PRACTICAL ANALYSIS OF ENCRYPTED NETWORK TRAFFIC

Andrew M. White

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2015

Approved by:

Fabian Monrose

Michael Bailey

Kevin Jeffay

Phillip Porras

Michael Reiter

© 2015
Andrew M. White
ALL RIGHTS RESERVED

ABSTRACT

Andrew M. White: PRACTICAL ANALYSIS OF ENCRYPTED NETWORK TRAFFIC
(Under the direction of Fabian Monrose)

The growing use of encryption in network communications is an undoubted boon for user privacy. However, the limitations of real-world encryption schemes are still not well understood, and new side-channel attacks against encrypted communications are disclosed every year. Furthermore, encrypted network communications, by preventing inspection of packet contents, represent a significant challenge from a network security perspective: our existing infrastructure relies on such inspection for threat detection. Both problems are exacerbated by the increasing prevalence of encrypted traffic: recent estimates suggest that 65% or more of downstream Internet traffic will be encrypted by the end of 2016. This work addresses these problems by expanding our understanding of the properties and characteristics of encrypted network traffic and exploring new, specialized techniques for the handling of encrypted traffic by network monitoring systems.

We first demonstrate that opaque traffic, of which encrypted traffic is a subset, can be identified in real-time and how this ability can be leveraged to improve the capabilities of existing IDS systems. To do so, we evaluate and compare multiple methods for rapid identification of opaque packets, ultimately pinpointing a simple hypothesis test (which can be implemented on an FPGA) as an efficient and effective detector of such traffic. In our experiments, using this technique to "winnow", or filter, opaque packets from the traffic load presented to an IDS system significantly increased the throughput of the system, allowing the identification of many more potential threats than the same system without winnowing.

Second, we show that side channels in encrypted VoIP traffic enable the reconstruction of approximate transcripts of conversations. Our approach leverages techniques from linguistics, machine learning, natural language processing, and machine translation to accomplish this task despite the limited information leaked by such side channels. Our ability to do so underscores both the potential threat to user privacy which such side channels represent and the degree to which this threat has been underestimated.

Finally, we propose and demonstrate the effectiveness of a new paradigm for identifying HTTP resources retrieved over encrypted connections. Our experiments demonstrate how the predominant paradigm from prior work fails to accurately represent real-world situations and how our proposed approach offers significant advantages, including the ability to infer partial information, in comparison. We believe these results represent both an enhanced threat to user privacy and an opportunity for network monitors and analysts to improve their own capabilities with respect to encrypted traffic.

ACKNOWLEDGEMENTS

I have been fortunate to receive aid and encouragement from a great number of people over the past seven years. First and foremost, I would like to thank Stephanie Malone, as well as my parents and family, for their support and understanding. Second, I cannot adequately express my gratitude to Prof. Fabian Monrose, who has been a mentor and colleague throughout and who has supported me in too many ways to count since our first days at UNC.

I would also like to thank the remainder of my committee (Prof. Michael Bailey, Prof. Michael Reiter, Prof. Kevin Jeffay, and Phil Porras) for their advice and their time. Special thanks are also due to my colleagues and collaborators at UNC (particularly Prof. Jan-Michael Frahm, Srinivas Krishnan, Austin Matthews, Prof. Elliott Moreton, Nathan Otternes, Rahul Raguram, Katherine Shaw, Kevin Snow, Teryl Taylor, Jan Werner, and Yi Xu); at SRI International (particularly Vinod Yegneswaran); and at IBM Research (particularly Reiner Sailer, Mihai Christodorescu, and Marc Stoecklin). In addition, Murray Anderegg, Jake Czyz, Alex Everett, Jim Gogan, Bil Hays, Jodie Turnbull, and Missy Wood provided invaluable assistance during the course of this work.

Finally, I would like to express my eternal gratitude to Todd Anderson, Brian Collier, and David Fortney, for keeping me off the wrong path, and to Prof. Barry Lawson, Prof. Douglas Szajda, and Prof. Jason Owen, for helping me find the right one.

This work was supported in part by the Department of Homeland Security (DHS) under contract number D08PC75388, the U.S. Army Research Office (ARO) under Cyber-TA Grant no. W911NF-06-1-0316, and the National Science Foundation (NSF) under award no. 0831245. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DHS, NSF, or ARO.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ARO	Army Research Office
ASCII	American Standard Code for Information Interchange
BDR	Bayesian detection rate
BR	Binary relevance
BuFLO	Buffered Fixed-Length Obfuscator
CBR	Constant bit-rate
CDF	Cumulative distribution function
CELP	Code-excited linear prediction
CPU	Central processing unit
CV	Cross-validation
DAG	Data Acquisition and Generation
DHS	Department of Homeland Security
DMA	Direct memory access
DNS	Domain name system
DPI	Deep packet inspection
DT	Decision tree
EFF	Electronic Frontier Foundation
EFR	Enhanced Full Rate
ELF	Executable and Linkable Format
FLLD	Fast Levenshtein-like distance
FP	False positive
FPGA	Field-programmable gate array
GIF	Graphics Interchange Format
GPU	Graphics processing unit
GSM	Global System for Mobile Communications
HMM	Hidden Markov model
HTER	Human-targeted translation edit rate

HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
HTTPOS	HTTPS with Obfuscation
HTTPS	HTTP-Secure (HTTP over SSL/TLS)
IDS	Intrusion detection system
IETF	Internet Engineering Task Force
IM	Instant messenging
IP	Internet protocol
IPA	International Phonetic Alphabet
IPS	Intrusion prevention system
IRB	Institutional Review Board
ISRG	Internet Security Research Group
JPEG	Joint Photographic Experts Group
K-S	Kolmogorov-Smirnov
k-NN	k-nearest neighbors
k-NN-WL	k-nearest neighbors with weight learning
LTE	Long Term Evolution
METEOR	Metric for Evaluation of Translation with Explicit Ordering
MIME	Multipurpose Internet Mail Extensions
MNB	Multinomial naïve Bayes
MoU	Memorandum of understanding
MTU	Maximum transmission unit
NB	Naïve Bayes
NIDS	Network intrusion detection system
NLTK	Natural Language Toolkit
NSF	National Science Foundation
OSAD	Optimal string alignment distance
OvR	One-vs-rest
PBX	Private branch exchange
PDF	Portable Document Format

PET	Privacy enhancing technology
PMF	Probability mass function
PRE	Precision
RBF	Radial basis function
REC	Recall
RF	Random forest
RFC	Request for Comments
ROC	Receiver operating characteristic
RTMP	Real Time Messaging Protocol
RTP	Real-time transport protocol
SIP	Session initiation protocol
SMTP	Simple Mail Transfer Protocol
SOCKS	Socket secure
SPRT	Sequential probability ratio test
SRILM	SRI Language Modeling
SRTP	Secure RTP
SSH	Secure shell
SSL	Secure sockets layer
TCP	Transport control protocol
TLS	Transport layer security
UDP	User datagram protocol
URI	Uniform resource identifier
URL	Uniform resource locator
UTF	Unicode Transformation Format
VBR	Variable bit-rate
VM	Virtual machine
VNG	Variable n-gram
VoIP	Voice-over-IP
VPN	Virtual private network
W3C	World Wide Web Consortium

WER	Word error rate
XMPP	Extensible messaging and presence protocol
XOR	Exclusive-“or”

CHAPTER 1: INTRODUCTION

Communications privacy is essential to modern society. Our financial infrastructure relies upon private transactions to operate while protecting our personal data and identities. Corporations and governments alike rely on the ability to conceal their intentions, while still transmitting those intentions to their agents, to protect their interests at home and abroad. Whistle-blowers are, at times, dependent upon such concealment for their very lives; even otherwise unremarkable people with unpopular beliefs rely on communications privacy to protect themselves from persecution. Our reliance upon private transactions becomes even more apparent when we consider the Internet, which has evolved into a omnipresent factor of everyday life for many people. As such, it is no surprise that a significant portion of Internet traffic is encrypted to ensure the privacy of the underlying communications: recent estimates suggest 29% of total downstream Internet traffic is encrypted, a proportion expected to increase to at least 65% after Netflix's transition to HTTPS in 2016 [52].

Public awareness of the need for encryption has heightened due to high-profile leaks of private data (such as the personal information of 70 million users of Sony's PlayStation Network in April, 2011 [127]) and the continuing increase in identity theft (see, e.g., [69]). We believe that the fraction of encrypted traffic on real networks will only continue to rise, as more web services follow in the footsteps of Google and Facebook in offering encrypted connections (as Netflix plans to do in 2016 [54, 61]), the movement toward ubiquitous end-to-end encryption (see, e.g., [14]) gains steam, and efforts to ease adoption for site operators take effect [e.g., 86]. In particular, private corporate networks, where secure communications are often required, undoubtedly see significantly higher proportions of encrypted traffic. Tools, such as the Electronic Frontier Foundation (EFF)'s *HTTPS Everywhere* [67] and the Internet Security Research Group (ISRG)'s *Let's Encrypt* [86], make securing personal web browsing and communications easier than ever. File-sharing programs like BitTorrent, which account for large proportions of

Internet traffic (16.5% by some estimates [e.g., 50]), offer connections encrypted at various levels to evade traffic shaping and institutional surveillance. Finally, governments and other organizations are increasingly mandating the use of encryption to protect the transmission of personal identifying information and other sensitive data.[1] These factors all contribute to altering the landscape of network traffic by increasing the prevalence of encrypted communications.

At the same time, encrypted traffic presents problems for network administrators and analysts. Policy, security, and quality-of-service requirements can be difficult to enforce when traffic is encrypted, since identifying characteristics are often obscured. No surprise, then, that the current default reaction to encrypted traffic is a defeatist stance, reflecting the prevailing point of view that encrypted traffic yields little or no information of value to an observer.

We believe this point of view to be misguided: in fact, substantial information regarding the underlying communications can be extracted from encrypted traffic under the right conditions. This is an example of a general problem in engineering secure systems: while the underlying cryptographic primitives may be sound, systems utilizing such primitives face a host of practical considerations which can result information leakage through *side channels*, such as the timing of certain cryptographic operations or the observable length of an encrypted message. This problem often occurs when security and efficiency requirements conflict, particularly when the security implications are not well understood—as has been the case with encrypted network traffic in the past. With respect to network communications, researchers have repeatedly demonstrated instances of side channels over more than a decade of previous work; for instance, prior work has shown: (a) that individual keystrokes typed during a secure shell (SSH) session can be identified [136], potentially leaking passwords and other sensitive information; (b) that the application protocol (e.g., HTTP) transported over encrypted channels, such as SSH tunnels or virtual private network (VPN) connections, can often be ascertained [e.g., 159]; and (c) that individual movies streamed over a wireless connection can be identified [130]. Our own work shows that encrypted Voice-over-IP (VoIP) calls, under certain conditions, can leak enough information to enable the reconstruction of conversation transcripts [155]. Prior work, and our own preliminary investigations, also indicate that web pages browsed over encrypted

[1]See, e.g., UNC's *Transmission of Protected Health Information and Personal Identifying Information Policy* [109]

connections can often identified [e.g., 43]. All of these attacks are based on properties (i.e., side channels, such as the sizes, timing, and direction of individual packets), of network traffic which are easily observable despite the use of encryption by the system in question.

Further, we believe that the increasing prevalence of encryption on the network necessitates a paradigm shift in the way we analyze and monitor network traffic. Deep packet inspection (DPI) engines, on which the prevailing intrusion detection/prevention systems (IDS/IPS) heavily rely, are of little use when payloads are encrypted. These engines are notorious for requiring extensive computational resources; encrypted traffic, in particular, imposes a heavy computational burden on DPI-based systems [22, 39]. In addition, IDS systems often utilize port numbers, both for filtering and for signature matching; unfortunately, port numbers are known to be unreliable due to the widespread use of non-standard and randomized ports [95, 105]. Lastly, while DPI systems may be able to apply signature matching to detect and filter some known encrypted protocols, some protocols, such as BitTorrent's Message Stream Encryption, are specifically designed to avoid signature matching and protocol identification [103]. Furthermore, identifying each new encrypted protocol requires the construction and deployment of a new signature specific to that protocol. Therefore, we argue that the resource constraints of DPI engines and the need to apply fundamentally different analysis techniques to encrypted traffic necessitate *encrypted traffic detection* techniques which are both *port-* and *protocol-agnostic*, and which can operate on in real-time on high-speed networks. Such techniques enable the rapid filtering of encrypted traffic, which can then be discarded or subjected to further analysis if desired. In Chapter 2, we develop and evaluate techniques for *winnowing*, or filtering, *opaque* (i.e., compressed or encrypted) traffic in real-time, which represent a substantial first step towards this goal.

The usefulness of such techniques is not limited to identifying opaque (or encrypted) traffic for further analysis and load reduction in DPI systems. Such techniques can enable policy enforcement on local networks, e.g., by flagging encrypted packets tunneled over unencrypted connections (an odd practice used by botmasters to hide command-and-control messages in otherwise mundane HTTP traffic [25]). Similarly, one might wish to flag unencrypted traffic where encrypted traffic is expected, such as might occur if an SSH connection is subverted [111], or on institutional networks where end-to-end encryption is a requirement. Identifying opaque (or encrypted) traffic may also help to discover applications tunneled over other protocols (e.g., video traffic tunneled over HTTP), or to provide sanity checking for strongly typed networking [107].

Our investigation into opaque traffic also reveals a startling fact, which underscores the need for specialized analysis: over a 24-hour weekday period (with peak loads of 1.2Gbps), almost 90% of payload-carrying TCP packets—and 86% of payload bytes—observed traversing a major university network were opaque. This staggering figure forces us to consider the challenges opaque, and particularly encrypted, traffic present for network security and forensics. Ultimately, there is a need to both improve our understanding of the properties and characteristics of encrypted network traffic and to explore new, specialized techniques for the handling of encrypted traffic by network monitoring systems on high-speed networks. Towards these ends, we examine two additional problems in this area: in Chapter 3, the approximate transcription of encrypted VoIP calls; and in Chapter 4, the large-scale identification of web resources retrieved over encrypted connections. These two problems are representative of major contexts in which encryption is often applied: streaming and request-response protocols. By exploring the practical limits of reconstructing information from leaks in both contexts, we provide a better understanding of the extent of the privacy threat these leaks represent, the steps necessary to mitigate that threat, and the ability of network and forensic analysts to turn that threat to their advantage.

Thesis Statement

> *Encrypted network connections can be detected in real-time and side channels exposed by those connections can be leveraged to provide significantly more useful information than previously believed, enabling both the reconstruction of approximate transcripts of encrypted VoIP conversations and the identification of HTTP resources retrieved over encrypted connections.*

In support, we first demonstrate (Chapter 2) that *opaque* traffic, of which encrypted traffic is a subset, can be identified in real-time and demonstrate how this ability can be leveraged to improve the capabilities of existing IDS systems. Second, we show that side channels in in encrypted VoIP traffic enable the reconstruction of approximate transcripts of conversations (Chapter 3). Finally, we propose and demonstrate the effectiveness of a new paradigm for identifying HTTP resources retrieved over encrypted connections (Chapter 4), which represents both a greatly enhanced threat to user privacy and an opportunity for network monitors and analysts to improve their own capabilities with respect to encrypted traffic.

1.1 Real-time Detection of Opaque Network Traffic

The analysis of encrypted traffic requires that we be able to identify and filter encrypted traffic from the masses of data transmitted across our networks every day. Towards this end, Chapter 2 develops and evaluates methods methods for quickly and accurately determining whether a packet is *opaque*, i.e., compressed or encrypted, in a port- and protocol-agnostic manner.

Our techniques are based on the fact that, in general, the processes of compression and encryption both impose necessarily high degrees of uniformity in the distribution of output bytes. We leverage this fact by formulating hypothesis tests to identify sequences of bytes which appear to be drawn from a uniformly random distribution. We focus on sequential analysis and small-sample fixed-size hypothesis tests to minimize the number of samples, i.e., the number of bytes, necessary to determine whether a packet is *opaque* or *transparent*. By operating on a minimal number of samples, we in turn minimize the overhead imposed by our techniques.

We evaluated a number of such techniques, including entropy-based tests which might be described as implementing "common wisdom". Ultimately, two simple tests, the sequential probability ratio and (fixed-size) likelihood ratio tests, provide impressive accuracy rates while examining $n = 16$ bytes or less per packet payload. In simplest form, both operate by counting the number of bytes m with value greater than 128, then comparing the likelihood of drawing m such bytes from a uniform distribution to that when drawn from a non-uniform distribution. The bytes are deemed uniformly distributed if the ratio between the two likelihoods exceeds a threshold determined by the desired error rates. The sequential test performs this check on a byte-by-byte basis, examining a single byte at a time and checking the ratio for the sequence observed so far, while the fixed-size test examines the full set of samples (e.g., all n bytes) as one batch. The advantage of the sequential test is that decisions can be made, in many cases, before examining all n bytes.

We demonstrate that these techniques enable the identification of opaque data with 95% accuracy through evaluations on both network traffic and static files. In addition, we demonstrate the usefulness of our techniques by identifying anomalous data streams in real-world traffic, including personal data such as account names and instant messenging (IM) conversations transmitted over port 443 (ostensibly reserved for HTTP connections over SSL/TLS). Finally, we show that winnowing opaque packets can increase the packet throughput of the Snort IDS by 147% on our campus network, enabling Snort to handle peak loads of 1.2Gbps with zero percent packet loss.

1.2 Reconstructing Transcripts of Encrypted VoIP Conversations

Chapter 3 shows how current practices for encrypting VoIP calls are insufficient to ensure privacy. In particular, the sizes of the transmitted packets, as observed on the network, are sufficient to reconstruct approximate transcripts of the encrypted conversation. The sizes of the encrypted packets leak information due to an interaction between two common design decisions in VoIP schemes: the use of variable bit-rate (VBR) codecs for compression of the voice signal and of length-preserving stream ciphers for encryption. These two technologies are chosen to minimize bandwidth consumption and requirements while maintaining call quality; however, this comes at the price of decreased privacy.

Previous work has shown that this information leak is sufficient to determine the language spoken in an encrypted call [157], identify the speakers [5, 92], and determine whether *known* phrases were spoken [156]. However, the previous work has not been considered a significant breach of privacy: the identity of the speakers and the language of a call can often be determined by locating the endpoints of the call (e.g., a call from Mexico to Spain is likely to be in Spanish). Determining the presence or absence of known phrases requires *a priori* identification of phrases of interest, a significantly limiting factor when considering practical breaches of privacy. Our work surpasses the previous work by constructing approximate transcripts of encrypted conversations without *a priori* knowledge of the contents of the conversation.

Our approach consists of four primary stages progressing from the identification of individual sounds to words and phrases. In the first stage, the sequence of packet sizes is segmented along the boundaries between individual *phonemes*, the fundamental sounds of which speech is comprised. Since each packet represents a single *frame* of audio, and frames have a short, fixed duration (e.g., 20ms), phonemes often span multiple packets; this means each phoneme in the input signal corresponds to a sequence of packet sizes. In the second stage, each sequence is classified, using two different classifiers, according to the phoneme represented. The third stage splits the resulting sequence of phonemes into smaller sequences, each representing a single word; the final stage matches those smaller sequences to dictionary words.

1.3 Identification of Encrypted Web Resources

Current network security and forensics platforms have often have little to no access to information about the communications underlying encrypted connections. In order to mitigate this weakness, we propose a data-driven approach to deriving information about HTTP traffic transmitted over encrypted channels. In particular, this information includes the network host to which the HTTP connection is directed (e.g., www.cs.unc.edu) as well as the path for the specific resource requested (e.g., ~amw/index.html).

Previous work has examined the problem of identifying the front page of web sites accessed over encrypted connections (see, e.g., Dyer et al. [43]). However, previous work has primarily focused on classifying the front pages of a limited number of web sites. Our work examines a significantly more complex space of labels by including both subdomains and the paths of the resource(s) requested in the set of labels to be modeled and predicted. In particular, we employ multi-label classification to predict not only the identity of a previously seen web page retrieved over an encrypted connection, but also to predict partial information (such as the domain name) about previously unseen URLs.

We collect a new dataset of traces of web pages retrieved over SSH and HTTPS tunnels which contains multiple URLs per domain name, enabling more realistic evaluations than in previous work. In particular, we evaluate the multi-label classification paradigm and the multi-class classification paradigm (employed by previous work) in scenarios including a single uniform resource locator (URL) per domain name (as in previous work) and multiple URLs per domain name. Our evaluation includes experiments under both closed-world (where the traces belong to a closed set of URLs known *a priori*) and open-world (with previously unknown, but unrelated, URLs) models. Furthermore, the multi-label paradigm enables a third experimental model which incorporates previously unknown but related URLs (e.g., previously unknown URLs belonging to a domain shared by known data).

Our experiments demonstrate that the adversary model used in previous work—that each page to be identified is unrelated to the every other—is unrealistic. Furthermore, we demonstrate that the multi-label paradigm is well-suited to the problem at hand as it supports "out-of-the-box" not only the closed-world and open-world models discussed in prior work, but also a third model—that of partial information via related URLs—not previously examined.

1.4 Contributions

In summary, the primary contributions of this dissertation are as follows:

1. Chapter 2 introduces the concept of *opaque*, i.e., compressed or encrypted, data as a distinct class of network traffic which is particularly difficult (in some cases, impossible) for DPI systems to analyze, resulting in substantial wasted effort on the part of such systems. Chapter 2 then develops and evaluates methods for quickly and accurately determining whether a packet is *opaque*, and demonstrates the accuracy and utility of these techniques in experiments on real-world network traffic (including through live experiments on our campus network). This work was previously published in:

   Andrew M. White et al. "Clear and Present Data: Opaque Traffic and its Security Implications for the Future". In: *ISOC Network and Distributed System Security Symposium – NDSS 2013*. The Internet Society, Feb. 2013.

2. Chapter 3 demonstrates how side channels in encrypted VoIP connections can be leveraged to reconstruct approximate transcripts of conversations. This work was previously published in:

   Andrew M. White et al. "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks". In: *2011 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2011, pp. 3–18.

3. Chapter 4 presents a novel approach to identifying HTTP resources retrieved over encrypted connections. In particular, we demonstrate how multi-label classification can be used to identify modeled resources and to partially identify previously unseen resources. We collect a new dataset containing multiple URLs per domain name, and use this dataset

to demonstrate how the single-URL-per-domain scenario of previous work leads to inflated results. Furthermore, we evaluate both our own approach and those of previous work in experiments under both closed-world and open-world models, as well as under a third model which allows for the inference of partial inference from related, but previously unseen, URLs.

CHAPTER 2: OPAQUE TRAFFIC[1]

2.1 Introduction

As our society becomes increasingly reliant on computer systems, the potential harms resulting from their insecure operation are far reaching—from legal, social, and economic to psychological and even physical (e.g., through cyber-physical systems) harms. Alongside *requirements* (e.g., confidentiality) and *mechanisms* (e.g., authentication), *policies* that define what actions are allowed, or disallowed, are a key building block of computer security and an important factor in minimizing these harms.

The successful monitoring of security policies via intrusion detection systems (IDSs) critically depends on scalable and accurate techniques for inspecting traffic. Deep packet inspection (DPI) is a common method for performing such inspection, especially when security policies require determinations based on information not accurately reflected by network ports, protocols, or hosts. In fact, the market for DPI appliances alone reached almost $1 billion in 2009, and continues to grow [161]. Since DPI must deal with huge volumes and significant heterogeneity of traffic, DPI designers often trade off accuracy of detection with resource demands [135]. DPI systems cannot generally derive useful information from *opaque* (i.e., encrypted or compressed) packets; thus, we propose improving the performance versus quality curve through the quick and accurate *winnowing*, i.e., filtering, of opaque traffic, and evaluate a number of techniques for doing so. Such techniques can improve the performance of DPI engines by quickly and accurately separating out low-value packets from the data stream they inspect; this is particularly important in high-performance environments, where the sheer volume of traffic can be staggering.

As previously discussed (Chapter 1), the class of opaque traffic encompasses not only encrypted connections, but also compressed entities, which, in the modern era of streaming

[1]The work presented in this chapter was previously published in: Andrew M. White et al. “Clear and Present Data: Opaque Traffic and its Security Implications for the Future”. In: *ISOC Network and Distributed System Security Symposium – NDSS 2013*. The Internet Society, Feb. 2013.

video, comprise an even more significant fraction of network traffic: some sources indicate that streaming video accounts for 63% of peak network usage [51], a proportion which has nearly doubled over the past three years (from 35% in the fall of 2011 [50]).

In fact, our experiments revealed a surprising preponderance of opaque traffic: in a 24-hour weekday period, nearly 90% of TCP packets traversing a major university network (with peak loads of 1.2Gbps) were found to be opaque. This truly staggering figure suggests a broader issue for the security community moving forward: as more and more payloads become opaque to DPI engines, how can we detect and prevent obscured threats? While content-based signatures will continue to be relevant as a detection mechanism for direct attacks (e.g., those exploiting buffer overflows in networking routines) against networked systems, bypassing current DPI engines can be as simple as encrypting the relevant exploit code, particularly for indirect attack vectors (e.g., exploits embedded in documents). Thus we, as a community, face both an immediate need to separate the wheat from the chaff—to winnow low-value, i.e., opaque, packets to enable our existing detection methods to operate in high-speed environments—and a long-term need to develop methods for coping with attacks embedded in opaque, and particularly encrypted, traffic. This work represents a first step toward solving both problems: our techniques enable the fast and accurate identification of opaque packets, either as chaff to be discarded, or as the inputs to specialized detection engines targeting opaque traffic.

Unfortunately, the identification of opaque network traffic is very challenging. While signatures can identify many known opaque protocols (e.g., SSL/TLS, SSH), some protocols (e.g., Bittorrent's Message Stream Encryption [103]) are specifically designed to avoid signature matching. In addition, signature-based approaches for identifying new opaque protocols require constructing and deploying new signatures for each new protocol. More importantly, existing techniques for identifying opaque data often require examination of a large number of bytes, which can be computationally and memory intensive on high-speed networks [22, 39]. Similarly, inspecting application-layer content-types to determine opacity requires resource-intensive flow reassembly. To compound matters, such detectors cannot rely on HTTP content-types as they are often inaccurate (see Section 2.3).

We take a first step toward addressing these challenges by providing novel methods for quickly and accurately determining whether a packet is opaque. Our techniques are both *port-*

and *protocol-agnostic*, allowing for the detection of opaque traffic regardless of the port(s) or protocol(s) involved. We concentrate on efficient techniques which can operate on limited portions of packet payload, such as small-sample fixed-size and sequential hypothesis tests, in order to minimize overhead and allow scarce computational resources to be reserved for high-value analyses.

In the present work, we necessarily evaluate our techniques in isolation, i.e., without attempting to fully integrate our approach with a particular system or methodology, in order to minimize the number of factors potentially biasing our results. That said, we envision the identification of opaque traffic not as the relatively standalone system portrayed herein — there is no "silver bullet" for network intrusion detection — but rather as an efficient new tool in the network analyst's toolbox. In particular, forensic recording systems, such as Time Machine [97], often avoid archiving full connections in order to reserve limited storage space for the higher-value packets at the beginning of a connection. An admitted weakness of this design decision is that attacks in the discarded portions of these connections go unnoticed. Our techniques mitigate this concern to an extent by providing a means for such a system to detect potentially high-value plaintext packets in lengthy connections. Similarly, our tests are simple and can be implemented on an FPGA, providing a benefit even to systems which make use of a hardware component, such as those following *shunting* approach proposed by González, Paxson, and Weaver [53].

As a final note, opaque traffic identification has value that extends beyond filtering to policy monitoring and enforcement. For instance, operators may wish to monitor and enforce policies within their network (e.g., flagging encrypted packets tunneled over unencrypted connections, an odd practice used by botmasters to hide command-and-control messages in otherwise mundane HTTP traffic [25]). Similarly, one might wish to flag unencrypted traffic where encrypted traffic is expected, such as might occur if an SSH connection is subverted [111], or on institutional networks where end-to-end encryption is required. Identifying opaque traffic may also help to discover applications tunneled over other protocols (e.g., video traffic tunneled over HTTP), or to provide sanity checking for strongly typed networking [107].

Our contributions include: 1) the concept of opaque traffic as a distinguishable class of network traffic; 2) the development, evaluation and comparison of multiple techniques for detecting such traffic (Section 2.2, Section A.1); 3) an operational analysis of modern network traffic with respect to opacity (Section 2.3.1); and 4) an evaluation, at scale, of the potential of our techniques for reducing the load on modern intrusion detection systems (Section 2.3.2).

2.2 Approach

An important requirement of ours is to minimize the amount of a packet's payload that we must inspect, as doing otherwise is both computationally and memory intensive on high-speed networks [22, 39]. These overheads severely restrict the numbers of samples available to us for any of the tests we explore. Therefore, for the remaining discussion, we propose *detectors* based on small-sample fixed-size hypothesis tests and sequential hypothesis testing. The latter allows us to make decisions quickly by examining only as many samples as needed to support a given hypothesis.

Our detectors are based on determining whether the bytes examined are drawn from a uniform distribution or some other, unknown, distribution. Our first instinct, when faced with the problem of measuring the uniformity of a set of samples, was to use entropy-based measures. However, as we show later, *accurate* entropy testing requires significantly more samples than is practical in our setting, and is less efficient than more direct methods based on the samples themselves rather than on a derived statistic such as entropy.

Both encrypted and compressed traffic will exhibit high entropy, i.e., the distribution of bytes will be close to uniform. In this paper, we consider these two cases as belonging to the same equivalence class of "*opaque*" objects as far as DPI engines are concerned. That is, regardless of whether the packets that belong to a session are compressed versus encrypted, they will be forced to go through the slow path wherein the engine must analyze all packets of these sessions, but will still fail to derive any useful information in the vast majority of cases. Hence, from the perspective of DPI engines, there is little value in attempting to analyze these packets. As Cascarano et al. [22] observed in their experimental evaluations, these slow paths incurred CPU overheads that can be *several orders of magnitude higher* than the average case for transparent traffic.

In our search for the best performing test for our goals, we examine several fixed sample-size hypothesis tests. There are a number of standard techniques for testing uniformity; however, many of these are designed for testing the uniformity of the outputs of a pseudo-random number generator, and thus assume access to millions of samples (e.g., [101]). These tests are unsuitable in our context because our sample size is extremely limited. We instead evaluate the appropriate fixed-size tests (the likelihood ratio test and the discrete Kolmogorov-Smirnov (K-S) test) and two variants of the sequential probability ratio test. We assess the effectiveness of each test in two scenarios, differentiated by the domain of the underlying distribution. In the first scenario, we directly test the distribution of byte values in the packet; for the second, we instead test the distribution of entropy values over n-byte "blocks" in the packet.

For pedagogical reasons, we leave descriptions of the less successful tests, along with the details of the parameter exploration experiment itself, to Section A.1.2, and discuss only the more effective tests here. In summary, the closely-related likelihood ratio and sequential probability ratio tests, operating directly on the byte values instead of on derived entropy values, consistently outperform the other tests. The poor performance of the entropy tests is related to the birthday problem, in that the (byte-)entropy of any n bytes is the same unless there is a collision (e.g., there are two bytes which share the same value). According to the birthday paradox, the *a priori* probability of a collision in 8 bytes, even when choosing only from only the 95 printable ASCII characters, is only about 26%. This means that the entropy of 8 ASCII bytes is indistinguishable from that of 8 unique bytes 74% of the time. Thus entropy-based tests require substantially more than the number of samples available in our context to be effective. However, our parameter exploration experiment (see Section A.1.2) reveals that the the more successful tests require examining only 16 bytes of payload to be effective.

Since our goal is to discard opaque traffic (albeit encrypted or compressed), we let our null hypothesis be that the packet is opaque and our general alternative be that the packet is transparent. Specifically, the null hypothesis is that the values of the bytes in the packet are approximately uniformly distributed (e.g., the packet is compressed or encrypted); the alternative is that the distribution is non-uniform (e.g., ASCII text).

When the need arises for a probability density function for the alternative hypothesis, we use a simple distribution based on the intuition that plaintext packets will have a higher frequency of bytes whose values are less than 128 (as are the ASCII values). We parameterize this distribution by setting δ to the cumulative density of those values. For example, at $\delta = 0.75$ the alternative hypothesis is that 75% of the bytes in the packet have values less than 128.

Likelihood Ratio Test

A well-known theorem of hypothesis testing is the *Neyman-Pearson* lemma, which states that the *most powerful* test, i.e., that with the lowest expected *false positive rate* for a given *false negative rate*, of one simple hypothesis against another is the *likelihood ratio test* [147]. For a single sample, the likelihood ratio statistic is simply the ratio of the likelihood of the sample under the alternative hypothesis to the likelihood of the sample under the null hypothesis. For multiple samples, these likelihoods are replaced with the corresponding joint likelihoods.

Suppose we wish to test the simple null hypothesis $\mathcal{H}_0 : \theta = \theta_0$ against the simple alternative $\mathcal{H}_1 : \theta = \theta_1$ given a random sample $\mathbf{x}$. Then the Neyman-Pearson lemma [147, Theorem 10.1] states that the most powerful test of $\mathcal{H}_0$ against $\mathcal{H}_1$ is that where one rejects $\mathcal{H}_0$ if $\Lambda(\mathbf{x}) \geq q$ and accepts $\mathcal{H}_0$ if $\Lambda(\mathbf{x}) < q$, where q is determined by the desired level of statistical significance and $\Lambda(\mathbf{x})$ is the ratio of the likelihood of $\mathbf{x}$ under the alternative to the likelihood of $\mathbf{x}$ under the null hypothesis.

Sequential Probability Ratio Test

Sequential analysis is often used when there is a desire, such as in our context, to limit the number of samples examined. In fact, Wald and Wolfowitz have shown that, among all tests of two simple hypotheses with fixed error probabilities, the sequential probability ratio test (SPRT) minimizes the expected number of samples required for a decision under either hypothesis [149]. Sequential testing works by examining samples one-by-one, rather than all at once, and evaluating a decision function at each sample. This allows the test to stop examining samples as soon as it has found enough "evidence" for one hypothesis or the other.

Specifically, let $\alpha = \mathbb{P}[\textsf{accept } \mathcal{H}_1 \mid \mathcal{H}_0]$ be the probability of a *false negative*, that is, an erroneous prediction that the population is not uniformly distributed; similarly, define $\beta = \mathbb{P}[\textsf{accept } \mathcal{H}_0 \mid \mathcal{H}_1]$ as the probability of a *false positive*. In order to perform the test, we iterate through some sequence of samples $x_1, x_2, \ldots$; according to Wald's theory of sequential hypothesis testing [148], we choose at each iteration m one of three actions: accept $\mathcal{H}_0$, accept $\mathcal{H}_1$, or continue, as follows:

$$\begin{array}{ll} \text{accept } \mathcal{H}_0 \text{ if} & \Lambda_m(x_1, x_2, \ldots, x_m) \leq g_0(m) \\ \text{accept } \mathcal{H}_1 \text{ if} & \Lambda_m(x_1, x_2, \ldots, x_m) \geq g_1(m) \\ \text{continue} & \text{otherwise.} \end{array}$$

Setting $g_0(m) = \frac{\beta}{1-\alpha}$ and $g_1(m) = \frac{1-\beta}{\alpha}$ gives the desired probabilities of false positives and false negatives.

A known drawback of this test is that it may not terminate within an acceptable number of samples. We alleviate this concern by exploring two variants, which we refer to as the *truncated* and *restricted* SPRTs. For the truncated SPRT, we specify a maximum number of samples which the test is allowed to examine; if this limit is reached without making a decision, then the fixed-size likelihood ratio test is applied to the samples examined so far. The restricted SPRT, on the other hand, works by sloping the decision boundaries such that they intercept the axis after the given number of samples. For the restricted case, we follow the approach suggested by Bussgang and Marcus [20].

Based on the results of the parameter exploration experiment (see Section A.1.2), we use the truncated sequential method ($\alpha = 0.005$, $\beta = 0.005$, $\delta = 0.85$) for the remainder of the experiments in this work.

2.3 Evaluation

In order to ascertain the effectiveness of our techniques in different scenarios, we perform a number of experiments in both offline and online settings. First we show, in a controlled, offline experiment, that our techniques are able to accurately identify the opacity of different file types. We then verify the accuracy of our techniques under real-world network conditions by evaluating our detectors on traffic logs and traces collected at two major universities, an analysis which produced a number of interesting anomalies which we investigated with the help of a network operator. Finally, we show the utility of winnowing by comparing two otherwise identical Snort IDS deployments under identical traffic loads.

2.3.1 Offline Analysis

File Type Identification

To gain an understanding of how well our techniques were able to label the opacity of various common data types, we collected a set of files with known ground truth, including compressed archives and streams, encrypted files, executable binaries, and text files. We also attempted to cover different sub-types; e.g., we included five different text file encodings. The details of this set, which we believe to be a reasonable cross-section of common file types, are presented in Table 2.1.

We gathered a base set of files from multiple sources, then applied standard compression and encryption algorithms to these files to create the remainder of the dataset. For executables, we used ELF binaries from the `bin` directories of a basic Ubuntu Server 10.04.3 installation (that we used for testing Bro and Snort). The files in each directory, including a number of Perl scripts, were then individually compressed using `tar` (which in turn uses `gzip` and `bzip2`) to provide `gzip` and `bz2` files and encrypted using `openssl` to provide examples of the RC4 and AES ciphers. The text files are the contents of the `man` path directory of the same Ubuntu installation, uncompressed and processed by `groff` into different encodings (ASCII, UTF-8, -16, and -32, and latin1); the PDF files are the archive of proceedings from a major computer security conference. Finally, the images were JPEGs scraped from public websites, including a departmental website and `nasa.gov`.

To simulate a network environment, we transmitted each object over an individual TCP connection on the loopback device of an Ubuntu virtual machine. We used `nc` to both send and receive, and `tcpdump` to collect the resulting packets. We report the proportion of opaque and transparent packets observed for each type of file in Table 2.1.

Our test labeled more than 95% of compressed, encrypted and image file packets as opaque, as expected. Similarly, even with multiple different encodings, our techniques labeled more than 99.99% of text file packets correctly as transparent. Our test is less consistent on the executables. Inspection of the binaries revealed a large number of null bytes, suggesting that a more targeted alternative hypothesis, perhaps counting only the set of printable ASCII bytes rather than simply those with value less than 128, may improve our results. However, this must be contrasted with the straightforwardness and efficiency of checking whether the value of a byte is greater than 128

Type	Objects	Size (MB)	True Positive Rate
compressed	1410	40	97.8
encrypted	1410	91	98.9
text	5802	176	100.0
images	205	12	94.4
executable	498	43	34.5
pdf	1006	75	13.5

Table 2.1: File Type Analysis

(which amounts to simply checking the high-order bit).

Content Type Matching

To assess the accuracy and performance of the techniques in Section 2.2, we instrumented the Bro IDS (version 1.6-dev-1491) to log packet-level information (with only coarse-grained payload information), providing a form of ground truth. Due to privacy restraints, we were unable to record full packet traces at scale; therefore, the experiments in this section are performed on logs generated by our instrumented version of Bro. For these experiments, we collected two logs (`log1` and `log2`) from two large university campuses.[2] Both logs were collected over several hours during a weekday. For simplicity, we only consider IPv4 TCP traffic. We labeled each packet by protocol using Bro's dynamic protocol detection [38], and restricted our analysis to two encrypted protocols (SSL and SSH) and two unencrypted protocols (HTTP and SMTP).

For each packet, we log the source and destination ports, the HTTP message type, the HTTP message encoding, and the payload length; we also store coarse-grained statistics in the form of a binary value for each byte of the payload (indicating whether the byte's value is less than 128), and the byte-value frequencies for each n-byte block of the payload. The latter are needed to calculate sample entropy at the block level and for the frequency-based tests (i.e., χ^2 and discrete Kolmogorov-Smirnov (K-S); see Section A.1). In all cases, we only log information for the first 256 bytes of the payload.

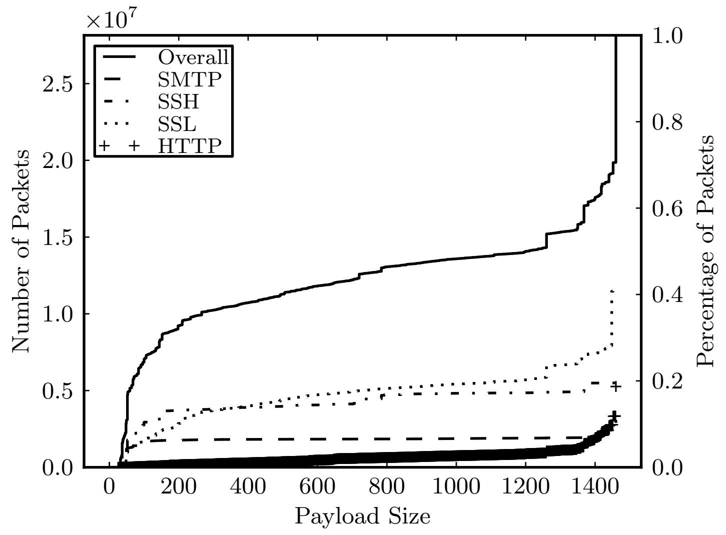
[2]The researchers and their respective Technology Service Offices have longstanding memorandums of understanding (MoUs) in place to collect anonymized network traffic. The MoU covers specific uses and types of networking data, as well as conditions for securing and accessing such data. To be compliant with our pre-existing Institutional Review Board (IRB) policies, all computations on payloads were performed in memory. For this specific collection effort, the Institutional Review Board (IRB) concluded that, as we collect only coarse-grained statistics regarding packet payloads, the activities in our application "do not meet the regulatory definition of human subjects research under the Common Rule" and are therefore not regulated.

We labeled each packet as “opaque” or “transparent” according to the expected data type for that packet: SSL and SSH packets are labeled opaque and SMTP packets are labeled transparent. For HTTP, we labeled packets based on the HTTP Content-Type and Content-Encoding header fields (as given by the sender) for the corresponding HTTP message (see Section A.2 for details of the labeling). This allows us to further restrict our attention to only those content-types for which opacity should be predictable and consistent. For instance, the HTTP 1.1 specification states that “HTTP entities with no specified Content-Type should be identified by inspection of the contents” [45]; we remove any such packets from our analysis because we have no way of determining ground truth. However, as we discovered during the course of this work, the HTTP Content-Type headers are often inaccurate and misleading (details are given in the following sections).

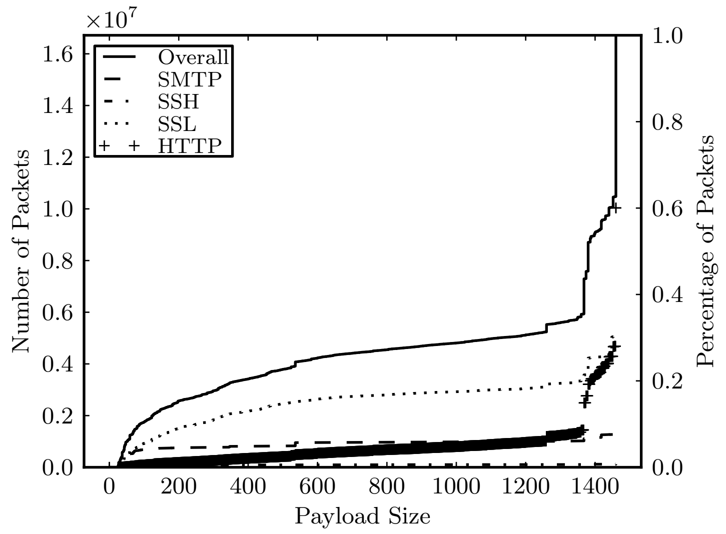
Unfortunately, Bro suffers from performance problems (see [39]) on high-speed networks, especially when port-based analysis is *disabled* (as is necessary to force Bro to determine protocols solely by inspection). Therefore, we discard any HTTP packets which belong to flows in which Bro experienced losses. We do so because the dropped packet(s) could have contained essential labeling information (e.g., a message header) necessary for determining the Content-Type and encoding.

After filtering the logs down to only those packets which met the criteria outlined above, over 39 million packets (across ≈ 3.8 million bi-directional flows) remained from log1 and over 24 million (across ≈ 2.3 million bi-directional flows) remained from log2. The traffic makeup for both logs is shown in Figure 2.1.

The Content-Type distribution (Figure 2.2) for the top content types in both logs reveals some interesting contrasts between the two. In particular, video types are prevalent in log2 while log1 consists mainly of web-browsing traffic (over 80% of log1 HTTP packets have Content-Type image/jpeg or text/html, compared to 40% in log2). The Content-Encoding field is only specified for a small proportion of packets in our logs, and the only significant encoding is gzip, at 4.0% in log1 and 6.5% in log2. All other encodings combined account for less than a tenth of a percent of packets in each log. We performed a large-scale analysis on both log1 and log2; the overall results are given in Table 2.2. Examining only 16 bytes per packet, offset 8 bytes from the start of each packet, we achieve a match rate, i.e., the percentage of examples for which our techniques



(a) `log1`



(b) `log2`

Figure 2.1: CDF of payload size for the protocols examined in the two campus network logs

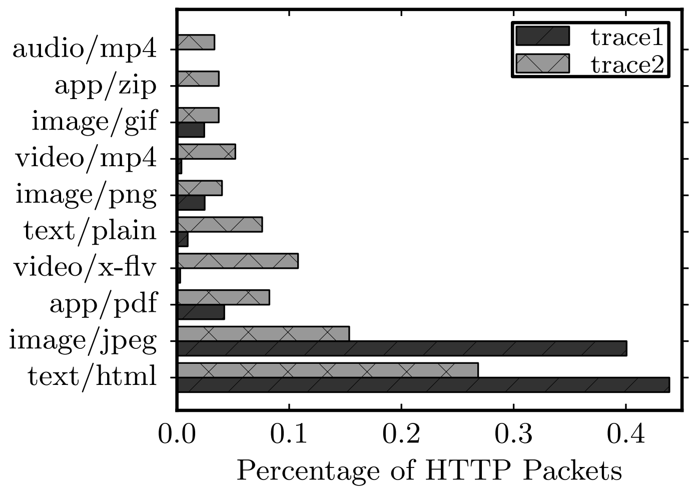


Figure 2.2: HTTP Content-Type Distribution

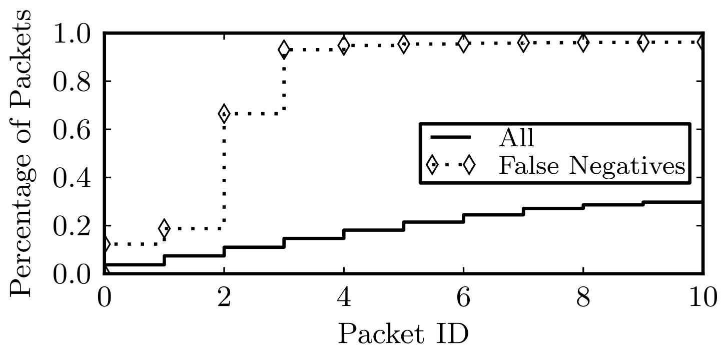
	`log1`		`log2`	
Protocol	Match Rate	Examples	Match Rate	Examples
SSH	94%	7.3m	97%	157k
SSL/TLS	96%	16.4m	94%	5.5m
SMTP	86%	3.35m	80%	1.4m
HTTP	91%	9.7m	85%	13.8m
Total	94%	36.7m	96%	20.8m

Table 2.2: Experimental Results (`log1` and `log2`)

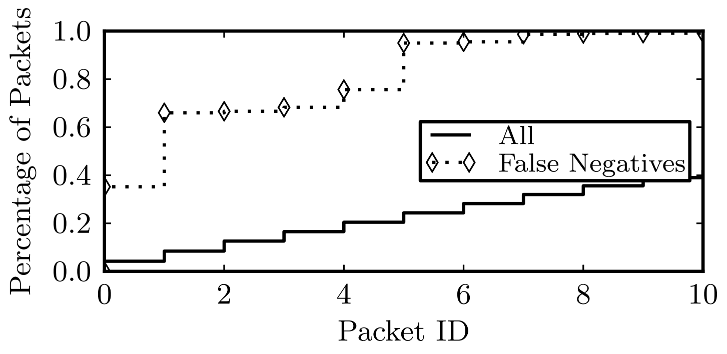
produced the same label as expected from the content type, of 95.1% on `log1` and 96.0% on `log2`. We refer to "match" rates here, rather than false-positive or false-negative rates, due to the large quantity of mislabeled content-types we encountered.

In the case of encrypted traffic (i.e., TLS/SSL) we accurately classified approximately 95% of the traffic. However, the mismatches are particularly interesting. Figure 2.3 shows the distribution of packet IDs, where a packet's ID is its position in the bi-directional flow (i.e., a packet with ID zero is the first packet sent by the originator of the connection). Notice that 94.8% of the mismatches for SSL/TLS occur within the first 5 packets, and 95% within the first 6 packets for SSH. These packets are all in the connection set-up phase and hence are not, in fact, encrypted. Moreover, these connection set-up packets, particularly any SSL certificates (typically around ID 2), may be of interest to DPI engines even when the encrypted payload is not.

A closer analysis of our overall results reveals that 50% of the transparent-as-opaque



(a) SSL/TLS



(b) SSH

Figure 2.3: CDFs of Packet IDs

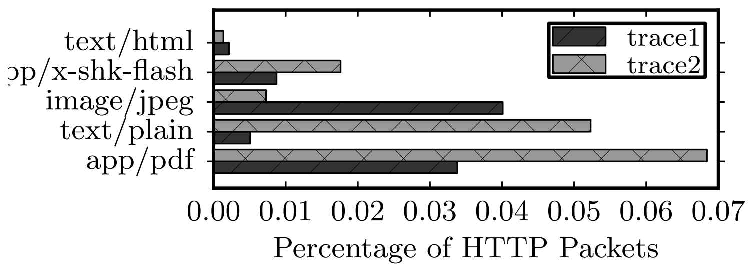


Figure 2.4: HTTP Mismatched Content-Types

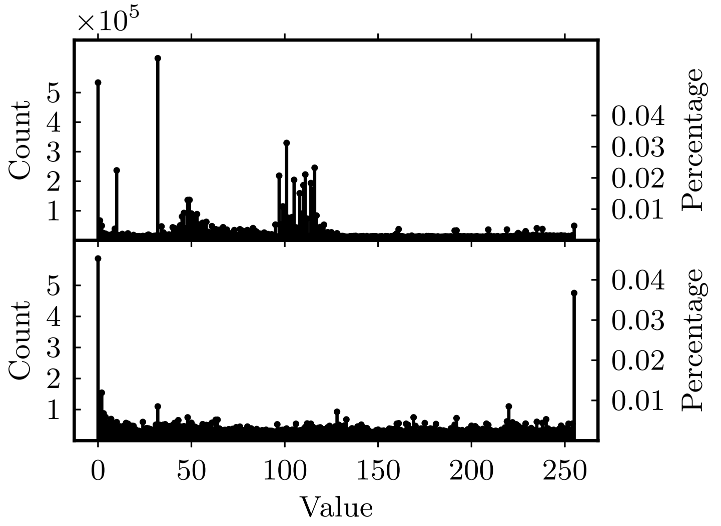
mismatches for `log1`, and 15% for `log2`, are from SMTP traffic, which we surmise includes opaque attachments for which we do not have accurate labeling information. Of the HTTP transparent-as-opaque mismatches (Figure 2.4), many are PDF and Flash (both of which can serve as a container for many other object types), but a surprising proportion are of type text/plain.

We investigated the text/plain mismatches from `log1` further and concluded, based on the observed byte-value distributions, that if these packets are in fact plaintext, the method used for encoding is non-standard. Figure 2.5a depicts the byte-value distribution for the mismatches and for the text/plain packets where the encoding was specified explicitly as one of UTF-8, UTF-16, US-ASCII or ISO-8859-1. As expected, the latter distribution has significant mass around the ASCII characters 'A'-'z' (values $65 - 122$ in Figure 2.5) while the former does not. A similar finding holds for the opaque-as-transparent mismatches in the image/jpeg case: the distribution (bottom, Figure 2.5b) has striking similarities to that observed for the case of known plaintext encodings (top, Figure 2.5a), and moreover, is quite different from that of the opaque JPEGs (top, Figure 2.5b). Unfortunately, without access to actual payloads for these two traces, we cannot say what was the underlying cause for the peculiar distribution in the content flagged by our methods, but we believe this underscores the utility of our techniques — i.e., we *successfully identified anomalous instances in both cases*. We revisit this issue in the next section.

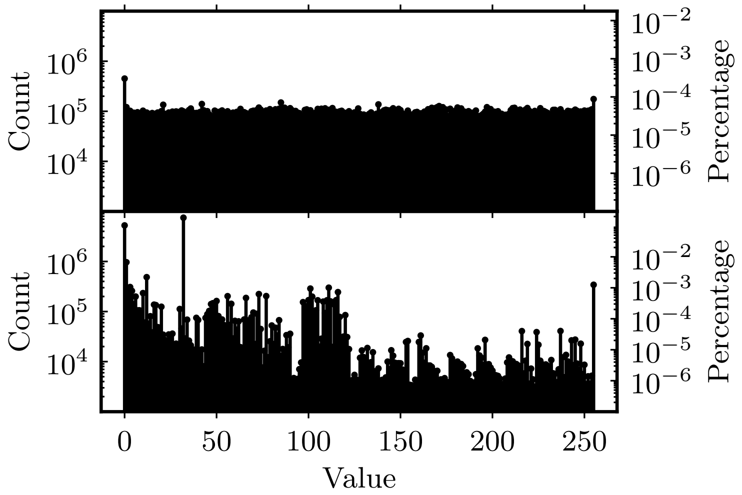
Finally, we examined the number of iterations needed by the truncated test to make a decision regarding each packet. With the maximum sample size set at 16 bytes, 45% of the packets can be classified in 12 bytes or less.

Operator Analysis

To gain deeper insights into the issue of Content-Type mismatches, we performed another experiment in which a resident network operator was able to manually inspect payload data from a third dataset, for which we were able to collect *full* payloads under the supervision of our local network operator. This trace covers four weekday afternoon hours of traffic from a university computer science department, and consists of 27 million packets. To enable this inspection, we instrumented Bro (version 2.0) to save both HTTP entities and the payloads of TCP connections to disk. We then ran the instrumented Bro against the port 80 (HTTP) and port 443 (SSL) traffic in the trace.



(a) text/plain (top: known encodings (transparent), bottom: opaque)



(b) image/jpeg (log-scaled; top: opaque, bottom: transparent)

Figure 2.5: Byte-value Distributions for Anomalies

We determined the opacity of each packet, saving those entities and connection payloads wherein the first five packets were mismatches. For the HTTP entities, we define a mismatch as having an opacity different from that implied by the Content-Type or Content-Encoding. For the streams on port 443, we consider transparent packets to be mismatches. We also captured relevant metadata, such as the uniform resource identifier (URI), Host, and MIME-type of the resulting file (determined using Bro's `libmagic` interface).

We discovered a number of mismatched HTTP entities in the trace, of both the transparent-as-opaque and opaque-as-transparent varieties. In the former case, many of these mismatches were HTTP entities labeled with Content-Type `text/plain` and no stated encoding; we determined from the metadata that most of these were either images or compressed bundles. These compressed bundles included what appear to be updates to Symantec antivirus, extensions to the Plex media center, and an installer for the DivX video codec (a `.cab` file). The images were identified by MIME-type as JPEGs. One interesting mismatch declared a Content-Type of `text/javascript`, with no encoding, while the MIME-type reported was `application/octet-stream` and the filename `.gz`.

Of the opaque-as-transparent mismatches, a number of entities were labeled as JPEG and GIF images by Content- and MIME-type but were flagged as transparent by our techniques. Many of these contain significant text, which we speculate may be metadata. We also found a large number of streams on port 443 with predominantly transparent packets. According to our network operator, these streams comprised: cleartext Yahoo Voice connection information (including account names), names and email addresses from a social networking chat site, and what appeared to be instant messenging conversations. As an aside, in looking at flagged mismatches while testing, one of the authors discovered his AIM contact list transmitted in the clear over port 443 by Adium!

2.3.2 Online Analysis

To further demonstrate the utility of winnowing opaque traffic in a real-world environment, we implemented our techniques as a preprocessor to the Snort IDS. Much of Snort's functionality, such as stream reassembly and HTTP inspection, is implemented as preprocessors which are executed after network and transport layer packet decoding but before engaging the rule-matching engine. We positioned our winnowing preprocessor to intercept packets before reaching the stream reassembly module.[3] This allows us to drop opaque packets early enough to avoid incurring the overhead (over 30% of Snort's run-time in some experiments) of stream reassembly and long before reaching the pattern matching engine.

The ruleset used was provided by a major university, which uses Snort primarily for post-facto forensics, and contains 1606 rules. As a sanity check, and to provide results on a public dataset which could therefore be easily reproduced, we evaluated both stock Snort and Snort with our winnowing preprocessor on the (admittedly controversial) DARPA/MITLL 1999 intrusion detection dataset. Both configurations produced exactly the same set of alerts.

For our online experiments, we made use of an Endace 9.2X2 DAG card to run two Snort (version 2.9.1.2) instances in parallel, one with stock Snort and the other with winnowing enabled, on live traffic. The DAG uses a packet processor to capture and timestamp packets at line rate, and employs an on-board direct memory access (DMA) engine to zero-copy packets into the host's memory (Figure 2.6). The DMA engine can also be programmed to duplicate traffic over multiple memory buffers (called "streams"), which enables us to simultaneously run multiple experiments, thereby comparing different approaches on the same traffic. We use a 10Gbps DAG connected to a 2.53 Ghz Intel Xeon 6 core host with 16GB of memory. As in our earlier experiments, we only examined traffic on ports 22, 25, 80 and 443; this filtering was performed on the DAG and therefore CPU resources are used exclusively for payload inspection. Beyond the winnowing preprocessor, there were no differences between the two configurations: both used the same ruleset with the default Snort options (including inspection of `gzipped` HTTP entities and the bypassing of the detection algorithms by SSL application data packets in established streams), and each was allocated 2GB of memory on the DAG.

Our primary experiment lasted for 24 hours and encompassed more than 7.6 terabytes of traffic; the load reached 1.2Gbps at peak. Figure 2.7a shows the number of packets which each instance of Snort was able to process in 15-minute intervals. Even at peak load, our winnowing Snort instance is able to handle the full volume with zero percent packet loss. In contrast, the stock Snort instance dropped nearly 60% of the 98.9 billion packets observed during the 24-hour

[3]Our preprocessor only drops packets with TCP payloads and therefore does not interfere with connection tracking at the transport layer.

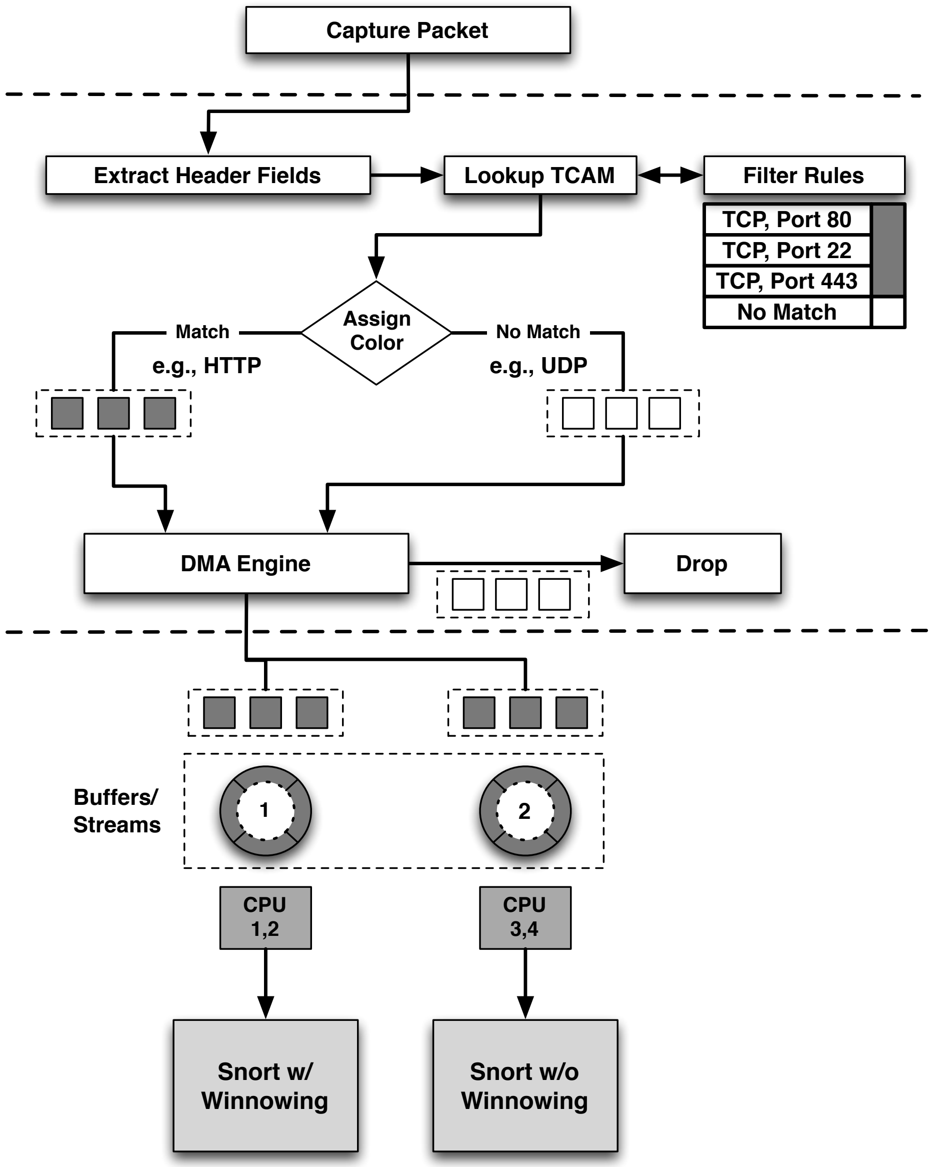
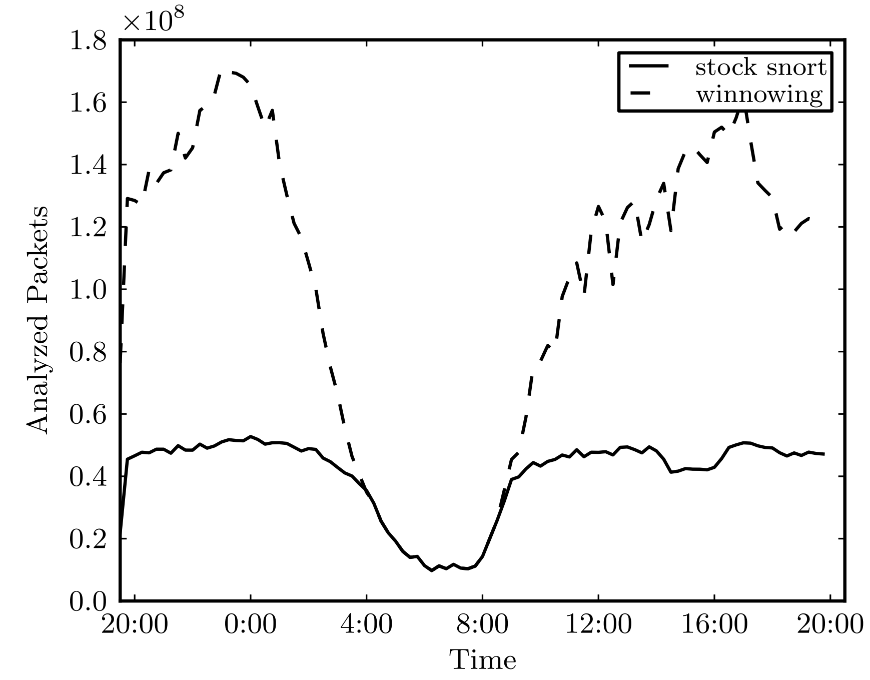
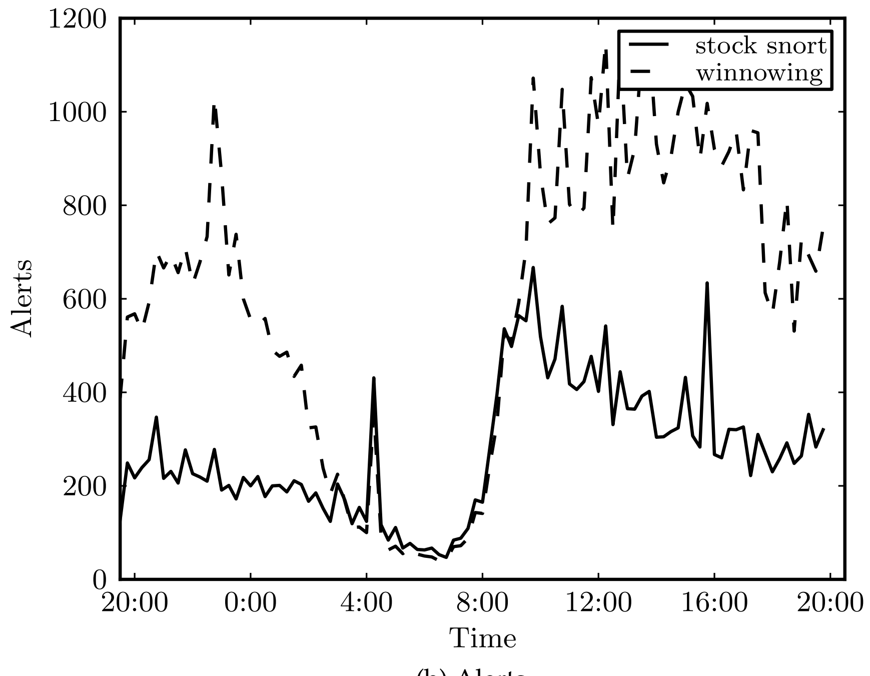


Figure 2.6: Packet duplication using an Endace DAG card.



(a) Packets



(b) Alerts

Figure 2.7: Number of alerts and number of packets processed by Snort with winnowing and Snort without (in 15-minute windows).

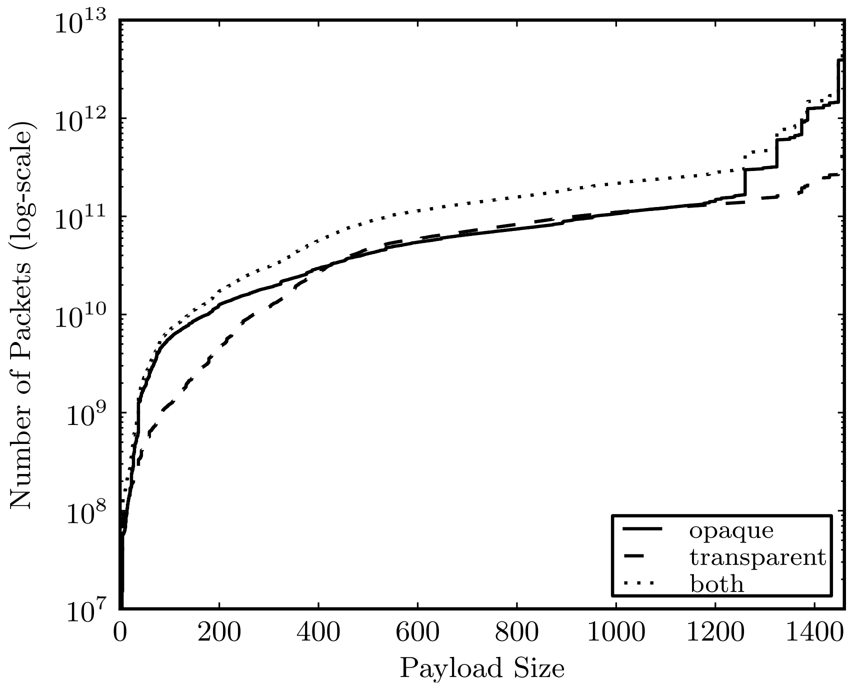


Figure 2.8: CDF of payload size for both opaque and transparent traffic.

window. In fact, the winnowing Snort instance processed 147% more packets, resulting in over 33,000 *additional alerts* (see Figure 2.7b). We note that as the drops suffered by the unmodified Snort are lost when Snort is unable to copy packets out of the network buffer fast enough to avoid old packets being overwritten by the kernel (or the DAG, in this case), and therefore Snort has no control over which packets are dropped [117].

In a second experiment, we partitioned one of the two identical traffic streams at the DAG into four disjoint streams, each of which was passed to a stock Snort instance running on a dedicated core. However, due to traffic bursts, the stock Snort instances still dropped significant numbers of packets despite our attempts to optimize its handling of the traffic. While this underscores the need for techniques like ours, we remind the reader that the advantages of winnowing traffic are not limited to situations of overload: the significant throughput increase allows systems to evaluate more rules than stock configurations under the same traffic conditions.

We stress that, apart from the winnowing preprocessor, the Snort instances in each experiment were *identically configured* and saw *exactly the same traffic*. The substantial improvement in capacity arises due to the surprisingly high proportion of opaque traffic, which, as discussed in Section 2.2, incurs high CPU overheads (confirmed in our experiments by an observed 30% increase in average per-packet processing time compared to transparent packets). In total, an astonishing 89% of the payload-carrying TCP packets observed in our primary experiment were classified as opaque, representing 86% of the bytes transmitted. We hypothesize that this is due to the prevalence of streaming video services such as Hulu, Netflix, and YouTube operating on port 80. Differences in the packet size distributions (see Figure 2.8) indicate that maximum transmission unit (MTU)-sized packets are far more often opaque than not, which suggests that large, compressed streams (e.g., the aforementioned streaming video), are prevalent. These streams may also account for the bursty nature of traffic observed in the multiple core experiment.

Operational Impact

Since winnowing opaque packets fundamentally changes the mix of traffic which reaches the rules-matching engine in an IDS, its influence could lead to deeper insights about network activities. To evaluate this further, we compared the alerts generated by both instances of Snort. In total, the stock Snort instance generated 25,081 alerts, while the Snort instance augmented with winnowing produced 58,297 alerts—a 118% increase. Of these, the three most prevalent in both cases remained the same; two overflow attack detections and a file-type identification alert (for portable executable binaries). We found the differences between the distributions of rules triggered by the stock Snort and by Snort with winnowing to be particularly interesting. One PowerPoint administrator privilege escalation attack rule was triggered 2000% more times by the winnowing Snort instance, an almost 10-fold increase over what one might expect just from the increased traffic volume. We also found a 760% increase in alerts for PDFs with embedded Javascript. While ascertaining whether winnowing induces any false positives is impossible due to the losses sustained by the stock Snort instance, we argue that intelligently dropping packets, as we do, is no more likely to induce false positives than dropping random packets due to overload. The only class of rules which produced fewer alerts when winnowing opaque traffic were for HTTP connections to blacklisted domains; these are the only clear false negatives. Since HTTP headers are in plaintext, and hence not dropped by Winnow, we suspect that the missed alerts are due to Snort failing to recover from missing (opaque) packets in a stream. Since parsing of HTTP headers is possible even in the presence of missing payload packets, we believe such alerts would be triggered if the IDS was better equipped to recover from such midstream losses. Alternatively, a more mature implementation of the winnowing preprocessor could inform Snort of the dropped packets.

Nevertheless, the instantiation of our prototype on a major campus network has already provided valuable information to its network operators, clearly showing that their network security exposure differs from what they originally thought.

2.4 Limitations

While our focus on minimizing payload inspection suggests a trivial method for an attacker to bypass DPI engines using our technique (by padding the beginning of the packet with ASCII bytes), in current systems an attacker need do nothing more than "encrypt" their exploit code (e.g., a simple XOR may suffice) to bypass content-based signature matching engines. Furthermore, our techniques comprise a method for flagging the likely presence of such encrypted objects based on nothing more than the ciphertext itself.

At first blush, it may appear that binary protocols present an insurmountable problem for our methodology. However, we point out that the majority of traffic (i.e., 80% on our network) is HTTP, a text protocol. Furthermore, the high-volume binary protocols (SSL, RTMP, and Bittorrent) transport primarily, if not exclusively, opaque data. We believe that a different alternative hypothesis, such as that mentioned earlier in the context of identifying file types, could provide improved accuracy for binary protocols.

Additionally, while it may seem that the flow level, as opposed to the packet-level, is the natural vantage point for identifying opaque traffic, our work concentrates on packet-level analysis. We argue that the presence of tunneled traffic, and container formats such as PDF, mandates identification of opaque traffic on a per-packet basis. In the specific case of encrypted connections, many protocols (e.g., SSH and SSL) begin with unencrypted, connection set-up packets; some protocols (e.g., using STARTTLS) are even designed specifically to enable upgrading a connection from unencrypted to encrypted mid-stream. Furthermore, packet-level techniques can be used in situations where flow state is not kept, such as on DAG cards. Finally, by performing packet-level analysis, we can winnow opaque packets before they reach the stream reassembly engine, significantly reducing overhead. That said, there are benefits to incorporating flow-level analysis. One interesting direction might be to limit packet-level misclassifications by utilizing information from prior and subsequent packets.

We remind the reader that our techniques are not intended to be used in isolation, but rather as components in larger systems. Therefore, the limitations of our techniques can be mitigated by the strengths of the remainder of the system architecture, such as by coordinating flow-level analysis with packet-level opacity checking. This would provide a layered defense against, e.g., attacks embedded in lengthy streams to evade systems using *selective packet discarding* [117], which is discussed in the next section, or approaches similar to the afore-mentioned Time Machine (see Section 2.1).

2.5 Related Work

The related problem of forensic file and data type identification has been extensively explored over the past decade. Many of these efforts have focused on analyses of *byte frequency distributions*, i.e., the frequency of each byte value in the data of interest. For the most part, these approaches work by creating signatures for known file types (i.e., HTML or PDF) based on byte frequency distributions, and then comparing unknown files to pre-existing signatures to determine the file type (see, e.g., Ahmed et al. [3]). Veenman [146] and Hall [56] examined entropy and compressibility measurements as indicators of file type, while others have explored techniques for directly modeling the structure of files [55, 60, 87, 128]. The closest work to the problem at hand is that of Conti et al. [29], who consider distinguishing between random, encrypted and compressed files using k-nearest-neighbor classification. Shannon entropy and the χ^2 statistic, among other measures, are evaluated as distance metrics over sliding windows of 1KB or more. Similar ideas were explored by Shamir and van Someren [132] for identifying cryptographic keys on disk. However, as our empirical analyses showed, the forensics scenario is fundamentally distinct from the setting we explore in this work: for real-time analysis on the network, the amount of data available is constrained and computational resources are scarce.

Approaches based on byte-frequency [150, 162] and entropy [94, 139, 141, 153] have also been applied in the context of malware identification and analysis, both on disk and in network traffic. These approaches are not well suited for our task, as they require large sample sizes for their statistical tests to be valid and/or impose high computational costs.

Most germane to our work is the scheme proposed by Olivain and Goubault-Larrecq, which uses hypothesis testing of the sample entropy statistic to determine whether packets are encrypted [111]. Similarly, Dorfinger proposed an approach for detecting encrypted traffic using entropy estimation, intended as a pre-filtering step for a Skype flow detection engine [17, 35–37, 144]. Their encryption detection scheme shares much in common with that of Olivain and Goubault-Larrecq, and is based on calculating the sample entropy for the packet payload and comparing that entropy value to the expected entropy value for a sequence of uniformly randomly distributed bytes of the same length. However, their entropy estimation approach does not scale well to situations where the number of samples is small [114–116]. For our entropy-based tests, we addressed this issue head-on by calculating the exact probability distribution function (see Section A.1) for the (byte-)entropy of n bytes, for small n. Malhotra compared a number of standard statistical tests, including the χ^2 and Kolmogorov-Smirnov tests, for identifying encrypted traffic [99]. As with Olivain and Goubault-Larrecq, their approaches required at least 1KB of data per trial. In any case, our byte-value tests outperformed all of these approaches.

From a systems perspective, similar notions have been suggested in the context of improving the ability of network intrusion detection system (NIDS) to weather inevitable traffic spikes by the *selective discarding* of packets when the system is under load: similar to Time Machine, Papadogiannakis, Polychronakis, and Markatos [117] propose discarding packets from lengthy flows, reasoning that these packets are less likely to trigger alerts. We believe our approach is more general, both in that we enable new policy decisions, as discussed earlier, and our techniques can operate on flows of any length. That said, our techniques are certainly ripe to be employed in a load-dependent fashion, but we focus on the more difficult load-independent setting in order to explore the limits of our techniques.

Hardware-based approaches for reducing the load on NIDS have also been proposed. That most closely related to our work is the notion of *shunting* [53], in which packets are matched in hardware against a dynamic list of rules which the IDS/IPS updates periodically. For each packet, the FPGA-based *shunt* decides, based on these lists, whether to drop, pass, or forward the packet to the IDS/IPS. Again, we see our techniques for the identification of opaque traffic as complementary to the shunting approach. Since the likelihood-ratio test can be, in the extreme, simplified to counting high-value bits, it can be easily implemented on an FPGA, allowing the NIDS to specify opacity-based rules for packet forwarding decisions.

Lastly, the application of sequential hypothesis testing to computer security problems is not new. For instance: Jung et al. [74] applied sequential hypothesis testing to portscan detection, Jaber and Barakat [70] proposed an iterative approach that used the size and direction of each packet to determine the most likely application generating the observed traffic, and Thatte, Mitra, and Heidemann [142] proposed a distributed denial-of-service detector based on a bivariate SPRT.

2.6 Discussion

In this chapter, we propose the notion of quick and accurate winnowing of opaque traffic as a mechanism for reducing the demands upon DPI engines, and introduce a number of statistical techniques for performing such winnowing. Our techniques are compared against those that might be considered common wisdom, and through extensive evaluation, we show that our statistical approaches perform best. Our results demonstrate that we are able to identify opaque data with 95% accuracy (Section 2.3.1). By implementing our approach with the Snort IDS, we demonstrate that winnowing vastly improves the rate at which an IDS can process packets without adversely impacting accuracy. Our experiments show that winnowing enables Snort to process 147% more packets and generate 135% more alerts over a 24-hour period featuring a peak traffic volume of 1.2Gbps.

It is our hope that the ability to classify traffic in the ways proposed in this chapter will provide significant benefits by enabling second-line (e.g., DPI) analysis engines to target each class with specialized approaches, rather than attempting to apply heavy-weight analyses to a general mix of traffic. That said, our real-world evaluations offer a cautionary tale; our experiments indicate that the vast majority—89% of payload-carrying TCP packets on our network—of modern network traffic is opaque. This finding was certainly a surprising result to us, and we believe it may have far reaching consequences in its own right. At the least, it calls into question the long-term viability of DPI techniques, and warrants revived interest in finding new ways for traffic monitoring and policy enforcement.

2.7 Future Work

Compressed vs Encrypted While we have demonstrated the usefulness of the identification and filtering of opaque traffic, we believe the specific identification of encrypted traffic would provide a greater benefit. Specifically, one proposal is a two-tiered approach to identifying encrypted packets: the first tier determining whether a packet is opaque, and the second tier making the further determination of whether the packet is encrypted. Such an approach allows an increase in the computational power used to determine compressed from encrypted traffic, since the second tier is only applied to packets already marked as opaque. Prior work [99] has indicated that statistics such as auto-correlation and index-of-coincidence can differentiate between compressed and encrypted files, and we therefore suggest the use of similar methods as the second tier.

Flow-level Analysis While we believe that decisions must fundamentally be made at the packet level (consider, e.g., STARTTLS), where a connection begins in the clear but switches to using encryption mid-stream), leveraging flow-level information should improve our decision accuracy as well as ease anomaly detection, interpretation by network operators, and coordination with IDS/IPS systems. We intend to evaluate a number of flow-level techniques (including the incorporation of a prior probability based on decisions made about previous packets) to determine the optimal approaches for the identification of both opaque and encrypted traffic.

2.8 Broader Implications

Moving forward, we are particularly concerned with the limitations of DPI engines with respect to encrypted traffic. While compressed traffic is certainly an issue for existing approaches which utilize DPI, the problem can be mitigated to an extent by increasing the resources available for capturing, storing, and decompressing such traffic before inspection. Encrypted traffic, on the other hand, represents a much more fundamental problem for DPI systems. While some see a solution in using network monitoring systems which intercept, decrypt, scan, and re-encrypt traffic, we see this approach as unacceptable for two reasons: 1) encryption is no longer end-to-end and thus user privacy is threatened; and 2) trust decisions (i.e., accepting a presented TLS certificate) are necessarily delegated to the system. While technical workarounds for the latter issue are no doubt possible, the former is a fundamental problem.

The previous paragraph notwithstanding, methods do exist for inferring information from encrypted traffic *without decryption*. However, such inference is not without it's own limitations: in particular, the amount of information gained has previously been seen as extremely limited, and the question of whether such information can be put to practical use has not—to our knowledge—even been raised. In the next two chapters, we attempt to answer these questions by exploring the extent to which information can be inferred from encrypted VoIP conversations (Chapter 3) and from HTTP traffic tunneled over encrypted connections (Chapter 4).

CHAPTER 3: PHONOTACTIC RECONSTRUCTION OF ENCRYPTED VOIP CONVERSATIONS[1]

If I have seen further, it is by standing on the shoulders of giants.

ISAAC NEWTON

3.1 Introduction

Over the past decade, VoIP telephony has witnessed spectacular growth. Today, VoIP is being used everywhere, and is making steady headway as a replacement for traditional telephony in both the residential and commercial sectors. The popularity of free online services such as Skype, Fring, and Google Talk is a case in point. Indeed, several analysts predict that VoIP will remain the fastest growing industry over the next decade, and some forecast that the subscriber base will top 225 million by 2013.[2] Yet, even with this widespread adoption, the security and privacy implications of VoIP are still not well understood. In fact, even with the attention VoIP security (or lack thereof) has received in the past, the concerns have mostly centered on the lack of authenticity in the call setup phases of the signal and session negotiation protocol(s) or susceptibility to denial of service attacks [79]. Regarding the confidentiality of the data streams themselves, the prevailing wisdom is that, due to the open nature of traffic traveling over the Internet, VoIP packets should be encrypted before transmission.

However, current practices for encrypting VoIP packets have been shown to be insufficient for ensuring privacy. In particular, two common design decisions made in VoIP protocols—namely, the use of VBR codecs for speech encoding and length-preserving stream ciphers for encryption—interact to leak substantial information about a given conversation. Specifically, researchers have shown that this interaction allows one to determine the language spoken in the

[1]The work presented in this chapter was previously published in: Andrew M. White et al. “Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks”. In: *2011 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2011, pp. 3–18.

[2]See, for example, Infonetics Research’s *VoIP and UC Services and Subscribers Report* at http://www.infonetics.com.

conversation [157], the identity of the speakers [5, 92], or even the presence of *known* phrases within the call [156].

Rightfully so, critics have argued that the aforementioned threats do not represent a significant breach of privacy. For example, the language of the conversation might easily be determined using only the endpoints of the call—a call from Mexico to Spain will almost certainly be in Spanish. While the identification of target phrases is more damning, it still requires the attacker to know (in advance) what she is looking for within the stream. In this work, we make no such assumption about *a priori* knowledge of target phrases. Rather, our ultimate goal is to reconstruct a hypothesized transcript of the conversation from the bottom up: our approach segments the observed sequence of packets into subsequences corresponding to individual phonemes (i.e., the basic units of speech). Each subsequence is then classified as belonging to a specific phoneme label, after which we apply speech and language models to help construct a phonetic transcription of parts of the conversation. To assess the quality of our reconstruction, we apply widely accepted translation scoring metrics that are designed to produce quality scores at the sentence level that correlate well with those assigned by human judges.

The approach we take has parallels to how infants find words in a speech stream. As Blanchard, Heinz, and Golinkoff [16] point out, adults effortlessly break up conversational speech into words without ever realizing that there are no pauses between words in a sentence. This feat is possible because we have a lexicon of familiar words that we can use to segment the utterance. Infants have no such luxury. Instead, they must use perceptual, social, and linguistic cues to segment the stream of sounds. Amazingly, the linguistic cues come from learned language-specific constraints (or *phonotactics*) that determine whether a word is well-formed or not; infants use this knowledge of well-formedness to help segment speech.

The fascinating problem here is that infants must learn these rudimentary, language-specific, constraints while simultaneously segmenting words. They use familiar words (e.g., their own names) to identify new words which are subsequently added to their small vocabulary. Interestingly, the Linguistic and Psychological Sciences literature abounds with studies (e.g., [18, 58]) which show that, as early as six months of age, infants use knowledge of which basic phonemes occur together, as well as learned knowledge of within-word versus between-word sounds, to segment perceived utterances into words. As we show later, we apply a similar methodology when tackling the problem of reconstructing words from strings of phonemes.

3.2 Background Information

Before proceeding further, we first present some necessary background that is helpful in understanding the remainder of this chapter. The background material covers basic notions in linguistics, pertinent VoIP details, and information about the datasets we use throughout the chapter.

3.2.1 Phonetic Models of Speech

The ideas in this chapter rely heavily on insights from modern theories of phonology. In particular, we draw from a vast body of work on phonetics — i.e., the study of linguistic sounds. From a computational perspective, phonetics involves studying how sound is produced by the articulators of the vocal tract and how they are realized acoustically [75]. In phonetics, the pronunciations of words are modeled as strings of symbols representing individual speech units called *phones*. While several alphabets exist for representing phones (e.g., ARPAbet for American English), the de facto standard is the International Phonetic Alphabet (IPA).

For the remainder of the chapter, what is particularly important is that each phone is based on articulatory processes, and that phones are divided into two main classes: consonants and vowels. Both kinds of sounds are formed by the motion of air through the mouth, throat and nose. Consonants, for example, are made by restricting airflow in some way, and can be both *voiced* (meaning they involve vibrations of the vocal cords) or *unvoiced*. By contrast, vowels usually involve less obstruction of air flow, and are louder and longer lasting than consonants. Moreover, because all consonants are sounds made by restricting airflow, they can be distinguished from each other by where the restriction is made (the *place* of articulation) as well as how the restriction is made (the *manner* of articulation). In English, for example, the "hissing" sound of [*f*] in "fish" is made by pressing the lower lip against the upper teeth. There are several major manners (e.g., stops, nasals, and fricatives) used to distinguish consonants.

Likewise, vowels can also be characterized by articulatory processes (see Figure 3.1), the most important of which are vowel *height* (i.e., roughly the height of the highest part of the tongue), *backness* (i.e., roughly indicating where the tip of the tongue is relative to the vocal track), and *roundness* (i.e., whether the shape of the lips is rounded or not). For example, compare how your

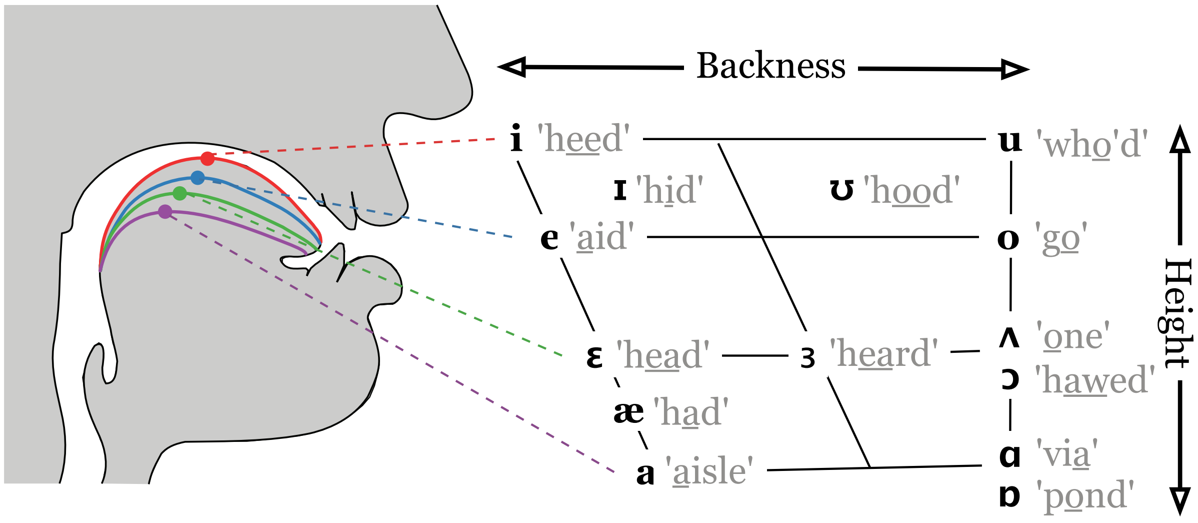


Figure 3.1: Vowels in American English (IPA format), differentiated by their *height* and *backness*. Left: the relative tongue positions.

mouth feels as you say "beat" and "boot". If you hold the vowels in these two words, you should be able to feel a difference in the *backness* of your tongue. Similarly, if you compare the words "beat" and "bat", you should feel your chin moving up and down; this is a difference in *height*. To feel a difference in *rounding*, compare the words "put" and "pool". As you say "pool", you should feel your lips pucker into a round shape; in "put", your lips should be loose.

Consonants and vowels are combined to make syllables, which are governed by the *phonotactics* of the language — that is, language-specific conditions that determine whether a word is well-formed or not. At a high level, phonotactics are constraints on which phones can follow which, i.e., rules that govern how phones may be combined to form well-formed words. In English, for example, there are strong constraints on what kinds of consonants can appear together: [st] (as in "stop") is a very common consonant cluster, but some consonant sequences, like [zdr] (as in "eavesdrop"), are not legal word-initial sequences in English.[3]

Lastly, in linguistics and speech processing, an abstraction called a *phoneme* (typically written between slashes) is used to represent similar phones with a single symbol. For example, the phoneme /t/ can be pronounced as any of three phones in English; which of these three phones is uttered depends on the position within a syllable: /t/ is pronounced as [t^h] at the beginning of

[3]Of course, [zdr] may exist word-initially in other languages, such as in the Bulgarian word [zdraf], which means "health".

a syllable (as in “top” = [t^{h}op’]), [t] in the middle of a syllable (as in “stop” = [stɒp’]), and [t’] at the end of a syllable (as in “pot” = [p^{h}ot’]). Phones belonging to the same phoneme are called *allophones*: [t^{h}], [t], and [t’] are allophones of the phoneme /t/.

In Section 3.5, we leverage such linguistic insights to build a string matching technique based on phonetic edit distance. In addition, we use phonotactics of English (e.g., what sequences of phonemes or allophones are allowable within words) to assist with phoneme classification.

3.2.2 Voice over IP

In VoIP, voice data and control messages are typically transmitted through separate channels. The control channel generally operates using an application-layer protocol, such as the extensible messaging and presence protocol (XMPP) used by Google Talk or the session initiation protocol (SIP). The voice channel typically consists of a real-time transport protocol (RTP) stream transmitted over UDP. We concern ourselves only with the voice channel in this work.

Typically, the audio for VoIP conversations is encoded using an audio codec designed specifically for speech, such as Skype’s SILK, the Enhanced Full Rate (EFR) codec specified by the GSM standard, or the open-source Speex used in many VoIP applications (including Google Talk). Speech codecs differ from general audio codecs since human speech can be represented much more efficiently than general audio due to the periodic nature of certain speech signals and the relatively limited number of potential sounds. For speech, sound is usually sampled at between 8 and 32 kHz (i.e., between 8,000 and 32,000 samples are recorded per second). This sample stream is then segmented into *frames*, or blocks, of a certain duration and each frame is compressed by the speech codec for transmission. The duration is a fixed value generally between 10 and 100ms; a typical value, and the one used in this work, is 20ms, which corresponds to 320 samples per frame when sampling at 16kHz.

Many modern speech codecs are based on variants of a well-known speech coding scheme known as code-excited linear prediction (CELP), which is in turn based on the *source-filter* model of speech prediction. The source-filter model separates the audio into two signals: the *excitation* or source signal, as produced by the vocal cords, and the *shape* or filter signal, which models the shaping of the sound performed by the vocal tract. This allows for differentiation of phonemes; for instance, vowels have a periodic excitation signal while fricatives (such as the [sh] and [f] sounds) have an excitation signal similar to white noise [145].

In basic code-excited linear prediction (CELP), the excitation signal is modeled as an entry from a fixed codebook (hence *code*-excited). In some CELP variants, such as Speex's VBR mode, the codewords can be chosen from different codebooks depending on the complexity of the input frame; each codebook contains entries of a different size. The filter signal is modeled using *linear prediction*, i.e., as a so-called adaptive codebook where the codebook entries are linear combinations of past excitation signals. The "best" entries from each codebook are chosen by searching the space of possible codewords in order to "perceptually" optimize the output signal in a process known as *analysis-by-synthesis* [145]. Thus an encoded frame consists of a fixed codebook entry and gain (coefficient) for the excitation signal and the linear prediction coefficients for the filter signal.

Lastly, many VoIP providers (including Skype) use VBR codecs to minimize bandwidth usage while maintaining call quality. Under VBR, the size of the codebook entry, and thus the size of the encoded frame, can vary based on the complexity of the input frame. The specification for secure RTP (SRTP) does not alter the size of the original payload; thus encoded frame sizes are preserved across the cryptographic layer. The size of the encrypted packet therefore reflects properties of the input signal; it is *exactly* this correlation that our approach leverages to model phonemes as sequences of lengths of encrypted packets.

3.3 Overview of Our Approach

The approach we pursue in this chapter leverages the correlation between voiced sounds and the size of encrypted packets observed over the wire. Specifically, we show that one can segment a sequence of packet sizes into subsequences corresponding to individual phonemes and then classify these subsequences by the specific phonemes they represent. We then show that one can segment such a phonetic transcript on word boundaries to recover subsequences of phonemes corresponding to individual words and map those subsequences to words, thereby providing a hypothesized transcript of the conversation.

Our work draws from advances in several areas of computational science. A simplified view of our overall process is shown in Figure 3.2. As an example, we use the phrase "rock and roll", the dictionary pronunciation for which is represented as [ɹɒk ænd ɹoʊl] in IPA. Our basic strategy

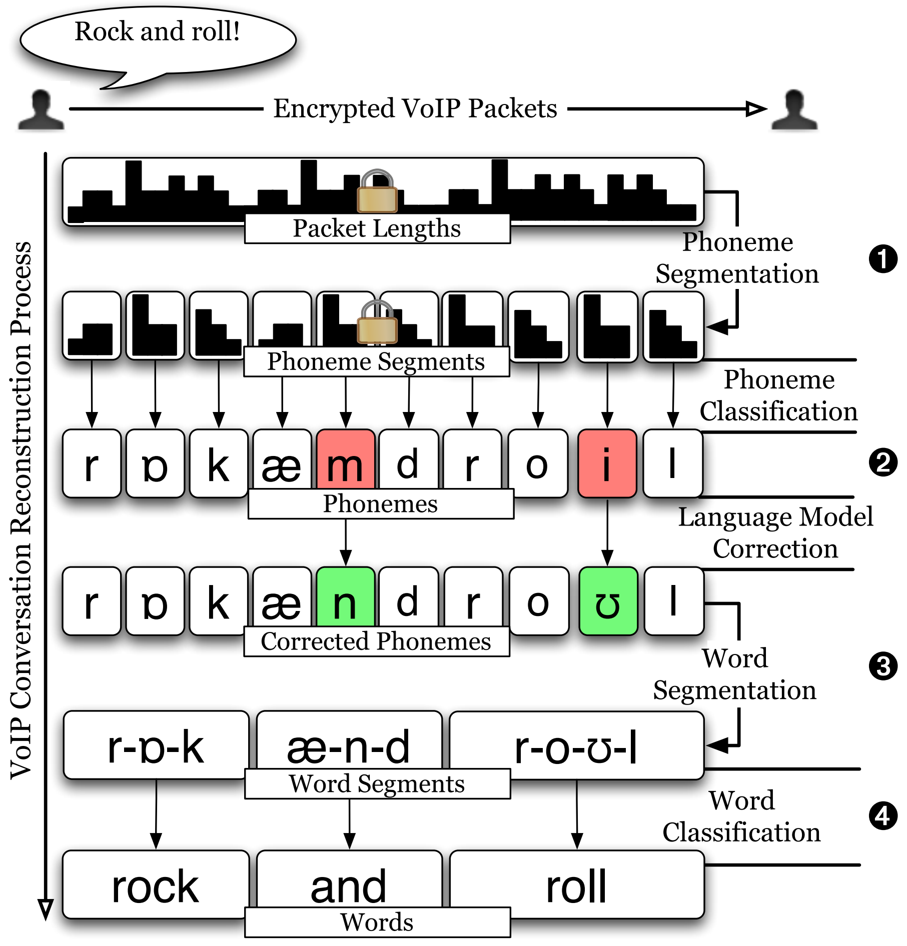


Figure 3.2: Overall architecture of our approach for reconstructing transcripts of VoIP conversations from sequences of encrypted packet sizes.

is as follows. First, we use a *maximum entropy* model (Stage ❶) to segment the sequence of packet sizes into subsequences corresponding to individual phonemes. We then apply (Stage ❷) a combination of maximum entropy and *profile hidden Markov* models to classify each subsequence of packet sizes according to the phoneme the subsequence represents, resulting in an approximate phonetic transcript of the spoken audio. In our example, this transcript is [ɹɒkæmdɹoil].

The hypothesized transcript is improved by applying a trigram language model over phonemes (and phoneme types) which captures contextual information, such as likely phoneme subsequences, and corrects the transcript to represent the most likely sequence of phonemes given both the classification results and the language model. In our example, this results in [ɹɒkændɹoʊl]. Notice the unlikely phonetic sequence [æmd] has been replaced with the far more likely [ænd]. Next, we segment (Stage ❸) the resulting transcript into subsequences of phonemes corresponding to individual words using a phonetic constraint model, resulting in the more recognizable string [ɹɒk ænd ɹoʊl].

Finally, we match each subsequence to the appropriate English word using a phonetic edit distance metric (Stage ❹), giving us the desired "rock and roll". In the general case, a trigram language model over words (and parts-of-speech) is then applied to the resulting transcript to correct tense and disambiguate between homophones (i.e., words which sound alike) by finding the most likely sequence of words given both the hypothesized transcript and the language model.

3.3.1 Data and Adversarial Assumptions

The *TIMIT Acoustic-Phonetic Continuous Speech Corpus* [48], a collection of recorded speech with time-aligned word and phonetic transcripts (allowing us to label segments by phoneme), provides the audio samples used in our experiments. The TIMIT corpus is comprised of 6,300 speech recordings from 630 speakers representing eight major dialects of American English. Each speaker reads ten pre-determined, phonetically-rich sentences, such as "Alimony harms a divorced man's wealth", "The drunkard is a social outcast", and "She had your dark suit in greasy wash water all year". The transcripts contain labels for 58 distinct phoneme-level[4] sounds. Following the standard approach used in the speech recognition community, we folded the original TIMIT classes into 45 labels [83] by combining some allophones and combining closures and silences. ARPAbet, the phonetic alphabet on which the labeling systems of TIMIT is based, does not map directly to the articulatory features in Section 3.2; therefore, we convert the phoneme sequences to their IPA representations for the latter stages of our evaluation. In order to generate sequences of encoded frame lengths from the (16kHz, single-channel) audio samples, we encode each sample using the reference version of the Speex encoder, instrumented to output the sizes of the encoded frames, in wideband (i.e., 16kHz) VBR mode. The phonetic labels from the time-aligned transcripts are then used to identify subsequences corresponding to individual phonemes for training; this encoding process gives us a number of sequences for each phoneme.

[4]In addition to phonemes, the corpus contain some labels for sounds, such as pauses and recording errors, unrelated to human speech.

We note that the approach we take assumes that the adversary has access to: 1) the sequence of packet lengths for an encrypted VoIP call; 2) knowledge of the language spoken in the call; 3) representative example sequences (or models derived therefrom) for each phoneme; and 4) a phonetic dictionary. The first assumption can be readily met through any number of means, including the use of a simple packet sniffer. Knowledge of the language of interest can be gleaned using the ideas in [78, 157] or by simple endpoint analysis. Lastly, obtaining representative example sequences for each phoneme is fairly straightforward: one can use prerecorded, phonetically-labeled audio files as input to a speech codec to produce the examples. In fact, using labeled examples from prerecorded audio is exactly the approach we take in this chapter in order to model phonemes. Note that our primary goal is to build speaker-independent models and thus we do not require speaker-specific audio. Finally, phonetic dictionaries (e.g., `CELEX`, `CMUdict` and `PRONLEX`) are readily available; we use data from TIMIT and from the `PRONLEX` dictionary (containing pronunciations from over 90,000 words) as our phonetic dictionary.

3.4 Related Work

Traffic analysis of encrypted network communications has a long and rich history. Much of that work, however, is focused on identifying the application protocol responsible for a particular connection (e.g., [12, 30, 42, 77, 98, 102, 159]). It was not until recently that researchers [23, 88, 130, 136, 140] began exploring techniques for inferring sensitive information within encrypted streams using only those features that remain intact after encryption—namely packet sizes and timing information. Song, Wagner, and Tian [136], for example, used the inter-arrival time between packets to infer keystrokes in SSH sessions; Sun et al. [140] and Liberatore and Levine [88] showed that identification of web sites over HTTPS connections is possible using the sizes of the HTML objects returned by HTTP requests; and Saponas et al. [130] showed how to identify the movie being watched over an encrypted connection.

More pertinent to this chapter, however, is the work of Wright et al. [156, 157] that showed that encrypted VoIP calls are vulnerable to traffic analysis wherein it may be possible to infer the spoken language of the call or even the presence of certain phrases. In the latter case, the approach of Wright et al. assumes that the objective is to search an encrypted packet stream for subsequences matching a target phrase or word, such as "attack at dawn", and therefore requires that a probabilistic model of likely corresponding packet length sequences (i.e., representing the target phrase in its entirety) be generated *in advance*. As discussed earlier, no such *a priori* information is necessary under our approach: we construct transcripts from the bottom up rather than matching phrases from the top down.

Several other approaches for exploring information leakage in encrypted VoIP calls, working under different environmental assumptions than Wright et al., have also been proposed. For example, if silence suppression is assumed (i.e., packet transmission is suppressed when a party is silent), researchers posit that the duration of talk spurts for words spoken in isolation makes identification of specific "speeches" [85, 92] possible. In a recent study with 20 speakers, Backes et al. [5] show that speaker-specific pause patterns might be sufficient to undermine the anonymity of speakers in encrypted VoIP calls. That said, it is well accepted in the speech community that continuous speech (i.e., everyday communication) lacks identifiable pauses between words [26]. In fact, speakers generally talk faster (and typically shorten or run sentences together) as speech becomes more natural and colloquial. This observation is even more important in our context where there are no within-word pauses. Hence, we make no assumptions about voice activation detection and/or silence suppression.

Lastly, Dupasquier et al. [40] investigate the extent of information leakage from Skype voice traffic. The authors conclude that the general concept we pursue here "seems quite difficult" because classification of phonemes is too challenging. Thus, they revert to the prior setting of knowing the target phrase in advance and use dynamic time warping to validate the work of Wright et al. A focus of this chapter is showing that such statements were premature, and that phoneme-level reconstruction can be successful in undermining the privacy of encrypted VoIP conversations.

For conciseness, the relevant literature on speech and language models will be presented elsewhere in this chapter.

3.5 Methodology

We now turn our attention to explaining the details behind the key ideas explored in this chapter. Wherever possible, we provide the intuition that drives our design decisions.

3.5.1 Finding Phoneme Boundaries (Stage ❶)

Given the sequence of packet sizes from a VoIP conversation, the first challenge is to identify which of these packets represent a portion of speech containing a boundary between phonemes. While automatic segmentation of speech waveforms on phonetic boundaries has received much attention in the speech recognition community, in our context we have no access to the acoustic information and must operate on the sequence of packet sizes. However, recall that many speech codecs, and Speex in particular, are based on CELP, which encodes speech with two different signals: the excitation signal and the filter signal. As mentioned earlier, the filter signal for a given frame is modeled as a linear combination of past excitation signals. Thus more information must be encoded for frames in which the sound changes drastically—such as at the transition between two phonemes. Similarly, less information is encoded for intra-phoneme frames, where the sound changes relatively little. Figure 3.3 illustrates how changes in frame size can indicate a phonetic boundary.

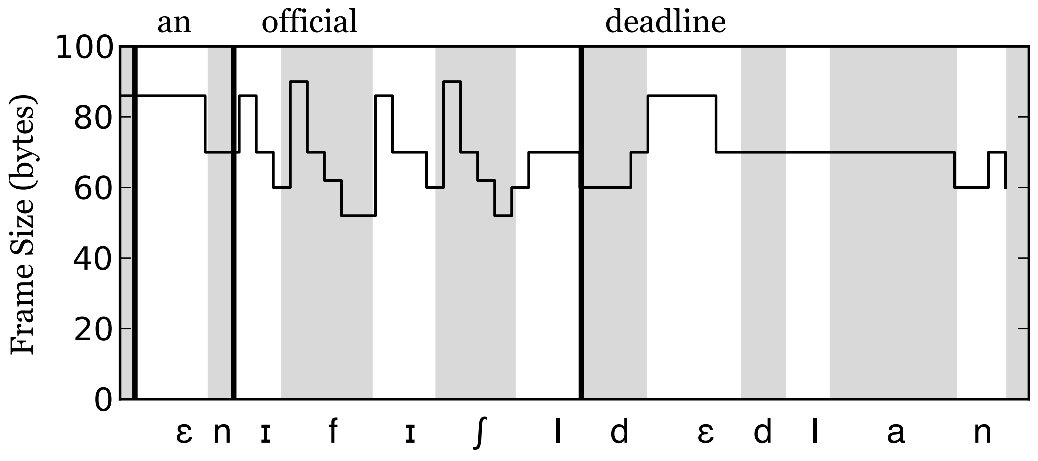


Figure 3.3: Frame size sequence for the first few words of an utterance of "an official deadline cannot be postponed", illustrating how the sizes of frames differ in response to phoneme transitions. Notice the distinct changes (e.g., a sharp rise) in frame sizes near some phoneme boundaries (e.g., [ɪ], [f], and [ʃ] in "official"). Near other phoneme boundaries (e.g., [d], [l], and [a] in "deadline"), however, frame size remains constant.

3.5.2 Methodology

To perform the segmentation, we apply a probabilistic learning framework known as *maximum entropy modeling*[5] [9, 71] that simultaneously captures many contextual features in the

[5]Also known as multinomial logistic regression.

sequence of frames, as well as the history of classifications in the sequence, to decide which frames represent phoneme boundaries. Such models have been successfully applied to problems like part-of-speech tagging [124] and text segmentation [8].

Maximum entropy modeling consists of finding a model $p(y \mid x)$, where x is an observation and y a label, to accurately estimate the posterior probability $\mathbb{P}[y \mid x]$. Many possible models exist; to find the best such model, we first impose a set of constraints on the model p in terms of binary indicator functions f_i (known as *feature* functions), each of which describes an aspect of the data relevant to classification. From the set of possible models which satisfy these constraints, we then select the model with the highest entropy.

In the case of phonetic boundary segmentation, we represent a given frame with w. The labels, i.e., *boundary* or *interior frame*, are represented by the binary variable v. An indicator function $f(w, v)$ then describes a feature of the frame which is relevant to whether that frame represents a phoneme boundary; for example:

$$f(w,v) = \begin{cases} 1, & \text{if } v \text{ is } \textit{boundary} \text{ and } w \text{ has size } n, \\ 0, & \text{otherwise.} \end{cases}$$

We now form a set of constraints on the final model with respect to such feature functions. To do so, we first calculate the empirical distribution $\tilde{p}(x, y)$ from the training data as the relative frequency of examples with value x and label y. Given an indicator function, we can then compute the expected value of a feature f with respect to the training data as:

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

We can thus represent any statistical phenomena in the training data with $\tilde{p}(f)$. The expected value of f with respect to the target model, $p(y \mid x)$, may be represented as:

$$p(f) = \sum_{x,y} \tilde{p}(x) p(y \mid x) f(x,y)$$

Requiring that $\tilde{p}(f) = p(f)$ imposes the constraint that the model agree with the training data with respect to feature f; over a set of n features, this yields a set of constraints for the target model:

$$\mathcal{C} = \{p \mid p(f_i) = \tilde{p}(f_i) \text{ for } i \in \{1, 2, \ldots, n\}\}$$

#	Phoneme Segmentation Feature Templates
1	size of frame w_i (i.e., the current frame size)
2	size of frame w_{i-1} (i.e., the previous frame size)
3	size of frame w_{i+1} (i.e., the next frame size)
4	bigram of sizes for frames w_{i-1}, w_i
5	bigram of sizes for frames w_i, w_{i+1}
6	trigram of sizes for frames w_{i-1}, w_i, w_{i+1}
7	sequence of frame sizes since the last hypothesized boundary
8	number of frames since since the last hypothesized boundary

Table 3.1: Feature templates for the phonetic segmentation, where w_i represents the ith frame.

Many models may satisfy the set of constraints. However, the principle of maximum entropy states that the model that best represents the data given the current state of knowledge is the one with the most entropy. This yields a constrained optimization problem of the form:

$$\arg\max_{p \in \mathcal{C}} H(p)$$

where $H(p)$ is the entropy of y conditioned on x in the model p.

For boundary identification, we define several feature *templates* which specify features that we hypothesize correlate with phoneme boundaries. The templates we use are given in Table 3.1, and some features are illustrated in Figure 3.4 for clarity. Although each frame only gives us one observable feature (namely, the size of the frame), we leverage the surrounding frames, and the history of previously classified frames, in the sequence to create a much richer feature set. The templates are used to automatically generate the full feature set directly from the data.

As per our hypothesis regarding the interaction between linear prediction and frame size, notice that feature templates 1–6 capture the frame sizes in the proximity of the current frame. The frame size unigrams, bigrams, and trigrams must be explicitly captured because maximum entropy models do not model feature interactions, i.e., they only consider individual features in isolation. Some phonemes may always exhibit a certain frame size sequence; we capture this behavior with feature template 7. Lastly, because some boundaries are not detectable by frame size changes (such as the long sequence of same-sized frames in Figure 3.3), we also model features such as phoneme length (feature template 8).

To efficiently solve the optimization problem posed by maximum entropy modeling, we use the *megam* framework with the limited-memory BGFS [89] algorithm to obtain the model $p(w|v)$.

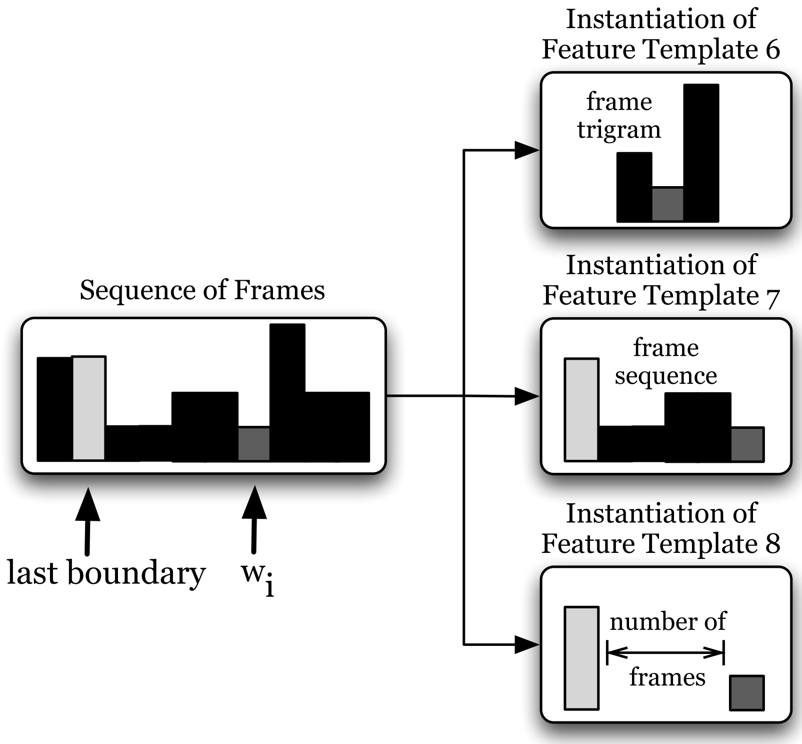

Sequence of Frames
last boundary
w_i
Instantiation of Feature Template 6
frame trigram
Instantiation of Feature Template 7
frame sequence
Instantiation of Feature Template 8
number of
frames

Figure 3.4: Illustration of features for frame w_i. The label for w_i is dependent on a number of features, including the frame size sequence since the last hypothesized boundary (shown here in gray). An example feature derived from each of templates 6–8 is depicted on the right-hand side.

Having built a model, we estimate the probability of each frame, in order, being a phoneme boundary by evaluating the estimated posterior $p(w|v)$. Since feature templates 7 and 8 depend on previously classified labels, we use a dynamic programming algorithm to maximize the likelihood of the sequence as a whole rather than greedily selecting the most likely label for each frame. The algorithm, a beam search, stores a list of the l most likely candidate segmentations up to the current frame; this list is updated after each frame is evaluated. We choose as our final segmentation the most likely candidate at the last frame.

3.5.3 Evaluation

In order to provide rigorous assessments of our methodology, we perform *cross-validation* in the segmentation and classification stages of our experiments. Cross-validation is a method for estimating the generalization performance of a classifier by partitioning the available data into complementary subsets, training with one subset, and testing with the other. In particular, we perform k-fold cross-validation, in which the data is partitioned into k complementary subsets. For each fold, one subset is selected for testing and the remainder used for training. The training and testing are performed as many times as there are subsets, with each acting as the testing set in one fold. The results of all iterations are then averaged to give the expected generalization performance, which mitigates the possibility of experimental results being unduly influenced by fortuitous selection of training and testing data.

To evaluate the performance of our phonetic segmentation model, we perform a 5-fold cross-validation experiment for each dialect in the TIMIT corpus. Using a holdout set of female speakers from the New England dialect, we experimentally determined an optimal value of 8 for the beam width l. We report the performance using precision (i.e., the fraction of boundaries in the transcription that are present in the reference transcription) and recall (i.e., the fraction of boundaries in the reference that appear in the transcription) as our metrics.

	$n = 1$		$n = 2$	
Dialect	Precision	Recall	Precision	Recall
New England	0.8539	0.7233	0.9443	0.8735
Northern	0.8555	0.7332	0.9458	0.8837
North Midland	0.8509	0.7372	0.9402	0.8901
South Midland	0.8452	0.7086	0.9352	0.8627
Southern	0.8525	0.7037	0.9405	0.8586
New York City	0.8530	0.7096	0.9386	0.8628
Western	0.8586	0.7259	0.9439	0.8652
Army Brat	0.8465	0.7540	0.9389	0.8985

Table 3.2: Phonetic segmentation performance for each dialect in the TIMIT corpus.

While interpreting these results, we note that Raymond et al. [125] have shown that phoneme boundaries are inexact even at the frame level—in fact, in their study, human transcribers agreed (within 20ms) on less than 80% of the boundaries. For this reason, a frame classified as a boundary is considered as correct if it occurs within n frames of an actual boundary; likewise, it is incorrect if there are no actual boundaries within n frames. Table 3.2 summarizes, for each dialect, our segmentation performance for $n = 1$ and $n = 2$. For the sake of comparison, we also note that state-of-the-art classifiers (operating on the raw acoustic signal) are able to recall approximately 80% (again, within 20ms) of phoneme boundaries in TIMIT [44, 106] with error rates similar to our own. Unfortunately, the comparison is not direct: our labels are necessarily at the granularity of frames (each 20ms), rather than samples, which means that the within-n-frame requirement for agreement is looser than the within-20ms requirement.

The results in Table 3.2show that our performance is on par with these other techniques. More importantly, the imprecision in the transcription boundaries does not negatively impact the performance of the next stage of our approach since the frames in question, i.e., the beginning and ending frames of each phoneme, are precisely those that will contribute the most variance to a phoneme model. In other words, the transition frames are likely to incorporate a significant amount of noise, due to their proximity to surrounding phonemes, and are therefore unlikely to be useful for classifying phonemes. It is for exactly this reason that we explicitly exclude the transition frames in the phoneme classification stage that follows. Finally, for the remainder of this chapter we make the simplifying assumption that phoneme boundaries can be recognized correctly; this assumption is revisited in Section 3.6.

3.5.4 Classifying Phonemes (Stage ❷)

We remind the reader that our overall approach requires that we segment a sequence of encrypted packet lengths into subsequences corresponding to individual phonemes, and then classify these subsequences based on empirical models derived from labeled training data. We therefore have a classification problem where the classes of interest are the various phonemes.

For classification, we employ a combination of two systems: one context-dependent, wherein the labeling of a segment is dependent on the labelings of its neighbors, and another context-independent, wherein a single segment is considered in isolation. We combine these two approaches in order to leverage the strengths of each. Our context-dependent classifier is also based on maximum entropy modeling, while the context-independent classifier is based on hidden Markov models (HMMs). Profile HMMs have been used widely in both the biological sequence analysis [41] and speech recognition communities [72].

Aside from the ability to incorporate contextual information, maximum entropy modeling is *discriminative*. Discriminative models are often used for classification tasks because they model only the parameters of interest for classification and thus can often encode more information. HMMs, on the other hand, are *generative* models. Generative models are sometimes preferable to discriminative models because they model the entire distribution over examples for a given class rather than just the information necessary to discriminate one class from another.

To combine the two models, we utilize a form of Bayesian inference to update the posterior given by the maximum entropy classifier with the “evidence” given by the HMM classifier. The updated posterior is then passed to a language model, as described below. By utilizing both types of models we enjoy increased classification accuracy while providing input to the language model with a valid statistical interpretation. Next, we discuss each stage in turn.

3.5.5 Maximum Entropy Discrimination of Phonemes

We discriminate between phonemes in a manner similar to the segmentation process described in Section 3.5.1. Specifically, we define a new set of feature templates over sequences of phonemes (which are themselves composed of sequences of frame sizes). For pedagogical reasons, the specifics are given in Table 3.3 and an example feature is illustrated in Figure 3.5.

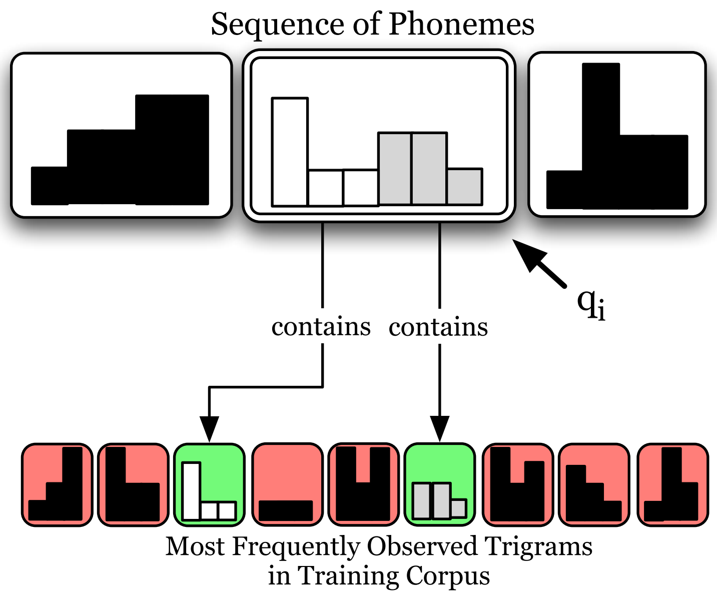


Figure 3.5: An example instantiation of feature template 10 which illustrates how the template models the presence of common trigrams.

Feature templates 1–3 capture the exact frame sequence of the current and surrounding phonemes to identify phonemes that frequently encode as exactly the same frame sequence. Feature templates 4 and 5 encode similar information, but drop the first and last frames in the sequence in accordance with our earlier hypothesis (see Section 3.5.1) that the beginning and ending frames of the phoneme are the most variable. Feature templates 6 and 7 explicitly encode

#	Template Description
1	q_i (i.e., the current phoneme's frame size sequence)
2	q_{i-1} (i.e., the previous phoneme's frame size sequence)
3	q_{i+1} (i.e., the next phoneme's frame size sequence)
4	q_i, excluding the first and the last frames
5	q_{i-1}, excluding the first and the last frames
6	length of q_i (in frames)
7	length of q_{i-1} (in frames)
8	frequency of frame size n in q_i
9	bigram b of frame sizes is in q_i, for top 100 bigrams
10	trigram t of frame sizes is in q_i, for top 100 trigrams
11	bigram b of frame sizes is in q_{i-1}, for top 100 bigrams
12	trigram t of frame sizes is in q_{i-1}, for top 100 trigrams
13	bigram b of frame sizes is in q_{i+1}, for top 100 bigrams
14	trigram t of frame sizes is in q_{i+1}, for top 100 trigrams

Table 3.3: Feature templates for the maximum entropy phoneme classifier. We denote as q_i the sequence of frame sizes for the ith phoneme. We limit the number of n-grams to 100 for performance reasons.

the length of the current and previous phonemes since some types of phonemes are frequently shorter (e.g., glides) or longer (e.g., vowels) than others. Feature template 8 captures the frequency of each possible frame size in the current sequence. Feature templates 9–14 encode the presence of each of the 100 most frequent frame size bigrams or trigrams observed in the training data; we limit the number of bigrams and trigrams to maintain manageable run-time performance. Finally, since we later incorporate high-level contextual information (such as neighboring phonemes) explicitly with a language model, we do not attempt to leverage that information in the classification model.

3.5.6 HMM Modeling of Phonemes

To provide generative models of the various phonemes, we train a profile HMM for each. A profile HMM is a HMM with a specific topology that encodes a probability distribution over finite sequences of symbols drawn from some discrete alphabet. In our case, the alphabet is the different sizes at which a speech frame may be encoded; in Speex's wideband VBR mode, there are 19 such possibilities. Given the topology of a HMM, we need to estimate the parameters of the model for each set of sequences. Towards this end, we utilize a well-known algorithm due to Baum et al. [7] that iteratively improves the model parameters to better represent the example sequences.

3.5.7 Classification

To label an observed sequence of packet sizes, we find the posterior probability $\mathbb{P}[r \mid q]$, where q represents the observed sequence of frame sizes, for each class label r. For the standalone maximum entropy classifier, the output for a given observation and label is an estimate of the desired quantity. For the HMM classifier, we calculate, using Bayesian inference, the posterior $\mathbb{P}[r \mid q] = \mathbb{P}[r]\mathbb{P}[q \mid r]$ using the likelihood[6] $\mathbb{P}[q \mid r]$, given by the HMM. This "updates" a prior probability $\mathbb{P}[r]$ with the new "evidence" from the HMM. For the stand-alone classifier evaluation, we estimate the prior $\mathbb{P}[r]$ as the proportion of examples belonging to the class in our training data. When using both the HMM and maximum entropy classifiers in conjunction, we use the estimated $\mathbb{P}[r \mid q]$ from the maximum entropy model as the prior $\mathbb{P}[r]$. In all cases, we choose the label whose model has the maximum posterior probability as the predicted label for a given sequence. These posterior probabilities also give a probability distribution over candidate labels for each phoneme in an utterance; these serve as the language model input.

3.5.8 Enhancing Classification using Language Modeling

Lastly, in order to incorporate contextual information on surrounding phonemes, we apply a trigram language model using the SRI Language Modeling (SRILM) toolkit. In particular, we train a trigram language model over both phonemes and phoneme types (e.g., vowels and stops). We disambiguate between candidate labels by finding the maximum likelihood sequence of labels given both the estimated distributions output by the classifier and the phonetic language model.

3.5.9 Evaluation

Our preliminary results show that we can correctly classify 45% of phonemes in a 10-fold cross-validation experiment on the New England dialect.[7] For this experiment, we operate on input with perfectly segmented phonetic boundaries so as to provide a baseline for our classifiers when evaluated independently from the other stages in our method. As can be seen from Figure 3.6, the combination of the HMM and maximum entropy classifiers with the language

[6]The likelihood given by an HMM is scaled by the marginal $\mathbb{P}[q]$.

[7]For brevity, we omit the other dialects as the results do not differ significantly.

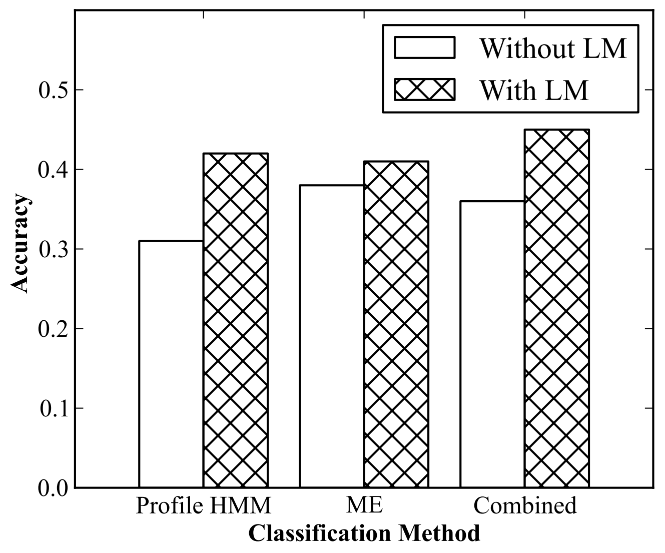


Figure 3.6: Phoneme classification accuracy on the New England dialect for the HMM and maximum entropy classifiers alone, in combination, and with the language model applied.

model outperforms the individual classifiers.

While this classification performance might sound lackluster, these results are quite surprising given the limited context we operate under (i.e., packet sizes only). For instance, recent approaches working directly on the *acoustic signal* report 77% accuracy on the TIMIT dataset in the context-dependent case (which corresponds roughly to our approach after application of the language model). In the context-independent case (analogous to our HMM classification approach without the language model), accuracy rates as high as 67% have been achieved [64] on the TIMIT dataset. Similarly, expert human transcribers achieve rates only as high as 69% [83].

3.5.10 Segmenting Phoneme Streams into Words (Stage ❸)

In this stage, our task is to identify likely word boundaries from the stream of classified phonemes. To do so, we follow the methodology suggested by Harrington, Watson, and Cooper [59] that, until very recently, was among the best approaches for word boundary identification. We also extend their approach to incorporate an additional step that makes use of a pronunciation dictionary.

Harrington, Watson, and Cooper identify word breaks with a two-step process. The first step consists of inserting potential word breaks into the sequence of phonemes in positions that would otherwise produce invalid phonemic triples, i.e., triples that do not occur within valid

Dialect	Precision	Recall
New England	0.7251	0.8512
Northern	0.7503	0.8522
North Midland	0.7653	0.8569
South Midland	0.7234	0.8512
Southern	0.7272	0.8455
New York City	0.7441	0.8650
Western	0.7298	0.8419
Army Brat	0.7277	0.8461

Table 3.4: Word Break Insertion Precision and Recall

words in English. Each such identified triple then causes the insertion of a pair of potential word breaks, one between each pair of phonemes in the triple. To resolve which of the potential word breaks are actual boundaries, we match the surrounding phonemes with all possible phonemes and pairs of phonemes which can begin or end words, and remove potential word breaks which would result in invalid word beginnings or endings.

We then perform an additional step whereby we use a pronunciation dictionary to find valid word matches for all contiguous subsequences of phonemes. For each such subsequence, we insert word breaks at the positions that are consistent across all the matches. For example, suppose the sequence [ɪnɔɪliɹæg] ("an oily rag") has the following three possible segmentations: [ɪn ɔɪli ɹæg] ("an oily rag"), [ɪn ɔɪl i ɹæg] ("an oil E. rag"), and [ɪn ɔ ɪl i ɹæg] ("an awe ill E. rag"). Since these choices have two words in common, we segment the phrase as [ɪn ɔɪli ɹæg].

The results of a 10-fold cross-validation experiment are given in Table 3.4. Overall, we achieve average precision and recall of 73% and 85%, respectively. Very recent results, however, by Blanchard, Heinz, and Golinkoff [16] and Hayes and Wilson [62] suggest that accuracy above 96% can be achieved using more advanced techniques than implemented here. Due to time and resource constraints, we make the simplifying assumption that word breaks can be correctly recognized. We revisit this assumption in Section 3.6.

3.5.11 Identifying Words via Phonetic Edit Distance (Stage ❹)

The final task is to convert the subsequences of phonemes into English words. To do so, we must identify words that best match the pronunciation dictated by the recovered phonemes. Towards this end, we design a novel metric of phonetic distance based on the difference in

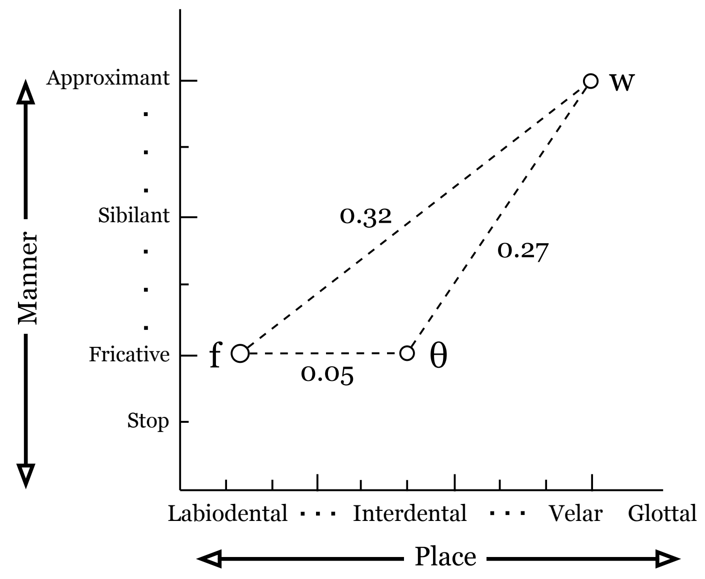


Figure 3.7: Illustration of distance between consonants [f], [θ], and [w].

articulatory features (i.e., the associated physiological interactions discussed in Section 3.2) between pairs of phonemes. Our approach has some similarities to ideas put forth by Oakes [110], which itself builds upon the work of Gildea and Jurasky [49] and Zobel and Dart [163, 164]. Oakes [110] proposes a phonetically-based alignment algorithm, though there is no notion of relative distance between various places or manners of articulation. In Zobel and Dart [164], the distances between phonemes are handcrafted, and their matching algorithm considers only the single most likely pronunciation.

In our approach, we define the distance between a vowel and a consonant as one unit, with a few exceptions: we assign a cost of 0 for converting an [i] to a [j] (or vice-versa) as well as for converting a [u] to a [w]. We do so because [w] and [j] are what are known as *semi-vowels*, and are essentially very short realizations of their corresponding vowels. Moreover, we assign a cost of 0 for [ɾ] (i.e., flap "r") and [t], as well as for [ɾ] and [d]. This is because [ɾ] is an allophone of [t] and [d]. Hence, we would like such minor phonetic alterations to have little effect on the distance between two pronunciations.

To measure the distance between two vowels or two consonants, we use three different articulatory features as axes and calculate the Euclidean distance (see Figure 3.7) between the points corresponding to the two phonemes (scaled to a maximum of one unit). For consonants

Word 1		Word 2			
Spoken	Written	Spoken	Written	Phonetic Distance	Primary Difference
bæt	bat	mæt	mat	0.0722	manner
bit	beat	bæt	bat	0.1042	height
did	deed	bid	bead	0.1050	place
bʌt	but	bɔt	bought	0.1250	rounding
bʌt	but	bæt	bat	0.1267	backness
bid	bead	bit	beat	0.5774	voicing
fɒðɝ	father	mʌðɝ	mother	0.7292	n/a
hʊkt	hooked	fɒnɪks	phonics	2.9573	n/a
hɛloʊ	hello	wɝld	world	3.1811	n/a

Table 3.5: Examples of our phonetic edit distance between pairs of example words. The last column lists the primary difference (in terms of articulatory processes).

these features are *voice*, *manner*, and *place* of articulation. For vowels they are *rounding*, *backness* and *height*. Thus we differentiate between substitution of a phonetically similar segment, such as replacement of [s] (as in "see") by [ʃ] (as in "she"), or of a completely different segment, such as of [s] (as in "seen") with [k] (as in "keen").

To compare two sequences of phonemes, we use the Levenshtein distance with insertions and deletions weighted at one unit and edits weighted according to their phonetic distance as defined above. Table 3.5 gives example word comparisons along with their primary differences (in terms of articulatory processes).

In order to determine the optimal values for the insertion and deletion weights for our phonetic edit distance metric, we performed a simple parameter space exploration. We hypothesized that the absolute insertion and deletion costs were less significant than the difference between them. As such we tuned based on two parameters, *base cost* and *offset*. Each insertion costs the base cost plus half the offset and each deletion costs the base cost minus half the offset. The effectiveness of each set of parameters is shown in Figure 3.8. Somewhat surprisingly, a base cost of 1.0 and offset of 0.0 (corresponding to insertion and deletion weights of 1.0) provided the highest average word accuracy.

To match a sequence of phonemes to an English word, we compute the phonetic distance between the sequence and each pronunciation in our dictionary in order to obtain a list of the closest pronunciations to the sequence. However, the existence of homophones means that, even

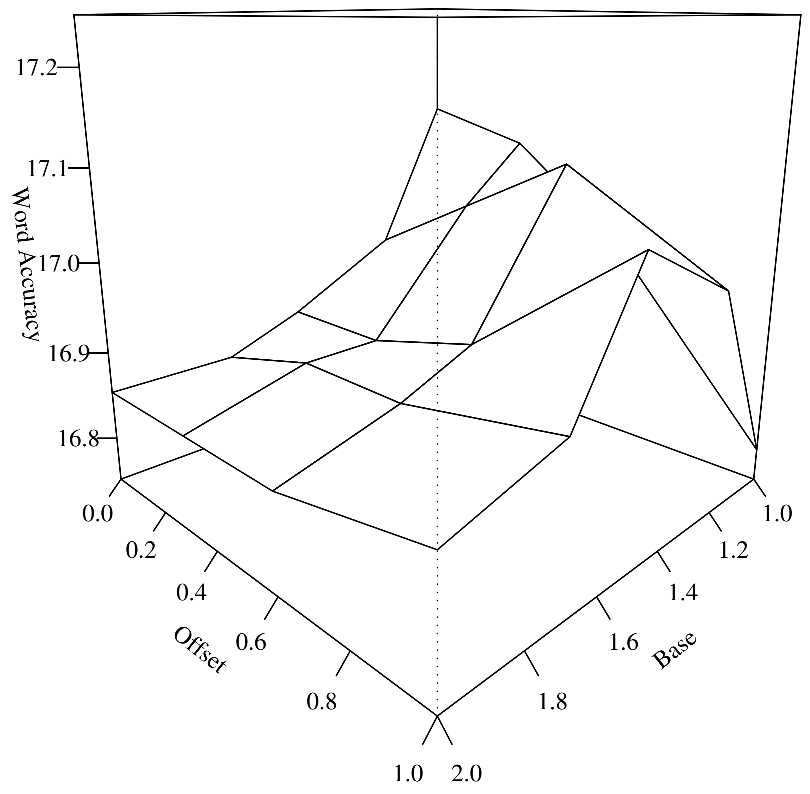


Figure 3.8: Parameter space exploration, in terms of average word accuracy, for our phonetic edit distance.

if the pronunciation is correct, we may have many choices for the word spoken. For example, “ate” and “eight” are indistinguishable phonetically: both are pronounced [eɪt].

In order to disambiguate between homophones, we incorporate a word and part-of-speech based language model to choose between the candidate words using contextual information from the sentence as a whole. Thus we can disambiguate between “ate” and “eight” by finding the most likely part of speech (e.g., noun, verb, pronoun, or adverb) for that position in the sentence. Using the SRILM toolkit, we train a trigram language model over both words and parts-of-speech on the well-known Brown corpus [47]. The part of speech tags used are those currently implemented in Natural Language Toolkit (NLTK). To improve the ability of the language model to disambiguate between candidate words, we assign each word a weight which estimates the conditional probability of the observed pronunciation given the candidate word.

To find these weights, we need a measure of how likely an observed pronunciation is given the phonetic distance to the actual pronunciation of the given word; therefore, we estimate the

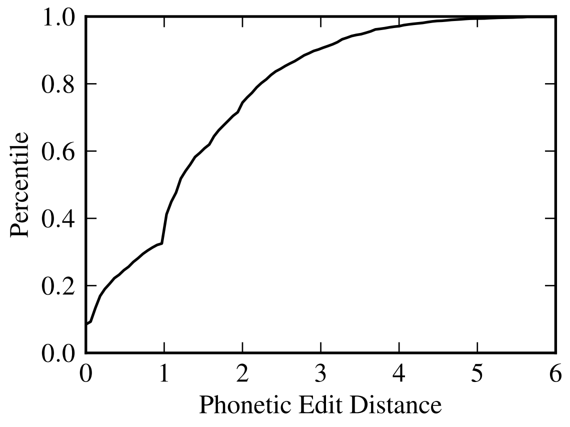


Figure 3.9: Empirical CDF of phonetic edit distance.

cumulative distribution function (CDF) over phonetic distances by deriving an empirical CDF (see Figure 3.9) from the distances of a large number of pronunciation pairs. We then transform the given distance between pronunciations into a probability estimate by evaluating the empirical CDF at that distance. For each pronunciation in the candidate list for an observed word, we weight the associated words with the probability estimate for that pronunciation.[8] Thus we have, for each word in an utterance, a list of candidate words with associated conditional probability estimates. Disambiguation is performed by finding the maximum likelihood sequence of words given the candidates, their probability estimates, and the language model.

At this point, the observant reader will have surely noted that the overall process is fundamentally inexact because, in the end, some sort of human judgement is required to evaluate the hypothesized output. That is, we need some way to measure the quality of our guesses, say, as assessed by a human judge who compares them to the actual transcript. Thankfully, the closely related problem of scoring machine translations has been extensively studied. In what follows, we discuss how we measure the accuracy of our guesses.

3.5.12 Measuring the Quality of Our Output

Since the early 1990s, much work has gone into finding appropriate metrics for scoring machine transcriptions from automatic speech recognition and transcription systems. In that context, the main task is to generate a literal transcription of every word that is spoken. The

[8]A word which is associated with multiple pronunciations is weighted according to the closest pronunciation, i.e., we take the maximum weight of all associated weights for the given word.

closer the machine transcription is to a human translation, the better it is. Early approaches for automatically measuring such performance simply relied on examining the proportion of word errors between the actual and transcribed conversations (i.e., the word error rate (WER)), but WER has been shown to be a poor indicator of the quality of a transcript since good performance in this context depends not only on the amount of errors, but also on the types of errors being made. For example, from the perspective of human interpretation, it often does not matter if the transcribed word is "governed" instead of "governing".

Hence, modern automatic scoring systems reward candidate text based on the transcription's *adequacy* (i.e., how well the meaning conveyed by the reference transcription is also conveyed by the evaluated text) and *fluency* (i.e., the lengths of contiguous subsequences of matching words). To date, many such scoring systems have been designed, with entire conferences and programs dedicated solely to this topic. For instance, NIST has coordinated evaluations under the Global Autonomous Language Exploitation (GALE) program since the mid-nineties. While the search for better metrics for translation evaluation remains an ongoing challenge, one widely accepted scoring system is the *METEOR Automatic Metric for Machine Translation* by Lavie and Denkowski [81]. METEOR was designed to produce quality scores at the sentence level which correlate well with those assigned by human judges. We evaluate the quality of our guesses using METEOR; for concreteness, we now review pertinent details of that scoring system.

Lavie and Denkowski's method evaluates a hypothesized transcription by comparison with a reference transcription. The two transcripts are compared by aligning first exact word matches, followed by stemmed word matches, and finally synonymous word matches. The alignment is performed by matching each unigram string in the reference transcription to at most one word in the hypothesis transcription. To compute the score from such an alignment, let m be the number of matched unigrams, h the number of unigrams in the hypothesis, and r the number of unigrams in the reference. The standard metrics of unigram precision ($P = m/h$) and recall ($R = m/r$) are then computed.

Next, the parameterized f-score, i.e., the harmonic mean of P and R given a relative weight (α) on precision, is computed:

$$F_{mean} = \frac{P * R}{\alpha * P + (1 - \alpha) * R}.$$

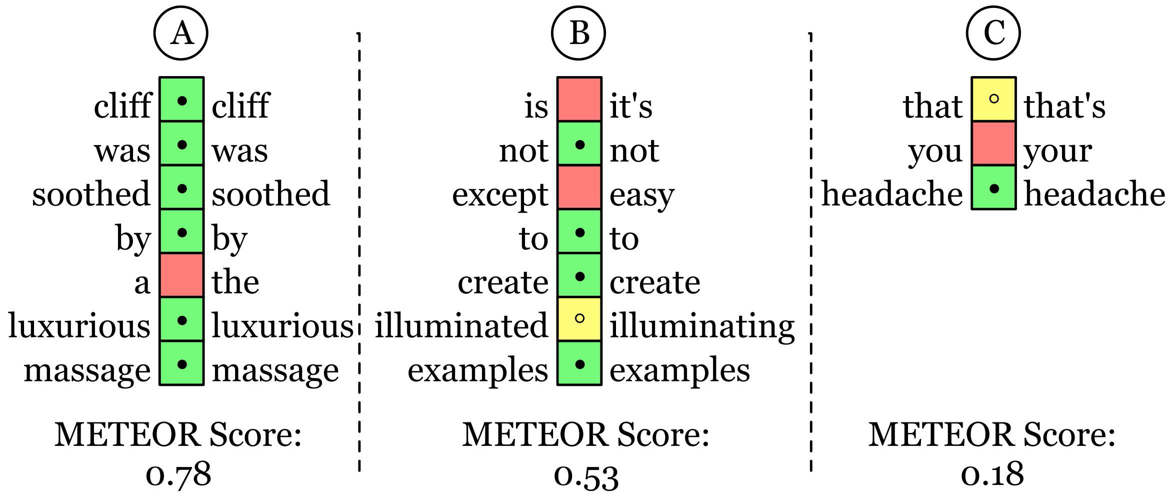
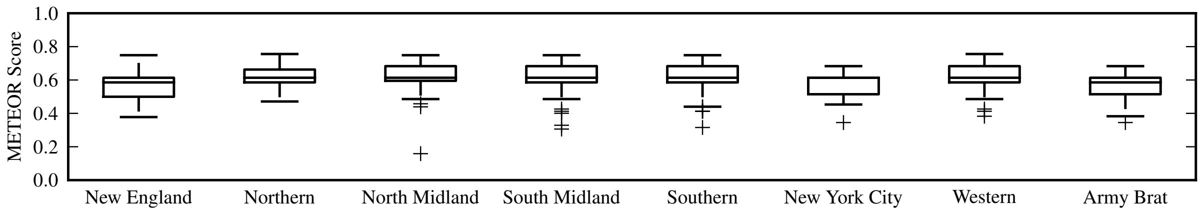


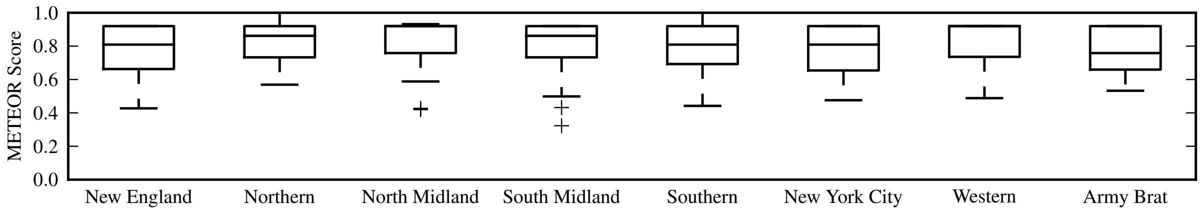
Figure 3.10: Example scoring of three hypothesized guesses. For each, the hypothesized guess is on the left, with the reference on the right. Filled circles represent exact matches. Hollow circles show matches based on stemming.

To penalize hypotheses which have relatively long sequences of incorrect words, Lavie and Denkowski count the number c of "chunk" sequences of matched unigrams which are adjacent, and in the correct order in the hypothesis. A *fragmentation penalty* is then computed as $P_{frag} = \gamma * (c/m)^{\beta}$, where γ and β are parameters determining the maximum penalty and relative impact of fragmentation, respectively. The final METEOR score is then calculated as $S_m = (1 - P_{frag}) * F_{mean}$ for each hypothesis.

Denkowski and Lavie [33] performed extensive analysis to determine appropriate values for the parameters α, β, and γ which optimize the correlation between METEOR score and human judgments. In our experiments, we use the parameter set that is optimized to correlate with the human-targeted translation edit rate (HTER) metric for human judgement on the GALE-P2 dataset [32]. We disable synonym matching as our system does no semantic analysis, and thus any such matches would be entirely coincidental. Some examples are shown in Figure 3.10. Notice that even a single error can result in scores below 0.8 (e.g., in part (a)). Moreover, in some cases, a low score does not necessarily imply that the translation would be judged as poor by a human (e.g., one can argue that the translation in part (c) is in fact quite decent). Finally, Lavie indicates that scores over 0.5 "generally reflect understandable translations" and that scores over 0.7 "generally reflect good and fluent translations" in the context of machine translation [80].



(a) SA1: "She had your dark suit in greasy wash water all year."



(b) SA2: "Don't ask me to carry an oily rag like that."

Figure 3.11: METEOR scores for all hypothesized transcripts of sentences SA1 and SA2 for each dialect in the TIMIT dataset.

3.6 Empirical Evaluation

In the analysis that follows, we explore both content-dependent and content-independent evaluations. In both cases, we assume a speaker-independent model wherein we have no access to recordings of speech by the individual(s) involved in the conversation. In the content-dependent case, we perform two experiments, each incorporating multiple different utterances of a particular sentence. We use TIMIT's SA1 and SA2 sentences for these experiments because each is spoken exactly once by each of the 630 speakers, providing a rare instance of sufficient examples for evaluation. In the content-independent case, we incorporate all TIMIT utterances.[9] Except where explicitly specified, all experiments are 10-fold cross-validation experiments and are performed independently on each dialect. As discussed in Section 3.5, for these experiments we assume that the segmentation of phonemes is correct to within human transcriber tolerances. However, the effects of this assumption are specifically examined in a small experiment described separately below.

Figure 3.11 shows the distributions of METEOR scores under each of the dialects for the two content-dependent experiments. For SA1, the results are fairly tightly grouped around a score of 0.6. The SA2 scores show significantly more variance; while some hypotheses in this case were

[9]We follow the standard practice in the speech recognition community and use the SA1 and SA2 sentences for training only.

Hypothesis	Score
She had year dark suit a greasy wash water all year.	0.67
She had a dark suit a greasy wash water all year.	0.67
She had a dark suit and greasy wash water all year.	0.67

(a) SA1: "She had your dark suit in greasy wash water all year."

Hypothesis	Score
Don't asked me to carry an oily rag like that.	0.98
Don't ask me to carry an oily rag like dark.	0.82
Don't asked me to carry an oily rag like dark.	0.80

(b) SA2: "Don't ask me to carry an oily rag like that."

Table 3.6: Top scoring hypotheses from the New England dialect.

Hypothesis	Reference Sentence	Score
Codes involves the displacement of aim.	Change involves the displacement of form.	0.57
The two artists instance attendants.	The two artists exchanged autographs.	0.49
Artificial intelligence is carry all.	Artificial intelligence is for real.	0.49
Bitter unreasoning dignity.	Bitter unreasoning jealousy.	0.47
Jar, he whispered.	Honey, he whispered.	0.47

Table 3.7: The five highest scoring hypotheses from the New England dialect under the content-independent model.

relatively poor, others attained perfect scores. To ease interpretation of the scores, we provide the three highest-scoring hypotheses for each sentence, along with their scores, in Table 3.6. In addition, recall that sentences with scores over 0.5 are generally considered understandable in the machine translation context; 91% of our SA1 reconstructions and 98% of our SA2 reconstructions exceed this mark.

The independent case, on the other hand, proves to be a more challenging test for our methodology. However, we are still able to reconstruct a number of sentences that are easily interpretable by humans. For instance, Table 3.7 shows the five highest-scoring hypotheses for this test on the New England dialect. In addition, a number of phrases within the sentences are exactly correct (e.g., "the two artists"). For completeness, we note that only 2.3% of our reconstructions score above 0.5. However, the average score for the top 10% (see Figure 3.12) is above 0.45. That said, we remind the reader that no reconstruction, even a partial one, should be possible; indeed, any cryptographic system that leaked as much information as shown here

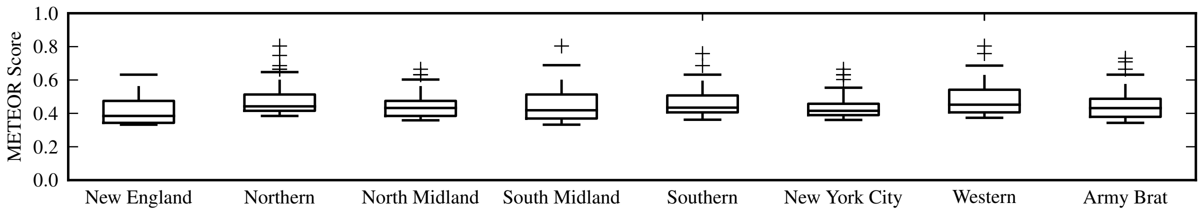


Figure 3.12: The top 10% of METEOR scores for hypothesized transcripts under the content-independent assumption.

would immediately be deemed insecure.

To mitigate any concern regarding our two previous simplifying assumptions, namely, the accurate segmentation of frame size sequences on phoneme boundaries and of (noisy) phoneme sequences on word boundaries, we perform one final experiment. We believe sufficient evidence has been given to show that we can accomplish these tasks in isolation; however, one possible critique stems from the potential effects, when these assumptions are lifted, on the efficacy of the methodology as a whole. Thus we remove these assumptions in a small, content-independent experiment comprised of the audio samples spoken by female speakers in the "Army Brat" dialect of the TIMIT corpus. The average score for the top 10%, in this case, is 0.19, with a high score of 0.27. We remind the reader that even such low scoring hypotheses can be interpretable (see Figure 3.10), and we stress that these results are preliminary and that there is much room for improvement—in particular, recently proposed techniques can be directly applied in our setting (see Section 3.5.10). Moreover, there are opportunities for extensions and optimizations at every stage of our approach, including, but not limited to, weighting the influence of the different classification and language models. In addition, other scoring systems for machine translation exist (e.g., NIST and BLEU), which may be appropriate in our context. We plan to explore these new techniques, optimizations and metrics in the future.

3.6.1 An Adversarial Point of View (Measuring Confidence)

Due to the difficult nature of our task (i.e., numerous factors influencing phonetic variation and the fact that we operate on *encrypted* data), an adversary is unlikely to be able to construct an accurate transcript of every sentence uttered during a conversation. Therefore, she must have some way to measure her confidence in the output generated, and only examine output with confidence greater than some threshold. To show this can be done, we define one such

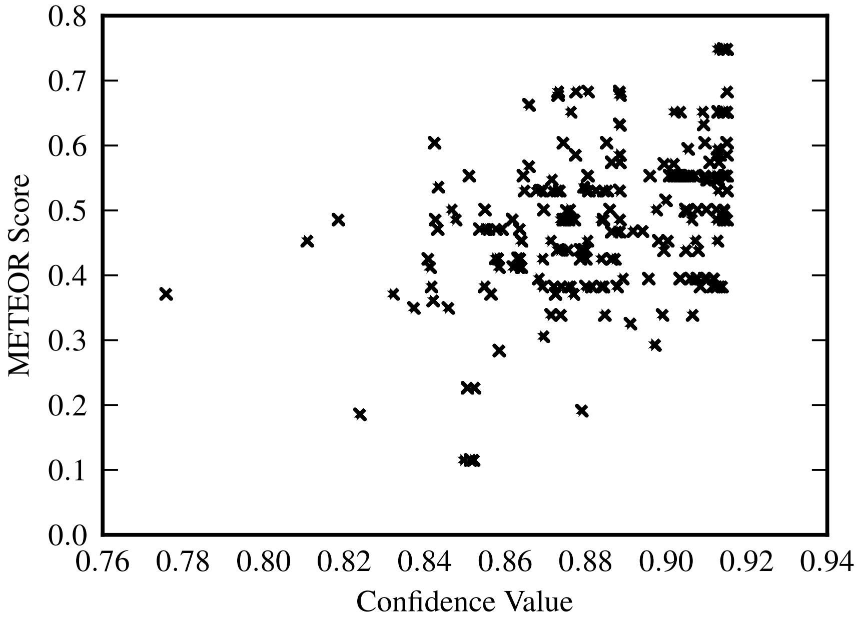


Figure 3.13: Scatter plot of METEOR scores against our confidence values (Pearson's r-value of 0.43).

confidence measure, based on our phonetic edit distance, which indicates the likelihood that a given transcript is approximately correct.

Our confidence measure is based on the notion that close pronunciation matches are more likely to be correct than distant matches. We use the mean of the probability estimates for each word in a given hypothesized transcript as our confidence value for that hypothesis. Analysis indicates that this confidence measure correlates (see Figure 3.13) with the maximum METEOR score obtained from the 10 best hypotheses output by the word-level language model (Stage ❹). This implies that, given a set of training data such as the TIMIT dataset, an adversary can determine an appropriate threshold for the calculated confidence values to suit her preference as to the balance between precision and recall in the hypothesized transcripts. The results in Figure 3.14provide one such analysis under the content-dependent model. We note that the threshold reduces the set of hypotheses to a subset with improved METEOR scores.

Unfortunately, the correlation of this particular confidence metric does not extend well to the content-independent model. However, we note that there are many other methods to measure confidence which an adversary could leverage, including those based on the posteriors output by

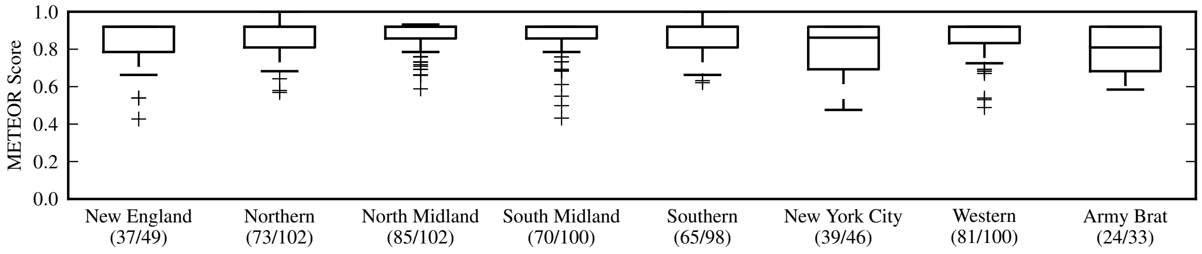


Figure 3.14: METEOR scores for hypothesized transcripts for sentence SA2 with confidence values above the threshold of .90 for each dialect in the TIMIT dataset. The number of transcripts with confidence values above the threshold compared to the total for each dialect is shown in parentheses.

the classification stage, the likelihoods given by the language models, and ex post facto analysis of the well-formedness (in terms of syntax, i.e., grammar) of the hypotheses. We hope to explore these strategies in the near future.

In closing, we note that one could also apply the notion of confidence values to interpreting the results at the word, rather than the sentence, level. In particular, we could filter our hypotheses at the word-level by only outputting those words for which we have high confidence. Preliminary results indicate that such “masking” may provide benefits to interpretation, for example, outputting “nonprofit ⋆ all ⋆ ⋆ raisers” instead of “nonprofit organizations all swiftly fairy raisers” as the hypothesis for “nonprofit organizations have frequent fund raisers”. We forego such analysis at this time since the METEOR metric does not allow for unknown words—an automated method of evaluating such hypotheses is necessary before we can make any claims.

3.6.2 Discussion & Mitigation

We note, like other work in this area (e.g., [5, 85, 92, 156, 157]), that we assume each packet contains a single frame. However, some recently designed codecs, such as Skype’s new codec (dubbed *SILK*), can vary the number of frames per packet based on network conditions. It therefore remains to be seen if the approach outlined herein can be adapted to that setting; exploring that, however, requires a substantial data collection effort that is beyond the scope of this work.

Further, our experiments assume that packets are observed in correct order and are not fragmented or combined, i.e., the adversary can observe packets at the level of the local network (e.g., between VoIP endpoint and hub or PBX) or can perform IP defragmentation or TCP stream reassembly. The inherently limited fidelity of the channel, however, suggests that our technique would be robust to reasonable noise in the form of packet reordering and fragmentation.

Lastly, a knee-jerk reaction to thwarting this and other aforementioned threats to VoIP is to simply use constant bit-rate codecs or block ciphers. However, VBR-encoded audio encrypted under a block cipher with a small block size is theoretically vulnerable to our attack. Packet sizes in that scenario still correlate with input signals, albeit at a reduced fidelity; thus relatively large block sizes are necessary to ensure privacy. For this reason, the use of constant bit-rate codecs is important to consider as an alternative to simple block ciphers for VoIP, since such codecs might improve call quality given a relatively large fixed packet size. Another alternative might even be to drop or pad packets [46, 68, 158], though, in that case, the effect on perceived call quality is unclear. We note, however, that VoIP providers have made no move to employ any such measures: Skype's `SILK`, for instance, is a VBR codec. Similarly, one of the leading proposals for 4G, the Long Term Evolution (LTE) Advanced standard, specifies a VBR codec for audio [1] and the use of SRTP to secure voice data channels.

3.7 Conclusion

In this chapter, we explore the ability of an adversary to reconstruct parts of encrypted VoIP conversations. Specifically, we propose an approach for outputting a hypothesized transcript of a conversation, based on segmenting the sequence of observed packets sizes into subsequences corresponding to the likely phonemes they encode. These phoneme sequences are then mapped to candidate words, after which we incorporate word and part-of-speech based language models to choose the best candidates using contextual information from the hypothesized sentence as a whole. Our results show that the quality of the recovered transcripts is far better in many cases than one would expect. While the generalized performance is not as strong as we would have liked, we believe the results still raise cause for concern: in particular, one would hope that such recovery would not be at all possible since VoIP audio is encrypted precisely to prevent such breaches of privacy. It is our belief that with advances in computational linguistics, reconstructions of the type presented here will only improve. Our hope is that this work stimulates discussion within the broader community on ways to design more secure, yet efficient, techniques for preserving the confidentiality of VoIP conversations.

In fact, since our original publication in May, 2011 [155], the Internet Engineering Task Force (IETF) released RFC 6562 (titled *Guidelines for the Use of Variable Bit Rate Audio with Secure RTP*), which recommends against the use of VBR codecs when conveying sensitive information [119].

3.8 Future Work

Skype Our evaluation in this work consists of reconstructing conversation transcripts using the open-source speech codec Speex. We feel that an evaluation using a second codec would corroborate the generality of our approach. In particular, Skype's speech codec (SILK) suggests itself due to it's popularity. However, SILK is significantly different from Speex; for our purposes, the primary difference is in the range of output packet sizes: SILK effectively can output packets of any size up to about 150 bytes, while Speex is restricted to 21 particular sizes in that range. This difference necessitates changes to our methodology; in particular, the phoneme classifiers must be replaced or adapted to consider the relative difference between packet sizes, rather than considering the possible packet sizes as an unordered set of symbols.

Conversational Speech Our analyses are based on a corpus of pre-recorded speech where the content of each utterance is known to the speaker in advance. An enhancement would be the application of our techniques to conversational speech samples, which more closely resemble real-world exchanges. Unfortunately, finding a dataset of conversational speech labeled at the granularity (i.e., at the phonetic level) required for our evaluations may present a problem.[10]

3.9 Broader Implications

In the broader context of inference from encrypted network traffic, this chapter demonstrates the potential magnitude of information which can be ascertained from a source as limited as a sequence of packet sizes. Furthermore, this work stands as an example (along with, e.g., the work of Saponas et al. [130]) of inference from streaming traffic—a class of traffic which comprised over 63% of Internet traffic in the fall of 2014 [51]. In Chapter 4, we explore another major class of network traffic (request-response) in the form of webpage visits (i.e., HTTP resource retrievals).

[10]One collection of conversational speech samples [122] contained numerous transcription errors; correction necessitates substantial effort on the part of a trained linguist.

The work presented in this chapter also demonstrates how the appropriate combination of methods from multiple disciplines can lead to results which surpass expectations. In particular, we stress that the development of the approach presented herein would not have been possible without significant contributions from the following disciplines (amongst others): linguistics (e.g., Section 3.5.10 and Section 3.5.11), natural language processing (Section 3.5.11), biological sequence analysis (Section 3.5.4), machine translation (Section 3.5.12), and machine learning (e.g., Section 3.5.1).

CHAPTER 4: PLAYING HIDE-AND-SEEK

4.1 Introduction

A network-security monitoring system such as an IDS, malware detector, content filter, or network forensic tool, is generally blind to the contents of encrypted traffic. This problem is amplified due to the multitude of options available for communicating into and out of an organization's network while hiding the contents of said communication from network monitoring systems. These options include HTTPS (often aided by browser extensions such as HTTPS Everywhere [67]), tunnels over SSH or similar protocols, and VPN or VPN-like mechanisms (e.g., Tor [34]). Currently, the only feasible way to make the contents of such traffic visible to a network monitoring system is through decryption (using, e.g., an escrowed key), throwing away any confidentiality guarantees.

Ideally, however, one should strike a balance between the visibility into network traffic needed by network monitoring and the confidentiality required by modern communication. As monitoring systems do not always require access to the complete content of network traffic, it might be sufficient to trade-off confidentiality for increased security by inferring any necessary information. One avenue for making such inferences is to leverage statistical characteristics of encrypted traffic. Previous work, concerned with the privacy implications of such inferences, has demonstrated that such statistical characteristics are sufficient to determine the web page being accessed in encrypted connections. Unfortunately, the vast majority of previous work only considers discriminating between pages from different domains, e.g., the *homepages of distinct web sites*, which is insufficient for network security and forensic applications.

The characteristics of the contemporary World Wide Web pose significant challenges to identifying web communication in encrypted traffic. One of these challenges is the sheer number of web pages: any practical approach to inference must be scalable. For instance, contrary to comments made in previous work [43], classification based solely on the size of pages does not scale—the range of likely page sizes is simply too small in comparison with the number of extant web pages to avoid collisions. Similarly, previous work has disregarded the question of scalability in terms of efficiency and run-time, in cases employing models which require hours to classify even small datasets. A second challenge is that HTTP connections are often persistent, with multiple HTTP requests and responses communicated over a single connection, obsoleting any techniques which assume that individual HTTP requests or responses are easy to isolate. In the same vein, many encryption technologies multiplex different application protocols over the same tunnel, further obscuring the picture. This challenge extends beyond just HTTP connections: for example, domain name system (DNS) traffic—which has seemingly been ignored by previous work—may be included in the traffic sent through a tunnel, peppering what might be seen as an orderly collection of TCP streams with occasional UDP packets.

Due in part to these difficulties, previous work has been restricted to classifying distinct pages from different domains. In contrast, the work presented in this paper examines a significantly more complex space of labels by including both the host name (domain) and the path components of requested pages in the set of labels to be modeled and predicted. In doing so, we are able to not only identify individual web pages for which we have training data (the extent of previous work), but also *provide information about previously unseen pages*. For example, having trained our system on data representing multiple domains and multiple pages for each domain, our model allows the system to predict the domain for new data representing a previously unseen page. Moreover, our model is not limited to domain names but also incorporates other components (e.g., the containing folder) of the page URL.

Our ability to infer additional information opens up new avenues for practical application of this line of research. In particular, we believe our model provides value to forensics and analytics platforms, which can use domain and path information to trace infection vectors, identify potentially dangerous connections, and generally gain insights into encrypted traffic which would otherwise be unavailable. Thus our efforts differ from previous work not only in methodology, but in motivation: our interest is in leveraging these techniques to provide security and forensics applications with a view into encrypted traffic, rather than further demonstrating the privacy threat such classification represents.

The problem we face in this work is to overcome the aforementioned challenges and enable network monitoring of encrypted web traffic without destroying confidentiality. Our approach is to employ a multi-*label* classification scheme, a paradigm shift away from the standard multi-*class* classification schemes explored in previous work, which allows our system to model and predict labels even for previously unseen pages. Our ultimate goal is enabling scalable, real-time forensic identification of web pages in encrypted traffic at the enterprise level. In designing our approach, we were guided in part by this goal of providing scalability; in particular, many of the classifiers explored in previous work (such as support vector machines, particularly those using edit-distance–based kernels) are not scalable in this context. Therefore, we chose to employ a classifier model which is novel to this domain: the random forest (RF). RFs, as ensembles of decision trees (DTs) provide us with inherent parallelization (and thus with scalability to real-world traffic volumes) and a proven ability to operate with a large number of classes. Additionally, the RF algorithm can be adapted to provide multi-*label* output at the classifier level; in contrast, most of the classifiers used in previous work are limited to multi-*class* classification, requiring *problem transformation* (Section 4.4.1) for multi-*label* classification.

Finally, we believe much of the prior work in this area suffers from a lack of scientific rigor. We are not the first to point out many of these issues (see, e.g., [120]), but we hope that enumerating these considerations in one place aids others in designing their evaluations. We stress that we do not mean to imply that all of the previous work in this area suffers from all of these issues, however, we were struck by the fact that some prior work does not even acknowledge the existence of many of these issues. In particular, we have identified the treatment of the following as deficient in least one prior publication each:

Heterogeneity of targeted pages Prior work has primarily considered scenarios where each page is distinct, overlooking the ramifications of multiple pages belonging to the same site. We address this issue head-on by collecting a new dataset with multiple URLs per domain name (Section 4.5.1).

Handling of DNS traffic The presence or absence of DNS traffic in an encrypted tunnel undoubtably affects observations; unfortunately, prior work has often failed to explicitly state whether or not such traffic is included in their datasets. We explicitly include DNS traffic when collecting our tunneled traffic dataset.

Problem definitions Prior work has emphasized the need for so-called open-world evaluations, wherein the set of possible pages is not known *a priori*; however, we found the definitions of the open-world model to be inconsistent across prior work. In Section 4.3.3, we present a taxonomy of different world models to aid in the disambiguation of this situation.

Suitability of accuracy measures Classification accuracy has consistently been used as the measure of performance for evaluations in prior work seemingly without regard to the suitability of this measure with regard to the specific task under evaluation. More recently, Bayesian detection rate (BDR) has supplanted accuracy (due in part to previous criticism [e.g., 120]) with the same disregard for context. We provide an overview of accuracy measures in Section 4.5.2, and discuss the scenarios under which each is appropriate.

Statistical validation of results The machine learning community has long recognized the need for validation (e.g., via cross-validation (CV)) of experimental results as a means of controlling the variance in classification error across experiments. At best, prior work has paid lip-service to this issue. In Section 4.4.5, we consider and reject the naïve use of CV in this context, instead designing an alternative scheme ("epoch validation") to overcome the problem of dependencies in time between samples.

Fairness of comparisons between algorithms Classification performance can vary significantly depending on the specific values used for the hyper-parameters on which most learning algorithms rely; one concern when comparing the results of different learning algorithms is that hyper-parameters values which result in good performance for a specific algorithm on one task and dataset may result in poor performance on another task or dataset. We address this issue through a disciplined search (Section 4.4.4) for optimal values for each hyper-parameter in each experiment we perform.

In summary, we make the following contributions in this work:

- A novel adaptation of multi-label classification to the problem of identifying encrypted web traffic, enabling inferences regarding, e.g., previously unseen URLs.

- The basic design for a scalable (both in terms of traffic volume, i.e., total number of web pages, and traffic diversity, i.e., number of distinct web pages) system for identifying web resources in encrypted traffic, independent of the encryption mechanism.
- A taxonomy of the various "world models", i.e., sets of assumptions defining specific tasks in experimental evaluations, used in prior work and the definition of a new, general class of world models designed to evaluate partial information scenarios.
- A novel form of statistical validation ("epoch validation") for scenarios in which a time dependency exists between samples.
- Empirical evidence indicating that the incorporation of multiple URLs per domain has a significant effect on classification performance.
- A rigorous and extensive evaluation covering multiple world models, classification paradigms, datasets (including two from prior work), learning algorithms, and feature sets.

4.2 Background & Related Work

Most germane to the work presented in this paper is the area of so-called *website fingerprinting* [13, 43, 63, 88, 113, 133, 152]. However, this designation is a misnomer; work in this category is instead generally concerned with *web page classification*. To avoid ambiguity in our own work, we define a *web page* as the collection of resources associated with a single URL and intended for simultaneous download and display. A *resource* is a single file, such as an image, text file, or HTML file, in isolation and located at a particular URL. A *web site* is the collection of web pages and resources which, linked together, form the content associated with a particular domain name. In general, we follow the World Wide Web Consortium (W3C)'s terminology [82]. The distinction between *web page* and *web site* in previous work has been unnecessary, however, as the collections of URLs on which these works base their experiments include only a single *web page* per *web site*. Due to the fact that web pages from the same web site are likely to share many characteristics, we hypothesize that discriminating between such pages is significantly more difficult than between web sites.

A surprisingly large body of work has examined the privacy threat posed by "website fingerprinting" when traffic is encrypted using various privacy enhancing technologys (PETs).

These technologies include tunneling proxies (e.g., those established by SSH's "dynamic forwarding") [13, 21, 43, 63, 88, 91], VPNs [63, 91], and anonymity networks (such as Tor) [21, 63, 65, 113, 133, 152]. Relatively little work has addressed HTTPS or similar measures (e.g., the non-proxying tunnels of SSH's "static forwarding") [24, 31, 65, 104, 140].

As far as we are able to ascertain from published sources, all prior work on classifying individual pages in tunneled connections has used datasets in which each web page is isolated from the others, i.e., no two pages belong to the same site. These isolated pages are unlikely to share many characteristics; we believe that this has lead to inflated estimates of one's ability to successfully identify web pages visited in encrypted tunnels. We pay particular attention to this issue in our analyis. Related, though unpublished, work has considered classifying pages in HTTPS connections (a model distinct from tunneling; see Section 4.3) where the domain is known [24, 31, 104].Conversely, Cai et al. [21] proposed an HMM-based system which identifies the domain corresponding to a *series* of linked page visits given a priori knowledge of the link structure of the domain's pages. Ultimately, however, *no prior work considers, as we do, the joint identification of page and domain nor the inference of partial information*.

Various countermeasures have been proposed to mitigate the privacy threat posed by previous work, including padding schemes [6, 43, 88, 90, 100, 104], traffic morphing [43, 158], buffering (e.g., Buffered Fixed-Length Obfuscator (BuFLO), Tamaraw), fragmentation (e.g., HTTPS with Obfuscation (HTTPOS)), randomized pipelining (Tor) [21, 121, 152] and randomized cover traffic [21, 93]. However, Dyer et al. [43] concluded that many of these defenses were ineffective. We return to the issue of defenses in Section 4.5.6.

In addition to identifying visited web sites, post-encryption features such as packet sizes and timing have been exploited to infer information about a wide variety of encrypted communications. In particular, these features have been leveraged to determine the application protocol underlying encrypted communications (e.g., [159]), search queries typed over HTTPS (e.g., [23]), keystrokes in interactive SSH sessions [136], and the content of encrypted VoIP calls (e.g., [155]), to name just a few. Relatively little prior work (save, e.g., [76]) has studied these features outside the context of assessing potential privacy leaks, however.

The remainder of this section summarizes the various learning algorithms and features explored in previous work.

4.2.1 Learning Algorithms

To provide a comparison with previous work, we evaluate various classifiers suggested by previous work in addition to the RF classifier which we suggest the use of herein. In addition, we use each classifier in conjunction with various feature sets proposed by previous work. Specifically, related work has considered the following classifiers:

- Jaccard index [88]
- naïve Bayes (NB) [43, 88]
- multinomial naïve Bayes (MNB) [63]
- support vector machine (SVM), by kernel:
  - radial basis function (RBF) [113]
  - optimal string alignment distance (OSAD) [21]
  - Damerau-Levenshtein distance [152]
- k-nearest neighbors (k-NN), by distance metric:
  - Levenshtein distance [91]
  - weighted Manhattan distance (weights learned using the algorithm given by Wang et al. [151])
- decision tree (DT) [73]

In this work, we propose the use of a random forest (RF) classifier (see Section 4.4.2).

4.2.2 Features

A wide range of potential features has been considered by previous work for classifiers which are formulated in the standard feature-vector representation. Features and sets of features considered in previous work include:

BurstLengthDistribution (BLD)	Distribution of burst lengths in each direction.
BurstSizeDistribution (BSD)	Distribution of burst sizes in each direction; referred to as *variable n-gram (VNG)* by Dyer et al. [43].

NumBytes	Total number of bytes transmitted in each direction; referred to as "bandwidth" in prior work.
PacketSizeDistribution (PSD)	Distribution of observed packet sizes in each direction.
Panchenko	Combines: NumBytes, BurstSizeDistribution, BurstLengthDistribution, PercentPacketsUpstream, NumPackets, NumUniquePackets, and HTMLResponseSize.
Time	Total connection time.
variable n-gram	Another name for BurstSizeDistribution.
VNG++	Combines: BurstSizeDistribution, Time, and NumBytes.

An alternative view of how to tackle this problem is to view the traces purely as sequences of packet sizes (or of paired packet size and inter-arrival time observations). Certain prior works therefore utilize learning algorithms which eschew the feature-vector representation in favor of edit-distance–based comparisons between these sequences. Specifically,: Lu, Chang, and Chan employ a k-nearest neighbors (k-NN) with the Levenshtein distance between sequences of inferred HTTP request and response sizes [91], Cai et al. use an support vector machine (SVM) with a kernel function based on the optimal string alignment distance (OSAD) [21], and Wang and Goldberg evaluate SVMs with kernel functions based on the OSAD, Damerau-Levenshtein distance, and fast Levenshtein-like distance [152]. While this is certainly a valid viewpoint, two issues spring to mind: 1) calculating edit-distance–based measures generally imposes runtime overhead of $O(st)$, where s and t are the lengths of the two sequences, for *each distance calculation required*; and 2) the feature-vector representation allows for the inclusion of summary statistics, such as NumBytes, which may provide enhanced discriminatory power. In fact, the former issue turns out to be damning in practice (see Section 4.5.3).

4.3 Assumptions and Threat Model

4.3.1 Networking Model

In general, we assume a model wherein a network-security monitor attempts to identify web pages visited by a client through an encrypted connection (or connections) which traverse part of

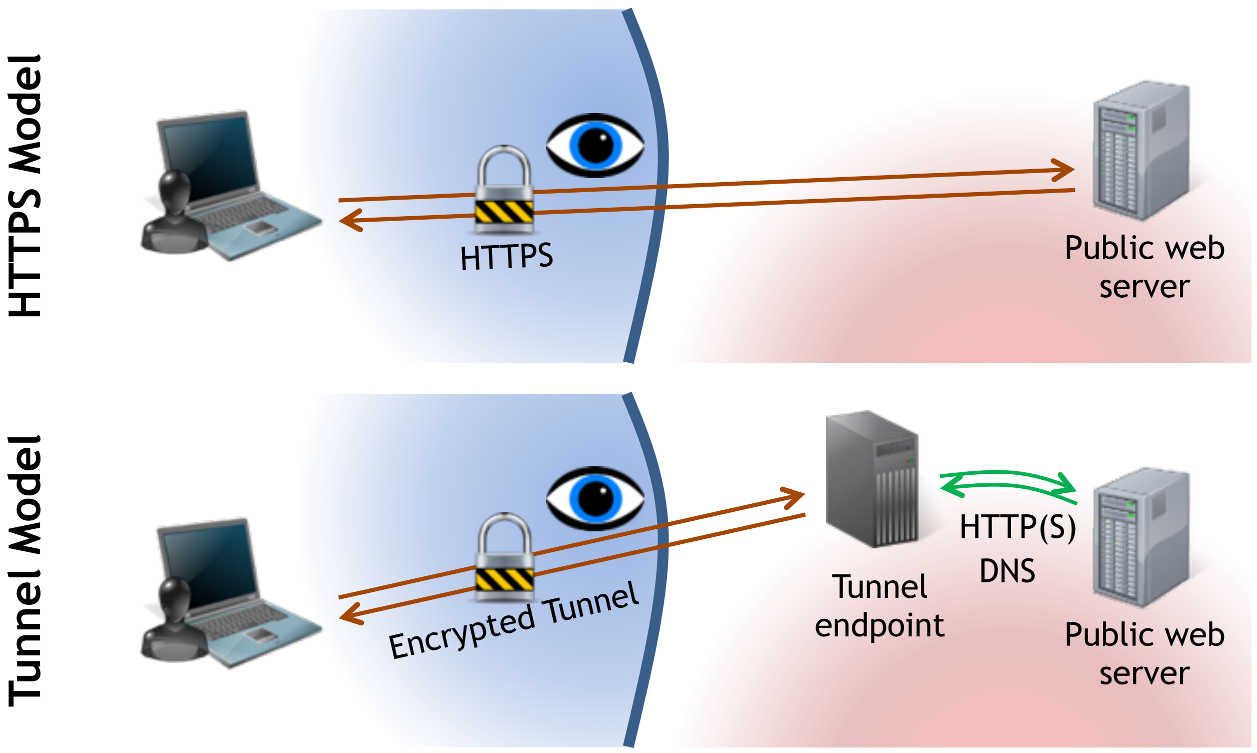


Figure 4.1: The two encryption models considered: HTTPS and web traffic tunneled over encrypted channels.

the monitor's network. The monitor is able to observe all encrypted packets at the network layer.

4.3.2 Encryption Model

In terms of the encryption scheme employed, there are two specific scenarios under which we believe our approach is applicable (Figure 4.1). The simpler of the two mimics HTTPS, and hence we will refer to it as the *HTTPS Model*; the other is that of a typical SSH tunnel, and hence we will refer to it as the *Tunnel Model*.

HTTPS model The HTTPS model is similar to that employed by Sun et al. in one of the earliest papers on the subject [140], and comprises the following assumptions governing the knowledge and abilities of the monitor: 1) IP addresses and DNS traffic are visible; 2) IP addresses can be mapped to DNS requests and hence domain names [11]; and 3) individual HTTPS connections are distinguishable. Under the HTTPS model, the labels of interest are primarily those derived from the path of the page requested. However, we note that an accurate model of a high-value website, e.g., bankofamerica.com, could be used to detect phishing websites attempting to masquerade as the original but served from rogue IPs or improper domains (e.g., typo-squatted domains like bankifamerica.com). By the same token, SSL hijacking attempts may be detectable, depending on the similarity between the actual website and the forgery.

Tunnel Model The tunnel model is similar to that explored by the majority of previous efforts in this area (e.g., [13, 21, 43, 88, 91]) and encompasses various tunneling technologies, including many VPN configurations and SSH's "dynamic" port forwarding (where an SSH client acts as a socket secure (SOCKS) proxy server, multiplexing connections over the encrypted channel). The tunnel model is significantly more restrictive than the HTTPS model, allowing the monitor access to much less information; in particular, all of the HTTP connections associated with accessing a web page are multiplexed through a single encrypted channel. We do make two simplifying assumptions: 1) the traffic through the tunnel is entirely DNS, HTTP, or tunnel control traffic (e.g., SOCKS control packets); and 2) the user does not simultaneously request multiple websites, i.e., we assume a reasonable delay between successive retrievals. We believe these assumptions to be sensible, particularly in the context of an SSH client used to proxy connections to a home or private network for unmonitored surfing or access to private resources.

DNS Traffic

An additional concern, which has not, to our knowledge, been explicitly discussed in prior work, is whether DNS traffic is tunneled or otherwise concealed from the monitor. We highlight this issue in order to clarify the model used herein and underscore one difference between our model and that used in previous work. In particular, we assume that DNS information is carried within encrypted connections and therefore inaccessible to the monitor.

DNS traffic may or may not be covered by the privacy-enhancing technologies discussed above; in many cases, configuring a PET to cover DNS requests is non-trivial and the difference is often undetectable without inspecting network traffic. This is a particular concern when using a proxy to secure connections (e.g., by tunneling over SSH). With the SOCKS protocol, for instance, DNS handling varies by version.[1] However, even if a suitable SOCKS version is used, DNS

[1]SOCKS v4 and earlier do not support UDP packets and require requests to specify a destination IP, forcing resolution to be done either on the client side or over TCP. SOCKS v4a added the ability to send requests by hostname, enabling DNS requests to take place on the server-side [134]. SOCKS v5 supports tunneling UDP connections, enabling DNS

requests may still be sent directly from the client. For instance, when Firefox is configured to use a SOCKS proxy server (as in the dynamic SSH tunnel scenario), an additional, hidden parameter must be set to force Firefox to use the proxy for DNS [108]. Even if a browser is configured to send DNS traffic through the proxy, extensions and plugins may ignore these settings [27, 143]. In particular, the DNS prefetching implemented in Chrome contacts DNS servers directly [27]. These nuances of proxy configuration are rarely disclosed openly to users. Similarly, VPN handling of DNS traffic varies by vendor, version and configuration.

Due to these issues of configuration, and the lack of relevant information in previous work, we believe that the impact of encrypted DNS on one's ability to identify web pages visited over encrypted connections is an open question. We attempt to remedy this oversight by explicitly including DNS traffic in our data collection and analysis.

4.3.3 World Models

One issue which is particularly pertinent to the ecological validity of any study in this area is the definition of the task to be accomplished, particularly with respect to the inputs (i.e., training data) and outputs (i.e., the set of possible labels for a test example) of the predictor. Previous work has focused on two distinct sets of assumptions governing the task of the predictor: the so-called *closed-world* and *open-world* models. This work defines a general class of *partial information* models which, as the name suggests, allow for the inference of partial information. We discuss each class of models in turn.

Closed-world Under the *closed-world* model, the task is to label a previously unseen trace as generated by one of N "web pages" (the classes), where the N classes are known *a priori* and the trace must belong to one of the known classes. For this model, the input is a set samples for each of the N classes; the output is single class label (identifying one of the N known pages). The closed-world model has been the subject of criticism [e.g., 120] on the grounds that having concrete knowledge of the entire set of pages likely to be visited is unrealistic.

Open-world An *open-world* model allows for the possibility that the new trace does not belong to any of the known classes: the task is to determine which, if any, of a set of N web pages

requests to be made through the proxy server [84].

of-interest generated a previously unseen trace (where the *of-interest* pages are known *a priori*). Under this model, the inputs are: 1) a set of samples for each of the N known classes; and 2) (optionally) a set of samples generated by none of the known classes. The output is either a single class label (identifying one of the N known pages) *or* $\perp$ (i.e., an indication that the test sample does not belong to one of the N known pages).

Binary Open-world A variant of the open-world model (seemingly confused, at times, for the open-world model as defined above) defines the task as a binary problem, rather than a multi-class problem: the task is to predict whether or not a trace was generated by one of N web pages (known *a priori*), without attempting to predict the specific page. This definition makes the task superficially easier; however, we argue that the application of standard classification techniques to this variant of the problem is unsound when $N > 1$ and the N pages are unrelated (as has been the case in previous work). Specifically, by grouping N unrelated classes into a single super-class, one implicitly assumes that the classes share some distinguishing characteristics.[2] However, there is no reason to believe *a priori* that the N pages *of-interest* share any characteristics which distinguish them from any page outside the set.

Partial Information An alternative class of models arises naturally from considering the problem as one of multi-label classification (as opposed to multi-class or binary classification). These models, which we refer to as *partial information* models, differ from the closed-world and open-world models in that the expected output of classifying a single trace is zero or more labels, each of which represents one piece of information about the trace. For instance, one such model considers two labels for each sample: the domain name of the page and the resource path, under that domain, at which the page is located. Then a fully correct labeling of a trace would consist of the domain and the resource path, while a partially correct labeling would consist of either the domain or the resource path. In our experiments, however, we explore a more complex space of possible labels: we treat each unique prefix of a URL in the data as a possible label (see Section 4.5.3). An important aspect of partial information models is that we define the expected

[2]This assumption may be reasonable when the grouping takes effect only in one of the two classes of a binary classification problem, as in anomaly detection: in such cases, the shared characteristic is that each of the grouped sub-classes is individually distinguishable from the other class. This is the situation when $N = 1$: the implicit assumption is that the single website *of-interest* is distinguishable from every other website.

output as *zero or more* labels: this definition makes the partial information models inherently "open-world" in that the set of possible page labels need not be fully defined *a priori* (in which case the classifier can simply return an empty set of labels to indicate that the trace does not represent any known labels).

4.4 Approach

The key hypotheses underlying our approach are: 1) assigning multiple labels (namely, domain and path components) to each connection enables finer-grained and more accurate classification than prior efforts; and 2) the performance aspects—both in terms of accuracy and scalability—and underlying theory of the classification model employed must meet the requirements of the problem at hand. We reiterate that, unlike previous work, we are not merely suggesting that such a model for disclosing information about web pages visited through encrypted channels is a threat to user privacy; instead, we are explicitly using this tactic to provide network operators with additional information which would otherwise be unavailable. It is our belief that such an approach provides a reasonable balance between preserving user privacy and informing network monitoring systems. In what follows, we describe our approach and the reasoning behind our design decisions.

4.4.1 Classification Scheme: Multi-label

A primary contribution of this work is the application of a multi-class classification scheme in the context of classifying encrypted web traffic. In a multi-label scheme, each example is assigned a *set* of labels and the objective is to predict the correct *set* of labels for each new example. Multi-*label* classification should not be confused with multi-*class* classification, the paradigm employed by previous work, where each example is assigned a *single* label from a set of class labels and the objective is to predict the *single* correct class label for each new example. While multi-label classification is not a new concept, the application to identifying encrypted web traffic is novel and enables far more interesting and in-depth analyses than in previous work.

One such type of analysis can be seen in cases where multiple websites use the same design template, and therefore have similar feature representations (e.g., packet size sequences). For the moment, we will use subdomains as our labels (we return to this scenario briefly in Section 4.5.6). Consider two departmental websites belonging to the same university (e.g., cs.example.edu and math.example.edu) which use the same template. Applying the traditional classification approach of previous work to this new scenario would imply the creation of a distinct class for each of the two subdomains. If the two classes are separable, then the traditional approach may well be able to accurately classify examples according to which subdomain was visited; if not separable, then many examples will be misclassified. In both cases, however, the traditional approach misses an important fact by treating these classes as distinct: both websites belong to example.edu. Our multi-label approach mitigates this concern: by including a third class corresponding to example.edu, we may be able to extract the additional information that would otherwise have been missed.

Furthermore, the multi-label approach enables us to garner information about pages for which we have no specific training data. In the university scenario, suppose an example is encountered for bio.example.edu. While we have no specific class for this subdomain—there were no examples of such in the training data—we may still be able to infer that the primary domain is example.edu. The same logic applies to the paths which, combined with domains, make up URLs.

4.4.2 Classifier Model: Random Forest

Previous work has primarily used either SVM or naïve Bayes (NB) classifiers. Unfortunately, both SVM and NB classifiers suffer from problems which limit their usefulness for tasks of the scale and complexity we face.

Support Vector Machine Classifiers

In particular, support vector machine (SVM) classifiers are notoriously difficult to parallelize, not well-suited to online formulations, and require numerous non-trivial design decisions (including choosing an appropriate kernel function along with its parameters) before use. Additionally, many of the SVM classifiers used in previous work employ an edit distance as the kernel function [21, 152]. In general, such an approach requires the computation of pair-wise distances between each pair of sequences, i.e., each trace. Each pairwise computation requires $O(mn)$ time (for m and n the lengths of the sequences). In our analyses, the lengths of the packet size sequences can extend into the thousands. The computational burden of N^2 such calculations (where N is the number of examples) is simply too much for any realistic scenario; in fact, the running time of these algorithms in our experiments has been excessive enough to prevent timely completion of even small-scale, offline experiments.

Prior work [113] has also explored using SVMs with the traditional feature-vector representation of data. A well-known concern in such cases, however, is that the feature values be scaled appropriately [66]. The primary reason for normalization of features is that SVMs are not invariant to scale in the input space, i.e., the objective function at the heart of an SVM classifier will more heavily weight features with higher mean values (thereby affecting the optimal decision boundary) [66, 137]. The question of feature normalization is of particular concern for many of the features proposed in previous work, since the scales on which the feature values are naturally distributed vary considerably. Consider the use of the total number of bytes transmitted as one feature and the number of packets of size 52 as another: by definition, these two features differ by a factor of 52 at the very least. However, to our knowledge, the issue of per-feature normalization has not been specifically addressed in prior work in this area. This oversight has serious practical implications, and so we return to the issue in the presentation of our results (Section 4.5.5).

Naïve Bayes Classifiers

While naïve Bayes (NB) classifiers are well suited both for parallelization and online tasks, these classifiers suffer from a different, though no less damning, set of drawbacks. In particular, NB classifiers are biased toward classes with more training examples [126]. Secondly, the NB paradigm achieves high classification accuracy and its parallelization ability through assuming that *the features used are independent*; in fact, NB classifiers are biased toward those classes which most violate this independence assumption. Since the features previously used in our context—packet size, direction, and timing—are hardly independent, this latter point may be a significant issue in practice.

Additionally, there is an informal argument to be made in favor of using random forests instead of NB classifiers, based on one of underlying assumptions made in the NB model. The NB classifier, like many classification algorithms, is based on modeling the conditional distribution $p(C|F_1, \ldots, F_n)$, where C is a random variable representing the classes and the F_i (for $0 < i < n$) are the feature variables. Classification is a matter of finding the class which maximizes the conditional given the values of the feature variables for a particular example. Calculating the conditional directly is generally computationally infeasible; NB is one of numerous methods for estimating this conditional which have been proposed. By applying Bayes's Theorem and assuming conditional independence between features (given the class), one can show that the conditional probability has the property $p(C|F_1, \ldots, F_n) \propto p(C) \prod_{i=1}^{n} p(F_i|C)$, which enables the conditional to be maximized in a computationally efficient way. Together, the application of Bayes' theorem and the assumption of conditional independence are the source of the name "naïve Bayes" ("naïve " referring to the independence assumption).

As discussed in Section 4.2, previous work has chosen scenarios in which the classes are relatively distinct (e.g., the front pages of google.com, microsoft.com, and yahoo.com). In this work, we have chosen to concentrate on scenarios in which examples from different classes may be very similar (e.g., Wikipedia pages for different topics). In the NB classifier, the individual conditionals $p(F_i|C)$ for each feature variable are generally assumed to have Gaussian distributions (for continuous data), as in Dyer et al. [43] and Liberatore and Levine [88], or multinomial distributions (for discrete data), as in Herrmann, Wendolsky, and Federrath [63]. When classes are distinct, this assumption poses no problem. However, when classes are similar (consider, e.g., different Wikipedia topics), these conditionals may easily overlap, potentially leading to a significant number of errors. The RF model does not assume any particular distribution for each feature variable, and therefore does not suffer from the same problem.

Random Forests

To mitigate these concerns, we turn to a different classification algorithm: random forests. A random forest (RF) is a collection, or *ensemble*, of decision trees. When classifying a new example, each decision tree (DT) assigns the example to a single class; the output of the RF is the *mode* of the outputs of the DTs [19].

Training a decision tree is an example of *recursive partitioning*. The tree starts with a root node to which is assigned the full set of training examples. This set is 'split' into two subsets based on a simple test (such as a threshold) of one particular feature. The feature used is chosen according to a criterion (commonly *information gain* [123] or *Gini impurity* [19]) which measures the ability of each feature to separate the node's examples into relatively homogeneous (in terms of label) subsets. Each subset forms the training set for a descendant node of the root note. This splitting is performed recursively for each node in the tree, stopping when the example sets for individual nodes are completely homogeneous, i.e., each leaf node has examples which all belong to a single class. The algorithm for training a decision tree for multi-label classification is the same, except that the splitting criterion used is averaged over the set of labels.

Training a RF classifier is a matter of training the individual decision trees, with a couple additional twists: 1) each tree is assigned a different training set; and 2) each node is assigned a randomly selected subset of features. If we have N examples in the training set, and given a value $m << M$, where M is the number of input variables (i.e., features), each decision tree is formed as follows:

- use a *bootstrap* sample from the full training set (i.e., sample N examples *with replacement*) for training
- for each node of the tree, choose m variables (features) at random from the set of input features; determine the split using this subset of features

The set of decision trees resulting from repeating this process is collectively the random forest.

Suitability for Our Approach

The RF algorithm thus has a number of nice features which are important in our context:

- inherent parallelizability: each (sub-)tree can be evaluated independently
- native multi-*class* classification (i.e., without binary classifier aggregation techniques like the *one-vs-all* or *one-vs-one* approaches generally used with, e.g., SVMs)
- adaptation to multi-*label* classification at the classifier level [2], i.e., without problem transformation (Section 4.4.1)
- availability of demonstrably effective *on-line* formulation, i.e., one which is designed to be iteratively updated over time [129]

4.4.3 Abstention and Thresholding

Given that our target deployment scenario is one in which we cannot model all possible labels, due in part to the infeasible amounts of training data and compute power which would be required to model the whole of the Internet, a method for identifying samples which do not match our models is important. With such a method, we can effectively refrain from classifying such an sample rather than simply assigning the most likely label (even if the most likely label is not significantly more likely than any other) as would otherwise be the case. Such a procedure, often referred to as abstaining classification, can significantly reduce error rates when classifying samples which do not match any of the labels in the training data.

Abstaining classification methods have previously been employed in the context of website fingerprinting. In particular, Juárez et al. [73] utilized a thresholding mechanism, similar to that we propose in Section 4.4.3, in order to combat the issue of false positives (FPs) and a low BDR in their work. Similarly, Wang et al. [151] modified the standard k-NN classifier methodology to abstain in cases where not all k neighbors agreed on which label to assign.

With respect to abstention, however, the multi-label paradigm again provides advantages over the multi-class paradigm employed by previous work. By it's very nature, a multi-label classifier is designed to assign to each sample not a single label but a (possibly empty) set of labels. In other words, a multi-label classifier can assign zero labels to a sample by definition—abstention is gained for free.

That said, for scenarios such as ours, in which many labels are over-represented in the training data relative to the test data (and the real world), some adjustment is required. To see why, consider a (multi-label) classification algorithm A which, when applied to a training data set $\mathcal{T}$, produces an estimator $\mathbb{P}_A[w \mid \mathbf{x};\ \mathcal{T}]$ of the conditional probability $\mathbb{P}[w \mid \mathbf{x}]$ that sample $\mathbf{x}$ has label w. The classifier then assigns w to $\mathbf{x}$ if and only if $\mathbb{P}_A[w \mid \mathbf{x};\ \mathcal{T}] > \frac{1}{2}$, i.e., if and only if the likelihood of $\mathbf{x}$ having label w is greater than random chance. However, for most common learning algorithms, if the label w is over-represented in $\mathcal{T}$, then $\mathbb{P}_A[w \mid \mathbf{x};\ \mathcal{T}]$ will over-estimate $\mathbb{P}[w \mid \mathbf{x}]$. One could correct for this bias by incorporating the prior probability of w either during classification or by adjusting the estimator post-classification. However, estimating priors for all the labels to be modeled is likely to be difficult and error prone. For this reason, we instead turn to a simple threshold optimization technique, which our experiments indicate is sufficient to achieve good classification performance (Section 4.5.5).

Post-Classification Thresholding

To reduce false positive rates, we employ a simple thresholding technique to discard any labels assigned with low likelihoods. A thresholding or like mechanism is particularly important in our experiments, where the classification models themselves are trained without reference to any samples with labels which are outside the interest set. We specifically exclude these labels from the training data in order to: 1) allow tuning of the false positive and false negative rates without relearning the models; and 2) reduce the amount of data which the classification algorithms must process.

This procedure is similar to the Classify-Verify [138] technique employed by Juárez et al. [73]. To determine a threshold for verification over n classes, Classify-Verify trains n multi-class classifiers (with $n - 1$ classes each). For each training example, two closed-world predictions are made: one using a classifier trained with the example's class (*in-set*) and one without (*not-in-set*). Two confusion matrices (one *in-set* and one *out-of-set*) are tabulated across all the training examples and these matrices combined in an average weighted by the expected proportion of *in-set* examples. The threshold is then found by optimizing with respect to the F_β score calculated from this aggregate confusion matrix. However, Classify-Verify is designed for small-scale multi-class classification. Training an additional classifier for each class (label) does is prohibitively expensive in scenarios, such as ours, with a large number of classes (labels).

Instead, we form a threshold training set by withholding from the training of the classifier a number of examples for each label to be modeled. To this set we add examples of other labels which are not represented in the training data. The trained classifier then assigns a set of likelihoods (one for each modeled label) to each example in the threshold training set. The optimal threshold is then the value which, when used as a cut-off to determine whether each particular label is predicted or withheld, maximizes some accuracy measure (e.g., F_β) across the examples in the threshold training set.

Formally, for a set of examples indexed by i, let $\mathbf{x}_i \in \boldsymbol{\mathcal{X}}$ denote the feature vector. Each classifier estimates the likelihood of a label w given an example (i.e., $\mathbb{P}[W = w \mid \mathbf{X} = \mathbf{x}_i]$). For each example i, we define the set of *predicted* labels (given some threshold value t) as:

$$\mathcal{Z}_i(t) = \{w \in \mathcal{W} \mid \mathbb{P}[W = w \mid \mathbf{X} = \mathbf{x}_i] \geq t\}$$

In other words, a label w is assigned to example i if and only if the likelihood of w given the feature values is greater than or equal to the threshold value t.

The threshold value t can be set manually or determined automatically by optimizing with respect to an accuracy measure or, equivalently, a loss function. For the latter approach, let $\mathcal{Y}_i \subseteq \mathcal{W}$ denote the set of *ground-truth* labels for example i and $\boldsymbol{\mathcal{Y}}$ the vector of all such sets for the given examples (i.e., $\boldsymbol{\mathcal{Y}} = \langle \mathcal{Y}_0, \mathcal{Y}_1, \ldots \rangle$). Similarly, let $\boldsymbol{\mathcal{Z}}(t)$ denote the vector of predicted label sets for all examples with respect to threshold t, i.e., $\boldsymbol{\mathcal{Z}}(t) = \langle \mathcal{Z}_0(t), \mathcal{Z}_1(t), \ldots \rangle$. Then the optimal threshold for classification with respect to a loss function $\lambda(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{Z}})$ (where *lower* values indicate less costly classification mistakes) is given by the expression:

$$\arg\min_t \lambda(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{Z}}(t))$$

Note that the same procedure can be used to determine an optimal threshold separately for each label, rather than globally.

Validation and Threshold Selection

We set the threshold value t as above using a separate validation step in which we inject examples of labels on which the classifier has not been trained. Specifically, for each fold in each experiment, a *threshold training set* is held-back from the normal training procedure. This set, denoted $\mathcal{R}_r$, consists of R_r examples each of N_h labels on which the classifier has not been trained. The classifier then assigns predictions to each example in both the original training set $\mathcal{T}_r$ and in the threshold training set $\mathcal{R}_r$. An optimal value of t is determined as described above using the union $\mathcal{T}_r \cup \mathcal{R}_r$.

4.4.4 Hyper-parameter Optimization

Nearly all supervised learning algorithms rely on hyper-parameters, such as the number of trees in a random forest, for which values must be set prior to training the final model. Often, the values used for these parameters are determined by a best guess at the optimal value or through ad-hoc exploration of the parameter space via experimentation. In this work, we employ a disciplined search methodology which combines the standard grid search with an extension known as randomized search [10]. By optimizing the set of hyper-parameter values within each experiment in a disciplined manner, we aim to provide a fair and reproducible comparison between algorithms.

A standard grid search takes as input a set of possible values for each hyper-parameter. An experimental trial is performed for each combination of hyper-parameter values; an accuracy value is calculated for each trial, and the highest accuracy value determines the “optimal” values for the hyper-parameters.

Unfortunately, this results in a number of trials equal to the product of the cardinalities of the sets of possible values for the hyper-parameters. Performing this many trials can easily be prohibitively expensive when a model has more than one or two hyper-parameters and/or a significant number of possible values for one or more hyper-parameters. For instance, for SVM learning with a radial basis function (RBF) kernel, a common suggestion for the regularization penalty parameter C and RBF kernel parameter γ is to use sets of logarithmically-spaced values such as $C \in \mathcal{C} = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma \in \Gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ (see, e.g., [66]). A grid search over just these two hyper-parameter value sets would require 110 trials ($|\Gamma| = 10$, $|\mathcal{C}| = 11$).

Another issue with the standard grid search is that the points in the hyper-parameter space defined by the combinations of possible values are spaced on a regular grid. For hyper-parameters defined over continuous ranges (or over discrete ranges of significant cardinality), using a small set of fixed values may easily miss optima located between points on the grid [10, Figure 1]. Furthermore, a user-chosen subset of hyper-parameter values for a continuous range can result in effectively arbitrary values, or, at best, values based on rules-of-thumb (consider the example of the SVM parameters, above).

Randomized search is designed to address these and other shortcomings of grid search and manual optimization. In a randomized search, the combination of values for a particular trial is created by drawing values independently at random according to distributions defined *a priori* for each hyper-parameter. Thus, the number of trials in the search can also be specified *a priori*, allowing the computation budget to be specified in terms of the number of trials rather than by the product of the sets of parameter values.

4.4.5 Epoch Validation

Robust methods for estimating the generalization error of a classifier, such as k-fold cross-validation, are important for preventing over-fitting and ensuring the repeatability and ecological validity of experiments. This is particularly true when the amount of data available, relative to the size and complexity of the problem space, is limited. We observe, however, that our problem scenario imposes a linear time dependency between samples, i.e., the content located at a particular URL may change over time. In modeling a URL, we must limit our training data to that collected at a point in time prior to the data used to test our models.

Unfortunately, such a linear dependency makes standard k-fold cross-validation inappropriate. In k-fold cross-validation, k trials are performed, each using a different subset of the data as the test set. Specifically, the full data set $\mathcal{T}$ is partitioned into k disjoint subsets ("folds") $\mathcal{T}_i$ (for $i \in \{1, \ldots, k\}$). For trial i, fold $\mathcal{T}_i$ is used as the test data while the remainder of the data ($\mathcal{T} - \mathcal{T}_i$) is used for the training set. However, since each fold is used to test models trained on data from every other fold, experiments formed using this method necessarily involve testing using data collected before at least some of the data used to train the models was collected.

Our solution, which we call *epoch validation*, adapts a technique known as *forecast evaluation with a rolling origin* from time-series analysis for use in estimating the generalization error of classifiers when such a dependency exists between samples. First, we partition the set of examples $\mathcal{T}$ into p disjoint subsets ("epochs") $\mathcal{T}_i$ such that a sample in $\mathcal{T}_i$ precedes a sample in $\mathcal{T}_j$ if and only if $i < j$ (for all $i, j \in \{1, \ldots, p\}$). Then we perform k trials in which, as with k-fold cross-validation, each trial uses a different subset as the test set. Specifically, for trial i, we use epoch $\mathcal{T}_{p-k+i}$ as the test set while the preceding $p - k$ epochs (i.e., $\bigcup_{j=i}^{p-k+i} \mathcal{T}_{j-1}$) form the training set.

4.5 Evaluation

4.5.1 Data Collection

As mentioned in Section 4.2, previous work in this area has focused on datasets limited to distinct pages from different domains, e.g., the homepages of distinct websites. In this work, we address this shortcoming by collecting a new dataset.

URLs

In order to collect a reasonably realistic set of traces which includes both multiple domains as well as multiple distinct URLs per domain, we extracted URLs from HTTP traffic generated by real users. As part of a different study, and under long-standing memorandums of understanding with a major university, HTTP headers were collected from a university department's network. Collection began at 2pm on a Tuesday and ended at 9pm the following day (a period of 31 hours); the traffic examined consisted of 141 million packets across more than 5 million flows.

From these records, we extracted the requested resource path, hostname, and returned content-type for each HTTP request-response pair, resulting in 2.2 million distinct URLs. After removing URLs corresponding to content-types other than `text/html`, 65,000 unique URLs remained across 3,768 domains. For reasons as follows, we then applied various filters to the list of URLs. First, to avoid excessive bias toward locally-oriented content, we applied Alexa's top 1-million domain list [4] as a white-list. Second, in order to abide by our university's acceptable use policy (which forbids knowingly downloading adult content) we applied a well-known blacklist [15]. Additional filters removed URLs with extensions corresponding to non-HTML content (e.g., `.exe` and `.tar.gz`) and URLs with invalid characters, non-ASCII encodings, non-standard delimiter schemes, or path components in excess of 255 characters. After applying these filters, 30,243 unique URLs across 2,225 domains remained, representing 215,000 distinct site-visits.

To reduce this list to a manageable size, we restricted our attention to those subdomains with at least 10 unique URLs remaining in the set, then selected the top 10 most-visited URLs for the top subdomain of each remaining domain, leaving 900 URLs across 90 distinct domains. This final list of URLs contains many unsurprising domains, including wikipedia.org, google.com, craigslist.org, and various ad networks, as well as domains oriented toward computer scientists (e.g., stackoverflow.com, sourceforge.net) and those of broader interest (e.g., espn.go.com, nbcnews.com, instagram.com).

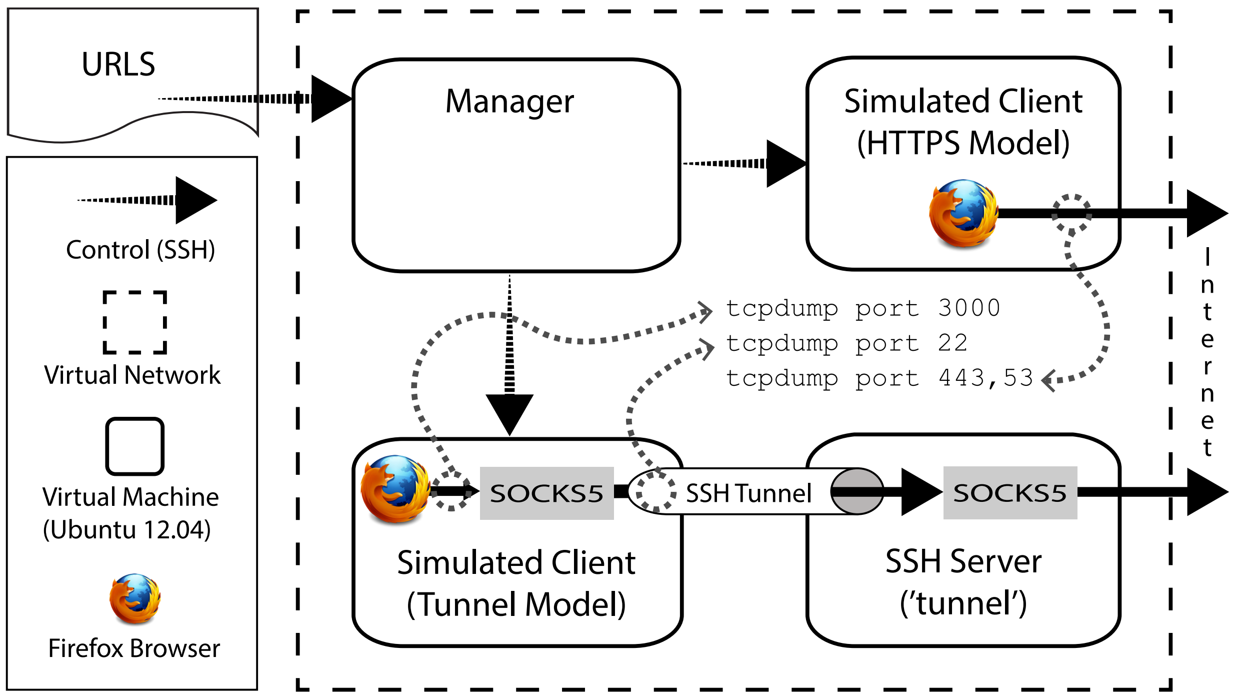


Figure 4.2: Collection process overview.

Scripted Retrievals

To gather the necessary data for our experiments, we performed a series of scripted retrievals. These retrievals were orchestrated by a collection of shell and Python scripts. At a high level, the primary script reads in the list of URLs, retrieves each URL in sequence, and repeats. To 'retrieve' each URL, the script first sets up an encrypted tunnel, if necessary. After waiting for the encrypted connection to initialize, the script begins recording network traffic. A browser is then launched and programmatically directed to the given URL. Once the page has loaded (or a timeout of 30s is reached), the browser takes a screenshot, then closes. Next the recording process(es) are stopped; finally, the encrypted tunnel is torn down. This process is then repeated for the next URL. A detailed description of the process is given in Section B.1.

To perform the retrievals, we created a small, isolated network (Figure 4.2) consisting of four virtual machines: a 'manager', two 'clients', and a 'server'. The 'manager' machine hosted a shared drive and provided the sole entry point into the network. One client virtual machine (VM) performed scripted retrievals over HTTPS connections and the other over 'dynamic' SSH tunnels (i.e., in the tunnel model). The 'server' VM acted as the SSH server for the second client.

All of the VMs were configured with Ubuntu 12.04LTS Server and ran on the same physical machine (with 12 cores and 64GB RAM, running VMware ESX 5.0).

Each retrieval iteration of 900 URLs took approximately 3.5 hours to complete. Between 2/13 and 2/14, we attempted to collect 20 instances of each URL. 884 URLs were successfully retrieved at least once; 785 were successfully retrieved all 20 times.

4.5.2 Evaluation Criteria

Multi-class Metrics

Previous work has almost exclusively used accuracy (defined as the number of correctly labeled examples divided by the total number of examples) as the metric for classification performance. In their experiments, the classes are balanced (i.e., each website is represented by the same number of traces), and so accuracy is a perfectly valid metric. For the sake of comparison, we follow the same philosophy in our multi-class experiments.

However, we remind the reader that when classes are unbalanced (i.e., the number of instances for each class varies greatly), the use of more sophisticated metrics becomes necessary. This would be the case in practice, since relatively few websites are responsible for the majority of network traffic. More importantly for the present work, the multi-label scheme produces an unbalanced distribution of labels (domains are represented many more times than individual pages, as each instance of a particular page is also an instance of its domain), necessitating the use of metrics designed to account for this non-uniformity. We describe the relevant multi-class metrics now both to prepare the reader for the multi-label metrics discussed next and to stress the importance of such real-world considerations.

To illustrate the need for different metrics when classes are unbalanced, consider a binary classification task where the majority class consists of 8 testing examples and the minority only 2 testing examples. Then a classifier which simply outputs the majority label for every example would have an accuracy of 80%, obviously misrepresenting the ability of the classifier. If we consider the same example on a per-class basis, the accuracy for the majority class would be 100% but the accuracy for the minority class would be 0%, highlighting the discrepancy.

Therefore, when class sizes are unbalanced, we turn to per-class *precision* and *recall*. Precision measures how often examples labeled as the given class actually belong to that class (disregarding the prevalence of the class), while recall measures how often examples from the given class are correctly labeled (disregarding how often examples from other classes are mislabeled as the given class). Precision for a class can be calculated as $tp/(tp+fp)$, where *tp* refers to the number of *true positives* (those examples from the class which were labeled correctly) and *fp* refers to *false positives* (those examples incorrectly labeled as belonging to the class). Recall is calculated as $tp/(tp+fn)$, where *fn* refers to *false negatives* (those examples belonging to the class which are incorrectly labeled). In the example above, the majority class would have a precision of 80% and a recall of 100%, while recall for the minority class would be 0% (precision for the minority class is undefined, as the classifier never labels examples as belonging to the minority class). The harmonic mean of precision and recall, known as the F-score, is often used to provide a single-value summary.

A second consideration, and one that is particularly important in practice, is the effects of false positives (e.g., labeling a user visit as to a censored website when the site is, in fact, benign) and false negatives (e.g., labeling a user visit to a malicious website as benign). Perry [120]'s recent critique of previous work emphasized this issue, which is exacerbated by, e.g., the massive discrepancy between the number of visits to benign websites versus that to censored or malicious websites. When the ratio between these quantities is high, as it is in practice, the accuracy metric is insufficient, as it gives little indication as to the rate of incidence of false-positives (and false-negatives). In the example above, the classifier achieves an accuracy rate of 80% despite never accurately labeling an example belonging to the minority class, further demonstrating the need for more sophisticated metrics.

Multi-label Metrics

Standard evaluation measures, such as precision and recall or accuracy, must be reconsidered in the multi-label case: since each prediction is a set of labels, we must account for partially correct predictions where the intersection of the predicted label set and the true label set is non-empty but the two sets are not equal. This is demonstrated in Table 4.1, which gives values for the multi-label metrics we consider when applied to a simple example. Multi-label classification accuracy measures can generally be grouped into two types, *example-based* and *label-based* [96].

Example-based measures examine the differences between predicted and true label sets averaged on a per-*example* basis. Similar considerations as for the multi-class case apply, however, necessitating multi-label analogs to the metrics discussed above. We define the precision for a single example as the ratio of the number of correct labels assigned by the classifier to the total number of labels assigned by the classifier (equivalent to the standard definition of precision as $tp/(tp+fp)$). Similarly, we define recall for an example as the ratio of the number of correct labels assigned by the classifier to the true number of labels for the example (equivalent to $tp/(tp+fn)$). These metrics are calculated for each example (Table 4.1) then averaged over the total number of examples to provide the per-example precision and recall.

As in the multi-class case, per-*label* metrics provide a potentially more accurate view of classification performance when the distribution of labels is non-uniform (per-example metrics are biased toward labels with more instances). To mitigate this bias, we turn to multi-label analogs of measures (i.e., precision, recall, and F-score) used in the multi-class case. For a given label, we define a *true positive* as an example whose true and predicted label sets both include the label. A *false positive* is an example whose true label set does not included the label but whose predicted label set does include the label; similarly, a *false negative* is an example whose true label set does include the label but whose predicted label set does not. Then the standard definitions in terms of these counts are used to define the precision and recall for a given label.

In terms of aggregate per-label measures, we consider both $micro-$ and $macro-$ averages of the measures described above. For macro-averaging, the value of a measure is calculated for each label and averaged over the total number of labels. For micro-averaging, the counts (i.e., of true positives, etc.) are summed over all labels and the totals used to calculate the value of each measure. The former gives equal weight to each label in the aggregation, whereas the latter weights the more numerous labels more heavily.

4.5.3 Experimental Setup

World Models Our first experiments are under the closed-world model for comparison with prior work; the remaining experiments are performed under a partial information model (Section 4.3.3). In the latter case, each experiment defines a set of *of-interest* URLs (or domains) and a set of *of-no-interest* URLs (or domains), as in the open-world model. We define the space of

Label	TP	FP	FN	Precision	Recall
foo	1			$1/1 = 1.0$	$1/1 = 1.0$
bar			1		$0/1 = 0.0$
baz			1		$0/1 = 0.0$
qux		1		$0/1 = 0.0$	
Per-example	1	1	2	$1/2 = 0.5$	$1/3 = 0.\overline{3}$
Per-label (micro)				$1/2 = 0.5$	$1/3 = 0.\overline{3}$
Per-label (macro)				$1/4 = 0.25$	$1/4 = 0.25$

Table 4.1: Values for the multi-label metrics applied to a single instance for which the true label set is {foo, bar, baz} and the predicted label set is {foo, qux}.

labels as comprising each unique prefix of a URL in the *of-interest* labels; the task for these experiments is to predict zero or more of these labels for each test sample. The inputs for the classifiers are labeled training data for each *of-interest* URL, and, where mentioned, labeled training data for a set of *of-no-interest* URLs. Classification performance is then evaluated using a separate set of examples of both *of-interest* and *of-no-interest* URLs.

Datasets We use three datasets in our analysis. The first two, due to Liberatore and Levine [88] and Herrmann, Wendolsky, and Federrath [63], were generously made available by their owners. The third is our newly-collected dataset (see Section 4.5.1).

We refer to the dataset of Liberatore and Levine [88] as LL. The domain names used for LL were derived from DNS traffic within the authors' department. The most-visited 2,000 names were selected for retrieval, and the authors collected a total of 240,000 samples over two months. The authors used Firefox 1.5 with OpenSSH 4.2p1 to perform the retrievals. As SOCKS v5 support was added in OpenSSH 3.7[3], it is possible that this data includes DNS traffic; however, the authors make no mention of the settings change required to force Firefox 1.5 to use the proxy for DNS traffic (see Section 4.3.2).

Our second dataset from previous work (denoted HWF) consists of the OpenSSH traces collected by Herrmann, Wendolsky, and Federrath [63]. The 775 domain names used by Herrmann, Wendolsky, and Federrath were derived from requests to a "medium-range proxy server used by approximately 50 schools". Over a period of two months, the authors collected

[3]http://www.openssh.com/txt/release-3.7

approximately 132,000 instances of OpenSSH retrievals (along with retrievals under other PETs, including OpenVPN and Tor); the authors performed the retrievals using Firefox 2.0 and "adjusted the proxy settings to relay all web traffic". However, no specifics regarding DNS traffic are included in [63]; furthermore, [63] refers to the collected traces as "HTTP traffic".

Our first set of experiments (Section 4.5.5) makes use of the `LL` and `HWF` datasets. Our remaining experiments are performed using only our own datasets as the datasets from previous work include only a single URL per domain.

Learning Algorithms In the following sections, we report the results of our experiments with the following classifiers (see Section 4.2.1): naïve Bayes (NB), multinomial naïve Bayes (MNB), support vector machine (SVM) with the radial basis function (RBF) kernel (denoted simply SVM), k-nearest neighbors (k-NN) with the Manhattan distance (denoted k-NN), k-nearest neighbors (k-NN) with Wang et al. [151]'s weight learning (denoted k-NN-WL), decision tree (DT), and random forest (RF). Note that our use of the RF classifier is novel, and that results for the k-NN classifier without weight learning have not, to our knowledge, been reported in previous work.In initial experiments, we also evaluated the Jaccard index classifier; we do not report those results due to uniformly poor performance (in terms of classification accuracy). Similarly, we do not report results of our preliminary experiments with the edit-distance–based classifiers (see Section 4.2.1) due to exceedingly long runtimes (orders of magnitude longer than the other classifiers) and poor performance.

Multi-label Classification As previously discussed (Section 4.4.1), most learning algorithms do not natively support the notion of multi-label classification. Specifically, the SVM, NB, and multinomial naïve Bayes (MNB) classifiers do not natively support multi-label classification. We employ a problem transform approach known as the binary relevance (BR) method to adapt these classifiers to the multi-label paradigm. In this approach, a multi-label classifier is built by training a binary classifier for each label which attempts to determine whether that label is present or absent for a given sample. Each binary classifier is trained using all the samples in the training data with the given label as the positive class and the remainder of the samples as the negative class. We implement this approach using `scikit-learn`'s one-vs-rest (OvR) meta-classifier.

Specifically, the implementations of the RF, DT, and k-NN algorithms in the machine learning toolkit we employ (`scikit-learn`) support multi-label classification [118] As our implementation of Wang et al. [151]'s k-nearest neighbors with weight learning (k-NN-WL) classifier is based upon `scikit-learn`'s k-NN implementation, the k-NN-WL classifier also natively supports multi-label.

Feature Sets We evaluate each classifier with two of the more effective feature sets from prior work (see Section 4.2.2): `PacketSizeDistribution` and `VNG++`.

Validation and Data Selection Each experiment (i.e., data point) consists of at least 5 trials with training and test data selected according to our epoch validation methodology. Where noted, an experiment may instead consist of n sub-experiments, each with a different set of URLs or domains and each consisting of 5 trials with training and test data selected according to our epoch validation methodology (for a total of $5n$ trials). Without the thresholding mechanism, 16 samples (per class, for each trial) are used for training and 4 for evaluation.

The selection methodology becomes complicated when thresholding and classes *of-no-interest* are included (see Figure 4.3). First, the data for each trial is selected using (stratified) epoch validation. Second, the classes are separated into *of-interest* and *of-no-interest* sets. Third, the data for each class is split into training, threshold training, and testing sets in linear-time order. Finally, the set of *of-no-interest* classes is split into (threshold) training and testing subsets. For experiments with thresholding, or when a comparison is made to an experiment with thresholding, 12 samples (per class, for each trial) are used for training, 4 for the threshold search, and 4 for evaluation.

Hyper-parameter Optimization Since classification performance can vary wildly with hyper-parameter values, each of our experiments incorporates a search for "optimal" values for the hyper-parameters of the classifiers in use. Our search methodology (Section 4.4.4) uses a predefined set of values (or distribution) for each hyper-parameter (given in Table 4.2).

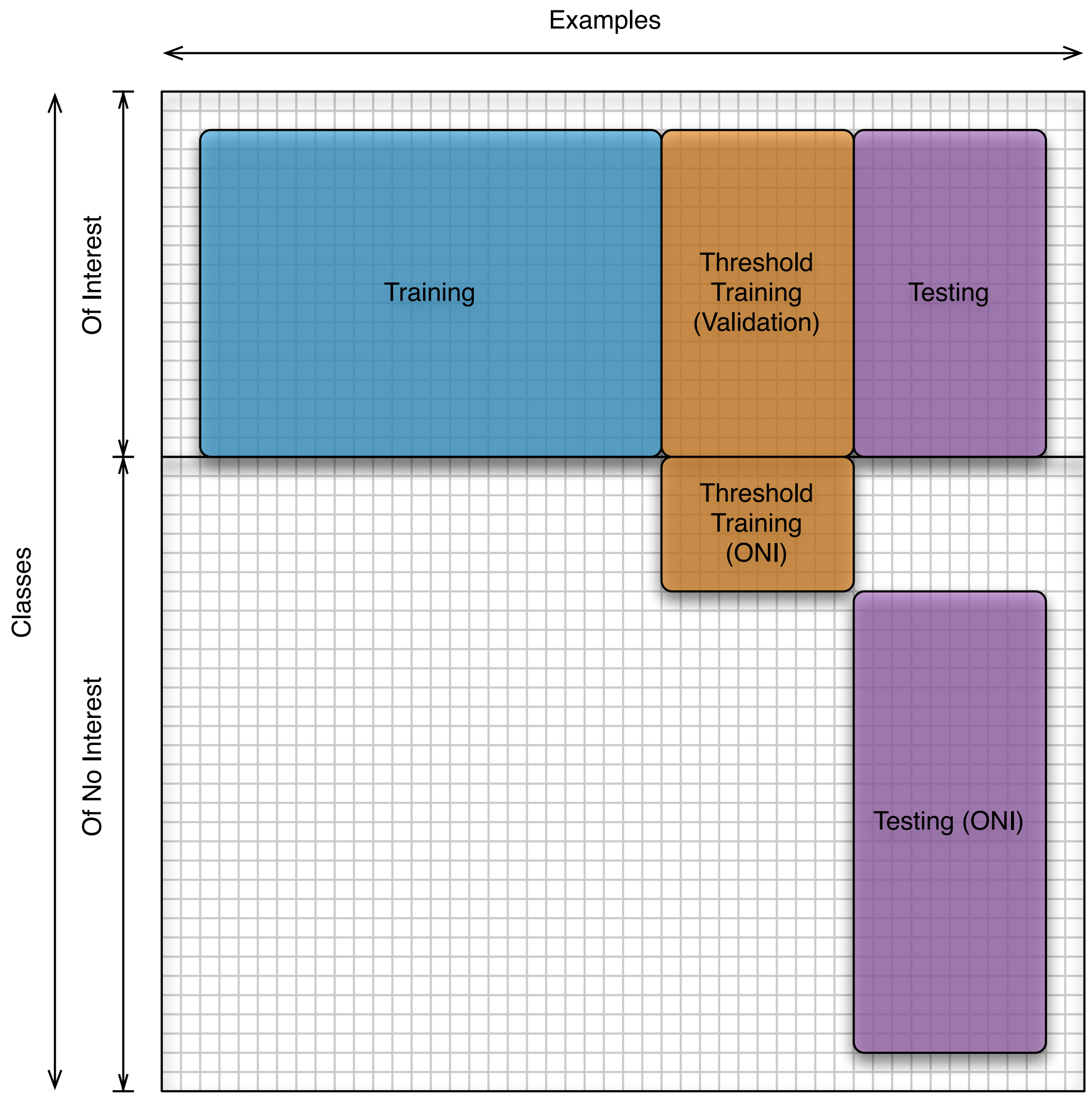


Figure 4.3: Data selection for experiments with thresholding and classes *of-no-interest*

Model(s)	Hyper-parameter	Value(s)	Description
RF, DT	criterion	Gini impurity, information gain	Criterion used in determining whether and how to split the samples at each node.
RF, DT	max_features	$F, \sqrt{F}, \log_2 F$	Number of features examined at each node (where F is the number of features given in the data).
RF, DT	min_samples_split	$2, 4$	Minimum number of samples required in prospective parent to split a node.
RF, DT	min_samples_leaf	$1, 2, 4$	Minimum number of samples required in each prospective leaf to split a node.
DT	splitter	best, random	Strategy used to choose the split at each node.
RF	num_trees	$8, 16, 32$	Number of trees in the forest.
MNB	α	$0.00, 0.01, 0.10, 1.00$	Smoothing parameter.
MNB	learn_priors	true, false	Whether or not to learn and use per-class priors.
k-NN, k-NN-WL	k	$1, 3, 5$	Number of nearest neighbors to use in classification.
k-NN-WL	k_{reco}	k	Number of nearest neighbors to use in weight learning.
k-NN-WL	num_rounds	$1, 5$	Number of rounds of weight learning to perform (one round corresponds to iterating through each sample once; note that this definition differs from that of Wang et al. [151]).
k-NN	metric	manhattan, euclidean	Distance metric.
SVM (RBF)	C	$\sim$ Exponential$(\lambda = 0.01)$	Regularization penalty parameter.
SVM (RBF)	γ	$\sim$ Exponential$(\lambda = 10)$	RBF kernel parameter.
SVM (RBF)	class_weight	weighted, unweighted	Whether or not to weight the regularization parameter C differently for each class (with weights inversely proportional to class frequency).

Table 4.2: Hyper-parameters for the models used, along with the values explored for each, in our experiments.

4.5.4 Implementation

We are indebted to Dyer et al. [43] for releasing the analysis framework used in their work, upon which we based our own. Our additions include new classifiers (the RF, k-NN, k-NN-WL, DT and edit-distance–based SVMs), the multi-label scheme, epoch validation, hyper-parameter optimization, and many other improvements totaling some 10,000 new lines of Python and Bash code (not including the retrieval scripts themselves). Our implementation is available upon request.

4.5.5 Results

In the analysis that follows, the term "classes" refers either to the set of distinct URLs or the set of distinct domains for each trial—which should be clear from the context. Error bars, when present, indicate the standard deviation of the presented metric's value across all trials for the given experiment.

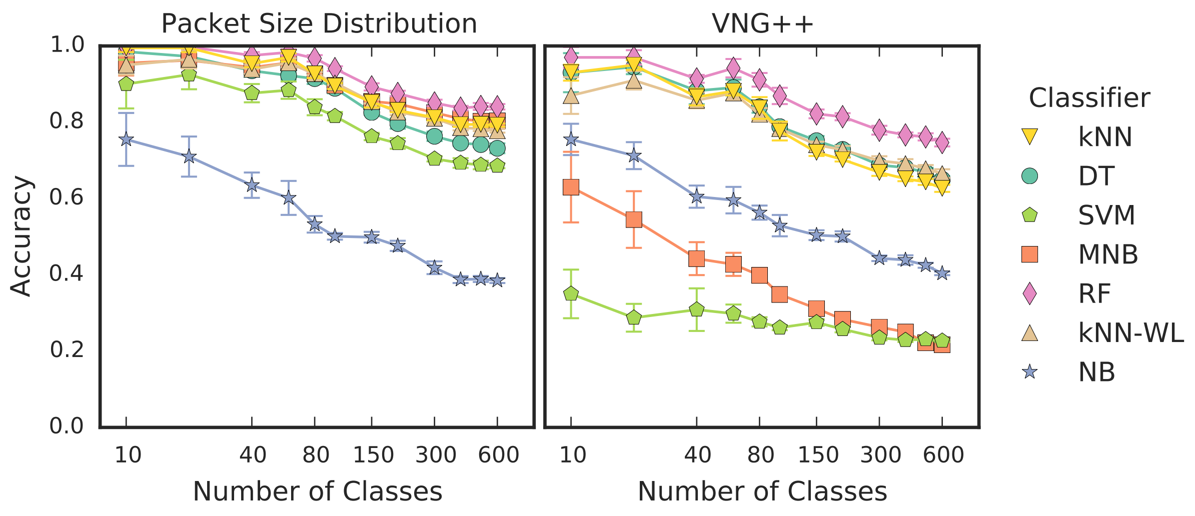
For our multi-label experiments, the term "classes" refers to the set of distinct URLs used for each trial. Note that the terms "class" and "label" are *not* interchangeable in this context: the former refers to a URL, while the latter refers to a URL *prefix*.

Multi-class Comparison with Previous Work

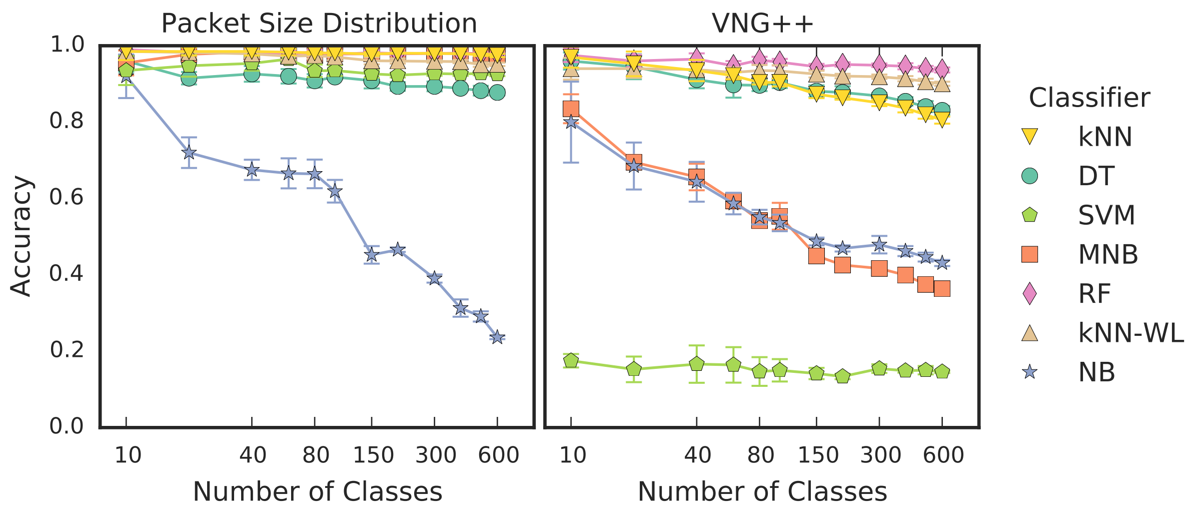
Our first two experiments are equivalent to many performed in previous work (e.g., those of Dyer et al. [43]), and provide an "apples-to-apples" comparison. In particular: the paradigm is multi-class classification, the datasets are `LL` and `HWF`, and the metric used is accuracy.

Our results (Figure 4.4) are generally on-par with those observed in previous work [43]. In particular, we achieve accuracies higher than 90% for most of the classifiers tested on the `HWF` dataset and lower accuracy rates for the `LL` dataset.

One difference, however, does stand out: results for the NB classifier with both `PacketSizeDistribution` and `VNG++` on both the `LL` and `HWF` datasets are substantially lower than those reported by Dyer et al. [43] (and Liberatore and Levine [88], to a lesser degree). We see two possible explanations for this discrepancy: 1) our implementation uses the NB classifier from `scikit-learn`, while Dyer et al. use that from Weka [57]; and 2) Dyer et al. perform no cross-validation in their experiments (i.e., their single trial might have an abnormally high



(a) LL



(b) HWF

Figure 4.4: Multi-class classification applied to two datasets from previous work, LL (top) and HWF (bottom), each with the PacketSizeDistribution feature set (left) and the VNG++ feature set (right). Reported values are for the accuracy metric.

accuracy rate compared to the mean across multiple trials).

Another point of interest is that the accuracy for the SVM classifier is substantially lower with the VNG++ feature set than with the PacketSizeDistribution feature set. We view this as aptly demonstrating the feature scaling issue often encountered with SVM classifiers (previously discussed in Section 4.4.2), as the VNG++ feature set includes BurstSizeDistribution, Time, and NumBytes, each of which is on a different scale than the others.

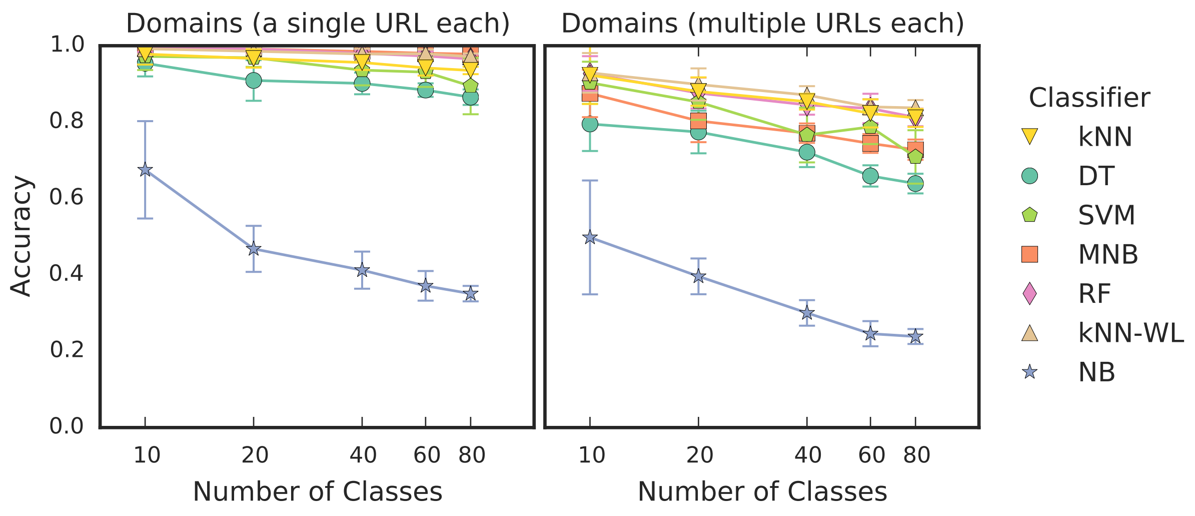
A similar problem occurs when attempting to apply the MNB classifier in combination with the VNG++ feature set (relative to PacketSizeDistribution). In particular, the assumption with the MNB classifier is that element i of a feature vector is the probability or frequency with which a particular event i occurs for the represented sample. Each class is then modeled as a *multinomial* distribution. For instance, in the *bag-of-words* model for document classification, each element is the number of occurrences of a particular word in the document being represented. The situation with the PacketSizeDistribution feature set is analogous to the *bag-of-words* model (using a unique packet size instead of a unique word for each position in the feature vector). However, the Time and NumBytes elements of the VNG++ feature set invalidate this assumption, and the difference in the achieved accuracy rates reflects this.

Multiple URLs per Domain Name

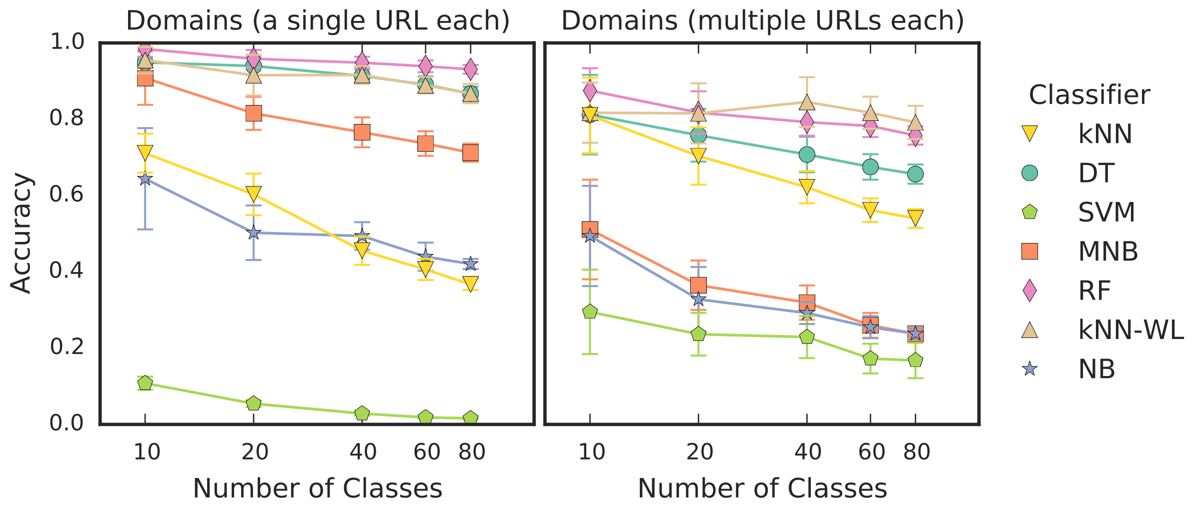
To illustrate the effects of including multiple pages per site in the data to be analyzed, we performed a pair of multi-class experiments using our newly collected SSH model dataset (described in Section 4.5.1) where the goal is to identify the domain to which a particular retrieval corresponds. The difference between the two experiments is that for the first experiment, each domain is represented (in both training and testing) using different instances of the same URL; in the second experiment, each domain is represented using 10 distinct URLs. The former experiment is the same scenario as has been used in the majority of previous work.

For these experiments, each reported value is the mean of 25 trials: each experiment (itself consisting of 5 trials configured via epoch validation) is repeated 5 times with a different randomly-selected subset of domains from our SSH dataset. We use this procedure to eliminate any potential bias due to the particular subset of domains selected in any one experiment.

Our results (Figure 4.5 and Figure 4.6) clearly demonstrate the problem with considering only



(a) PacketSizeDistribution



(b) VNG++

Figure 4.5: Multi-class classification applied to data which includes only a single URL per domain (left) vs data including multiple URLs per domain (right). Each is shown with the PacketSizeDistribution feature set (top) and the VNG++ feature set (bottom). Reported values are for the accuracy metric. See also Figure 4.6.

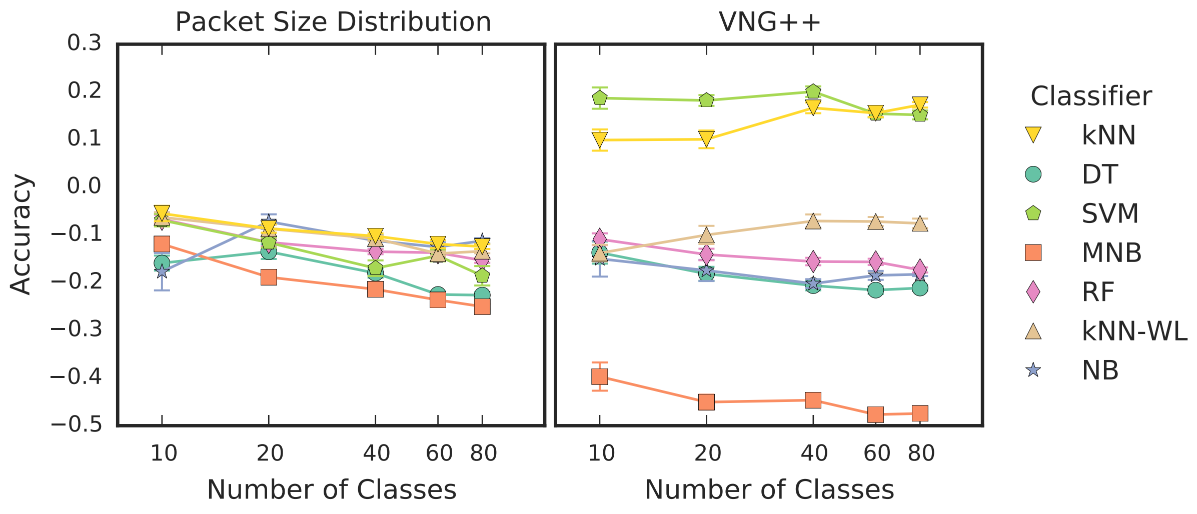
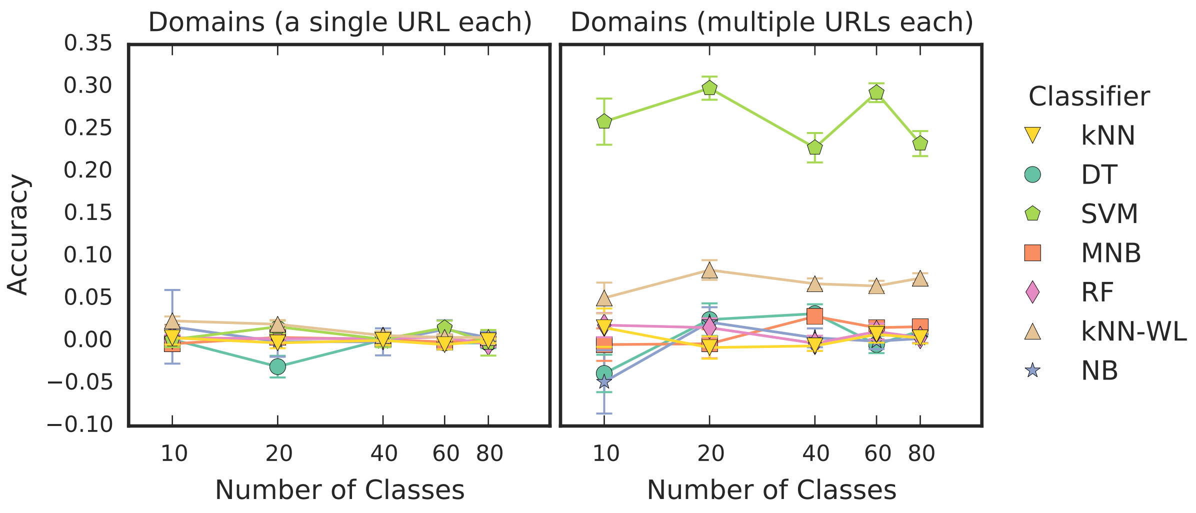


Figure 4.6: Multi-class classification: difference in accuracy between multiple URLs per domain vs a single URL per domain. See also Figure 4.5.

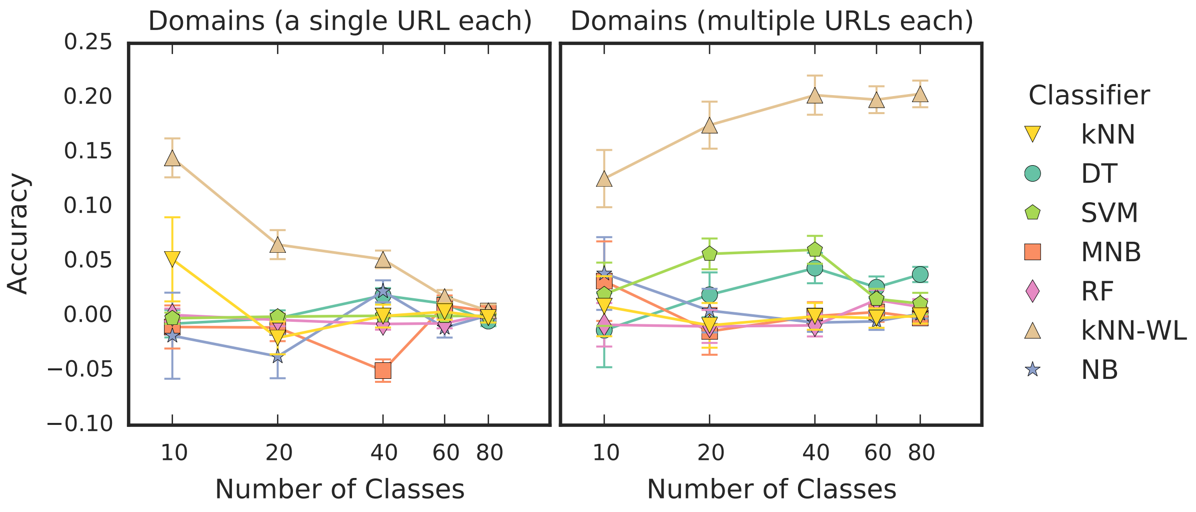
a single URL per domain. For both the PacketSizeDistribution and VNG++ feature sets, all but two classifiers (k-NN and SVM) suffered a decrease in accuracy of between 5% and 50% across the board. The highest-scoring classifier with each feature set (the RF in both cases) suffered decreases in accuracy of between 8% and 16.5%. The k-NN and SVM classifiers with the PacketSizeDistribution feature set suffered decreases similar to the others; however, with the VNG++ feature set, these two classifiers actually achieve higher accuracies in the multiple-URLs-per-domain scenario. We hypothesize that the cause of these anomalies is that the additional NumBytes feature in the VNG++ feature set provides extra discriminatory power between domains compared to between individual URLs; if this is the case, however, it remains unclear why the other classifiers suffered similar decreases as with the PacketSizeDistribution feature set.

Hyper-parameter Optimization We take this opportunity to comment on the effectiveness of automated hyper-parameter optimization via randomized search. The results from the optimized experiments (Figure 4.5) are, on the whole, quite similar to those from the unoptimized experiments (not shown); however, some classifiers show clear benefits from the optimization procedure (Figure 4.7).

In particular, the SVM classifier with the PacketSizeDistribution feature set and the k-NN-WL classifier with the VNG++ feature set show marked improvements in the



(a) PacketSizeDistribution



(b) VNG++

Figure 4.7: Difference in accuracy due to hyper-parameter optimization via randomized search (multi-class; SSH model).

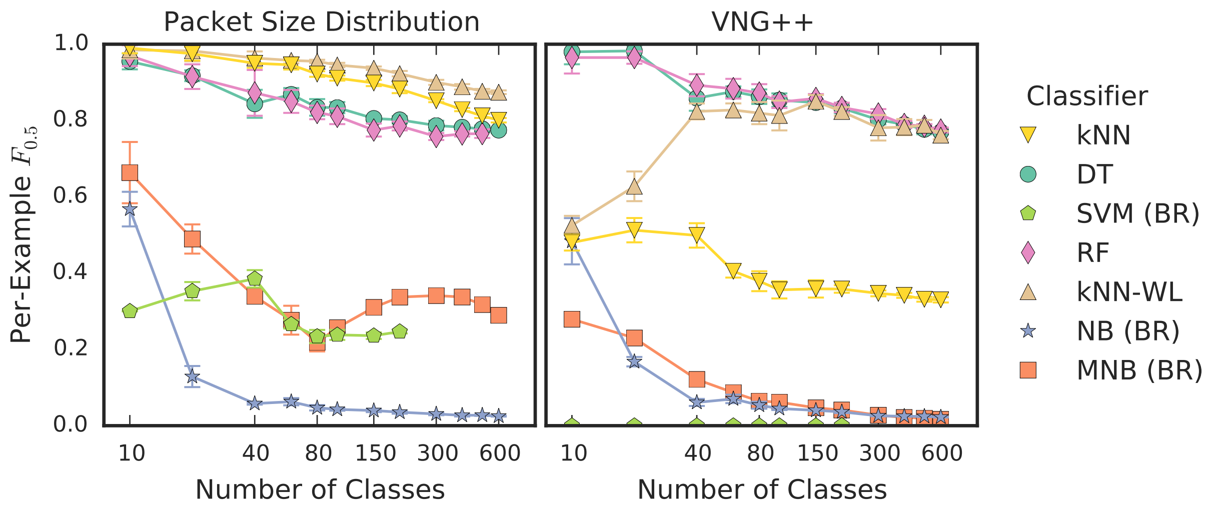


Figure 4.8: Multi-label classification applied to our SSH dataset with PacketSizeDistribution feature set (left) and the VNG++ feature set (right). Values shown are for the $F_{0.5}$ score (averaged per-example). Missing points indicate that the process exceeded our memory limit of 32GB. See also Figure B.1.

multiple-URLs-per-domain scenario. For the k-NN-WL classifier, the optimization procedure identified the combination of $k = 1$ and num_rounds $= 5$ as optimal in 21 of the 25 trials with 80 domains (instead of $k = 5$ with num_rounds $= 1$, which we originally selected based on results presented by Wang et al. [151]). For the SVM classifier, the optimization procedure identified different values for each trial for the C and γ parameters since these parameters are drawn from continuous distributions (see Section 4.4.4).[3] The final SVM parameter (class_weight) seemed to make little difference, with 13 trials weighted and 12 trials unweighted. Note that the NB classifier shows no changes as there are no parameters to optimize.

Labeling Traces with URL Components

Our first multi-label experiments demonstrate the feasibility of labeling traces with URL prefixes rather than attempting to each URL independently (as would be the naïve extension from prior work).

Our results (Figure 4.8) show that the proposed multi-label paradigm is not only feasible, but can be quite accurate when the proper classifiers are used. In particular, note that the classifiers

[3]For the trials with 80 domains, the means and standard deviations of the values identified as optimal in each trial are: 143.36 and 190.92, respectively, for C, and 0.0013 and 0.0012, respectively, for γ.

which natively support the multi-label paradigm (RF, DT, k-NN, and k-NN-WL) vastly out-perform those for which the BR method was used. Furthermore, for the experiments with more than 200 initial URLs, the SVM (BR) classifier vastly exceeded our memory limit (32GB). In this case, the macro- and micro-averaged results are similar to the per-example values depicted in Figure 4.8, and so we defer the presentation of those results to the appendix.

Abstaining from Classification

Our second set of multi-label experiments explore the benefits of abstaining from labeling an example if the classifier's confidence in it's prediction is low. In particular, we employ our abstention (Section 4.4.3) and threshold search (Section 4.4.3) mechanisms. For these experiments, we allow test examples with label sets where the classification model has not been trained on *any* of the labels. In other words, the label set for a test example may be completely disjoint from the set of labels on which the classifiers have been trained. This is analogous to the open-world scenario discussed in previous work (where URLs on which the classifier has not been trained are used for testing).

Since our dataset consists of traces from multiple URLs per domain, we isolate the training domains from those used for testing. Specifically, for each experiment, we select a set of domains which are *of-interest* for the experiment and a larger set of domains which are *of-no-interest*. For each such domain, we select one URL at random from our set of URLs for that domain to use for the experiment.

For the *of-interest* domains, we first use our epoch validation approach to determine which examples to use for training and testing for each trial. For each trial, we then set aside a portion of the training set to use for threshold determination. For the *of-no-interest* domains, we have no need of a training set because no models will be trained on examples from those domains. However, we do set aside a subset of those domains to use for threshold determination. The examples set aside for threshold determination (both *of-interest* and *of-no-interest*) are used neither for training the classifier models nor as part of the test data on which our final results are computed.

Due to the potential for bias in the selection of *of-interest* vs *of-no-interest* domains, each reported value is the mean of 25 trials: each experiment (itself consisting of 5 trials configured as above) is repeated 5 times with a different randomly-selected partitioning of the domains from the SSH dataset.

Our results (Figure 4.9a) in this scenario clearly demonstrate the ability of our approach to accurately label examples from domains *of-interest* and to ignore domains *of-no-interest*. This experiment also demonstrates (Figure 4.9b) the improvements in classification performance which our simple threshold-search mechanism (Section 4.4.3) delivers.

Summary of Findings

Our results also demonstrate several important lessons not only for the problem we examine herein, but for any application of machine learning:

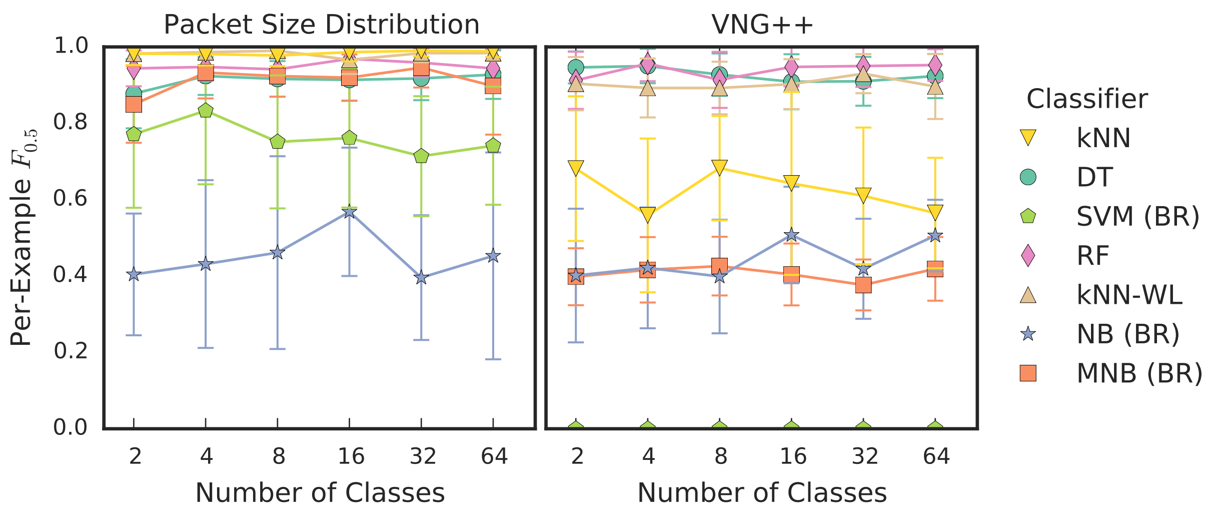
1. The classifiers for which the theory behind the classifier matches the problem at hand (RF, DT, k-NN, and k-NN-WL; see Section 4.4.2) consistently exhibit the best classification performance.

2. Our results indicate that choosing either feature set or classification algorithm in isolation leads to suboptimal results (see, e.g., Section 4.5.5); in particular, the underlying assumptions of classifiers should be considered before use in conjunction with a particular feature set.

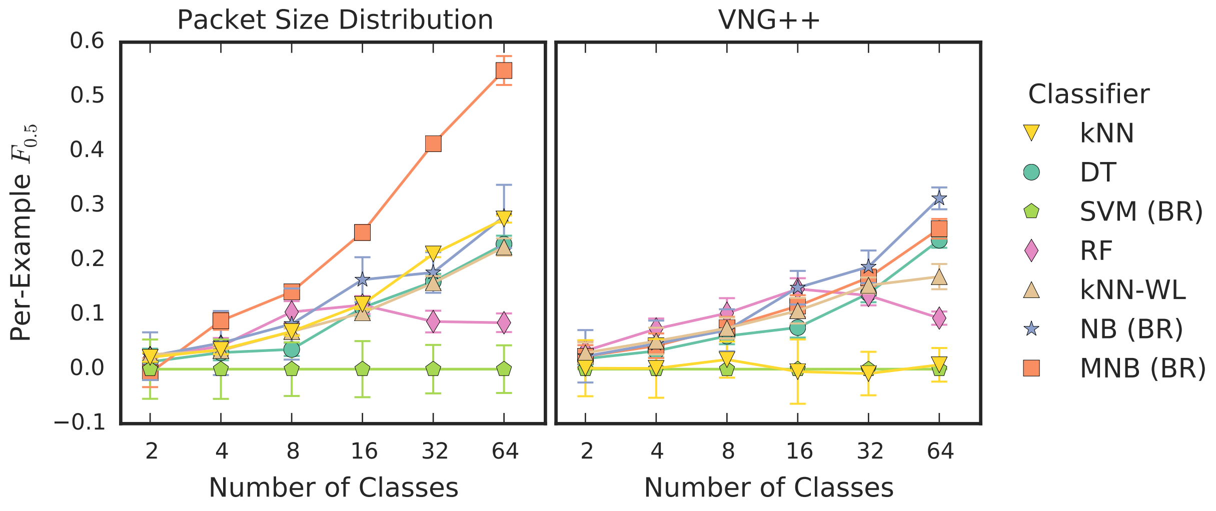
3. Different hyper-parameter values can yield significant differences in classification performance (see Section 4.5.5), particularly when attempting to apply the same parameter values to two different datasets.

We note that these lessons would not be news to the general machine learning community; rather, these are issues which have been overlooked by previous work in this particular area of research. In demonstrating the effects of these issues in this particular domain, we hope to educate security-focused audiences as to these considerations and improve the standard of practice for future work.

In addition to the general lessons described above, we can draw several conclusions from our results which are specific to the problem at hand:



(a) $F_{0.5}$ score (averaged per-example).



(b) Improvement in $F_{0.5}$ score (averaged per-example) with the threshold search.

Figure 4.9: Multi-label classification with abstention and threshold search for 8 domains *of-interest* with a varying number (on the x-axis) of domains *of-no-interest*. The dataset used is our SSH dataset with a single URL per domain and using the PacketSizeDistribution feature set (left) and the VNG++ feature set (right).

1. Assuming only a single URL per domain significantly overestimates the ability of a classifier to distinguish between traces representing different URLs.

2. The multi-label paradigm enables accurate labeling of traces even under the assumption of multiple URLs per domain.

3. When related URLs are included in the problem scope, significant partial information can be inferred even from traces of previously unseen URLs.

4. Incorporating an abstention mechanism can significantly decrease false positive rates.

In short, our findings validate our proposed paradigm and classifier combination in scenarios more complex and realistic than those considered in previous work.

4.5.6 Limitations and Future Work

Unfortunately, no study is perfect, and questions remain related to the internal validity of our study and those of previous work. These include a possible (in)sufficiency in the number of samples evaluated: the sample sizes in our experiments, which are comparable to those used in prior work, may not be sufficient to statistically validate whether over-fitting or under-fitting has occurred. It is difficult to determine whether enough examples are included to rightfully draw any strong general conclusions. One issue which we have not so far considered in this work is that of defenses. Preventing the type of analysis we propose in this work is an important goal given the unfortunate realities of internet monitoring and censorship. We remind the reader that our aim is not to prevent users from defending themselves against such monitoring; rather, our aim is to enable network administrators to better protect the networks—and, ultimately, the users—for which they are responsible. Unfortunately, properly ascertaining the impact on previously proposed defenses of the paradigm shift we outline in this work would require a separate study in and of itself. That said, we can make some informed predictions. In particular, while we propose the inference of additional information (e.g., subdomains and paths), the data from which these inferences are drawn (i.e., packet sizes, timing, and direction) remain the same. Therefore, techniques—such as the padding, buffering, fragmentation and camouflage techniques proposed in prior work [6, 21, 43, 88, 90, 93, 100, 104, 121, 152, 158]—which significantly reduce the fidelity of the signal represented by this data should also be effective against this new paradigm. Furthermore, we suggest that such defenses be evaluated not only on datasets such as those used in prior work, but also on datasets including multiple pages per domain, since we believe the latter to be the more realistic scenario.

A second issue which we have not directly considered in this work is that of the prior probabilities of visits to the URLs in question. In scenarios in which the prior probability of a "positive" example is low, FP rates calculated from experiments in which the priors are equal can significantly overestimate real-world accuracy. The security community has called for accuracy metrics, such as the Bayesian detection rate (BDR), which take these prior probabilities into account. Unfortunately, the Bayesian detection rate (BDR) and similar metrics are an over-simplification which are highly dependent on the specific values used for the priors. Since these priors are inherently a factor of the specific environment, we question the wisdom of reporting such summary statistics. Instead, we present various metrics (e.g., precision and recall) under various averaging mechanisms (e.g., per-example and per-label) along with the corresponding standard deviations to enable a nuanced interpretation on the part of the reader.

Finally, as future work we intend to explore the extent to which the multi-label approach allows for cross-domain identification of websites. As an example, both `stackoverflow.com` and `superuser.com` are part of the `stackexchange.com` network and use the same server-side software, though the templates are (slightly) different. Even if the classifier has never before seen `stackoverflow.com/questions`, the similarities between that page and `stackexchange.com/questions` may well allow the classifier to correctly label the page with `/questions` if we dissociate paths from domains.

4.6 Broader Implications

In the broader context of inference from encrypted network traffic, this work represents a substantial step towards the ability to ascertain useful, actionable information about tunneled HTTP connections. In particular, we demonstrate how a new paradigm (multi-label classification) can be leveraged to allow the inference of significantly more information than was previously believed, including partial information about URLs previously unseen by the system. We believe these new capabilities will allow future network monitoring and intrusion detection systems to make useful determinations with regard to encrypted connections.

CHAPTER 5: DISCUSSION & CONCLUSIONS

In Chapter 2, we demonstrated that a large and increasing proportion of network traffic is opaque, i.e., compressed or encrypted, rendering traditional IDS systems unable to detect and prevent security incidents in such traffic. Unfortunately, such increasingly ineffective systems are a major part of our network security infrastructure, particularly at the institutional level. As such, we believe that new, specialized techniques for analyzing opaque, and particularly encrypted, traffic are necessary to enable our network security infrastructure to cope with the threats we face now and in the future.

Also in Chapter 2, we presented techniques for identifying opaque traffic in real-time on high-speed networks and demonstrated how simply ignoring such traffic can significantly increase the throughput of IDS systems. The corollary to this increase in throughput is that IDS systems can more thoroughly examine other traffic. While this is a step in the right direction, it remains fundamentally a stopgap measure.

The remainder of this dissertation focused on encrypted traffic, in particular, and explored the extent to which one can extract information about the underlying contents of such traffic from side channels: in Chapter 3, we demonstrated how approximate transcripts of encrypted VoIP conversations can be reconstructed from packet sizes alone, and in Chapter 4, we showed how packet size and timing information can be leveraged to identify resources downloaded over encrypted tunnels (e.g., by visiting a webpage over a VPN connection). In both cases, our work represents a double-edged sword: such techniques can be used to provide information for security and forensic analysis and misused as methods for violating user privacy or censoring user activities. In Chapter 3, we focused on the threat to user privacy, but Chapter 4 focused on providing useful information about encrypted traffic for security and forensics.

APPENDIX A: OPAQUE TRAFFIC

This appendix presents various methods for opaque traffic identification, an extensive parameter space exploration experiment to compare these methods and provide optimal values for their parameters, the asymptotic efficiency of each method, and a description of the HTTP content-type labeling rules we used to determine ground truth in our experiments.

A.1 Comparison of Methods

A.1.1 Preliminaries

For all tests, let $\mathcal{H}_0$ be the hypothesis that the v_i are approximately uniformly distributed (e.g., the packet is compressed or encrypted), and let $\mathcal{H}_1$ be the alternative hypothesis, i.e., that the v_i are distributed according to some distribution that is not uniform (e.g., corresponding to a transparent packet).

In what follows, we examine approaches that can be broadly classified under two different models, which we refer to as operating on the *entropy* or *byte-value* domains. For the entropy domain, the basic unit is the (byte-) entropy of a *block* of bytes, where the number of bytes n in a block is parameterized. In other words, if X is a random variable in the entropy domain, then the support of X is the set of all possible values for the byte-entropy of n bytes. If X is a random variable in the byte-value domain, then the support of X is the set of integers 0 through 255. That is, for the byte-value domain, the basic unit is simply the byte.

Before delving into details of the various tests we explore, we first present necessary notation, summarized in Table A.1. To represent the packet payload, let $\mathbf{v} = \{v_1, v_2, \ldots, v_N\}$ be a sequence of observations (e.g., payload bytes), such that $v_i \in \{0, 1, \ldots, k-1\} \,\forall\, i \in \{1, \ldots, N\}$. That is, for a sequence $\mathbf{v}$ of bytes, $k = 256$. In the entropy domain, we operate on a sequence $\mathbf{w}$ of *blocks* of bytes $\mathbf{w} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N/n}\}$.

Notice that while the density function for the byte-value domain is well-known and easy to compute, the entropy domain is more complicated. For many of the tests we examine, we need to evaluate the probability mass function (PMF) over possible values for the sample entropy of n bytes for each hypothesis. Simple threshold checking of the sample entropy fails to leverage the distribution of sample entropy values, missing important information about the relative

Symbol	Meaning
n	size (in bytes) of a *block*
k	size of domain (e.g., 256 for bytes)
N	(maximum) number of payload bytes
M	(maximum) number of samples
v_i	observation (byte)
w_i	observation (block)
$\mathbf{v}$	sequence of observations (bytes)
$\mathbf{w}$	sequence of observations (blocks)
α	expected false negative rate
β	expected false positive rate
δ	alternative hypothesis weight
T	offset (in bytes)

Table A.1: Notation

likelihood of the values observed. For pedagogical reasons, we present the formal derivation of this mass function, for small n and a particular class of alternatives, in Section A.1.3. The essential idea is that the sample entropy of n bytes is the same regardless of the arrangement of those bytes. We can then enumerate the possible ways of arranging n bytes, for small n, to calculate the corresponding PMF. We implemented such an enumeration using NVIDIA's CUDA graphics processing unit (GPU) parallel programming toolkit, yielding a PMF for $n = 8$.

Discrete Kolmogorov-Smirnov Test

Another appropriate test for determining whether a sample is drawn from a particular distribution is the Kolmogorov-Smirnov (K-S) test. The K-S test is often used because of its generality and lack of restricting assumptions. The test is based on quantifying the difference between the cumulative distribution functions of the distributions of interest. However, we note that the traditional Kolmogorov-Smirnov test is only valid for continuous distributions, and Conover has already shown that the p-values for the continuous K-S test applied to discrete or discontinuous distributions can differ substantially from those of the discrete analog. Therefore, we apply the discrete version as described by Conover [28].

Lastly, we note that Pearson's χ^2 test also seems appropriate in this setting. However, Pearson's χ^2 test assumes that samples are independent, the sample size is large, and the expected cell counts are non-zero — but, the latter two assumptions do not necessarily hold in

	Domain	
Parameter	Entropy	Bytevalue[*]
α	$0.001, 0.005, 0.01, 0.05$	
β[†]	$0.001, 0.005, 0.01, 0.05$	
δ[‡]	$0.65, 0.75, 0.85$	
N	$8, 16, 24, 32, 48, 64, 80, 128$	$4, 12, 20, 28, 32, 40$
T	$0, 8, 16, 24$	$4, 12, 20$

[*] In addition to those for the entropy tests.
[†] Applicable only for sequential tests.
[‡] Not applicable for χ^2 or discrete K-S tests.

Table A.2: Parameter space explored for each domain

our setting as the goal is to examine as few bytes as possible. Nevertheless, we include such analyses for the sake of comparison to prior work [99], but omit the derivation since the test is in common usage and our application is no different.

A.1.2 Parameter Space Exploration

We now present the results of a parameter space exploration experiment examining all of the hypothesis tests with varying parameter values. We present receiver operating characteristic (ROC) plots that simultaneously examine the true positive rate and false positive rate as a parameter (e.g., a threshold) varies. A set of ROC plots, one for each classifier, can then be used to compare classifiers across a range of parameter values, allowing one to judge the relative performance of classifiers.

There are several knobs we can turn. For the sequential tests we explore the desired maximum false positive rate α, the desired maximum false negative rate β, and the maximum number of payload bytes N. The parameter values we examined for fixed tests included the desired significance level α and number of payload bytes N. In all experiments, the maximum number of samples for the sequential tests equals the sample size for the fixed tests.

We consider different alternative hypotheses by changing the relative weight, δ, of the lower 128 byte values versus the higher 128 byte values. The δ parameter takes values in $(0.5, 1.0]$, and indicates the expected percentage of byte values less than 128. By changing the value, we alter our model of transparent traffic. In addition to measuring uniformity, this class of alternative hypotheses has the advantages of being extremely efficient to implement (as determining whether a byte value is less than 128 is a simple bit mask operation) and of accounting for the prevalence of ASCII characters in network traffic.

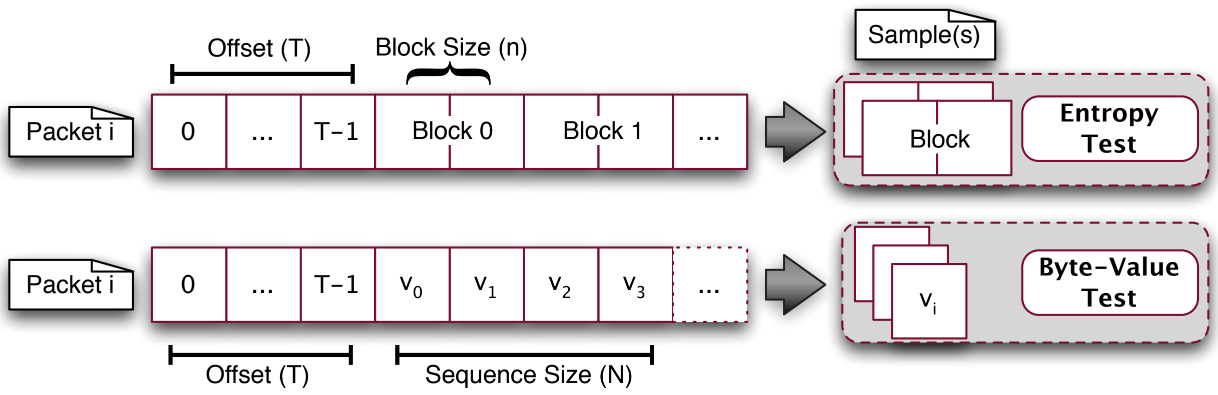


Figure A.1: Illustration of parameters length (N), offset (T), and block size (n)

Finally, we explore starting our analysis at different points within the packet, represented by the offset value T (in bytes). As shown pictorially in Figure A.1, T indicates how many bytes into the payload, i.e., past the end of the TCP header, our analysis begins. The specific values explored for each parameter are given in Table A.2. Our dataset for the exploration experiment consists of the first 100,000 packets from `trace1`, and we consider only *encrypted* data as positive examples.

Figure A.2shows a single point for each unique set of parameter values (over 10,000) in our experiments. Superior classifiers should evidence a high true positive rate and a low false positive rate, represented on a ROC plot by a predominance of points in the upper-left corner of the plot. Since the byte-value tests evidence the points closest to the upper-left corner, the plots indicate that the sequential tests and the likelihood-ratio test in the byte-value domain are able to more accurately classify packets than the other tests. We can also compare our techniques based on their theoretical computational efficiency (see Section A.1.4); again, the byte-value sequential tests are the clear winners.

We also determine which specific values for the various parameters provide the optimal performance. In order to do so, we make use of the so-called F-score, which is a weighted harmonic mean of the *precision* and *recall* metrics.[1] Precision is defined as the ratio of the number of packets correctly classified as opaque, i.e., *true positives*, to the total number of packets

[1]More precisely, we use the F_1-score, where recall and precision are evenly weighted.

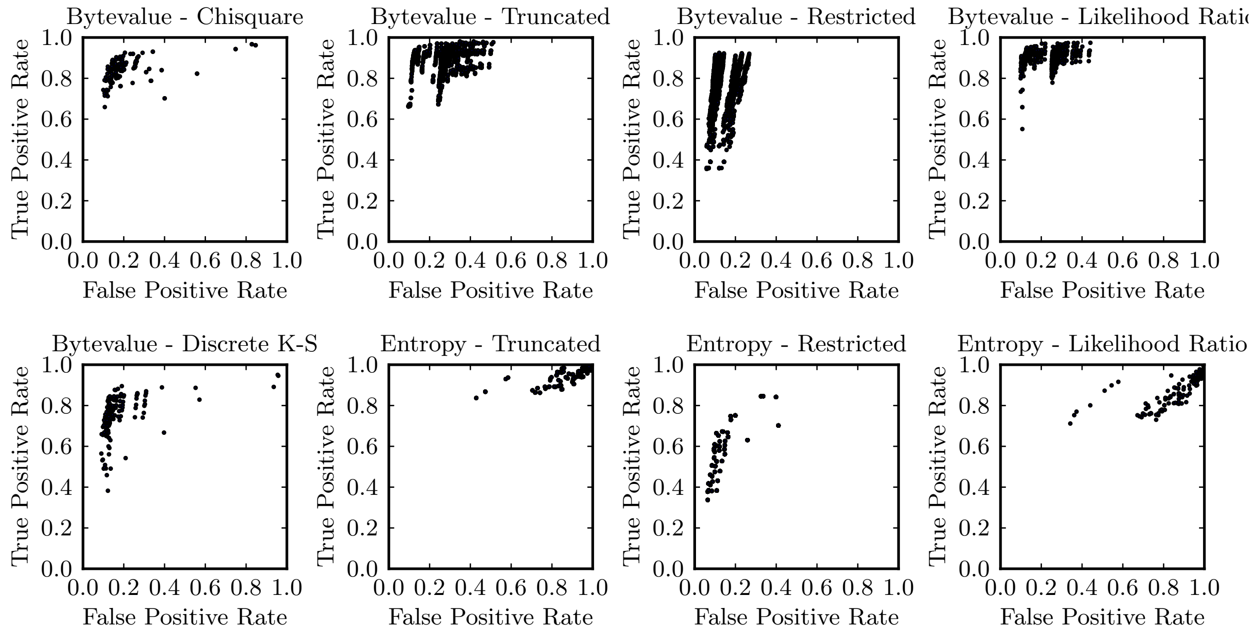
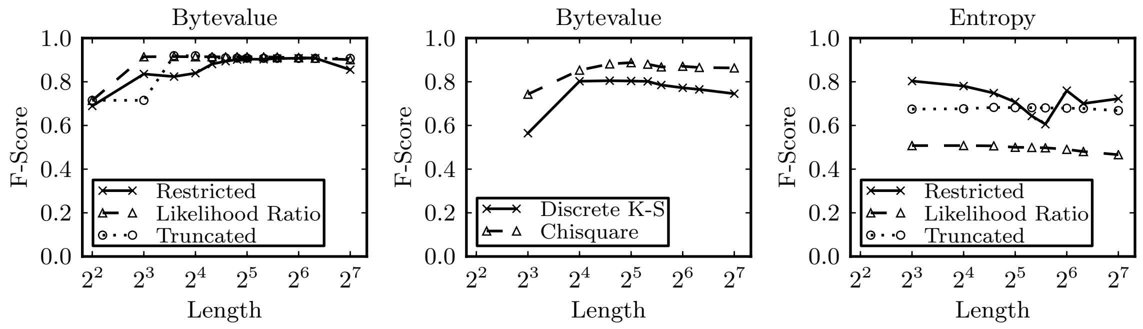


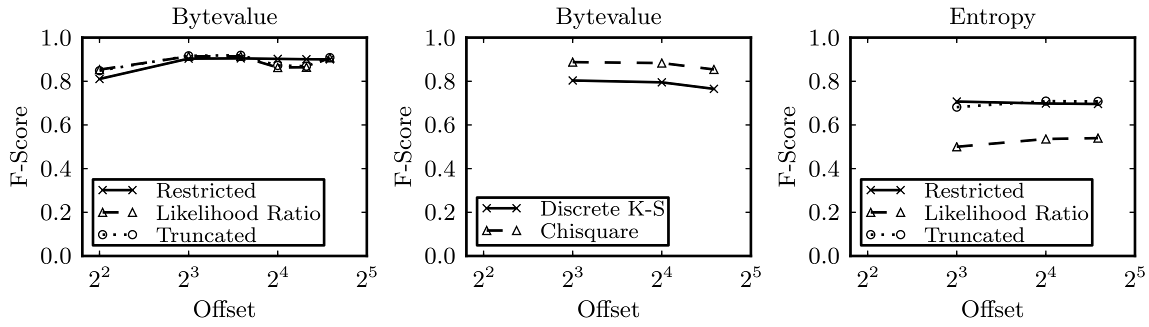
Figure A.2: ROC plots for the techniques examined in this work.

classified as opaque. Recall is defined as the ratio of the number of true positives to the number of packets which are opaque according to our ground truth. Said another way, precision can be seen as measuring how often packets labeled as opaque are actually opaque (but says nothing about how often opaque packets are correctly identified), while recall indicates how often the method correctly identifies opaque packets (but says nothing about how often transparent packets are mislabeled).

We now examine each parameter in isolation by varying the parameter of interest while fixing the other parameters at default values ($\delta = 0.85$, $\alpha = \beta = 0.005$, $N = 32$, and $T = 8$). As might be expected, the number of bytes examined has a substantial effect on the performance of the detectors (Figure A.3a); this effect drops off over time, suggesting that less than 32 bytes are needed in the general case to make a decision and less than 16 bytes are needed by the truncated test in the byte-value domain. The byte-value tests are the clear winners; all three sequential methods and the likelihood-ratio method performed equally well, each attaining precision over 90%. The entropy tests performed unexpectedly poorly, in few cases obtaining scores close to those of the other tests. In addition, we found that the restricted entropy test did not behave as



(a) F_1-Score for sample size



(b) F_1-Score for offset

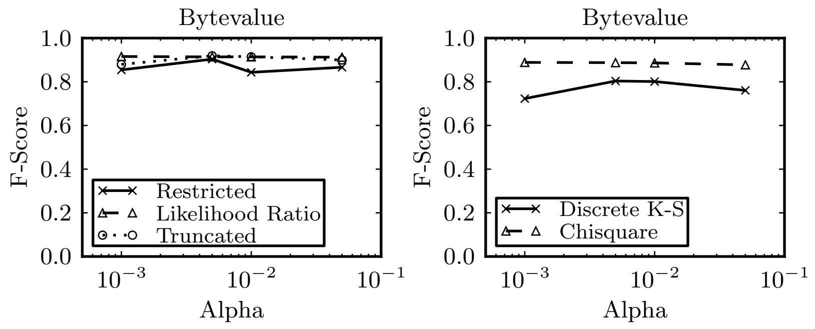
Figure A.3: Effects on accuracy when varying sample size and byte offset

we expected with regard to increasing the number of samples involved; an investigation revealed no underlying patterns in labels of the misclassified examples. We suspect that the particular decision boundaries (Section 2.2) used are simply poorly suited for the tests between entropy distributions.

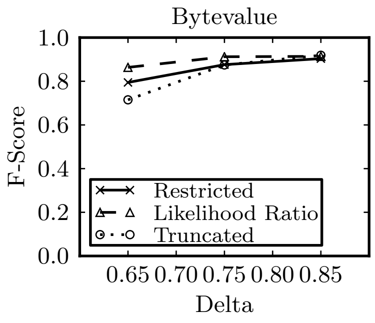
With regard to changes in offset (Figure A.3b), we see no clear improvement in classification for offset values larger than 8 bytes. Changes to the desired false positive and negative rates similarly had limit impact on the performance of the detectors, and so we omit those figures for brevity. Performance when varying the δ parameter, which controls the alternative hypothesis, peaks at $\delta = 0.85$.

Changes to the desired false positive rate (Figure A.4a), similarly, have little impact on the performance of the detectors (the case for the false negative rate β is the same; we omit the figure for brevity). In either case, the parameters have little effect on the entropy tests (due to the small number of samples for the entropy tests, e.g., 4 samples at $N = 32$ bytes).

We also examine changes to the δ parameter, which controls the alternative hypothesis.



(a) F_1-Score for significance level



(b) F_1-Score for alternative hypothesis

Figure A.4: Effects on accuracy when varying significance level and alternative hypothesis

Notice that in Figure A.4b, the sequential tests appear to fail most often when the byte value distribution under $\mathcal{H}_1$ is assumed to be close to uniform; this is due to a large number of trials failing to make a decision, or forcibly making the wrong decision, when faced with too few samples to effectively choose between two similar distributions.

Summary of Findings: Our analysis shows that we can narrow the field to the likelihood-ratio and sequential tests in the byte-value domain based on accuracy rates alone. Perhaps surprisingly, the entropy-based methods do not perform as accurately as those in the byte-value domain; we believe that this indicates that accurate entropy tests require more samples than are available in our context. In addition, examining more than 16 bytes provided little benefit in terms of accuracy. Interestingly, the offset and desired error rate parameters had little effect in our tests. Changes to the parameter (δ) controlling the alternative hypothesis did have a significant effect on the sequential tests; as the difference between the null and alternative hypothesis narrowed, the error rates for the sequential tests increased, suggesting that (as is to be expected) more samples are required for discrimination when the two classes to be distinguished are very similar. In summary, we found the truncated sequential test to one of the most accuracy and efficient tests, and therefore made use of this technique for the experiments in the body of the text.

A.1.3 Byte-Entropy Distributions for Small n

Formally, let $\mathcal{S}_n$ be the set of all possible values for the sample entropy of n bytes and let H be a random variable taking values in $\mathcal{S}_n$. We need to find the probability $\mathbb{P}[H = \mathrm{H}(\mathbf{x} \mid \mathcal{H}_i)]$ that the sample entropy $\mathrm{H}(\mathbf{v})$ of a sequence of bytes $\mathbf{v}$ equals some value H under each hypothesis (i.e., each of $\{\mathcal{H}_0, \mathcal{H}_1\}$).

More formally, define $\mathbf{x} = \{x_0, x_1, \ldots, x_{k-1}\}$ via

$$x_i = \sum_{j=0}^{n} \mathrm{I}(v_j = i),$$

where $\mathrm{I}(\cdot)$ is the indicator function, for all $i \in \{0, 1, \ldots, k-1\}$. Note that x_i is the count of the number of elements in $\mathbf{v}$ that are equal to i, i.e., the *multiplicity* of i. Then the sample entropy (in bits) is calculated as:

$$\mathrm{H}(\mathbf{v}) = h(\mathbf{x}) = \sum_{i=0}^{k-1} \frac{x_i}{n} \log_2 \frac{x_i}{n}$$

We can express our desired probability as the sum over all possible $\mathbf{x}$ which map to the same sample entropy:

$$\mathbb{P}[H = \mathrm{H}(\mathbf{v}) \mid \mathcal{H}_i] = \sum_{\mathbf{x} \text{ s.t. } \mathrm{H}(\mathbf{v}) = h(\mathbf{x})} \mathbb{P}[\mathbf{X} = \mathbf{x} \mid \mathcal{H}_i] \tag{A.1}$$

Since $\mathbf{X}$ is a vector-valued random variable with a multinomial distribution of length n and probabilities $\mathbf{p} = \{p_0, p_2, \ldots, p_{k-1}\}$, the corresponding probability mass function [147] is:

$$\mathbb{P}[\mathbf{X} = \mathbf{x} \mid n, \mathbf{p}] = \binom{n}{x_0, x_1, \ldots, x_{k-1}} p_0^{x_0} p_1^{x_1} \cdots p_{k-1}^{x_{k-1}}$$

In the equiprobable case where $p_i = 1/k$ for all i,

$$\mathbb{P}[\mathbf{X} = \mathbf{x} \mid n, k] = \binom{n}{x_0, x_1, \ldots, x_{k-1}} \left(\frac{1}{k}\right)^n \tag{A.2}$$

since $\mathbf{X}$ is a length-n multinomial and hence $\sum_{i=0}^{k-1} x_k = n$. Note that Equation A.2 *does not* depend on the order of elements in $\mathbf{x}$ (i.e., $\mathbf{x}$ can be viewed as a multiset). We note also that the entropy of $\mathbf{v}$ is the same *regardless of the arrangement of the values* in $\mathbf{v}$; therefore, the number of possible values of **v** for a given **x** is the number of ways of arranging the k elements of **x**. Define $\mathbf{y} = \{y_0, y_1, \ldots, y_n\}$ as

$$y_j = \sum_{i=0}^{k-1} \mathrm{I}(x_i = j)$$

for all $j \in \{0, 1, \ldots, n\}$ analogously to **x** above. Again, y_j is the count of the number of values in **x** that are equal to j, i.e., the multiplicity of j. Then the multinomial coefficient

$$\binom{k}{y_0, y_1, \ldots, y_n}$$

is the number of possible ways to arrange the k elements of **x**. Combining this with Equation A.2, we have (in the equiprobable case):

$$\begin{aligned}\mathbb{P}[H = \mathrm{H}(\mathbf{v}) \mid \mathcal{H}_0] &= \binom{k}{y_0, y_1, \ldots, y_n} \mathbb{P}[\mathbf{X} = \mathbf{x} \mid n, k] \\ &= \binom{k}{y_0, y_1, \ldots, y_n} \binom{n}{x_0, x_1, \ldots, x_{k-1}} \left(\frac{1}{k}\right)^n\end{aligned}$$

Log-likelihoods In their original forms, the probability mass functions are difficult to compute due to the extremely large values involved. However, we can calculate in log-space to mitigate the issue:

$$\begin{aligned}\log(\mathbb{P}[H = \mathrm{H}(\mathbf{v}) \mid n, k, \mathcal{H}_0]) &= \log\left[\binom{k}{y_0, y_1, \ldots, y_n} \binom{n}{x_0, x_1, \ldots, x_{k-1}} \left(\frac{1}{k}\right)^n\right] \\ &= \log\left[\left(\frac{k!}{y_0! y_1! \cdots y_n!}\right) \left(\frac{n!}{x_1! x_2! \cdots x_k!}\right) \left(\frac{1}{k}\right)^n\right] \\ &= \log\left(\frac{k! n!}{k^n}\right) - \log\left(x_1! x_2! \cdots x_k! y_0! y_1! \cdots y_n!\right) \\ &= \log\left(\frac{k! n!}{k^n}\right) - \left[\sum_{i=0}^{n} \log(y_i!) + \sum_{i=1}^{k} \log(x_i!)\right] \\ &= \log(k!) + \log(n!) - n \log(k) - \sum_{i=0}^{n} \log(y_i!) - \sum_{i=1}^{k} \log(x_i!)\end{aligned}$$

A.1.4 Theoretical Efficiency

We can also compare our techniques based on their theoretical computational efficiency (Table A.3), in terms of the number of online floating-point operations required for each case. In the following, any operations which can be precomputed are omitted. In the byte-value domain, the sequential tests each require only a single multiplication and comparison per iteration, so the

Test Type	Floating-Point Operations
Sequential	$\leq 2M$
Likelihood Ratio	$2M$
Pearson's χ^2	$M + 3k$
Discrete K-S	$O(M^2)$

Table A.3: Theoretical efficiency

Base Type	Sub-type	Action/Label
image, video, audio	*	opaque
text	*	transparent
application	pdf, xml, flash	transparent
application	gzip, zip	opaque
application	*	omit
*	*	omit

Table A.4: HTTP Content-Type Filtering Rules

number of standard floating-point operations is no more than $2M$ for M samples (in the byte-value domain, samples are bytes, so $M = N$); the likelihood ratio test is the same, but with equality in the relation. Working in the entropy domain introduces the overhead both of binning the samples ($M = N/n$ operations) and of computing the sample entropy ($3k$ standard operations plus k logarithms). Pearson's χ^2 test requires M operations to bin the samples and $3k$ operations, where k is the size of the domain (e.g., $k = 256$ for the byte-value domain), to sum the squared difference between the observed count and expected count over all bins. Finally, the discrete K-S test is $O(M^2)$. Obviously, from a performance standpoint, neither the K-S test nor the entropy tests are a good choice.

A.2 HTTP Labeling Rules

To determine ground truth for HTTP packets, we first examine the content-encoding: if the strings gzip or deflate appear in the content-encoding, the packet is labeled opaque; otherwise, the label is determined by content type. HTTP content-type fields contain a base type, such as text or video, a sub-type, such as html or mp4, and optional parameters, such as `charset=us-ascii`. For our filtering, if the base type is image, video, or audio, the traffic is labeled opaque; if the base type is text, the traffic is labeled transparent. For the application base type, we also consider the sub-type: zip and x-gzip are considered opaque, while xml, pdf, and x-shockwave-flash are considered transparent. While some of these sub-types (e.g., PDF) can be containers for many other formats, we choose to be conservative and simply classify them as transparent since we have no clear cut way of drawing the line between semi-structured and more opaque formats. All other sub-types are dropped, since hand-labeling of all potential content-types is infeasible and error-prone. Our HTTP filtering and labeling rules are summarized in Table A.4 (given in order of application).

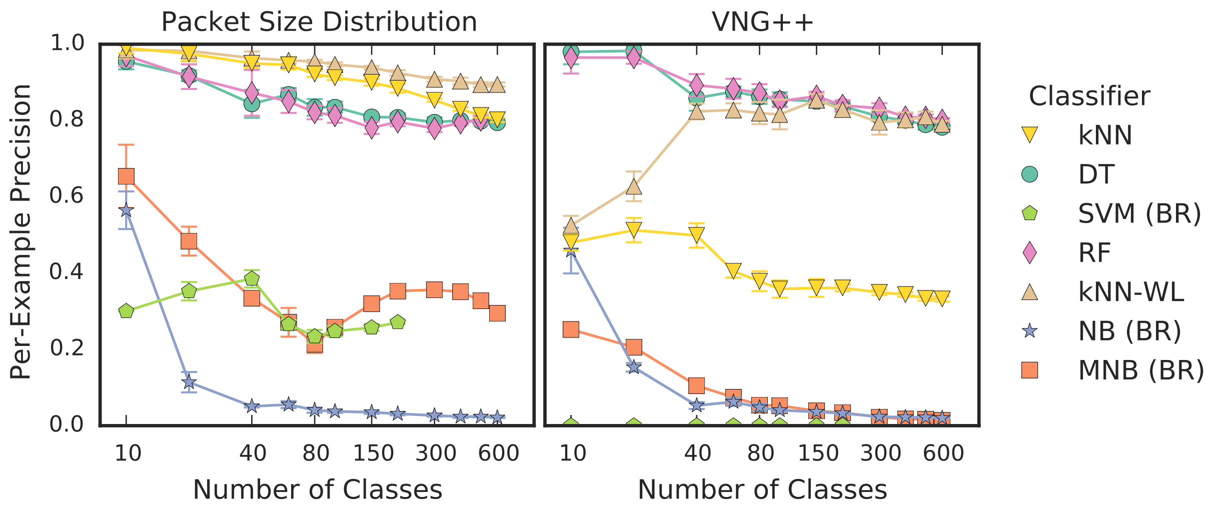
APPENDIX B: PLAYING HIDE-AND-SEEK

B.1 Scripted Retrieval Details

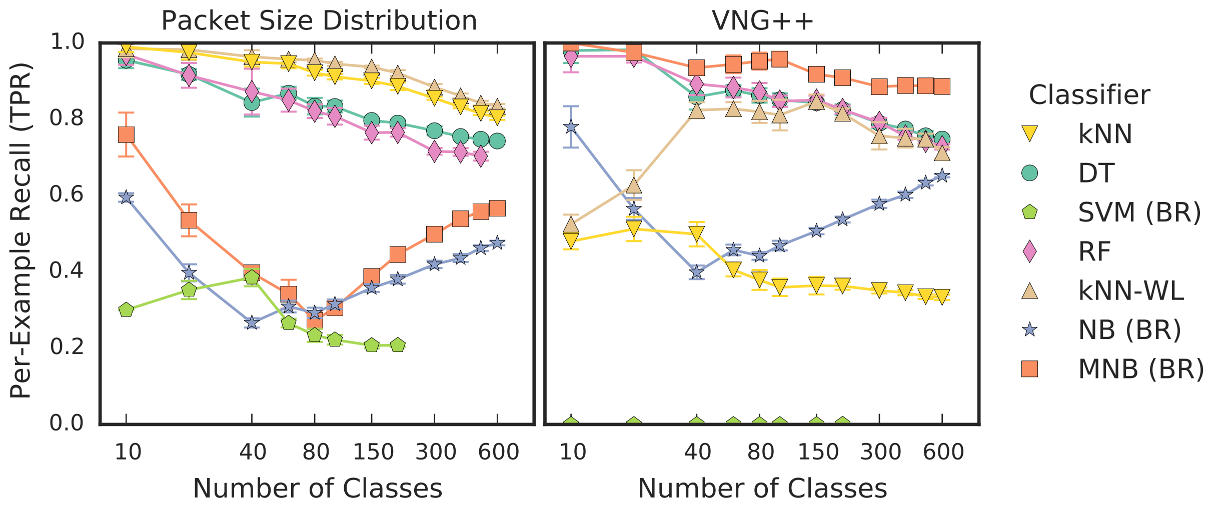
For the case of a *dynamic* SSH tunnel, our detailed process for retrieving a set of URLs is as follows. For each URL, the script establishes an SSH connection to a server using *dynamic* application-level port forwarding (OpenSSH [112]'s `-D` flag). This means that SSH acts as a SOCKS proxy server and forwards connections directed to a specified local port through the tunnel to the SSH server and from there on to the final destination. The script waits 2 seconds to ensure connection setup has completed, then starts to record the network traffic corresponding to the retrieval. The recording is done with three separate `tcpdump` processes (Figure 4.2):

1. on the `loopback` interface, monitoring the local proxy port, to collect the plaintext traffic
2. on the internal (intranet) interface, monitoring the SSH connection, to collect the encrypted traffic
3. on the external (internet) interface, monitoring ports 53, 80, and 443, as a sanity check to ensure that no traffic bypassed the tunnel

The script waits another 2 seconds to ensure that the capture processes have initialized appropriately, then launches a Python script which, using the Selenium browser automation framework [131], opens a Firefox browser process (configured to use the SOCKS proxy provided by the SSH process for both HTTP and DNS traffic) and directs the browser to visit the specified URL. A clean Firefox profile and temporary directory are used for each URL, ensuring that no detritus (e.g., cached files or cookies) remain from the previous retrieval. The browser is opened in a virtual framebuffer (i.e., an `X` server which does not require a screen) using `Xvfb` [160], avoiding the overhead of a full-blown window-manager. Once the requested page's JavaScript `onload` event is fired, Firefox takes and stores a screenshot, then closes (if the `onload` event does not fire within 30 seconds of starting Firefox, then the process is killed). Control is returned to the outer shell script, which waits 2 seconds, then terminates the `tcpdump` processes; once these have exited, the SSH connection is terminated. The script waits another 2 seconds, then begins again with the next URL.



(a) Precision



(b) Recall

Figure B.1: Multi-label classification applied to our SSH dataset with the PacketSizeDistribution feature set (left) and the VNG++ feature set (right). Missing points indicate that the process exceeded our memory limit of 32GB. See also Figure 4.8.

B.2 Precision and Recall

The precision and recall values for the multi-label experiment of Section 4.5.5 are shown in Figure B.1.

BIBLIOGRAPHY

[1] 3GPP. *Extended Adaptive Multi-Rate Wideband (AMR-WB+) Codec*. Tech. rep. 26.290. 3rd Generation Partnership Project (3GPP), 2009.

[2] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. “Multi-label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages”. In: *Proc. 22nd International Conference on World Wide Web (WWW)*. 2013.

[3] Irfan Ahmed, Kyung-suk Lhee, Hyunjung Shin, and ManPyo Hong. “Fast file-type identification”. In: *Proceedings of the Symposium on Applied Computing*. 2010, pp. 1601–1602.

[4] *Alexa - The Web Information Company*. URL: http://www.alexa.com (visited on 05/30/2013).

[5] Michael Backes, Goran Doychev, Markus Dürmuth, and Boris Köpf. “Speaker Recognition in Encrypted Voice Streams”. In: *ESORICS 2010: 15th European Symposium on Research in Computer Security*. Ed. by Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou. Vol. 6345. Lecture Notes in Computer Science. Springer, Sept. 2010, pp. 508–523.

[6] Michael Backes, Goran Doychev, and Boris Köpf. “Preventing Side-Channel Leaks in Web Traffic: A Formal Approach”. In: *ISOC Network and Distributed System Security Symposium – NDSS 2013*. The Internet Society, Feb. 2013.

[7] Leonard E. Baum, Ted Petrie, George Soules, and Normal Weiss. “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains”. In: *The Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171.

[8] Doug Beeferman, Adam Berger, and John Lafferty. “Statistical Models for Text Segmentation”. In: *Mach. Learn.* 34.1-3 (1999).

[9] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. “A Maximum Entropy approach to Natural Language Processing”. In: *Computational Linguistics* 22 (1996), pp. 39–71.

[10] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: *Journal of Machine Learning Research* (2012), pp. 281–305.

[11] Ignacio N Bermudez, Marco Mellia, Maurizio M Munafo, Ram Keralapura, and Antonio Nucci. “DNS to the rescue: discerning content and services in a tangled web”. In: *ACM Internet Measurement Conference (IMC)*. 2012.

[12] Laurent Bernaille and Renata Teixeira. “Early Recognition of Encrypted Applications”. In: *Passive and Active Measurement Conference*. 2007.

[13] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. “Privacy vulnerabilities in encrypted HTTP streams”. In: *Proc. 5th International Workshop on Privacy Enhancing Technologies*. 2006.

[14] Andrea Bittau, Michael Hamburg, Mark Handley, David Mazières, and Dan Boneh. “The Case for Ubiquitous Transport-Level Encryption”. In: *USENIX Security Symposium*. 2010.

[15] *Blacklists UT1*. Université Toulouse 1 Capitole. URL: http://dsi.ut-capitole.fr/blacklists/index_en.php (visited on 02/11/2014).

[16] Daniel Blanchard, Jeffrey Heinz, and Roberta Golinkoff. "Modeling the Contribution of Phonotactic Cues to the Problem of Word Segmentation". In: *The Journal of Child Language* 37.3 (2010), pp. 487–511.

[17] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. "Revealing Skype traffic: when randomness plays with you". In: *Comp. Commun. Review*. SIGCOMM (2007), pp. 37–48.

[18] H. Bortfeld, J. Morgan, R. Golinkoff, and K. Rathbun. "Mommy and Me: Familiar names help launch babies into speech-stream segmentation". In: *Psychological Science* 16 (2005), pp. 298–304.

[19] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[20] Julian J. Bussgang and Michael B. Marcus. "Truncated Sequential Hypothesis Tests". In: *IEEE Transactions on Information Theory* 13.3 (July 1967).

[21] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. "Touching from a distance: website fingerprinting attacks and defenses". In: *ACM CCS 12: 19th Conference on Computer and Communications Security*. Ed. by Ting Yu, George Danezis, and Virgil D. Gligor. ACM Press, Oct. 2012, pp. 605–616.

[22] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli. "An Experimental Evaluation of the Computational Cost of a DPI Traffic Classifier". In: *Global Telecommunications Conference*. 2009, pp. 1–8.

[23] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. "Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow". In: *2010 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2010, pp. 191–206.

[24] Heyning Cheng and Ron Avnur. *Traffic Analysis of SSL Encrypted Web Browsing*. 1998.

[25] Ken Chiang and Levi Lloyd. "A case study of the Rustock rootkit and spam bot". In: *HotBots'07: Proceedings of the First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007.

[26] Ronald A. Cole and Jola Jakimik. "A Model of Speech Perception". In: *Perception and production of fluent speech*. Lawrence Erlbaum Associates, 1980. Chap. 6, pp. 133–163.

[27] *Configuring a SOCKS proxy server in Chrome*. URL: http://www.chromium.org/developers/design-documents/network-stack/socks-proxy (visited on 02/19/2014).

[28] W. J. Conover. "A Kolmogorov Goodness-of-Fit Test for Discontinuous Distributions". In: *Journal of the American Statistical Association* 67 (1972), pp. 591–596.

[29] Gregory Conti, Sergey Bratus, Anna Shubina, Benjamin Sangster, Roy Ragsdale, Matthew Supan, Andrew Lichtenberg, and Robert Perez-Alemany. “Automated mapping of large binary objects using primitive fragment type classification”. In: *Digital Investigation* 7.Supplement 1 (2010), S3–S12.

[30] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. “Traffic classification through simple statistical fingerprinting”. In: *SIGCOMM Comput. Commun. Rev.* 37.1 (2007), pp. 5–16.

[31] George Danezis. *Traffic Analysis of the HTTP Protocol over TLS*.

[32] Michael Denkowski, Abhaya Agarwal, Satanjeev Banerjee, and Alon Lavie. *The METEOR MT Evaluation System, Version 1.2*. Carnegie Mellon University. Pittsburgh, PA, 2010.

[33] Michael Denkowski and Alon Lavie. “Choosing the Right Evaluation for Machine Translation: an Examination of Annotator and Automatic Metric Performance on Human Judgment Tasks”. In: *Proceedings of AMTA*. 2010.

[34] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The Second-Generation Onion Router”. In: *Proceedings of the 13th USENIX Security Symposium*. 2004, pp. 303–320.

[35] Peter Dorfinger. “Real-Time Detection of Encrypted Traffic based on Entropy Estimation”. MA thesis. Salzburg University of Applied Sciences, 2010.

[36] Peter Dorfinger, Georg Panholzer, and Wolfgang John. “Entropy Estimation for Real-Time Encrypted Traffic Identification”. In: *Traffic Monitoring and Analysis*. Vol. 6613. Lecture Notes in Computer Science. 2011.

[37] Peter Dorfinger, Georg Panholzer, Brian Trammell, and Teresa Pepe. “Entropy-based traffic filtering to support real-time Skype detection”. In: *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*. 2010, pp. 747–751.

[38] Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson, and Robin Sommer. “Dynamic application-layer protocol analysis for network intrusion detection”. In: *Proceedings of the 15th USENIX Security Symposium*. 2006.

[39] Holger Dreger, Anja Feldmann, Vern Paxson, and Robin Sommer. “Operational experiences with high-volume network intrusion detection”. In: *Proceedings of the 11th ACM conference on Computer and Communications Security*. 2004.

[40] Benoît Dupasquier Dupasquier, Stefan Burschka, Kieran McLaughlin, and Sakir Sezer. “Analysis of information leakage from encrypted Skype conversations”. In: *International Journal of Information Security* (2010), pp. 1–13.

[41] Richard Durbin, Sean Eddy, Anders Grogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[42] Maurizio Dusi, Manuel Crotti, Francesco Gringoli, and Luca Salgarelli. “Detection of Encrypted Tunnels Across Network Boundaries”. In: *ICC*. 2008, pp. 1738–1744.

[43] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. “Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail”. In: *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2012, pp. 332–346.

[44] Anna Esposito and Guido Aversano. “Text Independent Methods for Speech Segmentation”. In: *Nonlinear Speech Modeling and Applications*. Vol. 3445. Lecture Notes in Computer Science. Springer, 2005, pp. 261–290.

[45] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585. Internet Engineering Task Force, June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt.

[46] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. “Polymorphic blending attacks”. In: *Proceedings of the USENIX Security Symposium*. 2006, pp. 241–256.

[47] W. N. Francis and H. Kucera. *Brown Corpus Manual*. Tech. rep. Department of Linguistics, Brown University, 1979.

[48] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor Zue. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. 1993.

[49] D. Gildea and D. Jurasky. “Learning Bias and Phonological-Rule Induction”. In: *Computational Linguistics* 22.4 (1996), pp. 497–530.

[50] *Global Internet Phenomena Report: Fall 2011*. Sandvine, Oct. 27, 2011.

[51] *Global Internet Phenomena Report: Fall 2014*. Sandvine, 2014.

[52] *Global Internet Phenomena Spotlight: Encrypted Internet Traffic*. Sandvine, May 8, 2015.

[53] José M. González, Vern Paxson, and Nicholas Weaver. “Shunting: a hardware/software architecture for flexible, high-performance network intrusion prevention”. In: *ACM CCS 07: 14th Conference on Computer and Communications Security*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM Press, Oct. 2007, pp. 139–149.

[54] Dan Goodin. “It wasn’t easy, but Netflix will soon use HTTPS to secure video streams”. In: *Ars Technica* (Apr. 16, 2015). URL: http://arstechnica.com/security/2015/04/it-wasnt-easy-but-netflix-will-soon-use-https-to-secure-video-streams/ (visited on 08/27/2015).

[55] John Haggerty and Mark Taylor. “FORSIGS: Forensic Signature Analysis of the Hard Drive for Multimedia File Fingerprints”. In: *Proceedings of IFIP International Information Security Conference*. 2006.

[56] Gregory A. Hall. *Sliding Window Measurement for File Type Identification*. ManTech Security and Mission Assurance. 2006.

[57] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA data mining software: an update". In: *SIGKDD Explorations* 11.1 (2009), pp. 10–18. DOI: 10.1145/1656274.1656278.

[58] M. Halle. "Knowledge unlearned and untaught: What speakers know about the sounds of their language". In: *Linguistic Theory and Psychological Reality* (1978), pp. 294–303.

[59] Jonathan Harrington, Gordon Watson, and Maggie Cooper. "Word boundary identification from phoneme sequence constraints in automatic continuous speech recognition". In: *Computational linguistics - Volume 1*. 1988, pp. 225–230.

[60] Ryan M. Harris. "Using Artificial Neural Networks for Forensic File Type Identification". MA thesis. Purdue University, 2007.

[61] Reed Hastings and David Wells. *April 2015 Investor Letter*. Apr. 15, 2015.

[62] Bruce Hayes and Colin Wilson. "A maximum entropy model of phonotactics and phonotactic learning". In: *Linguistic Inquiry* 39.3 (2008), pp. 379–440.

[63] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier". In: *Proc. ACM Workshop on Cloud Computing Security*. 2009.

[64] Y. Hifny and S. Renals. "Speech Recognition Using Augmented Conditional Random Fields". In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.2 (2009), pp. 354–365.

[65] Andrew Hintz. "Fingerprinting websites using traffic analysis". In: *Proc. 2nd International Workshop on Privacy Enhancing Technologies*. Apr. 2002.

[66] Chih-wei Hsu, Chih-chung Chang, and Chih-jen Lin. "A practical guide to support vector classification". 2010. URL: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (visited on 08/12/2015).

[67] *HTTPS Everywhere*. URL: https://www.eff.org/https-everywhere (visited on 02/16/2014).

[68] A. Iacovazzi and A. Baiocchi. "Optimum packet length masking". In: *22nd International Teletraffic Congress (ITC)*. 2010, pp. 1–8.

[69] *Identity Fraud Survey Report*. Javelin Strategy & Research, 2010.

[70] Mohamad Jaber and Chadi Barakat. "Enhancing Application Identification by Means of Sequential Testing". In: *Proceedings of the 8th International IFIP-TC 6 Networking Conference*. 2009.

[71] E. T. Jaynes. "Information Theory and Statistical Mechanics". In: *Phys. Rev.* 106.4 (May 1957), pp. 620–630.

[72] Frederick Jelinek. *Statistical Methods for Speech Recognition*. Massachusetts Institute of Technology, 1997.

[73] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. "A Critical Evaluation of Website Fingerprinting Attacks". In: *ACM CCS 14: 21st Conference on Computer and Communications Security*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM Press, Nov. 2014, pp. 263–274.

[74] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. "Fast Portscan Detection Using Sequential Hypothesis Testing". In: *2004 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2004, pp. 211–225.

[75] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2008.

[76] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. "Transport layer identification of P2P traffic". In: *IMC '04: Proc. 4th ACM SIGCOMM conference on Internet measurement*. Oct. 2004.

[77] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. "BLINC: multilevel traffic classification in the dark". In: *SIGCOMM Comput. Commun. Rev.* 35.4 (2005), pp. 229–240.

[78] Timothy Kempton and Roger K. Moore. "Language Identification: Insights from the Classification of Hand Annotated Phone Transcripts". In: *Speaker and Language Recognition Workshop*. Jan. 2008.

[79] Angelos D. Keromytis. "A Survey of Voice over IP Security Research". In: *Proceedings of the 5th International Conference on Information Systems Security*. 2009, pp. 1–18.

[80] Alon Lavie. *Evaluating the Output of Machine Translation Systems*. AMTA Tutorial. 2010.

[81] Alon Lavie and Michael J. Denkowski. "The METEOR metric for automatic evaluation of machine translation". In: *Machine Translation* 23 (Sept. 2009), pp. 105–115.

[82] Brian Lavoie and Henrik Frystyk Nielsen, eds. *Web Characterization Terminology & Definitions Sheet*. 1999. URL: http://www.w3.org/1999/05/WCA-terms (visited on 12/12/2013).

[83] Kai-Fu Lee and Hsiao-Wuen Hon. "Speaker-independent phoneme recognition using Hidden Markov Models". In: *The Journal of the Acoustical Society of America* 84.S1 (1988), p. 62.

[84] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. *SOCKS Protocol Version 5*. RFC 1928 (Proposed Standard). Internet Engineering Task Force, Mar. 1996. URL: http://www.ietf.org/rfc/rfc1928.txt.

[85] T. Leila and R. Bettati. "Privacy of encrypted Voice-over-IP". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. 2007, pp. 3063–3068.

[86] *Let's Encrypt*. URL: https://letsencrypt.org (visited on 08/27/2015).

[87] Binglong Li, Qingxian Wang, and Junyong Luo. “Forensic Analysis of Document Fragment Based on SVM”. In: *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Dec. 2006.

[88] Marc Liberatore and Brian Neil Levine. “Inferring the source of encrypted HTTP connections”. In: *ACM CCS 06: 13th Conference on Computer and Communications Security*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, Oct. 2006, pp. 255–263.

[89] Dong C. Liu, Jorge Nocedal, and Dong C. “On the Limited Memory BFGS Method for Large Scale Optimization”. In: *Mathematical Programming* 45 (1989), pp. 503–528.

[90] Wen Ming Liu, Lingyu Wang, Kui Ren, Pengsu Cheng, and Mourad Debbabi. “k-Indistinguishable traffic padding in web applications”. In: *International Conference on Privacy Enhancing Technologies*. 2012.

[91] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. “Website Fingerprinting and Identification Using Ordered Feature Sequences”. In: *ESORICS 2010: 15th European Symposium on Research in Computer Security*. Ed. by Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou. Vol. 6345. Lecture Notes in Computer Science. Springer, Sept. 2010, pp. 199–214.

[92] Yuanchao Lu. “On Traffic Analysis Attacks To Encrypted VoIP Calls”. MA thesis. Cleveland State University, Fenn College of Engineering, 2009.

[93] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. “HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows”. In: *ISOC Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, Feb. 2011.

[94] Robert Lyda and James Hamrock. “Using Entropy Analysis to Find Encrypted and Packed Malware”. In: *IEEE Security and Privacy* 5 (Mar. 2007), pp. 40–45.

[95] Alok Madhukar and Carey Williamson. “A Longitudinal Study of P2P Traffic Classification”. In: *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS)*. 2006, pp. 179–188.

[96] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. “An extensive experimental comparison of methods for multi-label learning”. In: *Pattern Recognition* 45.9 (2012), pp. 3084–3104.

[97] Gregor Maier, Robin Sommer, Holger Dreger, Anja Feldmann, Vern Paxson, and Fabian Schneider. “Enriching network security analysis with time travel”. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2008, pp. 183–194.

[98] Gianluca Maiolini, Andrea Baiocchi, Alfonso Iacovazzi, and Antonello Rizzi. “Real Time Identification of SSH Encrypted Application Flows by Using Cluster Analysis Techniques”. In: *Proceedings of the International IFIP-TC 6 Networking Conference*. 2009, pp. 182–194.

[99] Paras Malhotra. "Detection of Encrypted Streams for Egress Monitoring". MA thesis. Iowa State University, 2007.

[100] Suhas Mathur and Wade Trappe. "BIT-TRAPS: Building Information-Theoretic Traffic Privacy Into Packet Streams". In: *IEEE Transactions on Information Forensics and Security* 6.3 (2011), pp. 752–762.

[101] Ueli M. Maurer. "A Universal Statistical Test for Random Bit Generators". In: *Journal of Cryptology* 5.2 (1992), pp. 89–105.

[102] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. "Flow Clustering Using Machine Learning Techniques". In: *PAM*. 2004, pp. 205–214.

[103] *Message Stream Encryption*. URL: http://wiki.vuze.com/w/Message_Stream_Encryption (visited on 08/20/2015).

[104] Brad Miller, Ling Huang, Anthony D. Joseph, and J. Doug Tygar. "I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis". In: *CoRR* abs/1403.0297 (2014). arXiv: 1403.0297 [cs.CR].

[105] Andrew W. Moore and Konstantina Papagiannaki. "Toward the Accurate Identification of Network Applications". In: *Proceedings of the 6th International Passive and Active Network Measurement Workshop (PAM)*. Vol. 3431. Lecture Notes in Computer Science. Springer, 2005, pp. 41–54.

[106] Iosif Mporas, Todor Ganchev, and Nikos Fakotakis. "Speech segmentation using regression fusion of boundary predictions". In: *Computer Speech & Language* 24.2 (2010), pp. 273–288.

[107] Chitra Muthukrishnan, Vern Paxson, Mark Allman, and Aditya Akella. "Using strongly typed networking to architect for tussle". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. 2010.

[108] *Network.proxy.socks remote DNS*. URL: http://kb.mozillazine.org/Network.proxy.socks_remote_dns (visited on 02/19/2014).

[109] University of North Carolina Information Technology Services. *Transmission of Protected Health Information and Personal Identifying Information Policy*. June 30, 2010. URL: https://its.unc.edu/files/2014/08/Transmission-of-Protected-Health-Information-and-Personal-Identifying-Information-Policy.pdf (visited on 08/20/2015).

[110] M.P. Oakes. "Computer Estimation of Vocabulary in Protolanguage from Word Lists in Four Daughter Languages". In: *Journal of Quantitative Linguistics* 7.3 (2000), pp. 233–243.

[111] Julien Olivain and Jean Goubault-Larrecq. *Detecting Subverted Cryptographic Protocols by Entropy Checking*. Research Report LSV-06-13. Laboratoire Spécification et Vérification, ENS Cachan, France, June 2006.

[112] *OpenSSH*. URL: http://www.openssh.com (visited on 08/07/2015).

[113] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. "Website fingerprinting in onion routing based anonymization networks". In: *Proc. 10th ACM Workshop on Privacy in the Electronic Society*. 2011.

[114] Liam Paninski. "Estimating Entropy on m Bins Given Fewer than m Samples". In: *IEEE Transactions on Information Theory* 50.9 (2004), pp. 2200–2203.

[115] Liam Paninski. "Estimation of Entropy and Mutual Information". In: *Neural Computation* 15.6 (2003), pp. 1191–1253.

[116] Liam Paninski and Masanao Yajima. "Undersmoothed Kernel Entropy Estimators". In: *IEEE Transactions on Information Theory* 54.9 (2008), pp. 4384–4388.

[117] Antonis Papadogiannakis, Michalis Polychronakis, and Evangelos P. Markatos. "Improving the accuracy of network intrusion detection systems under load using selective packet discarding". In: *Proceedings of the Third European Workshop on System Security*. EUROSEC. 2010.

[118] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[119] C. Perkins and JM. Valin. *Guidelines for the Use of Variable Bit Rate Audio with Secure RTP*. RFC 6562 (Proposed Standard). Internet Engineering Task Force, Mar. 2012. URL: http://www.ietf.org/rfc/rfc6562.txt.

[120] Mike Perry. *A Critique of Website Traffic Fingerprinting Attacks*. Nov. 7, 2013. URL: https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks (visited on 12/11/2013).

[121] Mike Perry. *Experimental defense for website traffic fingerprinting*. Sept. 4, 2011. URL: https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting (visited on 02/16/2014).

[122] M.A. Pitt, L. Dilley, K. Johnson, S. Kiesling, W. Raymond, E. Hume, and E. Fosler-Lussier. *Buckeye Corpus of Conversational Speech (2nd release)*. www.buckeyecorpus.osu.edu, Columbus, OH: Department of Psychology, Ohio State University (Distributor). 2007.

[123] J Ross Quinlan. *C4.5: Programs for Machine Learning*. 1993.

[124] Adwait Ratnaparkhi. "A Maximum Entropy Model for Part-Of-Speech Tagging". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1996, pp. 133–142.

[125] William D. Raymond, Mark Pitt, Keith Johnson, Elizabeth Hume, Matthew Makashay, Robin Dautricourt, and Craig Hilts. "An analysis of transcription consistency in spontaneous speech from the Buckeye corpus". In: *Proceedings of the International Conference on Spoken Language Processing*. 2002.

[126] J D Rennie, L Shih, J Teevan, and D R Karger. “Tackling the poor assumptions of naive Bayes text classifiers”. In: *ICML* (2003).

[127] Shane Richmond and Christopher Williams. “Millions of internet users hit by massive Sony PlayStation data theft”. In: *The Telegraph* (Apr. 26, 2011). URL: http://www.telegraph.co.uk/technology/news/8475728/Millions-of-internet-users-hit-by-massive-Sony-PlayStation-data-theft.html (visited on 08/20/2015).

[128] Vassil Roussev and Simson L. Garfinkel. “File Fragment Classification - The Case for Specialized Approaches”. In: *IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering* (2009), pp. 3–14.

[129] A Saffari, C Leistner, J Santner, M Godec, and H Bischof. “On-line Random Forests”. In: *IEEE ICCV Workshop on On-line Learning for Computer Vision*. 2009.

[130] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. “Devices that tell on you: privacy trends in consumer ubiquitous computing”. In: *Proceedings of the USENIX Security Symposium*. 2007, pp. 1–16.

[131] *Selenium*. URL: http://seleniumhq.org.

[132] Adi Shamir and Nicko van Someren. “Playing “Hide and Seek” with Stored Keys”. In: *FC’99: 3rd International Conference on Financial Cryptography*. Ed. by Matthew Franklin. Vol. 1648. Lecture Notes in Computer Science. Springer, Feb. 1999, pp. 118–124.

[133] Yi Shi and Kanta Matsuura. “Fingerprinting Attack on the Tor Anonymity System”. In: *ICICS 09: 11th International Conference on Information and Communication Security*. Ed. by Sihan Qing, Chris J. Mitchell, and Guilin Wang. Vol. 5927. Lecture Notes in Computer Science. Springer, Dec. 2009, pp. 425–438.

[134] *SOCKS 4A: A Simple Extension to SOCKS 4 Protocol*. URL: http://www.openssh.org/txt/socks4a.protocol (visited on 02/19/2014).

[135] Robin Sommer. “Viable Network Intrusion Detection in High-Performance Environments”. Doktorarbeit. Technische Universität München, Munich, Germany, 2005.

[136] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. “Timing analysis of keystrokes and timing attacks on SSH”. In: *Proceedings of the USENIX Security Symposium*. 2001, pp. 25–25.

[137] Andreas Stolcke, Sachin S Kajarekar, and Luciana Ferrer. “Nonparametric feature normalization for SVM-based speaker verification”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008.

[138] A Stolerman, R Overdorf, and S Afroz. “Classify, but verify: Breaking the closed-world assumption in stylometric authorship attribution”. In: *IFIP Working Group 11.9 on Digital Forensics* (2013).

[139] Salvatore J. Stolfo, Ke Wang, and Wei-jen Li. *Fileprint analysis for Malware Detection*. Tech. rep. Columbia University, 2005.

[140] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. "Statistical Identification of Encrypted Web Browsing Traffic". In: *2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2002, pp. 19–30.

[141] S. Momina Tabish, M. Zubair Shafiq, and Muddassar Farooq. "Malware detection using statistical analysis of byte-level file content". In: *KDD Workshop on CyberSecurity and Intelligence Informatics*. 2009, pp. 23–31.

[142] Gautam Thatte, Urbashi Mitra, and John Heidemann. "Parametric Methods for Anomaly Detection in Aggregate Traffic". In: *ACM Transactions on Networking* (2011).

[143] *Tor Project: FAQ*. URL: https://www.torproject.org/docs/faq.html.en (visited on 02/19/2014).

[144] Brian Trammell, Elisa Boschi, Gregorio Procissi, Christian Callegari, Peter Dorfinger, and Dominik Schatzmann. "Identifying Skype Traffic in a Large-Scale Flow Data Repository". In: *Traffic Monitoring and Analysis*. Vol. 6613. Lecture Notes in Computer Science. 2011, pp. 72–85.

[145] Jean-Marc Valin. *The Speex Codec Manual*. 2007.

[146] Cor J. Veenman. "Statistical Disk Cluster Classification for File Carving". In: *Proceedings of the Third International Symposium on Information Assurance and Security*. 2007, pp. 393–398.

[147] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. Sixth. Pacific Grove, CA: Duxbury Press, 2002, pp. 371, 467–473.

[148] A. Wald. "Sequential Tests of Statistical Hypotheses". In: *The Annals of Mathematical Statistics* 16.2 (June 1945), pp. 117–186.

[149] A. Wald and J. Wolfowitz. "Optimum Character of the Sequential Probability Ratio Test". In: *Annals of Mathematical Statistics* 19.3 (1948), pp. 326–339.

[150] Ke Wang and Salvatore J. Stolfo. "Anomalous Payload-Based Network Intrusion Detection". In: *Recent Advances in Intrusion Detection*. 2004, pp. 203–222.

[151] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. "Effective Attacks and Provable Defenses for Website Fingerprinting". In: *Proceedings of the USENIX Security Symposium*. San Diego, CA, Aug. 2014.

[152] Tao Wang and Ian Goldberg. "Improved Website Fingerprinting on Tor". In: *Proc. ACM Workshop on Privacy in the Electronic Society (WPES)*. 2013.

[153] Michael Weber, Matthew Schmid, David Geyer, and Michael Schatz. "PEAT - A toolkit for detecting and analyzing malicious software". In: *Proceedings of the 18th Annual Computer Security Applications Conference*. 2002, pp. 423–431.

[154] Andrew M. White, Srinivas Krishnan, Michael Bailey, Fabian Monrose, and Phillip A. Porras. "Clear and Present Data: Opaque Traffic and its Security Implications for the Future". In: *ISOC Network and Distributed System Security Symposium – NDSS 2013*. The Internet Society, Feb. 2013.

[155] Andrew M. White, Austin R. Matthews, Kevin Z. Snow, and Fabian Monrose. "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks". In: *2011 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2011, pp. 3–18.

[156] Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monrose, and Gerald M. Masson. "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations". In: *2008 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2008, pp. 35–49.

[157] Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson. "Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob?" In: *Proceedings of the USENIX Security Symposium*. 2007.

[158] Charles V. Wright, Scott E. Coull, and Fabian Monrose. "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis". In: *ISOC Network and Distributed System Security Symposium – NDSS 2009*. The Internet Society, Feb. 2009.

[159] Charles V Wright, Fabian Monrose, and Gerald M Masson. "On inferring application protocol behaviors in encrypted network traffic". In: *The Journal of Machine Learning Research* 7 (2006), pp. 2745–2769.

[160] *XVFB*. URL: http://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml.

[161] Greg Young and John Pescatore. *Magic Quadrant for Network Intrusion Prevention Systems*. market research report. Gartner, Inc., Dec. 6, 2010.

[162] Like Zhang and Gregory B. White. "An Approach to Detect Executable Content for Anomaly Based Network Intrusion Detection". In: *IEEE International Parallel and Distributed Processing Symposium*. 2007, pp. 1–8.

[163] J. Zobel and P. Dart. "Finding Approximate Matches in Large Lexicons". In: *Software—Practice and Experience* 25.3 (1995), pp. 331–345.

[164] J. Zobel and P. Dart. *Fnetik: An Integrated System for Phonetic Matching*. RMIT, Technical Report 96-6. 1996.