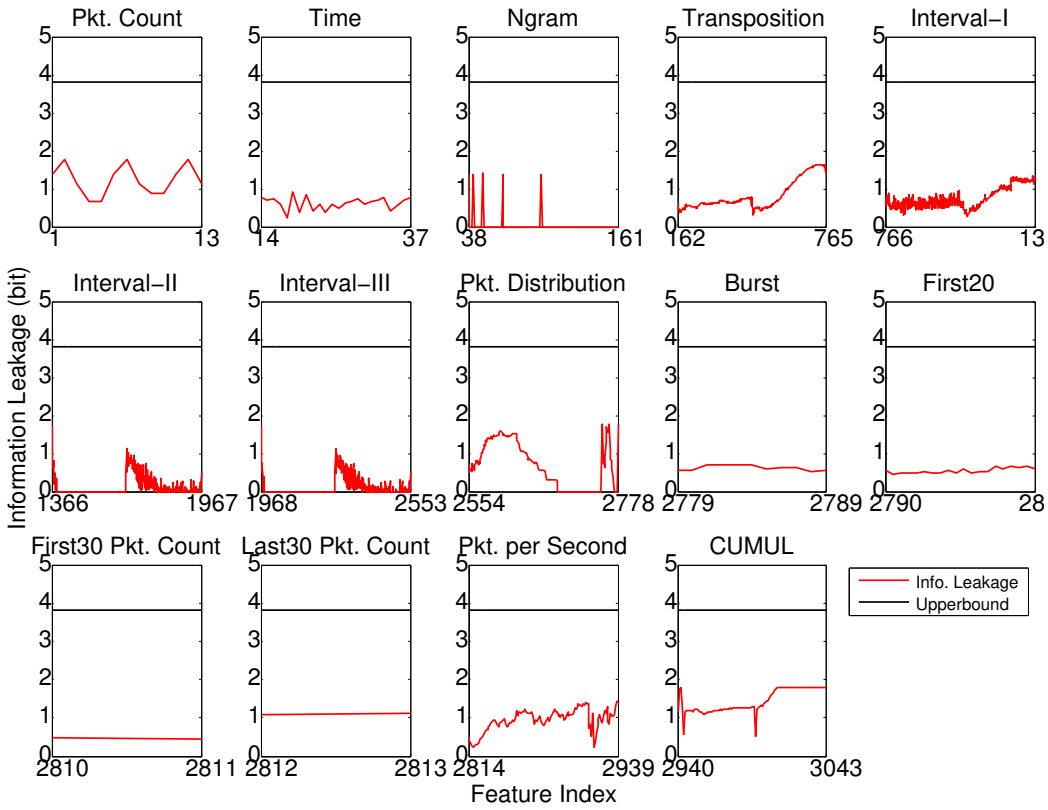


Figure 16: Information (bits) about the number of included Javascript libraries in a page given by each traffic feature

The success of website fingerprinting attacks is due to the fact that the varying contents of a web page cause variation in the traffic features an attacker can observe. While recent works [29,63,64] have found that some classes of contents can influence the fingerprintability of a page more than others, most works continue to focus on the indirect link between a web page and traffic features.

The link between content types and traffic features is unexplored. The goal of this proposal is to shed light on such link. Specifically, we will investigate: (a) how different types of contents influence the traffic features, (b) how to use traffic features to infer underlying types of contents.

The benefits of doing so are twofold. Firstly, the results will capture how an attacker might learn the users browsing behaviour even when the browsed webpage is unsupervised. Traditional website fingerprinting attacks usually adopt the supervised or semi-supervised learning in the sense that a set of pages is supposed to be visited or monitored. However, this assumption is unrealistic, the potential web pages that a user may visit is so many that

it is unrealistic to assume a small set of them. Our results may show that even a page is unsupervised, it is still possible to learn information about the users behaviours. Secondly, knowing enough about the contents of a download can enable linking back to a specific page.

4.8.1 Evaluation Results.

Our preliminary results calculate the information leakage from individual features about the webpage contents. We apply the WeFDE framework to measure the information leakage from features about the number of images embedded and Javascript libraries included in a page. The dataset consists of 157 websites, with 121431 instances in total. We count how many images or Javascripts are in each website page, and label it with this count, using binning to smooth the class sizes. The results are shown in Figures 15 and 16.

The results show that the individual features leak a signinficant amount of information about the contents. For example, total packet count alone leaks 1.81 bits information about the embedded image count. Our preliminary results on webpage classification indicate that there are enough independent features that by combining several similarly informative features, the image count should be uniquely identifiable. Since we observe a similar level of information about javascript count, this suggests that for many content classes, the content to feature mapping can be reconstructed.

5 Mailet: Social Networking under Censorship

5.1 overview

Social Media websites such as Twitter and Facebook have grown to play a prominent role not only in the social lives of their users, but as an important source of news about current events [51], communication hub in emergencies [80], and a coordination mechanism for social and political activism [77]. Correspondingly, governments in many countries have either permanently or temporarily blocked or threatened to block access to these sites; for example the *herdict.org* censorship data site reports at least 10 different countries blocking Facebook at some point in 2014 and 11 blocking Twitter. In response to this blocking, users in these countries often turn to circumvention tools; one survey of circumvention users found that accessing social media sites was the second most common reason for using these tools, with over 70% of respondents citing this intent [1].

In response to the popularity of some circumvention tools, several nation states have deployed technology that blocks access to these tools, through a combination of address blocking, protocol filtering based on deep packet inspection (DPI) and active probing to reduce false positives. This has led researchers to engage in an "arms race" of protocols and attacks for "unobservable transport." Steganographic "parrot" protocols [60, 82, 87] attempt to imitate protocols the censor will be reluctant to block, but many of these schemes were shown to suffer from imitation flaws [33]. Decoy Routing schemes [34, 46, 93, 94] use backbone routers as proxies to imitate connections to arbitrary unblocked hosts, but were shown to require impractically large and targeted deployments in order to avoid availability attacks [38, 73]. "Hide-within" systems [12, 36, 97] attempt to hide both the true destination and the covert nature of circumvention connections by tunnelling connections through popular services such as email, VoIP, and cloud storage. In addition to requiring high bandwidth overhead, these schemes were shown to suffer from detection and blocking attacks stemming from inconsistency between the data volume and loss tolerance of the proxy and cover applications [26].

As a result of this arms race, it is unclear whether there can be a single cover protocol that can handle arbitrary Internet content. An alternative strategy is to develop a small number of systems, each of which is difficult to detect or block when carrying a specific type of content. This strategy is supported by the finding that in China, most circumvention use is motivated by unfiltered search, unfiltered social media access, and video sharing sites [1].

An example of this strategy is Facet [55], which was designed to provide uncensored access to YouTube, Vine and Vimeo by playing the videos over an encrypted Skype call; since these videos are content-consistent the attacks on other "hide-within" schemes do not apply.

Given the important role of social media sites, finding an unobservable and difficult to block transport protocol for these systems is an important next step. Note that these services present several challenges not present in the social video context: they are not loss-tolerant, so they cannot rely on voice or video-based channels; they are authenticated, so the system should provide different privileges to different users; and the content provided is private, so it should be difficult for the circumvention to access or modify the content without the user's consent.

We present **Mailet**, an unobservable transport which provides unfiltered social website access by using email applications. Mailet servers and clients exchange the text content of a social media website via email. Specifically, the client sends an email to an inbox accessed by the server with the specified service details included; and on behalf of the client, the Mailet server communicates with the social website and emails back the response text, if any. This design guarantees channel consistency and has no imitation flaws. This makes Mailet immune to existing attacks. In addition, Mailet has several desirable features:

Mailet is secure against untrustworthy proxies. The Mailet design enables Mailet servers to provide privileged services without learning the social media login credentials of the client, using a threshold trust approach. In Mailet, the client is allowed to split and distribute the credential to a set of Mailet servers, and each server holds a share of the secret. Without learning the other shares, a single server can not recover the credential alone. When presenting the credential to the social website, Mailet servers should combine the shares privately. However, existing social websites do not support decentralized or privacy-preserving authentication, and expect to interact with clients through a single TLS session. Mailet solves this problem by proposing a highly efficient Galois/Counter Mode (GCM) based secure computation to enable the servers to recover the credential without disclosing their separate credential copies to each other. This computed TLS message contains the complete credential and is consistent with the social website interface. Comparing with the conventional secure two-party computation (2PC), which enables two parties to evaluate a function without revealing their inputs to each other [41, 58, 96], our approach can achieve a speedup of 120. This high efficiency can make a normal Mailet server to support up to 200 simultaneous sessions with each service request being completed in about 1 second.

Mailet users do not require additional software. Users can access Mailet through any standard mail client that supports STARTTLS, or any standard webmail service not controlled by the censor. This resolves the secure distribution or bootstrapping problem common in circumvention software.

Mailet resists proxy enumeration. Users interact with Mailet by sending email to mailboxes on widely-used mail services. Even if the censor learns the IP address of the Mailet servers, blocking direct connections is futile because users and servers never directly communicate. Thus blocking access to Mailet involves preventing users from sending and receiving email from users of all of the popular email hosting services on the web.

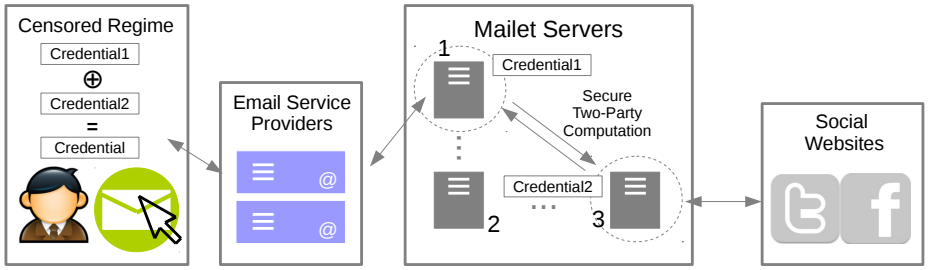


Figure 17: **Mailet System Architecture:** the user in the censored regime splits its credential into two copies, which are then distributed to two Mailet servers. This protects the credential of the user, while still being able to fulfill the user's service requests after the two servers running secure computation to recover the credential privately.

We have implemented the Mailet protocol for use with Twitter. We describe this implementation, which we release as open source software[4], and evaluate its performance and security as a circumvention tool both experimentally and analytically. We find that its performance is adequate and hope that other organizations will be willing to deploy Mailet servers as well, providing stronger security against server collusion and providing users with a free and secure alternative circumvention tool.

5.2 Mailet Design

The Mailet system uses email channels to facilitate communication between clients in a censored regime and social media websites. Instead of using email as a carrier for TCP/IP [97], Mailet directly transmits application-level content. This design reduces bandwidth requirements between clients and Mailet servers, while also increasing resistance to traffic analysis and differential channel attacks. The primary challenge associated with this design is protecting the clients' credential information. Social websites require a client to submit authentication credential in order to provide some privileged services, but cannot be relied on to provide a notion of limited delegation; thus for the Mailet system to represent a client it must have authentication credentials for that client's account. This poses a potential security threat to the client, since a malicious or compromised Mailet server could leak or otherwise misuse these credentials.

To protect the clients' credential information, Mailet is designed so that the system can *use* the credential even though no individual server will *know* it. In order to achieve this goal, we introduce a decentralized Mailet server structure, in which sub Mailet servers are controlled by separate parties, and clients distribute credential information among some of them, so that these servers can collectively represent the user while no individual server can access the credentials. In the following, we introduce the Mailet server design.

[4]https://github.com/magicle/Mailet

5.2.1 Architecture.

Mailet servers and clients communicate with each other by email. Clients send commands and message to be posted to servers via email, and servers use email to deliver site contents to clients. In the system architecture, there are four types of entities:

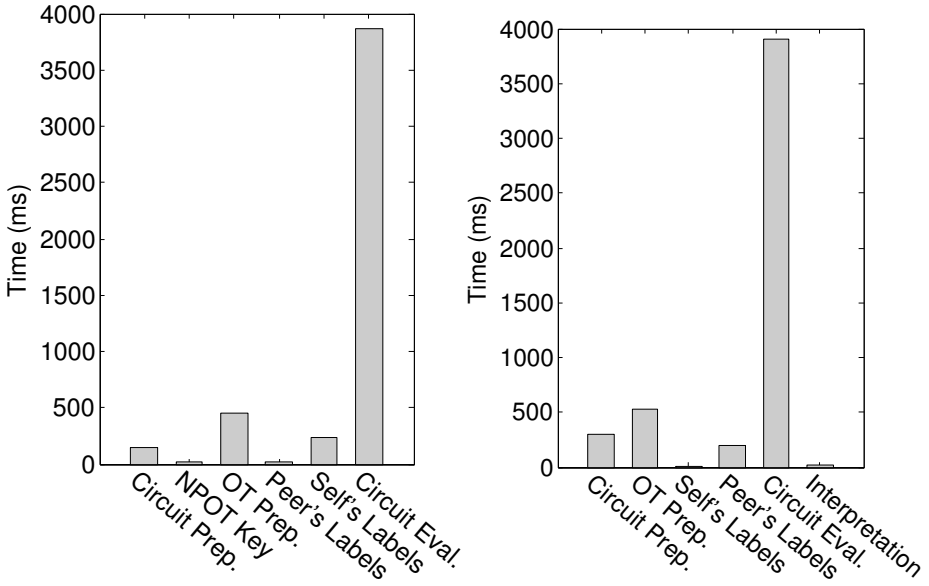
- Mailet *Clients* are assumed to be able to access an uncompromised email service, but are assumed to have no direct access to the social website, due to censorship or surveillance.
- *Email Service Providers.* The Mailet server and client can have different email service providers, and these providers are assumed to be uncompromised by the censor. We argue that this is a reasonable assumption. Since Mailet servers are outside of the censored regime, they can reach uncompromised email service providers. For Mailet clients, the abundance of independent email providers makes it difficult for a censor to compromise or filter all of them.
- Mailet *Servers.* The Mailet design includes multiple servers in order to protect the credential information of the clients. When a client first enrolls in Mailet, it distributes its credential information among two randomly selected servers, which collaboratively present the credential information to the social websites. Each server has its own (set of) email inbox(es) for communicating with clients.
- Social Websites. In our paper, we implement the Mailet system for Twitter. Social websites only accept direct connections via TLS, which cannot include third parties; thus Mailet's protocols must be designed to match this interface.

For connections that do not require credentials, such as searching tweets, the client can select a random server to provide the service. For example, if the user wants to search for tweets, the client can send an email with the command "searchtweet" plus the keyword to the server, which will search by the specified keywords and email the results to the client. However, most commands require the client to prove its identity to the website server by providing its credential. In these situations, Mailet servers use the decentralized credential mechanism described in the next section to allow transmission of the credential to the social website server while preventing its recovery by any Mailet server.

5.3 Decentralized Credential

Mailet clients need to share their credentials with Mailet servers when they requires privileged services from social websites, so that the Mailet servers can present the credential to websites on behalf of its clients. While this concept of the content proxy enhances Mailet in terms of unobservability and usability, it poses a potential threat to its clients' credentials if the Mailet servers are honest-but-curious or even malicious.

To protect the clients' credential, we propose the decentralized credential mechanism. Instead of "putting all the eggs in a single basket", Mailet design allows a Mailet client to



(a) Mailet Server A (b) Mailet Server B

Figure 18: **2PC Time Cost Breakdown:** Mailet server A holds the cryptographic keys of the TLS session and a part of the TLS plaintext, while Mailet server B having the other part of the plaintext. The computation ends up with a valid TLS message having the client's genuine credential. This TLS message is then presented by server B to social websites to finish the service request. RC4-SHA is used as the cipher suite of the TLS session.

split its credential into two copies, and distribute them in two separate servers (which are randomly selected by the client). For each server, its holding of one copy cannot enable it to recover the original credential. This decentralized credential design effectively protects the clients credential.

A challenge in decentralized credential mechanism is how to *privately* combine and represent the credential to social websites. In other words, the Mailet servers should collaboratively recover the original credential, and include it in a TLS connection to the social website, while still preventing each other from learning the other copy. This task is usually regarded as a secure two-party computation problem: two parties (Mailet servers) holding separate secret inputs (the credential copies) evaluate a common function (a TLS record message) without disclosing their inputs to each other. However, since a TLS record message in Mailet is usually large (several hundred bytes), a standard two-party computation is too costly. We implemented a credential recovery by using an optimized 2PC algorithm, and the costs (time, CPU and memory usage) are shown in Figure 18 and Table 3.[5] The results show that 2PC has to take nearly 6 seconds to finish and consumes about 6 MB bandwidth between Mailet servers. In addition, it uses about 90% CPU and 9% memory

[5]We adopted the FastGC framework [41] and implemented the 2PC in 1600 lines of Java.

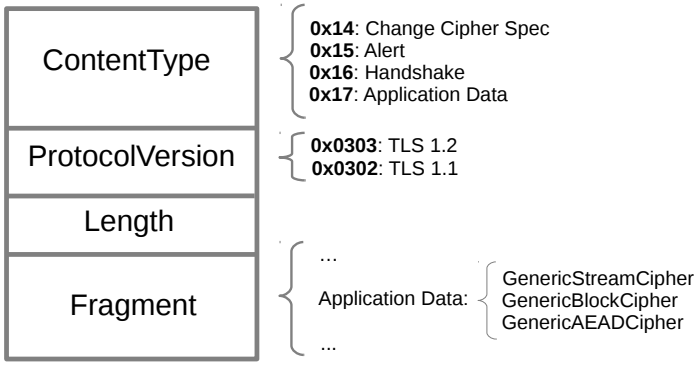


Figure 19: **TLS Record Format:** the general case

for each computer with a Quad-Core processor and 4GB memory.

To overcome this challenge, we propose a novel GCM based Credential Recovery (GCM-CR) approach to *secretly* combine the decentralized credential without using the high overhead secure two-party computation. This design uses Galois/Counter Mode (GCM) cipher suite in the TLS connection, and takes the advantage of Encrypt-then-MAC (EtM) of GCM mode to compute a valid TLS record message. Since this scheme involves no 2PC, a valid TLS record can be computed efficiently. Comparing with the conventional 2PC computation, our approach can achieve a speedup of 120. In our context, the speedup is equal to $\frac{L_{2PC}}{L_{GCM-CR}}$, where L_{2PC} and L_{GCM-CR} are the latency of the traditional 2PC computation and the GCM-CR approach, respectively. Furthermore, we propose Checking-by-Sampling (CbS) mechanism to enhance the Mailet design against a malicious Mailet server crafting malicious messages on behalf of the client. Before delving into the details about decentralized credential, we first briefly introduce the Transport Layer Security (TLS) protocol.

5.3.1 TLS Protocol

Social media users connect to the website server by TLS protocol. A TLS handshake protocol is firstly executed by both sides to negotiate the cipher suite to use, exchange random numbers, and agree on a common pre-master secret by public-key cryptography.

Category	Mailet Server A	Mailet Server B
OT Prep.	13.1	8.97
Label Transfer	0.79	0
OT Label Transfer	14.8	21.26
Circuit Evaluation	6631.13	0
Total	6659.82	30.23

Table 3: 2PC Downstream Bandwidth Consumption (KB)

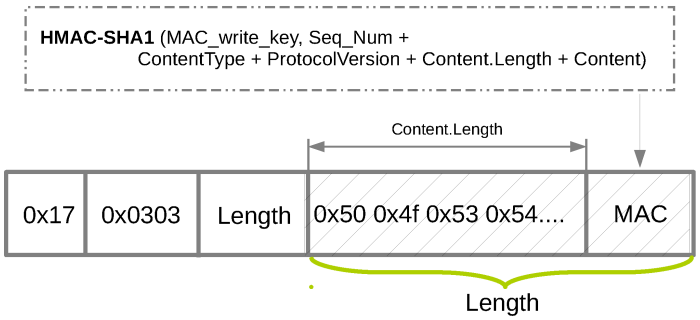


Figure 20: **Data Application Format:** under stream cipher encryption with HMAC-SHA1 as MAC algorithm

Afterwards, under the negotiated cryptographic parameters, users communicate with the website server by encrypted TLS application data, which may include the user's credential or the message to post. Then for an attacker, it can neither learn the application data nor forge a malicious message on behalf of the user or the website server.

The TLS message type can be figured out by any third party, which enables a Mailet server to intercept at the right time. The method is to examine the ContentType field. Figure 19 shows all the possible TLS message types and their ContentType field values. Besides of ContentType, a TLS message also contains ProtocolVersion field, Length field (the length of the Fragment in byte), and Fragment field. An example of the TLS format for application data under stream cipher encryption is shown in Figure 23. The content field is the encrypted application data, and the MAC field is the Message Authentication Code (MAC) for the application data, a sequence number, ProtocolVersion, ContentType, and length of the application data. The MAC is computed before the application data is encrypted. More details are given in the Appendix.

5.3.2 Credential Sharing and Recovering

Credential Sharing. Mailet's design incorporates multiple servers to store the client's credential information. For example, in Figure 17, the No. 1 and No. 3 servers are randomly chosen by the client to hold credential shares $Cred_1$ and $Cred_2$, separately. The credential generation method is as follows. First, the client generates a random string as the credential $Cred_1$, whose length is equal to that of the original credential. By XORing credential $Cred_1$ and the original credential, the client obtains the other credential $Cred_2$. Finally, the client distributes these two credentials to the chosen servers. Now neither of these two servers can recover the client's credential alone.

Credential Recovering. When the client initiates a command that requires credential use, the two Mailet servers S_1 and S_2 storing shares of the client's credential $Cred$ must collaborate to conduct a TLS session with the social website server that includes $Cred$. S_1 and S_2 could recover $Cred$ simply by jointly computing $Cred_1 \oplus Cred_2$, but this would not

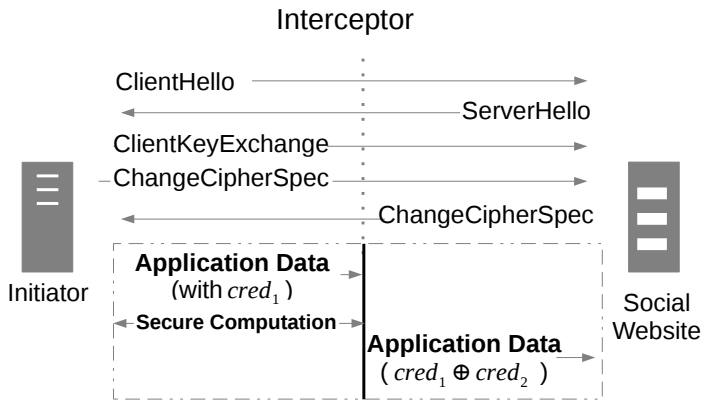


Figure 21: **Initiator and Interceptor Structure:** the Interceptor intercepts the TLS application data from the Initiator to the social media website. By collaborating with the Initiator, it regenerates a valid TLS application data which has the genuine credential.

be secure since then both servers would know $Cred$.

The approach in this section for privately generating the *valid* TLS record message relies on the *Initiator-Interceptor* structure. Particularly, we assign the servers asymmetric roles: one server is the *Initiator*, and the other is the *Interceptor*, shown in Figure 21. The initiator initiates the client side of the TLS handshake with the social website server, using the interceptor as a proxy to pass messages to the server, so that from the point of view of the social website, the connection originates from the interceptor. At the conclusion of the handshake, the initiator and the social website server share symmetric keys, so the interceptor is unable to decrypt the traffic passed to the social website.

After the TLS handshake, the initiator continues to forward TLS records through the interceptor to initiate the application-level session; once the TLS record containing only $Cred_1$ arrives at the Interceptor, the interceptor holds on this message, and regenerates a valid TLS record having $Cred = Cred_1 \oplus Cred_2$ with the help of the Initiator. Finally, the Interceptor sends the regenerated message instead of the original TLS message. The approach to regenerating a valid TLS record is described in the following.

5.3.3 GCM based Credential Recovery

We introduce the credential recovery without involving 2PC. The GCM-CR approach uses the cipher suites of GCM mode in the TLS session, and takes the advantage of GCM's stream cipher property and EtM feature.

Galois/Counter Mode (GCM). GCM is an operation mode for symmetric key block ciphers. It is an authenticated encryption algorithm, and can provide data confidentiality and integrity simultaneously. Due to its high speed with low cost and low latency, GCM mode has been included in TLS cipher suite list, and widely implemented by popular web-

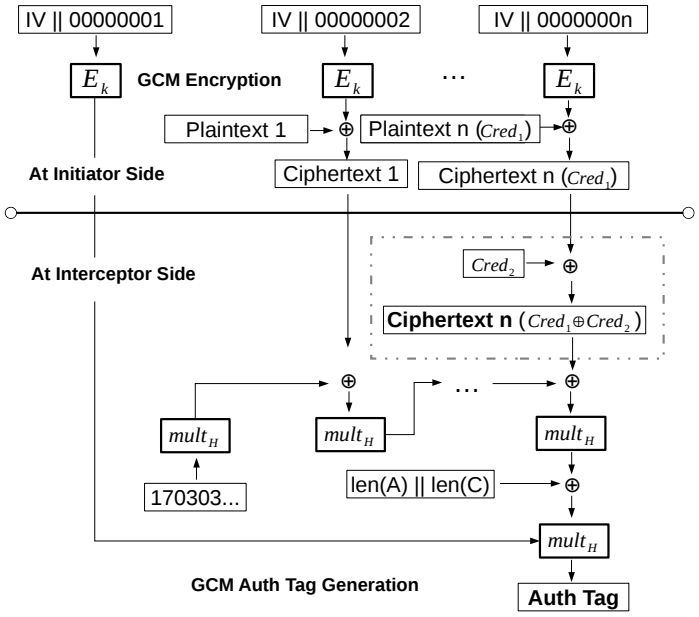


Figure 22: **Credential Recovery with GCM Mode:** the Interceptor receives H and $E_k(IV||00000001)$ from the Initiator and creates a valid Auth Tag for the ciphertext including the genuine credential.

site servers.

Recover the Credential in Ciphertext. For the encryption, GCM resembles the counter mode encryption, and turns a block cipher into a stream cipher. This makes the credential recovery in the ciphertext convenient. For the Initiator, it encrypts the TLS record message with $cred_1$ in place of the original credential $cred$, and passes this message to the Interceptor. The Interceptor can locate the credential $cred_1$ with the help of the Initiator, and XOR $Cred_2$ with the ciphertext bytes in this location, so that when the website server decrypts the TLS message, the plaintext will contain $Cred$. Since the Interceptor does not know the symmetric key, it cannot recover $Cred$, while the initiator does not see the completed record, and also cannot recover $Cred$. However, recovering the credential in the ciphertext/plaintext alone is not sufficient. The Interceptor has to generate a correct MAC to make the TLS record message valid.

Validate the TLS Record. In GCM mode, the plaintext is first encrypted, then an authentication tag is computed based on the ciphertext by a GHASH function. This Encryption-then-MAC (EtM) property enables the Interceptor, which has access to the ciphertext, to compute a valid authentication tag without having to do a 2PC with the Initiator. The authentication tag generation is shown in Figure 22. Note that the Initiator should share the H (the encryption of 128 bit 0s) and $E_k(IV||00000001)$ (the encryption of

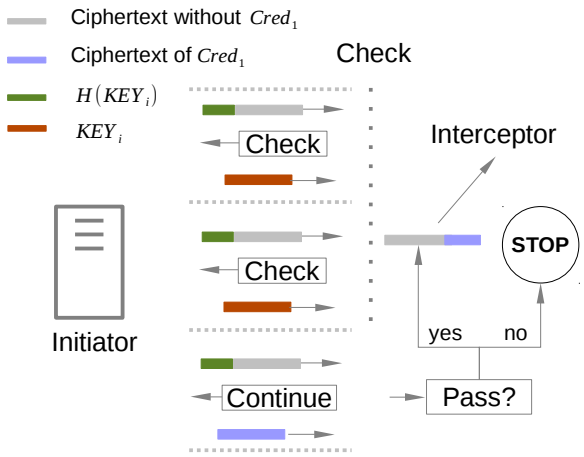


Figure 23: **Checking-by-Sampling:** the Interceptor randomly chooses $n-1$ sessions to check the correctness of the non-credential HTTP fields. The commitment $H(KEY_i)$ is required to force the Initiator to provide the true TLS session key in the latter phase.

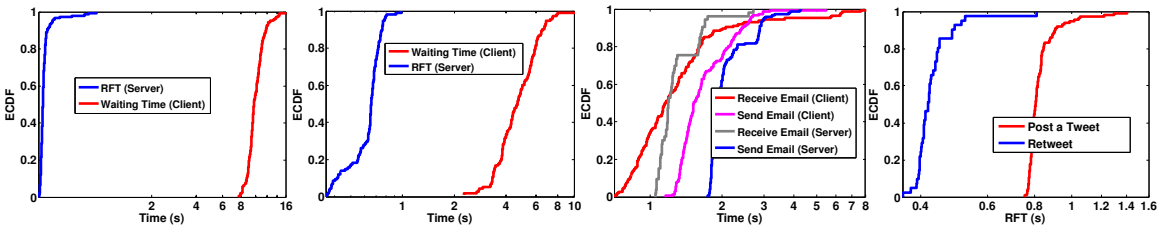
the first counter) with the Interceptor. This does not break the security of the cryptographic system. It leaks neither the TLS session key nor the Initiator's credential $Cred_1$.

Though GCM-CR can effectively protect the clients' credentials in the honest-but-curious attack model, it cannot prevent a malicious server from corrupting the protocols. Both Initiator and Interceptor can maliciously flip the bits in other fields of the TLS record message without being noticed by each other, so that they can change the APP ID to be authorized and the message to be posted, etc.. In the following, we give a solution to prevent the Initiator from crafting malicious messages.

5.3.4 Interaction Integrity

By applying the decentralized credential mechanism, A corrupted Mailet server is prevented from knowing the user's credential. However, this server might seek to modify the TLS message in a covert way to manipulate the outcomes of the protocol. An incorrect tweet may be posted, or a wrong third-party App is authorized because of this attack. In this section, we propose approaches to detect and prevent such attacks.

A Corrupted Mailet Interceptor. This Interceptor may flip the bits of the TLS ciphertext to manipulate a HTTP field value in the GCM-CR. To prevent such attacks, Mailet can randomize the order of fields in requests, and pad requests with separator strings of random length, making it difficult to predict the location of the desired field. In addition, an Initiator can screen the response from Twitter for a corrupted Interceptor. For example, when posting a tweet, the Initiator receives a response including the tweet ID, which allows it to retrieve the tweet. An inaccurate tweet indicates the presence of a corrupted Interceptor.



(a) Retrieve Self's Own Tweets (b) Search by Keywords (c) Send/Receive Emails (d) Comparison: Twitter API

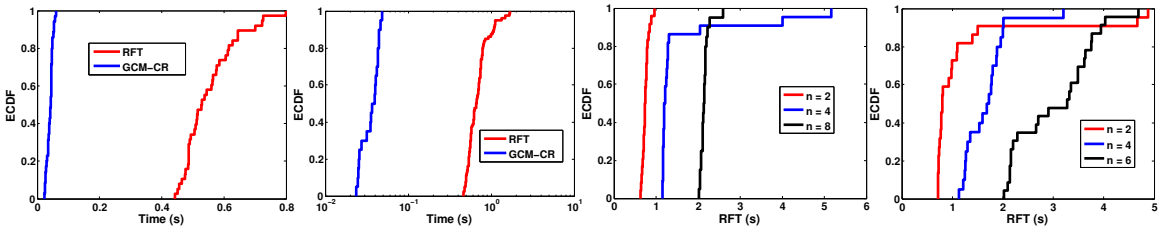
Figure 24: **Time for None-Privileged Mailet Services (s):** (a) and (b) give the ECDF of a client's waiting time and the Mailet server's Request Fulfillment Time (RFT); (c) shows the ECDF of the email channel's delay; (d) represents the ECDF of the RFT when the authorized Mailet server fulfilled the privileged services by the Twitter APIs as a comparison with the GCM-CR based approach.

A Corrupted Mailet Initiator. Violating the interaction integrity is much easier when the corrupted Mailet server takes the role of the Initiator. This is because this malicious Initiator can craft any arbitrary plaintext of the TLS message in GCM-CR. In order to enhance Mailet design against a malicious Initiator, Checking-by-Sampling (CbS) mechanism is proposed. Instead of initiating a single TLS session with the website server, the Initiator starts n parallel sessions which are all passed through the Interceptor. For all these sessions, the Initiator cuts off the ciphertext of its credential copy and passes only the rest of the TLS record messages with n commitments to corresponding TLS session keys. The commitment for each TLS session can be the cryptographic hash of the session keys. For the Interceptor, it chooses $n-1$ out of n TLS sessions to open by requesting the Initiator to provide $n-1$ corresponding session keys. The Interceptor checks the commitments, and if malicious messages are detected, it stops the collaborative computation with the Initiator. Otherwise, the Initiator is believed to be honest and the only TLS session left is used for credential recovery. Before doing so, the Initiator should complete this TLS record by providing the Interceptor with its credential ciphertext.

This enhancement can practically prevent an malicious Initiator. Even for a smart malicious Initiator, it only has $1/n$ probability to succeed for each session. Being malicious for multiple sessions, the malicious Initiator would be probably detected and reported.

5.4 Performance Analysis

This section measures the Mailet system from the perspective of performance. It evaluates the CPU, memory, and time cost of the Mailet servers, and demonstrates the feasibility of Mailet.



(a) Retweet (b) Post a Tweet (c) Post a Tweet with CbS (d) Retweet with CbS

Figure 25: **Time for Privileged Mailet Services (s):** (a) and (b) show the ECDF of Mailet servers' Request Fulfillment Time (RFT); (c) and (d) measure the ECDF of RFT when CbS is adopted.

5.4.1 Mailet Server Overhead

This part evaluates the Mailet server's overhead in terms of CPU and memory usage. Before giving the experimental results, we first introduce the experiment setup.

Experiment Setup. We started our experiment on two desktop computers A and B, which were connected by a local area network (LAN). The computers are Dell Precision T1500s, with the Ubuntu 14.04.1 LTS operating system. The processor for each computer is an Intel Core i7 CPU 860 (Quad-Core) 2.80GHz. The memory size for computer A is 8GB, and for computer B it is 4GB. They both have 8GB for swap space. The Initiator resided at B, and A was used as the Interceptor. The password in the credential is 10 bytes long, and the account is 17 bytes long.

We used a laptop as the Mailet user, which is a Lenovo Thinkpad T400 with an Intel T9900 processor and 6GB memory. This user sended service requests to the two Mailet servers over the Internet, and the services included authorization, posting a tweet, and retweeting. At the servers' sides, after receiving a request, they started to log the accumulative CPU and memory usages of the Mailet system until this request is fulfilled. The final measurement results are the average of multiple trials. In addition, the system cost with Check-by-Sampling mechanism was also logged as a comparison.

Evaluation Results. Table 4 gives a summary about the CPU and memory usage for the Initiator and Interceptor. In the table, the cost of the services (authorization, posting a tweet, and retweeting) is the cost of the Initiator and the Interceptor handling a single service request. Note that this measurement excludes the cost of the email clients for clarifications. The results show that the whole Mailet system (including email clients) only consumes less than 5.3% of the CPU and at most 1.0% of the memory space. If the Mailet servers adopt the Checking-by-Sampling mechanism to avoid a malicious Initiator, the overhead still remains low. Take retweet as an example. When no CbS approach is applied ($n = 1$), the CPU usage is 2.0% for the Initiator and 1.8% for the Interceptor. When CbS is applied with $n = 8$ (which means the Initiator starts 8 TLS sessions), the CPU usage only rises to 3.15% and 2.0%. In both cases, the Initiator consumes 0.1% memory while the Interceptor

occupies 0.2% memory. These results show the high efficiency of our GCM-CR approach.

5.4.2 Service Measurement

This part gives the performance analysis for Mailet. In the experiment, we measured the Mailet service in the following metrics:

- *Mailet User's Waiting Time.* This is the period of time between when a user makes a request and when this user receives the response from the Mailet servers. This time includes the time of sending and receiving the emails at both sides of the Mailet user and the server, and the time of the server fulfilling the request by the decentralized credential or Twitter API calls.
- *Request Fulfillment Time (RFT).* RFT is the period of time that the Mailet server needs to handle a request. For non-privileged services, RFT is the time that the server takes to complete the Twitter API call. For privileged services, RFT refers to the time cost of fulfilling a request by the decentralized credential mechanism. Note that RFT excludes the time cost of the email clients.
- *GCM-CR Time.* This refers to the period of time that the Interceptor spends on recovering the credential in ciphertext and calculating a valid authentication tag. Note that this time cost includes the time spent on the Initiator's transmitting parameters such as H. For privileged services, the GCM-CR time is part of the RFT, and therefore it is shorter than the RFT.

For each kind of requests, we logged the above metrics and had 100 trials. For each metric, the final measurement is the average of the 100 results. The experimental results are shown in Figure 24 and Figure 25.

None-Privileged Service. Figure 24 (a) gives the time in retrieving the client's own tweets. Since collecting a user's tweets does not necessarily require the permission from this user, we implemented by using the Twitter API without having the user's credential. For the client, the waiting time is from 8 seconds to 16 seconds. On the server side, with 90%

Initiator		Category	Interceptor	
CPU	MEM		CPU	MEM
0.4%	0.8%	Email Client	0.5%	0.2%
4.88%	0.2%	Authorization	1.71%	0.1%
2.25%	0.1%	Post a Tweet, n=1	1.3%	0.2%
4%	0.1%	Post a Tweet, n=8	1.7%	0.2%
2.0%	0.1%	Retweet, n=1	1.8%	0.2%
3.15%	0.1%	Retweet, n=8	2.0%	0.2%

Table 4: CPU and Memory Consumption for Mailet

probability, the API calls can be completed in less than 0.2 seconds. These results show that the time cost of the email conversations contributes most to the client's waiting time.

Figure 24 (b) shows the time for the service of searching by keywords. The Mailet server is configured to return 20 search results in a session, and RFT is less than 1 second. At the client side, the waiting time is in the range of 3 seconds to 8 seconds. It is worth noting that the search service can be completed without the user's credential. Consequently we implemented this service by using the Twitter API.

Figure 24 (c) presents the time of sending/receiving emails for the Mailet client and the server. The x-coordinate is set in log scales, and the y-coordinate is the Empirical Cumulative Distribution Function (ECDF). Both the Mailet client and server have to spend at least 1 second sending or receiving emails with 50% probability.

To compare GCM-CR with the normal Twitter service access, Figure 24 (d) gives RFT for posting a tweet/retweeting by Twitter API at the server side when this server is authorized by the user. The server uses about 0.5 seconds to retweet, less than the time of its posting a tweet, which is in the range of 0.8 to 1.2. For GCM-CR, the server uses about 0.6 seconds to retweet, and 0.8 seconds to post a tweet. This comparison demonstrates the high efficiency of GCM-CR.

These experimental results show that the email conversations contribute the most to the client's waiting time, while most Twitter API calls can be completed by the Mailet server in less than 1 second. These results demonstrate that Mailet can provide quick access to non-privileged services of Twitter for its clients.

Privileged Service. The measurements for privileged services are given in Figure 25.

Figure 25 (a) (b) shows how fast GCM-CR can be completed, and how quickly the Mailet servers handle the services of posting a tweet or retweeting. It shows the GCM-CR time is under 0.05 seconds, which has a speedup of 120 when being compared with the conventional 2PC computation. In addition, the figures show that the Mailet servers can handle a post request in 0.8 seconds and a retweet request in 0.6 seconds on average.

Figure 25 (c) and (d) shows the RFT when the Checking-by-Sampling strategy is applied. For the post request, when a larger n is adopted (which means the Initiator starts more TLS connections), the RFT is increased. If $n = 4$, it takes the Mailet servers about 1.2 seconds to complete a post request. For the case $n = 8$, the RFT is about 2 seconds. For the system deployer, it can tune the parameter n to make a trade-off between usability and security. It is worth noting that if long-term detection is applied in Mailet deployment, a small n (for example $n = 2$) is suitable.

The experimental results demonstrate the high performance of the Mailet design, while having small CPU and memory usages.

6 Streaming over Videoconferencing for Anti-censorship

6.1 Introduction

As the Internet has become a more useful tool for communicating information between individuals, censors working for Nation State Adversaries have responded by blocking access to the unfiltered versions of these sites. Additionally, these censors have also deployed increasingly sophisticated tools to identify and block access to the tools designed to circumvent this censorship, such as Tor [20] and other proxy services. These tools include sophisticated protocol fingerprinting via deep packet inspection (DPI), and even active probing attacks, in which suspected relays are contacted by the censors in order to confirm participation in the Tor protocol.

In response, researchers have developed systems that attempt to provide proxy steganography, which intend to make proxy connections resemble innocent "cover" protocols. For example, "decoy routing" [35, 47, 95] systems make connections to relays resemble TLS connections to random websites by hiding the relays in routers; SkypeMorph [61], CensorSpoofer [83], StegoTorus [88] and FreeWave [37] attempt to make proxy connections look like VoIP calls; and Collage [13] hides information in photos posted to content-sharing sites such as Flickr.

However, recent research [27, 33, 74] has revealed that these systems fail against more sophisticated censors due to several different inconsistencies between the proxy protocol and the cover protocol:

- *Emulation* inconsistencies can occur because either the client or proxy does not perfectly implement the cover protocol. StegoTorus' HTTP module does not respond to requests in the same way as any well-known HTTP server, and SkypeMorph does not simulate the TCP control connections [33].
- *Channel* inconsistencies can occur because the cover protocol responds differently to channel behavior than the proxy protocol, allowing an adversary to disrupt proxy connections while minimally impacting innocent connections. Decoy routing implementations can fail if the censor distributes packets across multiple AS paths to the "overt" destination, whereas TLS connections will not be affected [74]; VoIP and videoconferencing protocols are typically loss tolerant whereas proxies fetch general data and cannot tolerate packet loss [27].

- *Content* inconsistencies can arise when the behavior of a cover protocol depends on the characteristics of the traffic it carries and proxies do not match content to these characteristics; for example, since the FreeWave server tunnels modem traffic instead of voice signal over a VoIP channel, its communication session can be identified by traffic analysis [27, 89, 91, 92].

In light of these potential problems, finding a single cover protocol to carry arbitrary Internet content seems difficult. However, a recent survey of Chinese users of circumvention tools [2] found most users circumvent the Chinese "Great Firewall" to use three services: unfiltered search engines such as Google, uncensored social networks such as Facebook and Twitter, and video sharing sites like YouTube and Vine. This raises the possibility of serving most circumvention needs through a small set of unobservable transports.

In this paper, we present Facet, a system that enables the clients in a censored regime to watch YouTube, Vine and Vimeo videos in real-time. The basic idea of Facet is to send videos from these sites as the video content of a videoconferencing call – in the case of our prototype, a Skype call – between a Facet server and a client. Like all proxy steganography systems, it relies on the assumption that the censor is unwilling to indiscriminately block all or most sessions of the cover protocol (Skype) to avoid "collateral damage". Under this assumption, Facet provides the following features:

- Facet is **Emulation Consistent:** because the video is transmitted over an actual two-way Skype call, there is no difference between implementations to allow identification.
- Facet is **Channel Consistent:** we transmit videos over a channel intended for videos, so any disruption to a Facet session would cause the same disruption to a regular call.
- Facet is **Content Consistent:** Arbitrary videos may have different characteristics from videoconferencing calls, leading to detectable differences in packet sizes. We implement a binary classifier similar to the approach from Wright *et al.* [92] and show that unaltered YouTube videos sent over Skype are distinguishable from Skype calls. To defeat this, we introduce "video morphing," in which the Facet server frames the requested video within a randomly selected videoconference call. This increases the false positive rate of a classifier that can recognize 90% of Facet calls to nearly 40%.
- As a result, Facet provides **Unobservability**. Since videoconferencing streams are encrypted in transmission, it is difficult for censors to detect Facet sessions. Even if the videoconferencing software (or servers, in case calls are not routed directly between peers) is compromised, the use of randomized video morphing forces the censor to decode and analyze all video calls in real-time to detect Facet sessions.
- Facet's throughput is high enough to provide **real-time video delivery.** Most steganographic anti-censorship tools are designed for regular web browsing, and often have limited bandwidth for clients. In contrast, Facet is aimed at delivering real-time video service for clients, and achieves the same throughput as the videoconferencing service.

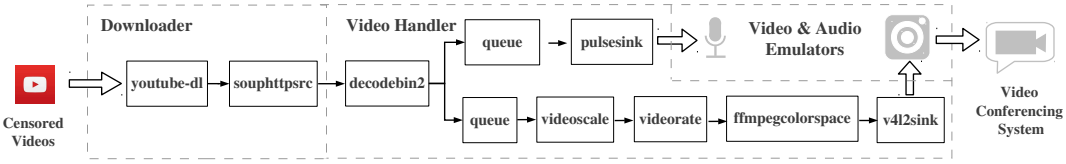


Figure 26: **Facet Pipeline:** to deliver censored videos in real-time

- Our approach is **provider independent.** Since the emulator devices in Facet are built independently from the videoconferencing systems, Facet can be adopted widely on any conferencing platform, such as Google Hangout, Skype, FaceTime or QQ. This feature not only provides accessibility to users who have access to different videoconferencing systems, it also provides the ability to evade blocking targeted at a single protocol or implementation.
- **No deployment at client side.** For Facet clients, there is no need to install any client software (which is often blocked), or to pre-share secrets with the server. This property makes Facet easier to use and maintain, since software updates only need to be applied by servers outside of the censored region.

We built a proof-of-concept implementation based on Skype videoconferencing service, and tested it in a real-world environment.

6.2 The Facet Design

Facet delivers censored videos in real-time. As is shown in Figure 17, the procedure of a Facet connection is:

1. The Facet server distributes its conferencing ID for service discovery. The distribution can be public or private, depending on the architecture of the videoconferencing system.
2. A Facet client sends a contact request to the server. After the server accepts the request, they establish initial connections.
3. The client sends the Uniform Resource Locator of the censored video to the server by an instant message or an email.
4. The server extracts the client's request, initiates audio & video emulators, and places a conferencing call to the client.
5. Simultaneously, the server forwards the URL to the Facet pipeline, which will download, decode, and resize the requested video, finally streaming it into the emulator devices.
6. After accepting the videoconferencing request, the client can watch the video in the videoconferencing session.
7. The client can also send control commands to the server for video playback or adjusting video speed.

8. After the video is over, or the client ends the conferencing, the Facet server destructs the emulators, and ends the session.

Navigation. For the Facet clients, an important question is how to navigate and discover the URL of the censored video. There are three methods for discovery.

Encrypted Video Search. For regimes where encrypted web search services (provided by Google, etc.) are not blocked, the client can use such a service for navigation. The client can specify the keywords and websites for video searching, and the search engine will return the results with the URLs of the videos.

Video Subscription. The client can also use a subscription service. For websites such as YouTube, the client can make a subscription to the videos, and periodically it will receive emails including subscribed video information, such as URLs and titles.

Search Engine Proxy. Facet implementation also includes a search engine proxy using email tunnels. The client can email search keywords to the Facet server, which will fetch the search results, and email a screenshot of the page back to the client. Then the client can find the videos to watch and their URLs The email address of the Facet server can be publicly distributed, and the unobservability of the email tunnels guarantees its security [97].

Service Discovery. The Facet server has two strategies to distribute its conferencing ID.

Public Distribution. For centralized videoconferencing systems, such as Google Hangout and FaceTime, the Facet conferencing ID can be publicly distributed. Though this strategy also discloses the ID to the censor, it does not increase the censor's ability to block the service. Since the videoconferencing traffic is encrypted the censor can not link the conferencing ID to a specific session to block. Even though the censor may proactively probe the service, it can not pinpoint the Facet server IP address, because this address is hidden behind the videoconferencing server. Thus, the censor's ability to distinguish and block the Facet session is not improved, even when it knows the server's conferencing ID.

Private Distribution. For decentralized videoconferencing systems such as Skype, the two entities send traffic to each other directly. Consequently, the Facet server ID should only be distributed privately. Otherwise, the censor can pinpoint and block the Facet server IP address by proactively probing the service.

Security. The censor may block the potential Facet session in which the video stream is only unidirectional. In this situation, the Facet client should enable the camera in the conferencing. To prevent denial-of-service (DoS) attacks, the Facet server is configured to not accept strangers' requests. Thus, a potential Facet client is required to register with the server, by sending an "add contact" request to the server's conferencing ID. Only after this request is proved by the Facet server, can the client access to the service. Also, the Facet server enforces usage limits on each registered client ID to further defend against DoS attacks.

Element	Fuctionality
souphttpsrc	receive HTTP network data as a libsoup client
decodebin2	decode and demultiplex the data stream from souphttpsrc
queue	audio/video stream queue
videoscale	resize the received video frame to match the camera emulator
videorate	manipulate the timestamps on video frames to adjust frame rate
pulsesink	direct audio to the PulseAudio server
v4l2sink	play the video in v4l2 device
ffmpegcolorspace	convert the video from one colorspace to another

Table 5: Gstreamer Elements

6.3 Implementation

The Facet server is implemented by selecting Skype as the videoconferencing system. The server is built on Ubuntu 12.04 Precise Pangolin, and can support most popular video sites, such as YouTube, Vine, and Vimeo.

6.3.1 Facet Pipeline

The real-time delivery of the censored videos requires Facet to construct a pipeline to handle video downloading, decoding, and playing in parallel. The pipeline implementation mainly relies on Gstreamer [3], an open source multimedia framework. A typical Facet pipeline is shown in Figure 26, which consists of the Downloader, Video Handler, and Camera & Microphone Emulators.

Downloader. Popular video websites usually utilize HTTP based dynamic video streaming [40], so the downloader needs to decode the video URL to obtain the actual data streaming address. This address is obtained by utilizing `youtube-dl`, which is an open source video downloading toolkit, and can support websites such as YouTube, Vimeo, Vine, and MetaCafe. Then, this obtained streaming address is forwarded to a Gstreamer element `souphttpsrc` to download the video. It is worth noting that `souphttpsrc` can directly forward the received stream to the video handler, without having to wait until downloading the entire video.

Video Handler. The video handler is implemented with the Gstreamer framework to convert the downloaded stream live. The functionality of Gstreamer elements is listed in Table 5. The stream is split by `decodebin2` to obtain the video & audio stream. Each is placed into its respective emulator for playing. Since the downloaded video stream may fail to satisfy the emulator's requirement in colorspace, resolution, and frame rate, additional elements such as `videoscale`, `videorate`, and `ffmpegcolorspace` are utilized to make the conversion.

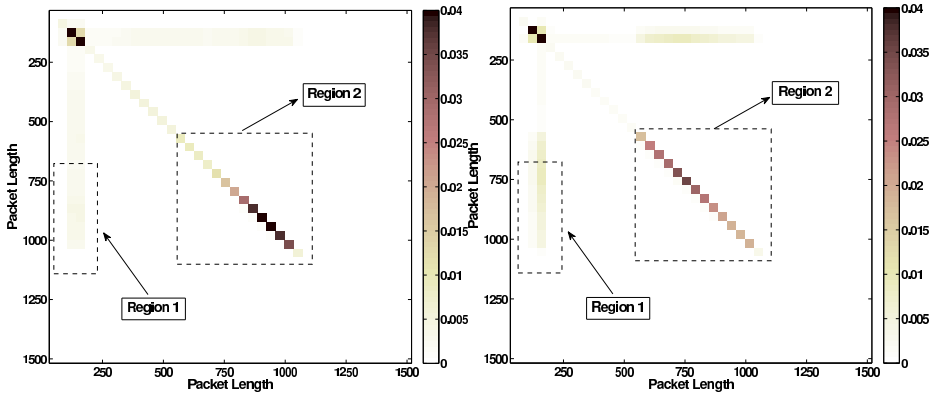
Emulator. Facet initiates two emulators to deliver the video & audio stream. For the camera emulator, our Facet implementation utilizes `v4l2loopback`, a kernel module to

create a v4l2 device emulator. For the microphone emulator, Facet utilizes `pactl`, a program controlling PulseAudio sound server, to initiate a microphone device instance. Both of these two emulators can be recognized by the conferencing systems.

6.3.2 Other Implementation Details

The Facet server needs to initiate or end videoconferencing request automatically. For our implementation, we use `skype4py` [6] for automation, which is a python wrapper for the Skype API. Also, the server can run multiple videoconferencing sessions to serve more clients. Our implementation shows for a Facet server which has 15 Mbit/s bandwidth and 4 virtual cores, it can support up to 20 simultaneous sessions.

Another implementation detail is URL submission. Facet supports instant message submission: the user can give the video URL to server by using the videoconferencing's instant message service. Facet also supports email submission, which is more secure when the videoconferencing provider is considered to collude with the censor. A detailed analysis is given in Section 8.



(a) 2-gram Distribution for Chat Videos (b) 2-gram Distribution for YouTube Videos

Figure 27: **Traffic Analysis:** Chat and YouTube videos have different traffic patterns.

6.4 Traffic Analysis

Since videoconferencing adopts VBR codecs and length preserving encryption, the packet length can leak information about the content being transmitted. Previous research shows the rate of distinguishing phrases, languages, and even speaker identity in a VoIP conversation. Thus, it is necessary to study whether the censor can distinguish the Facet connection by traffic analysis.

Classifier. [92] reveals a χ^2 classifier to distinguish the language in a VoIP call. With about 90% accuracy, the classifier can narrow down from the two possible languages to one. We adopt this best known binary classifier, and investigate whether a censor can

identify the Facet session, or in other words, how accurate it can determine whether the videoconferencing is genuine or not.

The χ^2 classifier takes the packet length as input, and adopts n-gram as feature extraction, which is a contiguous sequence of n packet lengths from the time series traffic. Suppose the traffic is (a, b, c, d), where a, b, c, and d are the packet lengths, then the 2-grams are (a, b), (b, c), and (c, d). The reason for not including traffic delay in the classifier is that this feature is not stable. It can be easily affected by network conditions. Besides, the delay in VoIP traffic is usually fixed [91]. The packet length is discretized into equal partitions of size K before calculating the n-gram, to avoid the curse of dimensionality and improve the classification accuracy. $G_K(n)$ is used to denote the set of all the possible discretized n-grams. Then, for a given traffic, the probability over each element of $G_K(n)$ can be used as its fingerprint for classification.

Training. Let T_0 denote the set of genuine chat videos, and T_1 denotes the set of the censored videos in the training process. The models for the genuine chat and censored videos are built as follows:

$$\bar{Pr}(i,g) = \frac{1}{\sum_{v \in T_i} N_v} * \sum_{v \in T_i} N_v * Pr(i,v,g), \;\; g \in G_K(n) \tag{16}$$

where N_v denotes the number of grams for video v.

Classifying. The original classifier assigns a test case to the category whose model is closer to that of the test case in terms of χ^2 distance. This decision rule only allows fixed false positive/negative rates. Considering a censor should be able to adjust its aggressiveness in blocking unwanted connections, we use the following rule:

$$\Theta = \frac{\Delta(v, T_0, G_K(n))}{\Delta(v, T_1, G_K(n))} \quad \begin{matrix} \mathcal{H}_0 \\ \lesseqgtr \\ \mathcal{H}_1 \end{matrix} \quad \delta \tag{17}$$

where $\mathcal{H}_0$ represents the video is classified as chat video, and $\mathcal{H}_1$ denotes it is determined to be a censored video. Θ is the χ^2 distance ratio, and $\Delta(v, T_i, G_K(n))$ is the χ^2 distance between video v and training set T_i:

$$\Delta(v, T_i, G_K(n)) = \sum_{g \in G_K(n)} \frac{[\bar{Pr}(i,g) - Pr(i,v,g)]^2}{Pr(i,v,g)} \tag{18}$$

This allows the censor to adjust the value of δ to change its blocking strategy. In the experiment part we will discuss how the censor chooses a proper δ value in order to block a given percentage of unwanted traffic. In addition, the classifier includes the gram selection algorithm introduced in [92]. The general idea is to exclude the grams which have negative influence on classification results.

For our dataset, we use the 2 fold cross-validation in evaluation [48]. The dataset is randomly separated into two groups d_0 and d_1 with equal size. Each group will alternatively be the training set in two rounds.

Accuracy. The censored video dataset consists of 1013 popular YouTube videos, and 1045 YouNow videos are used as the chat video set (details are given in section 9). With $n = 2$ and $K = 50$ the experimental results are shown in Figure 27 and Figure 28. The classification is measured by false positive rate, the probability of wrongly determining the traffic of chat videos as that of YouTube videos, and the false negative rate, the probability of determining the traffic of YouTube videos as that of chat videos [76]. It shows with a rate of 90%, the censor can correctly distinguish the traffic pattern of the streamed YouTube video, with only the cost of 10% false positive rate. Another observation is that the censor can even adjust needs to disrupt only 2% of genuine videoconferencing connections to block 80% of Facet connections. This result is further demonstrated by the difference on averaged traffic patterns between chat videos and YouTube videos in Figure 27(a) and 27(b). Thus, traffic morphing is necessary to protect Facet from blockage.

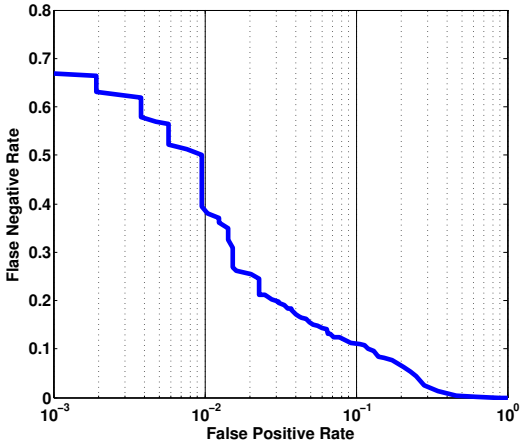


Figure 28: **Traffic Analysis:** the receiver operating characteristic (ROC) curve of the classifier

6.5 Morphing

Video & audio morphing are introduced to defend against the censor's traffic analysis. An intuitive method for traffic shaping is to manipulate packets, but [33] demonstrated the flaws in such morphing. In Facet, the property of video & audio is transformed to simulate that of chat video & audio, modifying the traffic pattern without packet dropping.

6.5.1 Audio Morphing

Observation. Region 1 in Figure 27 reveals for genuine conferencing, there are more packets with a length in the range of 100 to 150 bytes and less packets with a length in the range of 150 to 200 bytes as compared to Facet streaming the YouTube videos. The primary reason for this difference is that the conferencing audio has lower quality, and shorter packets can be used to transmit the audio. This difference is further demonstrated by the discrepancy in sampling rate. The relation between audio quality and sampling rate

is that the higher the sampling rate, the better the quality [4]. For our chat video dataset, the sampling rate is 11,500hz, but a typical YouTube video has a sampling rate of 44,100hz. In addition, the chat videos have only one channel with 16 bit width, while YouTube videos have two channels with 32 bit width.

Audio Morphing. The Facet audio is resampled live to simulate the quality of the chat audio, and this requires Facet pipeline to include a `audioresample` element. In the following, we take the YouTube audio as an example, and show how to determine the resampling rate empirically. For the chat video dataset, the video & audio is streamed simultaneously, while for the YouTube dataset, YouTube audio and chat video are streamed. Thus, the difference in traffic pattern is only related with the audio. Figure 29 (a) shows when the sampling rate is 3,000, the morphing is at its best. Also, the YouTube audio which usually has two channels is converted into mono-channel, and the bit width is set to 16.

It is worth noting that though the classifier can still do better than random guessing, the audio morphing can *practically* disable the detection as is analyzed in Section 8.

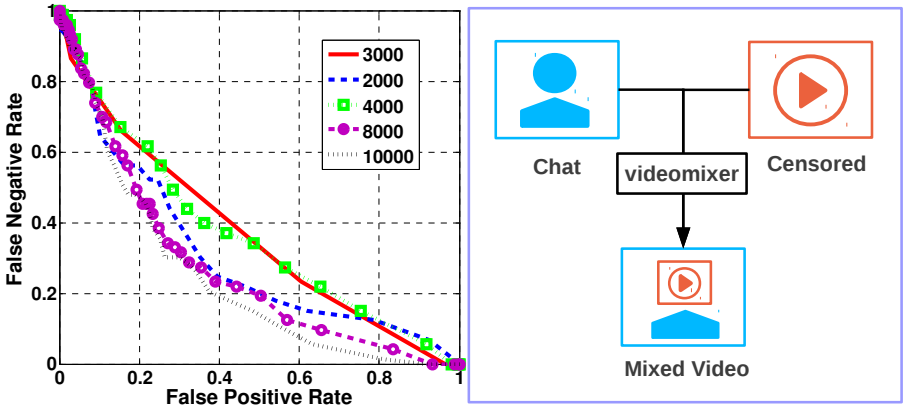
6.5.2 Video Morphing

Observation. Region 2 of Figure 27 shows chat video traffic has more large packets than that of YouTube videos. Our results show chat videos have more packets in the range of 400 to 600 bytes, and 800 to 1000 bytes than YouTube videos. The possible explanation is that the chat videos are usually slower motion which causes the video encoder to handle them differently.

The encoder can use the temporal redundancy of the slow motion video to optimize its coding efficiency. Take the H.264 codec as an example. There are mainly three types of frames in H.264: the I frame, B frame, and P frame [5, 45]. The I frame is encoded independently from other frames, only exploiting the spatial redundancy to compress the video stream. Differently, the P frame and B frame depend on the previous frames or even future frames, taking advantage of temporal redundancy. Typically, the I frame is significantly larger than the P and B frames. Thus, for videos with different temporal redundancy, the encoding results are different.

Video Morphing. The video morphing in Facet is based on the block-oriented feature of H.264 codecs. In H.264, each frame is divided into multiple small square blocks called macroblocks. The coding tools or kernels are applied to these macroblocks rather than the whole frame. Facet performs its traffic shaping by manipulating these macroblocks. Specifically, it places the censored video on several adjacent blocks, with a randomly selected chat video being played on the remainder of the blocks. By adjusting the width and height of the censored video, the traffic pattern is expected to shift between that of the censored video and the chat video. This mechanism is illustrated in Figure 29 (b).

The Facet server can make the trade off between steganography and the video quality. We define the steganography level s to be the scale of the censored video. Suppose the original width and height for the video is w and h, then the embedded video has width



(a) (b)

Figure 29: (a) Audio Morphing: choose resampling rate, and (b) Video Morphing: embed the censored video in a chat video

$w \cdot s$ and height $h \cdot s$. The experimental results in Section 9 show the video morphing can effectively defeat the censor's traffic analysis.

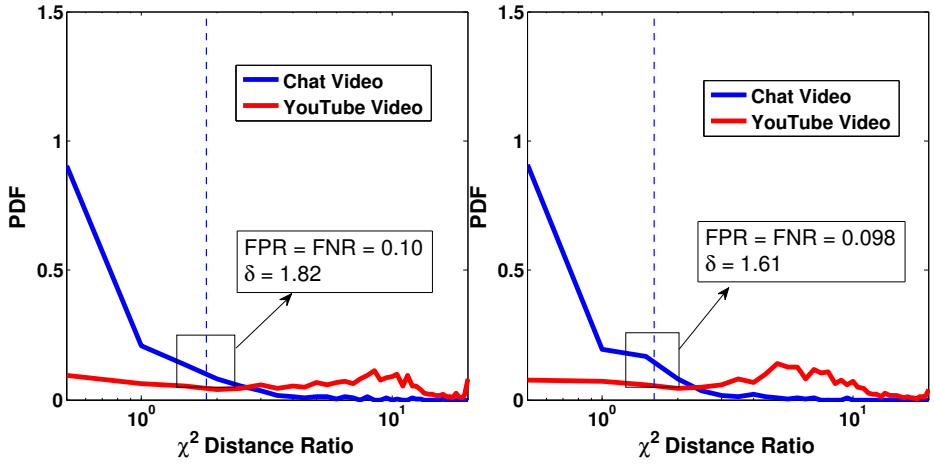
6.6 Experiment

6.6.1 Dataset

Chat Video Set. The chat videos are from YouNow.com, a popular live video blogging website. YouNow videos can simulate the chat video well for the following reasons. First, it does not lack interactions. A special feature of YouNow is that it allows the audiences to interact with the blogger by instant messages. Consequently, the blogger often pauses and answers questions, simulating the interactions in genuine video chat. Second, the users seldom use any video editing in the live video blogging (the reason could be the inconvenience of live editing), the videos hold the nature of webcam video chats. Third, the YouNow has a large amount of diversified webcam videos representing different situations in video chats. All these features make YouNow videos suitable.

The videos were collected on Sep 17, 24, and Oct 5. The users were picked out by using YouNow's "recently Broadcasted" feature, and for each user only their latest video is included. Finally, 1045 YouNow videos are included in the chat video set.

Censored Video Set. Highly viewed YouTube videos are chosen to construct the censored video set. YouTube charts list the most popular videos by category for a time period of one week, one month, or all time. For each category, we harvested all the videos with length more than 1.5 minutes. 1013 YouTube videos under 15 categories are included in the dataset, as is shown in Table 6. The video format is selected to be FLV, and the resolution is 360p.



(a) $K = 20$, $n = 2$ (b) $K = 20$, $n = 3$

Figure 30: **YouTube Video without Morphing:** probability density function of χ^2 distance ratio Θ

6.6.2 Experimental Setup

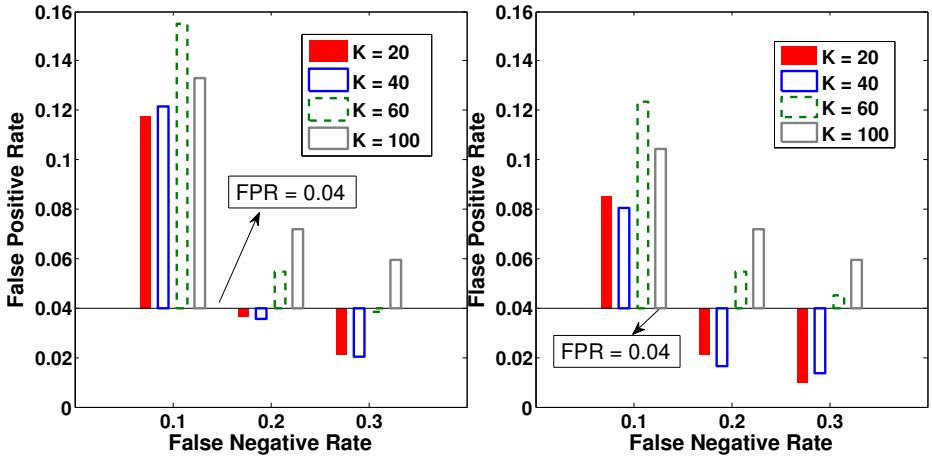
The experiment is run on a MacBook Pro with a 2.3Ghz Intel Core i7 processor, and OS X version 10.9. A guest operating system, Ubuntu Precise Pangolin (12.04), is installed on a VMware virtual machine, in which the Facet server is implemented. The host machine acts as the Facet client. The connection between host and guest machine is through a private virtual network [9].

Skype is selected as the videoconferencing system. The traffic between the two videoconferencing clients is exchanged through the VMware private virtual network. For the host, the Skype version number is 6.9 (701), and for the guest machine, the version is 4.2.0.11. Skype uses H.264 as the video codec, and SILK_V3 as the audio codec. The Frames Per Second (FPS) is 30.

Both of chat videos and YouTube videos are streamed into camera & microphone emulators. For each video, the packet capture is started after the video has been playing for 30 seconds, and the capture lasts for one minute. Skype's technical call information shows the packet loss rate is 0% in the experiment. For the camera emulator, we set the timeout to 1000 seconds, with YUY2 format. The emulator resolution is selected to be 320×240.

6.6.3 Morphing Effectiveness

This part evaluates the effectiveness of Facet morphing. The experiment captures the traffic of the chat video set, morphed YouTube videos with $s = 0.125, 0.25$ and 0.5, and non-morphed YouTube videos. Then, the chat video set is grouped with each of these four YouTube video sets individually for binary classification.



(a) $n = 2$ (b) $n = 3$

Figure 31: **False Positive Rate (no Morphing):** if the censor blocks 70%, 80%, or 90% Facet connections.

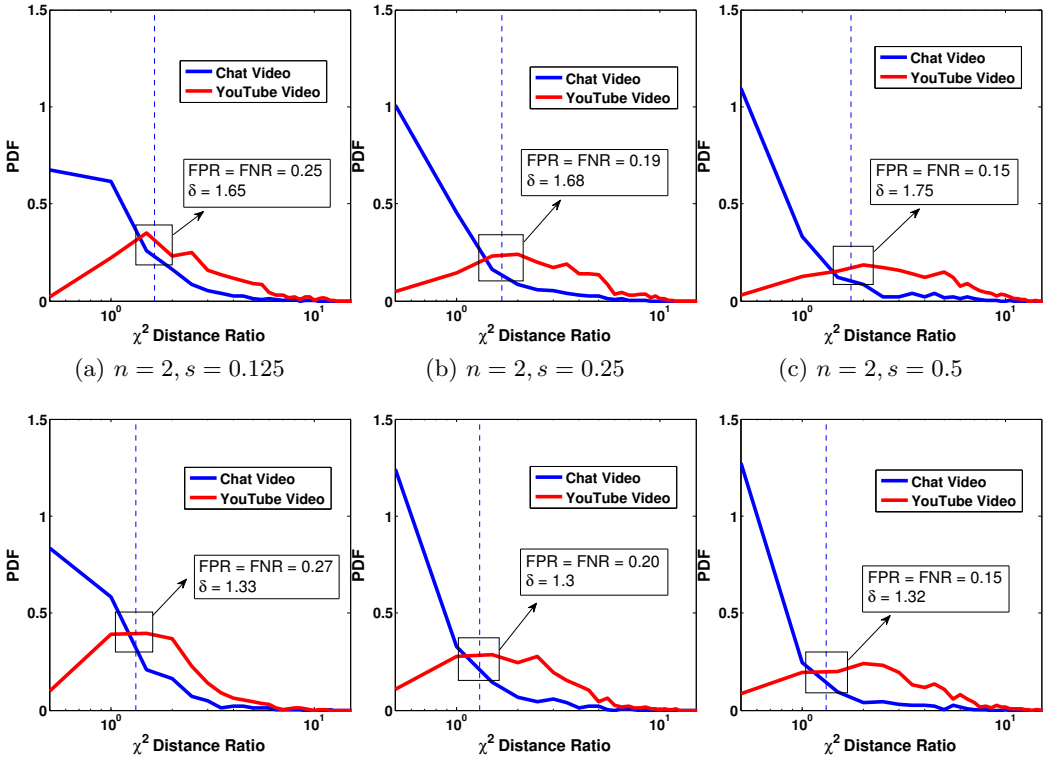
Category	No.	Category	No.	Category	No.
Animation	80	Autos	39	Travel	38
Comedy	85	Edu	84	Sci	51
Entertain	74	Gaming	98	Sport	59
Howto	86	Music	94	People	64
News	57	Nonprofit	58	Pet	46

Table 6: YouTube Video Set

We set $n = 2$ and $n = 3$ respectively. We do not include a larger n case, because the classifier with $n > 3$ performs poorly (and therefore not representative) due to the curse of dimensionality [72]. Suppose $n = 4$. If for each packet, there are 20 discretized lengths, then $20^4 = 160000$ grams are possible for videoconferencing traffic. When dimensionality is high, the volume of the space becomes so large that the the training samples are too sparse to represent the model, therefore, the classification in high dimensional space performs poorly. Also, we tune the discretization parameter K to be 20.

Figure 30 and Figure 32 give the probability density function (PDF) of the test statistic (χ^2 distance ratio Θ) for different cases, which indicates how the classifier performs for binary classification. Also, the false positive rate (or false negative rate) is given, when it equals to false negative rate (or false positive rate). This value can be used to compare the classifier performance for different cases.

Figure 30 shows the PDF of Θ when the classifier is used to distinguish genuine YouTube and chat video traffic. (a) uses 2-gram as the feature extraction, and (b) 3-gram. Both of them have $K = 20$. These figures show the traffic of chat and YouTube videos have distinct



(a) $n = 2, s = 0.125$ (b) $n = 2, s = 0.25$ (c) $n = 2, s = 0.5$

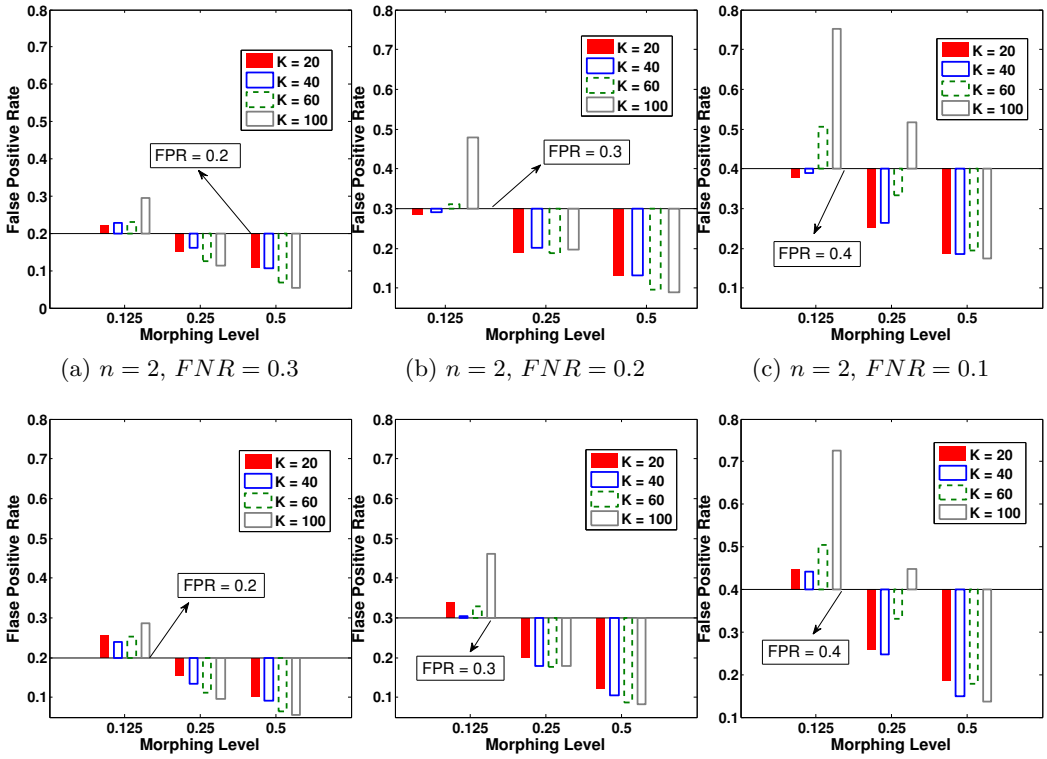
(d) $n = 3, s = 0.125$ (e) $n = 3, s = 0.25$ (f) $n = 3, s = 0.5$

Figure 32: **YouTube Video with Morphing:** probability density function of χ^2 distance ratio Θ

Θ distributions, thus the censor can specify a proper threshold to block a majority of unwanted traffic while keeping most of the genuine conferencing connections alive.

Figure 32 is the PDF of Θ when the classifier is used to determine morphed YouTube video traffic from chat video traffic with $K = 20$. For (a) and (d) $s = 0.125$, and the figures show Θ of the morphed YouTube video has more distribution in the 0 to 1 range, which means by morphing, the YouTube video is more likely to be regarded as the chat video. From the perspective of false positive/negative rate, the classifier only has $FPR = FNR = 0.25$ for 2-gram and $FPR = FNR = 0.27$ for 3-gram. The results demonstrate the morphing effectiveness.

Also, the distribution of Θ for different morphing levels is given in (b) (c) (e) (f). We show with a smaller morphing level s, the distribution of Θ for morphed YouTube videos resides more on the range of 0 to 1, and the false positive/negative rate is higher. This demonstrates when morphing level s is smaller, the morphed YouTube video session is more secure against traffic analysis.



(a) $n = 2$, $FNR = 0.3$ (b) $n = 2$, $FNR = 0.2$ (c) $n = 2$, $FNR = 0.1$

(d) $n = 3$, $FNR = 0.3$ (e) $n = 3$, $FNR = 0.2$ (f) $n = 3$, $FNR = 0.1$

Figure 33: **False Positive Rate (with Morphing):** if the censor blocks 70%, 80%, or 90% Facet connections.

6.6.4 Security Against Blockage

In our attack model, the censor is assumed to be unwilling to block or disrupt the genuine conferencing. Here, we investigate in order to block a given fraction of Facet connections, how likely the censor is to affect the genuine conferencing.

The censor is set to block $0.9, 0.8$ and 0.7 Facet connections. The corresponding false positive rate is investigated for morphed YouTube videos as well as the genuine YouTube videos. The experiment results are given in Figure 31 and Figure 33.

In Figure 31(a) and (b), the false positive rate is given when the censor tries to block 70%, 80%, or 90% Facet sessions playing genuine YouTube videos. We can see without morphing, the censor only has to disrupt 4% genuine videoconferencing to block 80% Facet connections. If it is aimed at blocking 70%, the false positive rate is even lower, only 2%. The figures show the necessity of adopting morphing mechanisms in the Facet design.

Figure 33 (a) to (f) show the false positive rate when the censor attempts to block 70%, 80%, or 90% Facet sessions playing morphed YouTube videos. For (a) and (d), the false negative rate is set to 0.3, and we can see even if the censor chooses the optimal parameters,

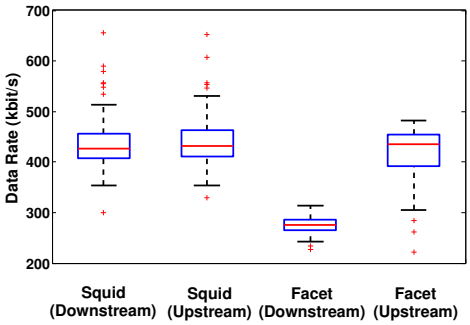


Figure 34: **Bandwidth Consumption:** Facet vs. Squid

	Downstream		Upstream	
	Skype	Gstreamer	Skype	Gstreamer
Mean	42.49	232.62	409.11	8.71
Max	51.83	265.32	471.91	10.81
Min	30.85	189.63	215.90	6.48
Med.	45.36	236.58	424.94	8.45
Std.	7.11	12.88	49.96	1.39

Table 7: Facet Traffic Break Down (kbit/s)

such as $K = 20$ and $n = 2$, it has to disrupt more than 20% genuine videoconferencing. For (c) and (f), when the censor attempts to block 90% of the Facet connections, it has to block 40% of the genuine videoconferencing connections. To conclude, if the censor wants to massively block the Facet connections, it has to disrupt at least 20% genuine videoconferencing connections. This cost, under our assumption, can make Facet less vulnerable to blockage, especially considering genuine videoconferencing connections are enormously more frequent than those of Facet.

Also, the figures show the smaller s is, the more secure the Facet server is against blockage. But considering the aggressiveness and capability of the censor may vary, the value s can be adjusted by the Facet server to provide a higher quality service. In addition, it shows the classifier with $n = 2$ in general outperforms the classifier with $n = 3$, though 3-grams can have more complete feature extraction. The reason for this is the curse of dimensionality.

6.6.5 Performance Analysis

Setup. 106 YouTube videos (length below 180s) are selected from our dataset. When the Facet server plays these videos, the upstream & downstream bandwidth, together with CPU & Memory usage are recorded. The client is set to disable the camera. For comparison, a traditional web proxy `Squid` is adopted. In the experiment, the client uses the Squid and plays the YouTube videos by Firefox. Also, the video resolution is set to 240p in both cases.

Experimental Results. The bandwidth costs are given in Figure 34 and Table 7. It shows the downstream bandwidth of the Facet server is lower than the Squid server. If the client uses Facet, about 150 kbit/s downstream bandwidth is saved. A possible reason for this difference is that for web browsing, extra information (such as advertisements, etc.) are fetched, and for a Facet connection, the client only downloads the video. For upstream, the bandwidth costs of these two systems are close. It is worth noting that although the downstream bandwidth of the Facet server will be increased if the client enables its camera, Table 3 shows the downstream bandwidth is still less than 700 kbit/s if the client's camera has the same resolution with the server. These experimental results show Facet server has high efficiency in bandwidth usage. For a server with 15 Mbit/s bandwidth, it can support up to 20 simultaneous sessions.

The computational costs for these two systems are shown in Table 8. For Facet server, the costs comes from making a Skype video call (Skype), downloading and redirecting video & audio stream (Gstreamer), and executing a Skype wrapper (Python). Though the table shows the Facet server consumes more CPU cycles and memory than Squid (most of which comes from the Skype video call), this cost is acceptable. Still, a computer with one virtual core can support up to 5 Facet sessions, and for a computer with 4 virtual cores, it can support 20 sessions.

Category	Program	CPU	Memory (4GB)
Facet (s=1)	Skype	14.4%	2.4%
	Gstreamer	3.7%	0.5%
	Python	0.6%	0.1%
Web Proxy	Squid	0.4%	0.3%

Table 8: Facet CPU & Memory Usage

7 Conclusion

Information leakage in anonymous traffic allows attackers or censors to exploit such leakage to de-anonymize the traffic. This thesis conducts the first larg-scale information leakage measurement upon Tor netowrks, which is one of the most popular anonymous networks. We find that the traditional way of validating a defense by classification accuracy alone is flawed. The second part of this thesis focuses on how to prevent information leakage in anonymous traffic. Targetting at anti-censorship design, we propose Mailet and Facet systems to provide realtime video watching and social networking with minimual information leakage.

References

[1] Collateral freedom: A snapshot of chinese internet users circumventing censorship, https://openitp.org/news-events/collateral-freedom-a-snapshot-of-chinese-users-circumventing-censorship.html.

[2] Collateral freedom: A snapshot of chinese internet users circumventing censorship, https://openitp.org/news-events/collateral-freedom-a-snapshot-of-chinese-users-circumventing-censorship.html.

[3] gstreamer: open source multimedia framework, http://gstreamer.freedesktop.org.

[4] http://music.columbia.edu/cmc/musicandcomputers /chapter2/02_05.php.

[5] Skype encoding camera specification, http://developer.skype.c om/resources/skype_encoding_camera_specification.pdf.

[6] skype4py, http://sourceforge.net/projects/skype4py/.

[7] `https://github.com/wtfpad/wtfpad`.

[8] `https://www.cse.ust.hk/~taow/wf.html`.

[9] Vmware virtual networking concepts, http://www.vmware.com /files/pdf/virtual_networking_concepts.pdf.

[10] A crawler based on tor browser and selenium. `https://github.com/webfp/tor-browser-crawler`, 2015. Accessed: 2015-12-04.

[11] L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[12] C. Brubaker, A. Houmansadr, and V. Shmatikov. Cloudtransport: Using cloud storage for censorship-resistant networking. In *Proceedings of PETS'14*, 2014.

[13] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *Proceedings of USENIX Security'10*, 2010.

[14] X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A congestion sensitive website fingerprinting defense. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2014)*, November 2014.

[15] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 21th ACM conference on Computer and Communications Security (CCS 2014)*.

[16] X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 19th ACM conference on Computer and Communications Security (CCS 2012)*, 2012.

[17] J. Cheng and R. Greiner. Comparing bayesian network classifiers. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 101–108. Morgan Kaufmann Publishers Inc., 1999.

[18] G. Cherubin, J. Hayes, and M. Juarez. Website fingerprinting defenses at the application layer. *Proceedings on Privacy Enhancing Technologies*, 2017(2):186–203, 2017.

[19] T. Chothia, Y. Kawamoto, and C. Novakovic. A tool for estimating information leakage. In *International Conference on Computer Aided Verification*, pages 690–695. Springer, 2013.

[20] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of USENIX Security'04*, 2004.

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

[22] B. Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.

[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[24] B. S. Everitt. *Mixture DistributionsI*. Wiley Online Library, 1985.

[25] S. Frühwirth-Schnatter. *Finite mixture and Markov switching models*. Springer Science & Business Media, 2006.

[26] J. Geddes, M. Schuchard, and N. Hopper. Cover your acks: Pitfalls of covert channel censorship circumvention. In *Proceedings of CCS'13*, 2013.

[27] J. Geddes, M. Schuchard, and N. Hopper. Cover your acks: Pitfalls of covert channel censorship circumvention. In *Proceedings of CCS'13*, 2013.

[28] R. C. George Danezis. Introducing traffic analysis. 2007.

[29] C. Giovanni. Bayes, not naïve: Security bounds on website fingerprinting defenses. *Proceedings on Privacy Enhancing Technologies*, 2017, 2017.

[30] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, and N. Feamster. The effect of dns on tor's anonymity. 2017.

[31] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[32] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, Austin, TX, Aug. 2016. USENIX Association.

[33] A. Houmansadr, C. Brubaker, and V. Shmatikov. The parrot is dead: Observing unobservable network communications. In *Proceedings of IEEE S&P'13*, 2013.

[34] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of CCS'11*, 2011.

[35] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of CCS'11*, 2011.

[36] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want my Voice to be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *Proceedings of NDSS'13*, 2013.

[37] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want my Voice to be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *Proceedings of NDSS'13*, 2013.

[38] A. Houmansadr, E. L. Wong, and V. Shmatikov. No direction home: The true cost of routing around decoys. In *Proceedings of NDSS'14*, 2014.

[39] F. House. Manipulating social media to undermine democracy, 2017.

[40] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of IMC'12*, 2012.

[41] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of USENIX Security'11*, 2011.

[42] I. T. Jolliffe. Principal component analysis and factor analysis. In *Principal component analysis*, pages 115–128. Springer, 1986.

[43] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 21th ACM conference on Computer and Communications Security (CCS 2014)*.

[44] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*. Springer, 2016.

[45] B. Juurlink, M. Alvarez-Mesa, C. Chi, A. Azevedo, C. Meenderinck, and A. Ramirez. Understanding the application: An overview of the h.264 standard. In *Scalable Parallel Programming Applied to H.264/AVC Decoding*, SpringerBriefs in Computer Science, pages 5–15. Springer New York, 2012.

[46] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable internet communication. In *Proceedings of FOCI'11*, 2011.

[47] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable internet communication. In *Proceedings of FOCI'11*, 2011.

[48] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1995.

[49] I. Kononenko. Semi-naive bayesian classifier. In *Proceedings of the European Working Session on Learning on Machine Learning*, EWSL-91, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[50] T. O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):517–519, 1987.

[51] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of WWW'10*, 2010.

[52] S. Li, H. Guo, and N. Hopper. Measuring information leakage in website fingerprinting attacks and defenses. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18.

[53] S. Li and N. Hopper. Mailet: Instant social networking under censorship. *Proceedings on Privacy Enhancing Technologies*, 2016.

[54] S. Li, M. Schliep, and N. Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, 2014.

[55] S. Li, M. Schliep, and N. Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of WPES'14*, 2014.

[56] M. Liberatore and B. N. Levine. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, November.

[57] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[58] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella, et al. Fairplay-secure two-party computation system. In *Proceedings of USENIX Security'04*, 2004.

[59] L. Mather and E. Oswald. Pinpointing side-channel information leaks in web applications. *Journal of Cryptographic Engineering*, pages 1–17, 2012.

[60] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. Skypemorph: Protocol obfuscation for Tor bridges. In *Proceedings of CCS'12*, 2012.

[61] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. Skypemorph: Protocol obfuscation for Tor bridges. In *Proceedings of CCS'12*, 2012.

[62] R. Nithyanand, X. Cai, and R. Johnson. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 12th Workshop on Privacy in the Electronic Society (WPES)*.

[63] S. E. Oh, S. Li, and N. Hopper. Fingerprinting keywords in search queries over tor. *Proceedings on Privacy Enhancing Technologies*, 2017(4), 2017.

[64] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz. How unique is your .onion? an analysis of the fingerprintability of tor onion services. *Proceedings of the 24th ACM conference on Computer and communications security*, 08 2017.

[65] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *Proceedings of the 23rd Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016)*.

[66] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)*.

[67] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)*. ACM, October 2011.

[68] M. Perry. Experimental defense for website traffic fingerprinting.

[69] D. Politis, J. Romano, and M. Wolf. *Subsampling*. Springer Series in Statistics. Springer New York, 1999.

[70] Reuters. Google's schmidt predicts end of censorship within a decade. 11 2013.

[71] M. Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[72] J. Rust. Using randomization to break the curse of dimensionality. *Econometrica*, pages 487–516, 1997.

[73] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *Proceedings of CCS'12*, 2012.

[74] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *Proceedings of CCS'12*, 2012.

[75] Y. Shi and K. Matsuura. Fingerprinting attack on the tor anonymity system. In *International Conference on Information and Communications Security*, pages 425–438. Springer, 2009.

[76] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[77] Z. Tufekci. Networked politics from tahrir to taksim: Is there a social media-fueled protest style? In *Digital Media and Learning Central*. http://dmlcentral.net/blog/zeynep-tufekci/networked-politics-tahrir-taksim-there-social-media
-fueled-protest-style, June 2013.

[78] B. A. Turlach et al. *Bandwidth selection in kernel density estimation: A review*. Université catholique de Louvain, 1993.

[79] P. Van Kerm et al. Adaptive kernel density estimation. *The Stata Journal*, 3(2):148–156, 2003.

[80] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: What twitter may contribute to situational awareness. In *Proceedings of CHI'10*, 2010.

[81] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, Dec. 2010.

[82] Q. Wang, X. Gong, G. T. K. Nguyen, A. Houmansadr, and N. Borisov. Censorspoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of CCS'12*, 2012.

[83] Q. Wang, X. Gong, G. T. K. Nguyen, A. Houmansadr, and N. Borisov. Censorspoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of CCS'12*, 2012.

[84] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proc. 23th USENIX Security Symposium (USENIX)*, 2014.

[85] T. Wang and I. Goldberg. Improved website fingerprinting on tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2013)*.

[86] T. Wang and I. Goldberg. Walkie-talkie: An effective and efficient defense against website fingerprinting. Technical report, 2017.

[87] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: A camouflage proxy for the Tor anonymity system. In *Proceedings of CCS'12*, 2012.

[88] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. StegoTorus: A camouflage proxy for the Tor anonymity system. In *Proceedings of CCS'12*, 2012.

[89] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *Proceedings of IEEE S&P'11*, 2011.

[90] C. Wright, S. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network and Distributed Security Symposium NDSS '09*.

[91] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *Proceedings of IEEE S&P'08*, 2008.

[92] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob. In *Proceedings of USENIX Security'07*, 2007.

[93] E. Wustrow, C. M. Swanson, and J. A. Halderman. Tapdance: End-to-middle anticensorship without flow blocking. In *Proceedings of USENIX Security'14)*, 2014.

[94] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of USENIX Security'11*, 2011.

[95] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of USENIX Security'11*, 2011.

[96] A. C. Yao. Protocols for secure computations. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[97] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov. Sweet: Serving the web by exploiting email tunnels. In *Proceedings of HotPETs'13*, 2013.