LOW-LATENCY ANONYMITY SYSTEMS: STATISTICAL ATTACKS AND NEW APPLICATIONS

A Dissertation Presented

by

MARC D. LIBERATORE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2008

Computer Science

LOW-LATENCY ANONYMITY SYSTEMS: STATISTICAL ATTACKS AND NEW APPLICATIONS

A Dissertation Presented

by

MARC D. LIBERATORE

Approved as to style and content by:

Brian Neil Levine, Chair

Mark Corner, Member

Dennis Goeckel, Member

David Jensen, Member

Andrew Barto, Department Chair
Computer Science

ACKNOWLEDGMENTS

First and foremost, thanks are due to my advisor, Brian Levine, whose years of guidance, support, and friendship made this dissertation possible. I would also like to thank the members of my committee, Mark Corner, David Jensen, and Dennis Goeckel, for their advice and feedback. I owe David a particular debt of gratitude for helping me realize the benefits of an structured, empirical approach to security.

Thanks go to past and present members of the PRISMS group, and especially to George Bissias, Boris Margolin, and Matthew Wright, without whom some of the ideas in this dissertation would never have been developed. Matt's encouragement was much-needed toward the end of the writing process.

I also wish to thank my friends and family for their support and encouragement during the writing of this dissertation. My parents, Frederick and Jeannine, my good friend, David Clooney, and my fiancée, Katherine Kelly, all provided invaluable help, encouragement, and love, every step of the way.

ABSTRACT

LOW-LATENCY ANONYMITY SYSTEMS: STATISTICAL ATTACKS AND NEW APPLICATIONS

FEBRUARY 2008

MARC D. LIBERATORE

B.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Brian Neil Levine

In this dissertation, we study low-latency anonymity protocols and systems. Such systems enable anonymous communication where latency is not tolerated well, such as browsing the web, but also introduce new vulnerabilities not present in systems that hide timing information. We examine one such vulnerability, the profiling attack, as well as possible defenses to such an attack. We also examine the feasibility of using low-latency anonymity techniques to support a new application, Voice over IP (VoIP). First, we show that profiling attacks on low-latency anonymity systems are feasible. The attack we study is based upon pre-constructing profiles of communication, and identifying the sender of encrypted, anonymized traffic on the basis of these profiles. Second, we present results from a large-scale measurement study, and the application of this attack to the measured data. These results indicate that profiling is practical across sets of thousands of possible initiators, and that such profiles remain valid for

weeks at a time. Third, we evaluate defenses against the profiling attack and their effects upon system performance. We then demonstrate the feasibility of supporting anonymous VoIP; specifically, we show supporting measurement data and outline the changes current anonymity systems would require to carry such traffic. We also show how such systems are potentially more vulnerable to known attacks, and start to examine the tradeoffs between VoIP performance and anonymity inherent in such systems.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

INTRODUCTION

End-to-end communication through the Internet is not secure, private, or anonymous. Insecurities abound in and between all layers of the protocol stack. The presence of these insecurities is directly at odds with people's desire to communicate securely, privately, and anonymously.

Anonymous and pseudonymous communication protocols are particularly sensitive to these insecurities. Once the contents or even the existence of a communication is linked to an identity, the game is up. Whatever cost is associated with the loss of anonymity is now due from the user: A member of a minority political party might receive a visit from the local police, chilling their future political speech; an ill person researching their condition might be discriminated against; an abuse victim seeking help might be discovered; or a human rights worker might be deported or imprisoned.

People who want to communicate anonymously through the Internet thus require the use of specially designed software. Such software must be carefully designed and analyzed to avoid insecurities in the algorithms it implements. Moreover, both the environment in which the software runs and the data it transports must be considered. Insecurities in the environment abound: At the physical layer, wireless transmissions and unsupervised cables can be listened in on by third parties. Higher in the protocol stack, encryption is often optional or unspecified. The data itself can be the source of insecurity: Even when communication protocols specify encryption, side channels can leak sufficient information to allow observers to infer much about the information being communicated.

These leaks, and the costs of plugging them, are the subject of this dissertation. In particular, we set out to show that:

Low latency anonymity system are subject to failure in the face of side channel attacks.

These attacks can be mitigated, at some cost, by removing or obscuring the information present in such side channels.

Anonymity systems which support low latency, jitter sensitive applications such as telephony, are possible on the Internet.

To demonstrate these ideas, we show that current low latency anonymity systems are vulnerable to profiling attacks that are low in both computational and bandwidth cost. We describe several deterministic approaches to limiting the scope of these vulnerabilities, and examine their effectiveness. Finally, we present the design of an anonymity system that can support low latency, jitter-sensitive applications, and evaluate the tradeoffs present in such a system.

This dissertation is organized as follows:

Chapter 1 provides an overview of existing, low latency anonymity systems that are feasible with current and near-future communication systems, and corresponding attacker models. We evaluate these systems on the basis of latency and review their vulnerability to various attacks. We show that existing systems provide either low latency or strong anonymity and resistance to side channel attacks, but not both.

In Chapter 2, we examine the feasibility of inferring the source of encrypted HTTP streams on the basis of observable characteristics of the streams. In particular, we examine a profiling attack, attempting to discover the identity and contents of a web page that a user attempts to retrieve privately. In this attack, an observer builds a large library of profiles, each corresponding to a possible communication a user might engage in. We build profiles of packet streams corresponding to retrievals of web pages. These profiles are built on the basis of the size and interarrival times of the packets in the streams. By matching a user's communication with one of these profiles, we can predict with some accuracy the identity of the remote site.

This information leakage violates the responder anonymity property that most low latency anonymity systems seek to provide. On the other hand, it enables a new type of digital forensics. Much as viruses, malware, and other objectionable material are fingerprinted and cataloged in various digital libraries, the traffic generated by accessing a web site can be fingerprinted and cataloged. Such a catalog would have significant expected value to forensic investigators operating in the digital domain. On the basis of this idea, we further investigate profiling attacks in Chapter 3, and focus our attention on attacks that rely only on the size of the packets. We show that such attacks can be extremely effective. Further, we show that such profiles can be gathered in a fully decentralized manner, and that the profiles degrade slowly enough over time that profiling nearly all front pages on the web is quite feasible.

Chapter 4 describes several methods to disrupt the attack presented in Chapter 3. We examine in trace-driven simulation the effects of various packet padding techniques upon our profiling techniques. Unsurprisingly, increasing the amount of padding in packets renders them less distinguishable, lowering the attacker's accuracy at the cost of increasing bandwidth consumption.

Chapter 5 describes our investigation of the feasibility of an anonymous overlay network on the Internet supporting telephony, an extremely latency-sensitive application. We present the results of a measurement study conducted on PlanetLab. We find that providing such a service is feasible, with only marginally higher latency than is typically tolerated in traceable Internet telephony, and modestly increased risk of exposure as compared to traditional low-latency path selection algorithms.

CHAPTER 1

LOW LATENCY ANONYMITY SYSTEMS AND VULNERABILITIES

Users of the Internet desire secure communications. In this chapter, we examine what this implies in the context of the Internet, with an emphasis on the specifics of various forms of anonymity. We survey various theoretical and real-world communication systems that attempt to provide some level of anonymity to their participants. We describe several ways in which these systems can be compromised, and present assessments of existing systems in light of these attack models. This material will serve as a reference for the remainder of this dissertation.

Communication security on the Internet has several components. The most important of these are confidentiality, integrity, and access.

Confidentiality is commonly defined as a guarantee that information is only accessible to parties authorized to have access to the information. Implicit in this definition is the concept that there are at least two parties involved in the communication, and the opportunity for third party observation exists. This situation is nearly universal on the Internet, where communications routinely pass along untrusted paths. Typically, confidentiality is preserved through some form of keyed encryption, rendering its contents unreadable for all but the possessors of the decryption key. Confidentiality of the contents of message, the parties involved, and even the existence of the message are at the core of most anonymity systems. We will return to these concerns in greater detail shortly.

Integrity is a property guaranteeing that information cannot be changed without the changes being detected by authorized parties. A simple message checksum will not

suffice to provide this property, as a third party could alter the checksum to correspond to the altered message. Depending upon other parameters of the communication, various options such as digital signatures or hashed message authentication codes (HMACs) may be used. In an anonymity system, integrity of the communication may mean more than integrity of the data: disruption of the timing of the communication may be used by a third party to transmit information, as we shall see later in this chapter.

Access is a property guaranteeing that parties can access information if and only if they are authorized to do so. Various forms of authentication, based on passwords, shared keys, or biometric information, often serve to prove a user's identity. Such authentication systems are obviously antithetical to anonymity systems. There is another facet to access that is vital to anonymity systems: availability. If a third party is able to deny the user of an anonymity system access to some or all of the system, then the strength of the system is reduced, sometimes drastically. We will see this effect later in this chapter.

1.1 Anonymity

There are many parties who would prefer their communications on the Internet to remain anonymous. Journalists, human rights activists, and political dissidents are all examples of users that might require anonymity on the Internet, and who are generally regarded as being operating ethically if they do so. On the other, an opponent of anonymous communication technology might cite terrorists or pedophiles as reasons to block the development and deployment of such technology. Regardless of the users and their intentions, anonymous communication technology exists. The American Academy for the Advancement of Science states that "anonymous communication online is morally neutral," that it "should be regarded as astrong human right; [and] in the United States it is also a constitutional right." [52]

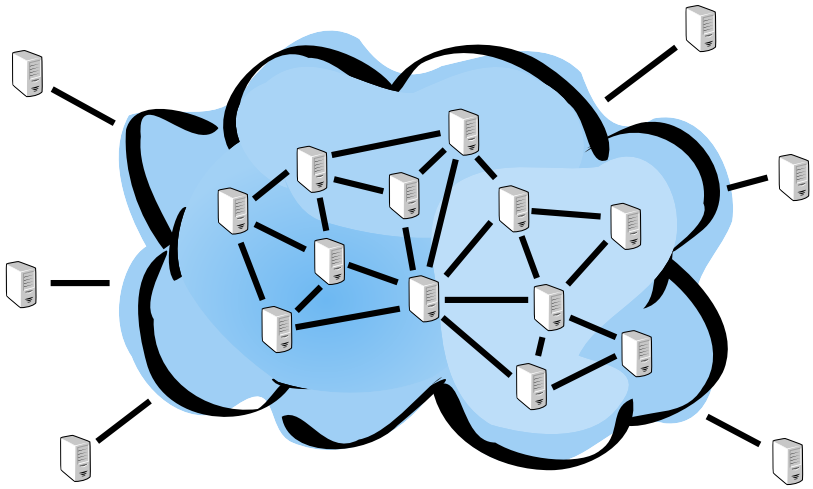

Figure 1.1. A hypothetical anonymity cloud. Initiators and responders make connections to nodes within the cloud, who pass messages among themselves before relaying them to their final destination. Connections between nodes are encrypted with ephemeral keys, and messages may be delayed or altered along the way, obstructing an observer's attempts to correlate incoming and outgoing messages at each node.

But what does it mean to be anonymous? Informally, being anonymous is being indistinguishable from a crowd. An observer might know that a member of the crowd yelled "Fire!" but not know who. Intuitively, anonymity requires such a crowd. When implementing anonymous communication systems on the Internet, the crowd in question is a large number of proxy servers, typically run by those wishing to form such a crowd. Figure 1.1 shows such a crowd.

Several properties of increasing strength or degree are often assumed when referring to anonymous communcations. Here we make them explicit, using Pfitzmann and Hansen's proposal for anonymity terminology [40], and expanding upon their definitions as appropriate:

- **Anonymity** is the state of being indistinguishable from a set of possible subjects, the anonymity set. This property is sometimes parameterized as k-anonymity, where k is the cardinality of the anonymity set.

A more discriminating measure of anonymity is probabilistic: an observer can estimate the probability that two entities are tied: for example, the probability that a particular message originated from a subject of interest In a perfect anonymity system, this probability is equal to $1/k$; the goal of an attacker is to drive this probability to 1 through various attacks and observations. Díaz et al. [14] and Serjantov and Danezis [47] independently noted that this process is exactly analogous to information gain, and proposed defining the effective size of an anonymity probability distribution as follows.

Given a model of the attacker and a finite set of all users Ψ, let $r \in \mathcal{R}$ be a role for the user ($\mathcal{R} = \{sender, recipient\}$) with respect to a message $\mathcal{M}$. Let $\mathcal{U}$ be the attacker's *a posteriori* probability distribution of users $u \in \Psi$ having the role r with respect to $\mathcal{M}$. The effective size $\mathcal{S}$ of an r anonymity probability distribution $\mathcal{U}$ is defined as the entropy of the distribution:

$$\mathcal{S} = -\sum_{u \in \Psi} p_u \log_2(p_u) \tag{1.1}$$

$\mathcal{S}$ can be interpreted then as the number of bits of information necessary to identify user u with role r for a particular message $\mathcal{M}$. We can state that new information corresponds exactly to changes in the probabilities in $\mathcal{U}$. Observations generally lead to increases in information, and increased certainty as to the identity of participants in the anonymity system. Deliberate misinformation, such as cover traffic, can reduce an observer's confidence in their estimates.

- **Unlinkability** is a property of an anonymity system, where an observer gains no information regarding two or more entities in an anonymity system. These entities can be messages, initiators, recipients, paths, or any other item in the system, and the unlinkability is with respect to these entities. To provide this

property, a system must generally be allowed to delay messages arbitrarily, or inject cover traffic.

- **Undetectability/Unobservability** is an even stronger property than unlinkability. Unobservability is a property of an anonymity system where observers cannot even determine if the entity or transaction of interest exists. To provide this property, a system and all participants must be able to inject arbitrary amounts of cover traffic and delay into the communications.

The above properties are used in the description of anonymity systems, though unlinkability and unobservability are not typical properties of real-world systems. As we discuss later in this chapter, these properties are hard to design for in systems that are of general interest on the Internet. For the remainder of this text, when we refer to "anonymity" without other qualification, we are using it in the sense of probabilistic anonymity above, where we expect an observer to be able to improve their estimates of probabilities on the basis of their observations.

In addition to the degree of anonymity offered, systems provide anonymity to different entities:

- **Initiators** are entities in the system that originate communications, for example, an end user making a web request or sending an email. A system providing initiator anonymity attempts to prevent an observer of a set of initiators from determining which of the initiators in involved in a communication, while not necessarily anonymizing the responder.

- **Responders** are entities in the system that respond to initiator requests, for example, a web site replying to a request. A system providing responder anonymity is complementary to one providing initiator anonymiyt: the responder to a request is anonymized, while not necessarily anonymizing the initiator.

- **Mutual** anonymity is provided when a system attempts to provide both initiator and responder anonymity.

- **Network** or total anonymity is even stronger than mutual anonymity. It is provided when a system attempts to prevent an observer from determining not just the endpoint, but the intermediate parties in an anonymous transaction. Typically this cannot be accomplished without total cover traffic.

Finally, there is one other property of interest in anonymous communication systems: low latency. An implicit property of communication systems is that that they be timely. In some sense, this is at odds with perfect anonymity.

Consider an example mutual anonymity system: users put write messages on paper, and place them in an envelope addressed to the recipient. The envelopes are placed in a box. The box is opened once, just before then end of the universe, and messages are handed out to the recipients. There is no way to identify who sent which message by observing the set of recipients, (discounting side-channel attacks like dusting for fingerprints). If the box is emptied earlier, such as once halfway until the end of the universe, and once at the end, we can divide the senders into two groups: those that sent before the halfway point, and those that sent after. Extending the example to the worst case, if the mailbox is opened after each letter is deposited, there is no anonymity — though recipients don't need to wait until the end of the universe to receive their mail. This tradeoff between mixing of messages and latency is a fundamental tension present in all real-world anonymity systems.

1.2 Anonymity Systems

Chaum's seminal paper [8] proposed the use of a chain of traffic-shaping mixes to hide the correspondence between inputs and outputs of a mail system, and the use of public key cryptography to provide disposable pseudonyms for users of the system.

All anonymity systems resemble Chaum's proposal, though they differ in their detail. Such systems provide some degree of anonymity through four fundamental functions: encryption, mixing, proxying, and traffic shaping.

Encryption serves to hide the contents of messages from observers. If a message were unencrypted, an observer could trivially link initiators and responders by watching for identical message within the anonymity cloud. Unless otherwise noted, all of the anonymous communication systems described in this section employ link encryption between directly connected nodes in the system. Link encryption typically adds a small, fixed CPU cost at each node. Encryption can also be used to prevent portions of the anonymity system from knowing the contents of messages; such encryption forms the basis for onion routing, detailed later in this section.

Mixing of messages occurs when a node in an anonymity system waits to send messages until a system-specific interval has elapsed. When the time arrives, the messages that have arrived thus far are sent out in a randomized order. If the messages have no identifying characteristics other than their order of arrival, then they are effectively mixed: An attacker cannot link incoming and outgoing messages. As mentioned in the previous section, the longer the duration of the interval, the less linkage there is between messages, at the cost of overall time it takes for each message to arrive at its destination.

Proxy nodes relay messages on behalf of other nodes in the system. In many anonymity systems, participants form a path of several proxies, and send their messages along this path. The purpose of such proxying is to obscure the relationship between the initiator and responder. An observer watching a set of nodes that does not include the initiator, responder, and all proxies involved in the communication cannot link the initiator and responder with certainty. Thus, more proxies make the observer's task more difficult, though correlating traffic flows observed at different points in the network is feasible. Adding proxies to a path increases the number

of nodes an adversary must observe to tie initiators and responders, at the cost of increasing message delivery times. This cost is due to longer network paths and the additional mixing delays that some anonymity systems impose. Longer path lengths have other effects on the overall performance of the system, which we examine in Section 1.3.

A secondary purpose of proxies is to distribute trust, and thus risk of exposure, throughout the anonymity cloud. If it is possible for individual nodes to be compromised — as is the case in any publicly accessible system — then having only one proxy necessitates trusting that proxy completely. If such a single proxy is under the control of an observer, then anonymity is lost. A chain of proxies distributes this trust and makes compromising nodes a less fruitful endeavor for the attacker.

Traffic shaping, conceptually a superset of mixing, serves to hinder an observer from linking messages observed at one point in a system from messages observed in another. One form of traffic shaping is to force all messages to be of a system-determined size or sizes, padding small messages and fragmenting larger ones. An observer attempting to correlate messages on the basis of their sizes is thus stymied. Another form of traffic shaping involves modifying the interarrival times of messages within the system. Messages can be delayed and dummy messages can be sent in an attempt to prevent observers from correlating streams of messages on the basis of their interarrival times. A key observation about traffic shaping: in the face of an adversary watching sets of possible endpoints, only traffic shaping is sufficient to prevent linking the actual endpoints. Further, such traffic shaping will increase end-to-end latency. Thus, there is a fundamental tension between low-latency anonymity and resistance to such end-to-end correlation attacks.

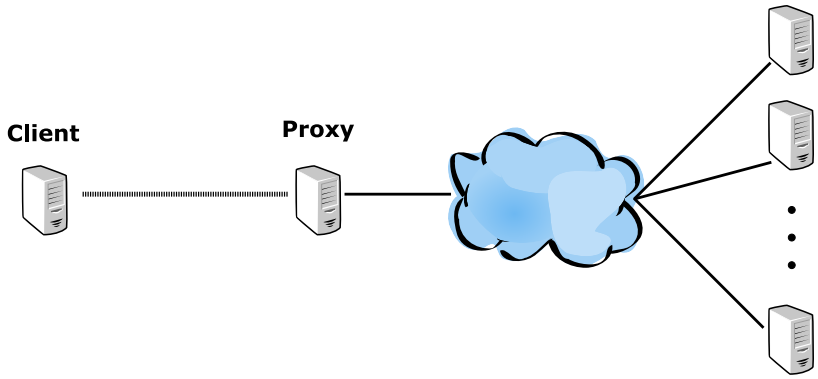

Figure 1.2. A single-proxy anonymity system. The initiator (or client) connects to the proxy, which in turn connects to services on behalf of the client.

1.2.1 Single Proxy Systems

The simplest way to implement the four functions of anonymity systems is to do so at a single point, in the client-server model. This system abstraction, which is known as a relay server or mix in the case of high latency systems [8] and as a proxy in low latency systems, works as follows (shown in Figure 1.2): It implements encryption when communicating with clients, thus preventing simple traffic sniffing from discerning the contents of the communication. Due to its location in network, it prevents the responders from identifying the initiators through identifying routing information in packet headers. Mixing further obfuscates these links in the case of an observer watching the network links at the proxy itself. Further, traffic shaping prevents statistical analysis from matching traffic flows that aren't delayed by mixing.

Such a single proxy system, implementing all of the above techniques, is still not a desirable system, as it has a single point of failure. The motivated attacker will shift from attempting to compromise the protocol to attempting to compromise the node itself, as we show in Section 1.3. Still, there were and are several systems that fit this description. We describe several below.

Not all anonymity systems are constructed with full attention to possible adversaries. A deployed and highly-used system in the early nineties, the Penet re-

mailer [25], functioned only as a proxy. It stripped identifying information from the mails and substituted a pseudonym before remailing to allow for replies through the system. It failed to provide strong anonymity in a variety of ways, as an adversary watching communication links could have easily linked initiators and recipients. Ultimately, though, Penet was shut down due to vulnerability to a legal attack: A subpoena from the Church of Scientology gained access to the file mapping email addresses to pseudonyms. The operator of Penet, realizing that the design of the system could not withstand such attacks, shut it down. Newer anonymous remailers, described later in this section, were designed with more care and are resistant to such attacks.

Various commercial systems exist and claim to provide anonymity of one level or another. `Findnot.com`, Anonymizer, `Anonymization.net`, and Zero Knowledge Systems, among many others, have offered some form of anonymity service, as have many volunteer systems. Their services generally take the form of a proxy that is under control of the company in question which relays traffic for a customer. As with Penet, these systems generally do not have strong resistance to an adversary that can watch their network connections, nor to out-of-band information discovery, such as through the legal system. From a scientific perspective, they are of interest only as examples of extremely weak anonymity.

1.2.2 Mixnets

As noted above, Chaum proposed the use of public key cryptography and a series of nodes between sender and recipient, deemed mixes, to hide the relationships between senders and recipients. The basic mechanism is as follows. Assume I wishes to send a message to R via a mix A. The following messages are sent:

$$I \rightarrow A : K_A^+\{r_1, R, K_R^+\{r_2, \text{data}\}\}$$
$$A \rightarrow R : K_R^+\{r_2, \text{data}\}$$

where K_X^+ denotes encryption with the public key of X, and r_i is a random string, to prevent replays, traffic analysis, and the like. Not shown in the protocol is a vital step: waiting for several other messages to arrive, and then resending all messages in some order independent of their arrival order. In this fashion, the messages are "mixed," and their originator obscured. Obviously, enough messages must be passing through the system from independent sources to perform this mixing. This batch processing of messages renders them unlinkable, so long as they have no other identifying characteristics.

Chaum also proposed several extensions to this basic system. A network of mixes can be used, where the original message is wrapped in a series of public key encryptions. This idea of mixnets is the basis of onion routing (see Section 1.2.3). It has the advantages of increasing the mixing across the anonymity system and making traceback and traffic analysis harder. In particular, compromising a single node in a single-node system allows an observer to tie all inputs to all outputs and thus break anonymity; compromising a single node in a multiple node system allows ties of inputs and outputs, but at least one side of the equation will be to another mix, and thus in theory not break anonymity. A variety of applications of this basic technique exists [1, 24, 27, 28, 37, 38]; the details of each are not relevant to the problems of low-latency anonymity and we do not discuss them further here.

Several mail systems descended from the original mixnets were actually implemented, following the failure of the `anon.penet.fi` single proxy system. First were the systems retrospectively deemed Type I remailers, implemented independently by Hughes [39] and Finney. These systems functioned as simple mail relays, wrapping mail and headers in a straightforward fashion with public keys. Later, Cottrell im-

plemented the Mixminion system [11, 33], also known as the Type II remailer. This system implemented several improvements that improved overall anonymity and resistance against attacks. In particular, it included message padding, which helped hide relationships among messages and their progress through a series of mixes; and message pools, which are buffers of messages in the mix, designed to mitigate the effects of an adversary flooding mixes with dummy messages so as to pick out the destination of the few non-dummy messages; along with a set of other minor changes intended to make traffic more uniform and the system less abusable. Unfortunately, the design of Mixmaster did not include support for replying to anonymous senders, or for anonymous recipients.

The development of Internet remailers culminated in the Type III remailer Mixminion, developed by Danezis, et al. [13]. Mixminion supports replies through the network to anonymous senders, anonymous recipients, replay prevention and key rotation, and support for user-specificied dummy traffic. Further, its design presaged many aspects of the design of later low-latency systems (such as Tor, described in Section 1.2.3, including link encryption, application layer hop-by-hop path construction for perfect forward secrecy, control of exit policies so as to address complaints of abuse from third parties, and integrated directory servers to prevent partitioning attacks. The design of the Type III remailer is extremely secure; as described in the design document, Mixminion is highly resistant to virtually all known attacks on anonymity systems, and fails only in the presence of a extremely powerful adversary. This robustness to attack is due to several factors; most relevant are the use of arbitrary delays, padding, and path length. Combined, these make practical use of each of the attacks described in Section 1.3 virtually impossible in isolation, and extremely difficult in concert. The cost, of course, is potentially arbitrary network delay between initiator and responder.

More recent developments remain largely theoretical. Re-encryption mixnets [20, 37, 4] to build mixnets with subtly different properties from public-key cryptography,

culminating in mixnets where nodes do not require key generation, distribution, or management[23]; we do not describe the differences here as they are not relevant to low-latency networks.

1.2.3 Onion Routing

Onion routing, like Crowds, is designed to enable low-latency anonymity on the Internet. Onion routing was originally proposed by Goldschlag, et al. [22] as a way of hiding routing information from the participants in a communication network. Hiding this routing information is essential in preventing traffic analysis and providing a form of unlinkability for messages observed in the middle of the anonymity cloud.

Onion routing systems are an attempt to hybridize proxy-based and mix-based systems, and provide the best of both worlds. Onion routing presupposes a set of proxies willing to forward traffic for others. Each such proxy is accessible to all other proxies, and has a widely-known public key. At a specified time (for example, user request or a timer), a proxy will choose a path consisting of nodes in the network. Layered public-key encryption hides everything but the routing information from each node along the path. For example, consider an initiator I sending a message through nodes A, B, and C to responder R. A sketch of the protocol looks like the following:

$$
\begin{aligned}
I \rightarrow A &: K_A^+\{B, K_B^+\{C, K_C^+\{R, K_R^+\{\text{data}, \}\}\}\} \\
A \rightarrow B &: K_B^+\{C, K_C^+\{R, K_R^+\{\text{data}, \}\}\} \\
B \rightarrow C &: K_C^+\{R, K_R^+\{\text{data}, \}\} \\
C \rightarrow R &: K_R^+\{\text{data}\}
\end{aligned}
$$

where K_X^+ denotes a public key encryption using the public key of X. As shown, each node along the path removes a layer of encryption and learns where next to send the message. The layering, and the "peeling" that each node must perform, gives onion routing its name. As a result of the layered encryption, each node learns only the

where the message came from and where it is going, but nothing about the contents of the message and only limited information about the path.

The above protocol sketch elides certain details, such as the specifics of path construction, public key distribution, constructing return paths. It also has certain weaknesses: The use of expensive public-key operations at each node, possibly on a per-packet basis, imposes relatively high computational overhead and delay. Nonuniform packet sizes, as a result of the path information being included in each packet, might leak information about the paths.

Reed, et al. [44] provide a specification for a functional system that works at the application layer and that addresses many of the unspecified details above. This system, known now as the first generation onion router, still suffered from some weaknesses in its design, notably the use of public key cryptography at each node and implementation of application-layer proxying. Syverson, et al. [51] presented a security analysis of the system against varied attacker models, and note that the adversary does best when controlling the two endpoints of the route, an event which happens with probability proportional to c^2/n^2 in a system where routes are chosen randomly, where c is the number of compromised nodes and n is the total number of nodes in a system.

The second-generation onion router [16], described by Dingledine et al., addresses many of these issues. The design was modified from the original routing to set up circuits among nodes, rather than perform per-packet switching. These circuits reflect the path of data through the network, and incur some cost as they introduce state to the network. The benefit is that while circuit setup is performed by public-key cryptography, the layered encryption is based on symmetric keys. All traffic sent along circuits is split into 512-byte packets, called "cells," and encrypted with layers of ephemeral symmetric keys, where each key is shared only between the initiator and the node it is created for. Circuits are created in a hop-by-hop fashion, so an

observer of the initiator cannot simply watch to see with whom it exchanges key material. Nodes in a circuit decrypt each cell with the symmetric key corresponding to its circuit. If the decrypted cell can be parsed, then it is acted upon – for example, route extension or teardown messages are acted upon, or packets are sent to the public Internet. If the cell cannot be parsed, it is assumed that it is further encrypted, and it is forwarded to the next node in the circuit. Further, a detailed, byte-level specification of the system was released. This specification allowed for formal proofs of correctness of various parts of the protocol to be made, such as by Goldberg [21]. Finally, Tor has been implemented and is in large-scale use, which exposes problems with the protocol and its implementation, as well as spurring further research interest in the problem of low-latency anonymity – such as this dissertation. We survey some of the known attacks against Tor in Section 1.3.

In addition to Tor, there are several other systems that specify or implement aspects of onion routing. MorphMix [46], Cebolla [5], and P2P Anonymization (formerly known as Tarzan) [19] build circuits in a fashion similar to Tor, though use peer-to-peer techniques to decentralize the list of available nodes. JAP[1] uses fixed, shared routes in an attempt to increase the size of the anonymity set, and padding to help make the traffic indistinguishable.

1.2.4 Crowds and Hordes

Reiter and Rubin's Crowds system [45] was an early implementation of a low-latency anonymity system. The system attempts to keep latency low by mixing in realtime (that is, by not delaying each messge), and by forming paths probabilistically at regular intervals. The Crowds system as implemented functions as an HTTP proxy, but the basic design is generalizable to other connection-oriented protocols. The system works as follows:

---

[1] `http://anon.inf.tu-dresden.de/index_en.html`, retrieved 05 December 2006

Once per interval, a group of n nodes (a so-called crowd) forms a chain of proxies to hide the initiator from the responder. Each node that wishes to communicate sends an initialization message to another node, chosen at random, within the crowd. This chosen node is the first proxy in the chain. With some probability p_f, the chosen node will repeat this action, extending the proxy chain by one hop. This process repeats at each such chosen node until a node does not choose to extend the path. This node becomes the exit node for the given path. All communication along the paths is encrypted with ephemeral keys. These paths persist until the end of the interval, whereupon any new node wishing to join the system is added to the crowd.

Through this algorithm, Crowds attempts to protect against several threats to anonymity. In the case of a local eavesdropper, the proxy system provides the same level of anonymity as in the single-proxy case. Similarly, a malicious responder or eavesdropper local to the responder is unable to identify the initiator. The randomness of the path formation, both in choosing the next hop and in its overall length, provides probabilistic protection against c collaborating malicious nodes. Malicious nodes are generally not sure if a request originates from the node preceding them or not due to this randomness. When the condition $n \geq \frac{p_f}{p_f - 1/2}(c+1)$ holds and $n \to \infty$, the initiator remains "probably innocent," in Reiter and Rubin's formulation – that is, the probability that the originator under suspicion is the true initiator is less than or equal to 0.5.

This probabilistic anonymity is guaranteed by mathematical proof, in the face of a single round of path formation. As we discuss in Section 1.3, there are attacks across rounds of path formation that will compromise Crowds. The general form of this attack, known as the predecessor attack, is in fact a danger to all anonymity protocols that rely on a typical mix structure and that reform paths between such mixes periodically.

Levine and Shields developed a successor to Crowds, known as Hordes [30]. Hordes is similar in design to Crowds, but it relies on IP multicast to send messages to the initiator from the responder. There are two other key differences between the Crowds and Hordes.: First, each node in Hordes uses a small random subset of other nodes to forward messages. The size subset is chosen such that a Horde is no more vulnerable to the predecessor attack than a Crowd of equivalent size.[2] Second, Hordes must use UDP for messages from the responder to the originator, as Internet multicast does not support standard TCP connections. This restriction is a non-trivial for an implementor, as it implies that connections will be likely be highly asymmetric, in terms of the path length, nodes along the path, loss rate, and latency. The theory of reliable and congestion-controlled transmission protocols for this situation is not well understood. Finally, Hordes suffers from the weakness that Internet multicast is not widely deployed, though that may change with the widespread adoption of IPv6.

1.3 Attacks and Attacker Models

The purpose of anonymous communication systems is to hide relationships among participants and their messages. But from whom are these relationships hidden, and what sorts of attacks are undertaken to reveal them? In this section, we examine the who and the what of attacks on anonymity systems, with a focus on attacks that are relevant to low-latency systems. We first describe the most commonly assumed attacker models, which describe the powers and limitations of an adversary (or set of adversaries) that may be attempting to compromise the anonymity of a system. We then survey specific known attacks on anonymity systems.

There are a variety of ways in which an attacker might attempt to compromise an anonymity system. These various methods can be broadly characterized as passive

[2]This limited size set is a precursor to the more general notion of entry nodes, first proposed by Wright, et al. [55].

or active. A passive adversary is characterized as being limited to observing network traffic, but having no information about encryption keys, message contents, paths, etc., but for what can be inferred from its observations. An active adversary is credited with the powers of a passive adversary, but additionally with the power to modify network traffic in various ways.

1.3.1 Passive Adversaries

Passive adversaries have existed for as long as communication systems have attempted to provide security properties to their users. Intuitively, a passive adversary is an entity with the power to observe a system, but not to change it. Examples include eavesdropping on an isolated meeting, perhaps through use of non-obvious technology such as shotgun microphones; radio-based triangulation of transmitters, pioneered in Europe during World War II to capture spies and locate shipping vessels; and tapping telephone.

The last example is analogous to the typically assumed passive adversary. In the simplest case, a passive adversary can observe a single link in an anonymity system. Virtually all such systems, even the single-proxy system, were until recently believed to be largely secure against such an attacker so long as the contents of the link were encrypted. A main contribution of this dissertation is to show that this belief was incorrect: Chapters 2 and 3 demonstrate the feasibility and limits of an attack that a one-link passive observer can mount. This adversary is approximately the weakest that is typically considered; all other models are supersets of this one.

On the other end of the spectrum is an observer able to watch all such links in a system. Such an observer is known as a global passive adversary, and is remarkably powerful, despite its reliance upon inference based on traffic patterns. As we describe in Section 1.3.3, the global passive adversary is known to be able to compromise

low-latency anonymity systems in short periods of time, and be a threat to many high-latency systems as well.

In between these two models are anonymity systems with varying levels of saturation of passive observers. Serjantov and Murdoch [48] describe to so-called partial passive adversary, where an observer only sees some of the links among mixes. They also describe the adaptive adversary, that is, an adversary that can move its limited number of observation points over time. Their work implies that a partial adversary's power approaches that of a global adversary as paths are re-formed in a mix network.

1.3.2 Active Adversaries

Active adversaries, as the name implies, are more powerful than their passive counterparts. In particular, active adversaries are modeled as having the ability to actively manipulate an anonymity system in various ways. We can divide active adversaries into two classes: network-level and node-level. Network-level adversaries are capable of various forms of traffic modification, while node-level adversaries actually control nodes in a system and have correspondingly greater power.

Network-level active adversaries are directly analogous to passive adversaries: we consider them to have access to one or more communication links in an anonymity system, which they can observe freely. In addition to observing these links, an active adversary may delay, drop, or modify traffic on the link, as well as injecting traffic of their own. Delays and drops in particular are not easy for a network to compensate for, and result in trivial denial-of-service attacks. In this sense, even a single-link active adversary can be devastating for a user of an anonymity system, if it is on the right link. A network-level active adversary can also inject packets into the network. Even if such packets are not correctly-formed and are immediately dropped by the attacker, they may result in delays in processing other packets. Particularly in low-

latency systems, this will increase the amount of information available to the attacker, as we discuss in Section 1.3.3.

Finally, we turn to the concept of a node-level adversary. Such an adversary exists when a node in a system is under the control of an adversary. When this occurs, the adversary is assumed to control all communication links leading into our out of the node, to have access to all keying material on the node, and to be able to selectively follow or disregard the anonymous communication protocol the node is expected to be executing. We note that such nodes are supposed to come into being in one of two fashions. The nodes can be compromised by an attacker: computers connected to the Internet and inadequately protected are routinely compromised; the SANS institute's Internet Storm Center[3] estimates that unpatched Windows systems are compromised in about 20 minutes on average. A dedicated adversary could script a host of known attacks against a node, or attempt to discover new attacks to compromise such a node.

Alternatively, an adversary could circumvent the prospect of compromising a node, and just run their own. Many of the recent anonymity systems described in Section 1.2 are distributed, and not typically deployed under a single administrative domain, for reasons discussed in Section 1.3.4. As a result, there is no guarantee that nodes in an anonymity system aren't malicious from the start. At worst, every node along the path chosen by an initiator could be compromised before the communication even begins. This potentially disastrous scenario is at the root of many of the attacks describe below.

Finally, we note there is another class of adversary, which exists in a dimension orthogonal to those listed above: the external, typically legal adversary. Through force of law (or just through force), an adversary may attempt to act as one or more

[3] `http://isc.sans.org/`

of the adversaries described above, without regard for the technical and algorithmic strength of the anonymity system. The Church of Scientology functioned in this fashion against the `anon.penet.fi` remailer. There are some ways to mitigate such an adversary: removal of old logs, ephemeral keys, and the like. Regardless, such an adversary cannot be defended against through solely technical means. As mentioned above, spreading nodes across administrative and legal domains helps to mitigate this class of adversary.

1.3.3 Traffic analysis

Traffic analysis is the standard approach to compromising anonymity systems, and with good reason. In the absence of total, uniform, constant bitrate cover traffic and static membership among the anonymity set, an anonymous communication system leaks information when messages are sent through it. Traffic analysis is the name for a family of techniques that attempt to extract and infer this information as efficiently as possible given a set of observations.

Several authors, including Chaum [8], Pfitzmann and Waidner [42], Pfitzmann et al. [41], Rackoff and Simon [43], noted early on that such attacks were feasible and worthy of consideration when designing anonymous communication systems. Pfitzmann and Waidner in particular designed a system, ISDN-Mixes, that was largely immune to most of the traffic analysis attacks later described, though infeasible to deploy on the Internet due to reliance on features found the ISDN (Integrated Services Digital Network) medium.

The classic traffic analysis attack is the so-called "Brute Force" attack. Upon observing a message from a sender, the observer traces it to the next node, and waits for that node to resend the message. The details vary depending upon the protocol in use, but in general, there might be t_1 other nodes that will receive a message, of which on is the actual message ($t_i \geq 1$, where $t_i = 1$ implies only the actual message

was re-sent). These t_1 nodes arise due either to the batching that mixes perform or to dummy messages. At the second layer of such nodes, the path is traced again to t_2 nodes, and so on, along a path of length p. Eventually, the observer knows $n = \prod_{i=1}^{p} t_i$ possible nodes that could be the actual recipient of the message. As more messages are sent, paths are reformed, nodes leave the system, etc., an attacker can increase their certainty in their estimate of the probability of two particular nodes communicating. This is the basis of the intersection attack, where the intersection of sets of possible recipients at different times eventually results in the identification of the recipient.

There are mechanisms to improve this attack. Marking the messages makes traffic analysis trivial, but is equally trivial to defend against. More subtly, an adversary can flood a mix node with messages, attempting to force the mix node to "flush," that is, resend all its messages. If the adversary knows which messages are its own, it can drive t down toward 1. Chaum acknowledged this problem; the design of Mixmaster includes messages pools to defend against it, and Kesdogan, et al. [28] provide an analysis of the probability of adversary success in Stop-and-Go Mixes, which wait a random duration before releasing messages from a mix. This work provided the first proof that continuing communication between an initiator and responder would be de-anonymized after a sufficient number of observations and path changes occurred.

As described in Section 1.1, Díaz et al. [14] and Serjantov and Danezis [47], this problem can be formally modeled, and the amount of information leakage measured. Danezis later did just that [12], and Mathewson and Dingledine [32] explored the attack in simulation; the latter work assumed the use of the intersection attack. Murdoch and Danezis [35] have also shown the efficacy of a similar attack on Tor, where the adversary floods dummy traffic through a node suspected to be along a path of interest, and measures the higher latencies that result. An observed sudden

increase in latencies along the path refines the brute force attack, and makes it quite practical.

Low-latency anonymity networks on the Internet of the foreseeable future cannot hope to resist global adversaries, for two reasons. First is the inability to delay traffic, and second is the inability to have constant bitrate, constant packet rate cover traffic. Thus any global adversary can, with near certainty, track packet flows as they cross the anonymity cloud. On the other hand, the global nature of the Internet helps make it possible for paths through this logical cloud to cross geographical and political barriers, making it less likely that one real-world entity can truly observe all the traffic on any given path. There are several attacks that are still feasible for more limited attackers, so long as potential endpoints of chains of node (or even hops along long-lived paths) can be observed.

The first is a family of straightforward packet timing and correlation attacks. In essence, the attacker watches the initiator and some set of possible responders, treating the anonymity cloud as a black box. When there is a correlation between traffic seen at the initiator and responder(s), the attacker updates his estimate that the two nodes are communicating. For example, attacker could watch for the start of connections, as suggested by Serjantov and Sewell [49], or watch for correlations between interarrival times of sequences of packets, as suggested by Levine, et al. [29]. When such correlations occur with in a short time of one another, the attacker can conclude with high probability that then connections are part of the same traffic stream. More traffic between the pair, and associated correlations, leads to higher attacker certainty of correctness.

A second attack that is feasible for limited adversaries attacking low-latency systems is the intersection attack. This attack, which is also effective against high-latency systems, works as follows. Using some mechanism, the adversary watches a message from an initiator and finds all possible recipients. The specific mechanism is irrel-

evant; brute force tracing, correlations, or packet marking would all suffice. The attacker then has a set of possible recipients. Some time later, after something has changed in the system – typically path reformation as the result of nodes joining and leaving – the attack is repeated. For a given initiator, the set of possible responders likely changes. If the same responder is being communicated with, the intersection of the two sets contains the responder, and likely is smaller than either of the original sets. This process is repeated until sufficiently few possible responders remain for the adversary to take the desired action.

The third such attack, introduced by Wright, et al. [54] and later analyzed in more detail [56] is known as the predecessor attack. This attack, which is a generalization of an attack first introduced by Reiter and Rubin in their work on Crowds, is based on the simple observation that the initiator is always on the path. The attacker assumes that repeated communications occur between the initiator and responder, and that such communication occur across path reformations. Finally, the attacker must be able to differentiate messages from separate communications. This condition implies the attacker must have compromised the node where observation is taking place, as described in Section 1.3.4. If these assumptions hold, then the attack knows that of all possible parties likely to be sending the message, the initiator is slightly more likely under realistic circumstances (that is, paths that contain nodes that are small subsets of the set of all nodes). Over time, the attacker can degrade the anonymity of participants by refining their estimates that various nodes are communicating. Wright, et al. derived theoretical bounds on this degradation in terms of the number of path reformations, shown in Table 1.1, and validated these bounds in simulation.

1.3.4 Traceback and Compromise Attacks

Most recent anonymous communication systems, such as Tor, MorphMix, and Tarzan, are somewhat distributed or peer-to-peer in design, and do not strongly

Protocol	**Rounds to attack, with high probability**	**Rounds to attack, expectation**	**Work required of participants**	**Latency from I to R**
Crowds [45]	$O\left(\frac{n}{c}\log n\right)$	$O\left(\frac{n}{c}\right)$	$O\left(\frac{1+1/n}{(1-p)^2}\right)$	$\left(\frac{p}{1-p}+2\right)$
onion routing	$O\left(\left(\frac{n}{c}\right)^2 \ln n\right)$	$O\left(\left(\frac{n}{c}\right)^2\right)$	$O(l)$	$O(l)$
mix-net				
•fixed path length l	$O\left(\frac{n^l}{c^l}\ln n\right)$	$O\left(\frac{n^l}{c^l}\right)$	$O(l)$	$O(l)$
•variable path length	$O\left(\frac{n^{l_{min}}}{c^{l_{min}}}\ln n\right)$	$O\left(\frac{n^{l_{min}}}{c^{l_{min}}}\right)$	$O(l_{ave})$	$O(l_{ave})$
DC-Net				
•fully connected, $c=(n-1)$	1	1	$O(n)$	$O(\lg n)$
•fully connected, $c<(n-1)$	(see [56]	(see [56]	$O(n)$	$O(\lg n)$
•ring connection	$\Theta(n)$	$\Theta(n)$	$O(n)$	$O(\lg n)$

Table 1.1. A summary of the analysis for variations on each of four protocols.

verify the identity of the each node operator. As a result, it is trivial for malicious parties to add nodes to the network that do not operate in good faith. These nodes can leak information, perform the protocol incorrectly or not at all, collaborate by sharing information about carried messages out of band, mark packets, and the like. Worse, it is impossible to detect such mischief or prevent it from happening without an out-of-band identification and authentication channel. As Douceur noted, the so-called Sybil attack [17] is trivial in any case where one entity can masquerade as multiple entities, as is certainly the case on public, switched networks such as the Internet.

Alternatively, the adversary can attempt to compromise running nodes. As noted above, unpatched Windows systems are compromised by automated attackers in an average of 20 minutes when connected to the Internet. New vulnerabilities are discovered frequently for virtually all modern operating systems, and many computers are not kept completely up to date with the latest security fixes. Any node in a peer-

to-peer anonymity system might be so vulnerable, and thus come under the control of an adversary.

As noted above, if the adversary is in control of one or more nodes in an anonymous communication system, certain attacks become possible or easier. Exploiting compromised nodes in a brute force fashion is straightforward: If an adversary controls all nodes along the path between a sender and initiator, then anonymity is trivially compromised. It is easy to see that an attacker that controls c nodes out of a total of n has probability roughly proportional to c/n of controlling each node along the path. As noted by Syverson, et al. [51], an adversary in a low-latency anonymity system need only compromise the endpoints, as timing correlations generally suffice to tie initiators to responders. This will happen with probability $\frac{c(c-1)}{n^2}$ for each path formation. The predecessor attack, which generalizes even to high-latency anonymity systems, is equally effective and can work with only a single compromised node, if communications persist across enough path reformations.

The existence of compromise attacks is a difficult problem in the face of user-chosen paths. Wright et al. [55] proposed the following mechanism to help mitigate the power of these attacks. Upon first joining a peer-to-peer anonymity system, a node learns a large list of peers that could be on its paths. The node chooses a small set of these peers, called entry guards, and forever after uses only these nodes as the first node on their paths. If the none of these nodes are compromised, the user is immune to traceback. This occurs with probability $\frac{n-c}{n}$. If not, the user is no worse off than without entry nodes.

Finally, the adversary could engage in a different sort of brute force, and actively compromise nodes while communications are occurring. This scenario is most likely when an initiator is engaging in near-real-time communication, such web-browsing. The attacker compromises a node on one end of the path, learns the next node in the path, and compromises that node until the complete path between initiator

and responder is compromised. Obviously, this requires a powerful adversary, with technical expertise, or subpoena power, or the like.

In all of the above attacks on low-latency systems, as well as the more passive traffic analysis attacks, one observation tends to hold – attacks by less-than-omniscient adversaries cannot succeed unless an initiator engages in long-term communication with a sender. Put another way, the presence of observers in any real system[4] will degrade anonymity over time: these systems buy their participants time, but the amount of such time is limited.

1.3.5 Other Attacks

There are many other attacks that can affect anonymity system. Many are indirect, and attack the infrastructure of an anonymity system rather than its algorithmic basis. We survey several classes of attacks here.

One class of attacks, called directory attacks, attempts to manipulate a node's view of the system. In particular, the adversary controls some subset of nodes, and would like nodes joining the system to use compromised nodes when forming a path. Rather than wait across many path formations for this to occur by chance, the adversary can skew the process by presenting new nodes with an incomplete list of nodes in the system. This attack is a serious concern in any peer-to-peer system. Currently, Tor avoids this attack by centralizing the directory servers with a trusted source. Nambiar and Wright [36] propose a system they call Salsa, which allows new nodes to select randomly from the full set of nodes through use of a distributed hash table. Salsa requires that the proportion of attackers be less than 20% of the nodes in the system as it is currently designed; it unclear whether this is a realistic roadblock to the Sybil attack.

---

[4] A fully connected DC-Net avoids this problem, but with totally unrealistic overhead.

Related to many legal attacks is a class of attacks based upon removing exit nodes from the system. This attack is a form of denial of service, through which the adversary is attempting the remove certain nodes from the system, perhaps with the goal of leaving only compromised or easily observable nodes remaining. The attacker forms paths through the system with the target nodes functioning as exit nodes, and then proceeds to engage in objectionable behavior, as defined by the jurisdiction in which the exit node is located. The attacker relies on the anonymity system itself to prevent traceback, and the legal system to shut down the node. Defending against this attack is hard; Tor allows nodes to advertise and enforce a destination-based exit policy, though it performs no content-based filtering. Locating nodes in jurisdictions with strong free speech protections is an expensive option that is also not scalable.

There is a class of attacks that attempt to exploit side channels – that is, information leakage due to the physical implementation of a system. These attacks can be analogous to Murdoch and Danezis's flooding attack in Tor. Murdoch presented an attack [34] that measured remote node clock skew through TCP timestamps. From this information he inferred CPU temperature, and from that, the presence or absence of encrypted traffic. Side channel attacks are often surprising in this fashion, and are relatively unstudied in the context of anonymous communication systems.

Related to the above, there are attacks that utilize out-of-band information and assumptions about the contents of the communication in order to degrade the participants anonymity. Hintz [26] proposed the technique of fingerprinting or profiling web sites on the basis of packet sizes in underlying HTTPS traffic. To implement the attack, the adversary connects to a site, which is assumed to respond identically to all initiators, in the same manner as a real initiator would, and records a trace of the connection. The attacker then examines the trace of a connection under suspicion. If the two packet traces contain a sufficient number of identical counts of packets of a given size, then they could be considered a match. Hintz assumes a single proxy, low-

latency system performing no additional mixing or padding. Sun, et al. [50] extend this attack, still assuming that the underlying transport is HTTPS but examining a set of 1,000 web sites. Their results, based on object sizes extracted from the HTTPS connections on the basis of think times and server delay, were quite good, allowing them to identify the responder over 90% of the time. However, their assumption about the underlying infrastructure is overly simplistic and not in line with deployed systems. First, all real-world anonymity systems utilize some form of traffic shaping to prevent this attack, and their study did not account for this. Second, virtually all privacy-preserving technologies, such as Virtual Private Networks, multiplex over a single TCP link. This prevents easy extraction of object size, again not reflected in their work. Regardless, the approach of Sun et al. is quite promising.

1.4 Assessment

In the preceding sections, we have surveyed current work on anonymous communication systems, discussed models of the adversaries that such systems face, and presented many of the attacks that such systems face. Here, we summarize these topics, and draw attention to the areas where interesting problems remain. These existence of these areas motivate the work in this dissertation, both what we describe, and what we propose to do.

Anonymous communication systems are designed to prevent adversaries, be they passive or active, from learning about the participants and their messages between one another. This obfuscation is accomplished through encryption, proxying, mixing, and traffic shaping. Much of the early work on anonymous communication systems posited the global passive adversary. Many early systems were designed to counter this adversary. Deployment of such systems in the real world significantly limits the design space of anonymity systems, that is, the extent to which each of these methods

can be applied. Correspondingly, the power of the adversary that can be realistically defended against is limited.

As a result, recent work has started to examine attacks that are more feasible for adversaries with less power, such the Sybil attack, the predecessor attack, and various side-channel attacks.[5] We find three areas to be of particular interest.

First, prior to our work, the fingerprinting attack had yet to be evaluated in a realistic context. We show later in this dissertation that despite the limited traffic shaping in currently deployed anonymity systems, such traffic is vulnerable to an extended version of the fingerprinting attack. Further, we show that such an attack can defeat responder anonymity, in some scenarios up to 90% of the time, even with large sets of responders (2,000, in this study). We present results that support the hypothesis that such attacks are scalable with the number of responders, that fingerprints remain usable for long periods of time (over four weeks, in our experiments), and that the fingerprints can be gathered after the communication under suspicion is observed.

Next, we examine several methods of mitigating the attack we describe, focusing on deterministic schemes for altering the identifying characteristics of collected fingerprints. We find a straightforward tradeoff between the amount of padding necessary and its efficacy.

Finally, we observe that there is a class of applications that deployed low-latency systems have not addressed: delay-intolerant, soft realtime applications such as Voice over IP. While the techniques of onion routing are appropriate for such applications, their efficacy has not yet been evaluated. We present in Chapter 5 a study of the feasibility of anonymous VOIP (aVOIP) on the Internet. We show that there exist on the Internet paths of sufficient quality to support aVOIP. We also describe a path

[5]There has even been recent meta-work on reasoning about which attacks are most economical and focusing on defending against those first; see Acquisti, et al. [2] for an example.

selection algorithm that provides reasonable performance on the paths in exchange for a modest decrease in anonymity.

CHAPTER 2

ON THE FEASIBILITY OF PROFILING ATTACKS

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

Maintaining user privacy is not such a simple matter. In this chapter,[1] we show that an encrypted connection is not sufficient for removing traffic patterns that can reveal the web site that a user is visiting. Specifically, we examine the success rate of traffic analysis attacks used by an eavesdropper that test against learned profiles of web site traffic inter-arrival times and packet sizes.

We examine real traces of encrypted, proxied HTTP traffic, and our attacks attempt to discern the responder to each web request from among a set of candidate responders. The method of our attack is straightforward. In advance, the attacker gathers a *profile* of specific web pages according to some criteria, which may be their popularity or level of interest to the attacker. The profile is composed of two features from the encrypted HTTP request and response stream: the packet size and inter-arrival time distributions. The attacker then monitors the traffic of a wireless link or

[1]Some of the work in this chapter was done in collaboration with George Bissias.

a wired link to which he has access. When a burst of traffic occurs, the attacker tests the trace against a library of profiles of web pages looking for a good match.

This type of attack on web traffic is an area of concern for advocates of privacy enhancing technologies. This includes the developers of Tor, who have recognized that this type of web page fingerprinting is a significant problem [15]. Currently deployed low-latency anonymity systems do not significantly adjust traffic to prevent comparison to profiles.

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

The remainder of this chapter is organized as follows. We first present our data collection methodology. We then describe the basis of the profiling attack, as well as the specific method of profiling and matching under study, based upon cross-correlation of packet sizes and interarrival times. We evaluate the performance of the profiling attack on our dataset, and show that some sites are more consistently identifiable than others. Finally, we discuss areas for further study, some of which are addressed in Chapters 3 and 4.

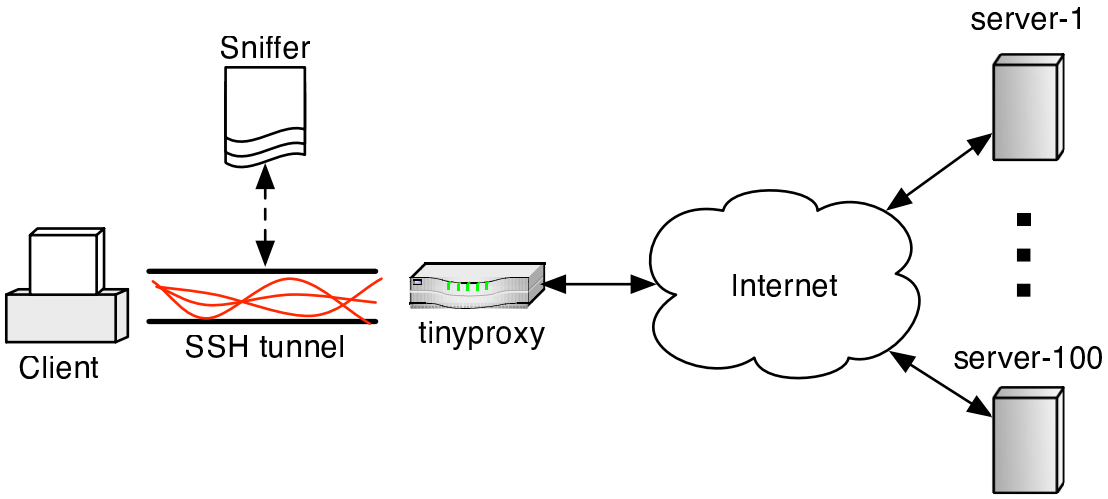

Figure 2.1. Measurement setup

2.1 Data Collection

To begin our study, we collected data from January 01, 2004 until March 31, 2004. In this section, we describe the procedure we used to collect our data. We used the results of a prior user study by Wright, et al. [55] to determine the sites most visited by users of the computers in our department. The study tracked the web traffic of 24 volunteers for 214 days. We examined the proxy logs from Wright's study and used the front pages of the 100 most-visited sites for experiments in our study.

To retrieve a baseline version of each page, we scripted an instance of Mozilla Firefox 1.0 to retrieve a site's main page and all graphics and other associated objects. Fig. 2.1 illustrates our measurement setup. We configured Firefox to connect through an instance of tinyproxy 1.6.3 bound on a local port via an SSH tunnel (OpenSSH 3.5p1 with compression and encryption enabled). All processes involved in the collection were running on the same machine. Our script retrieved the main page of each of the 100 sites every hour. We used tcpdump 3.7.2 to sniff and record the encrypted traffic over the SSH tunnel.

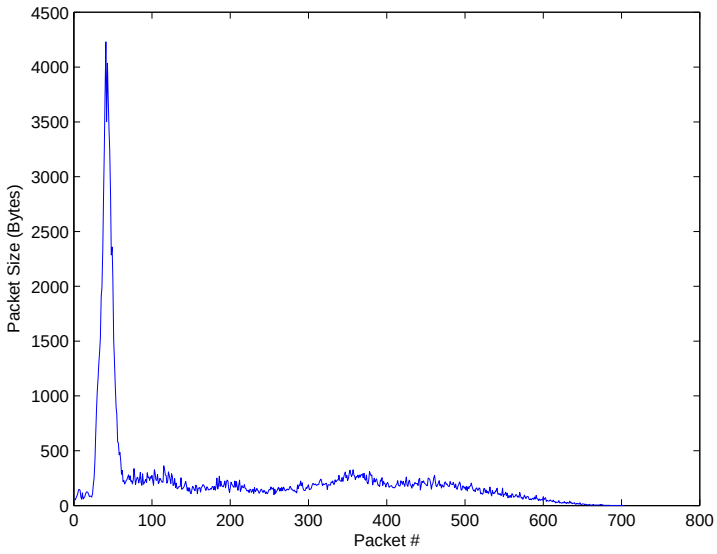

Figure 2.2. An example of a size profile for `www.amazon.com`

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

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

2.2 Identifying Encrypted Traffic

The main goal of this study is to answer the following question: does a trace of web traffic, sent through an encrypted tunnel to a proxy, leak sufficient information to allow us to determine the site being accessed? As this section details, we have found that many popular web sites are reasonably identifiable even with relatively old training data (see Figure 2.4), and that some are extremely distinctive.

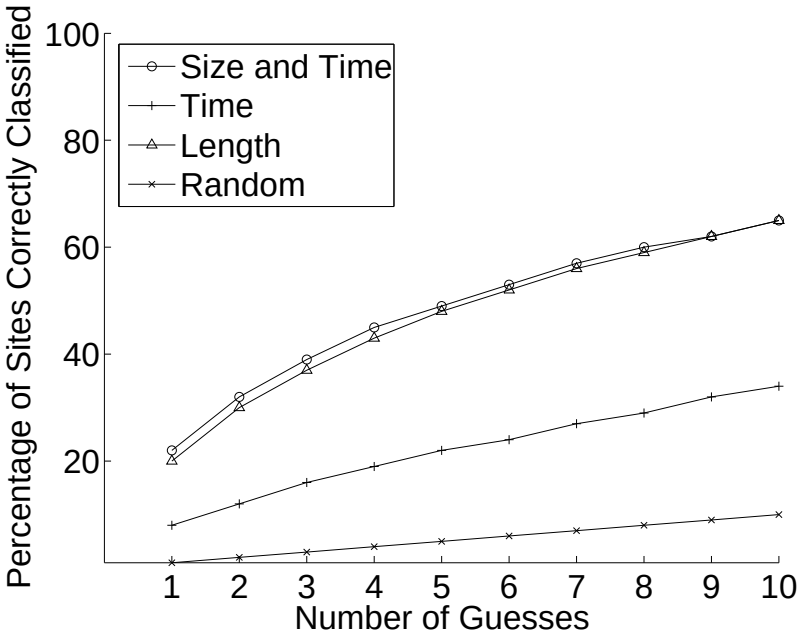

Figure 2.3. Accuracy per number of guesses with a one hour gap and different classification methods. In all figures, the gap refers to the time between a 24 hour training period and a single test.

2.2.1 Performing the Attack

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

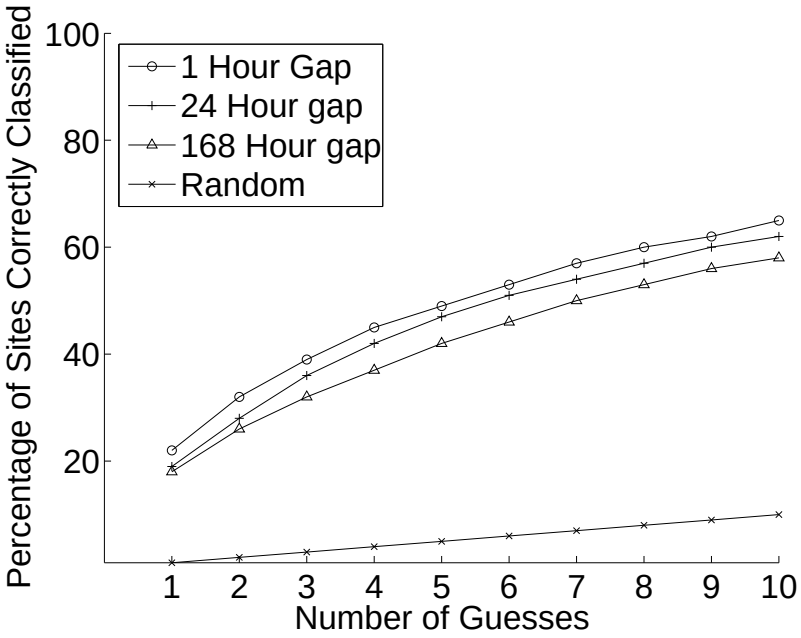

Figure 2.4. Accuracy per number of guesses. The random line is the result of choosing site labels uniformly at random. In this and all following figures, we are using the combination of both the size and time profiles.

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

We define an *inter-arrival time trace*, I, as the sequence of n inter-arrival times, $\{t_1, t_2, \ldots, t_n\}$, between packets for a given trace. To construct a *time profile*, we coalesce a set of inter-arrival times, each corresponding to the same site, N, but a different time, D, into a single, new inter-arrival time trace. We take the arithmetic mean of each t_i in the set for each $1 < i < n$, and use this as the new t_i in the time

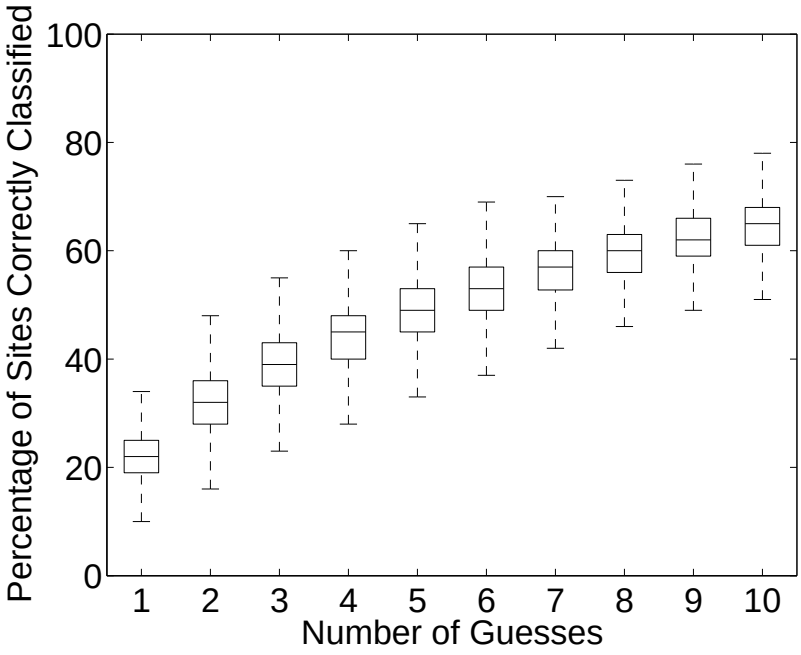

Figure 2.5. Accuracy per number of guesses with a one hour gap.

profile. A corresponding trace and profile exist for packet sizes, which we denote as the *size trace* and *size profile*, respectively. Figure 2.2 shows, as an example, a profile of `www.amazon.com`.

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

$$r = \frac{\sum_{i=1}^{n} [(x_i - \bar{x})(y_i - \bar{y})]}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})}} \tag{2.1}$$

where $\bar{x}$ and $\bar{y}$ are the means of the corresponding series. Intuitively, the cross correlation estimates the degree to which two series are correlated. It is a summary statistic indicating whether the individual numbers in two sequences have similar magnitude.

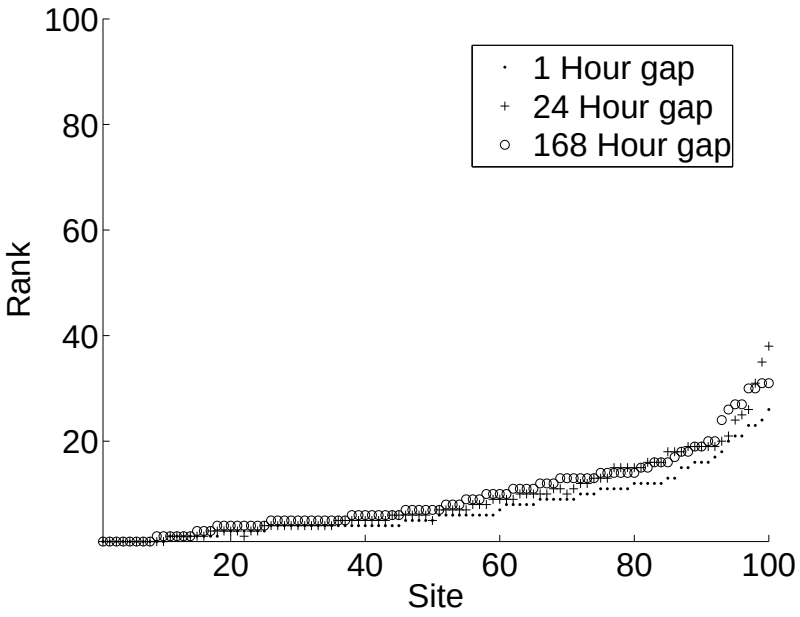

Figure 2.6. Per-site rank in the ordered list of guesses.

We call the cross correlation of a trace and a profile the *similarity* of the trace to the profile. When attempting to classify an unknown data set, we compute its similarity to each of the profiles we have previously created. A sequence of guesses is returned, each corresponding to a particular site and ordered by the magnitude of the similarity.

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

- Size profile only;

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

We found the third method, shown in Figure 2.3, to be most effective overall, and utilized that method in all further results presented here. As shown in Figure 2.4, accuracy decreases as the gap between training and testing grows, but the trend remains consistent. The implications for the attacker are clear: training immediately before the attack is best, but even old data allows for some identification.

We also note that the combined size and time profile has better performance than either methods alone, and that this performance difference from using size alone is statistically ($p = 0.01$) but not substantively significant. The statistical significance of the small performance difference is to be expected with the large sample size, but leaves the following questions: Why does the size profile perform so much better than the time profile? And why does the combination of the two not improve performance more substantively?

We surmise that the performance difference between the size profiles and the time profiles can be attributed to the much higher variance present in the time profiles. In particular, the sizes of packets tend to remain largely similar across retrievals to a web site, while the interarrival times of these packets varies more significantly. This effect implies that there is a larger source for error in profiles built atop packet interarrival times.

We quantify this effect as follows. Recall that each trace for a web page is a time series of packet sizes or interarrival times. We can characterize the variability of a set of n traces, each containing i values, by considering the variance for each of the ith values across the n traces. To make comparisons meaningful across packet sizes and interarrival times, we normalize the size values from the range (8, 1500) to (0,1), and we normlize the time values from the range (min(time[i_1] ... time[i_n]), max(time[i_1] ... time[i_n])) to (0,1). We then compute the mean variance across all i for sizes and

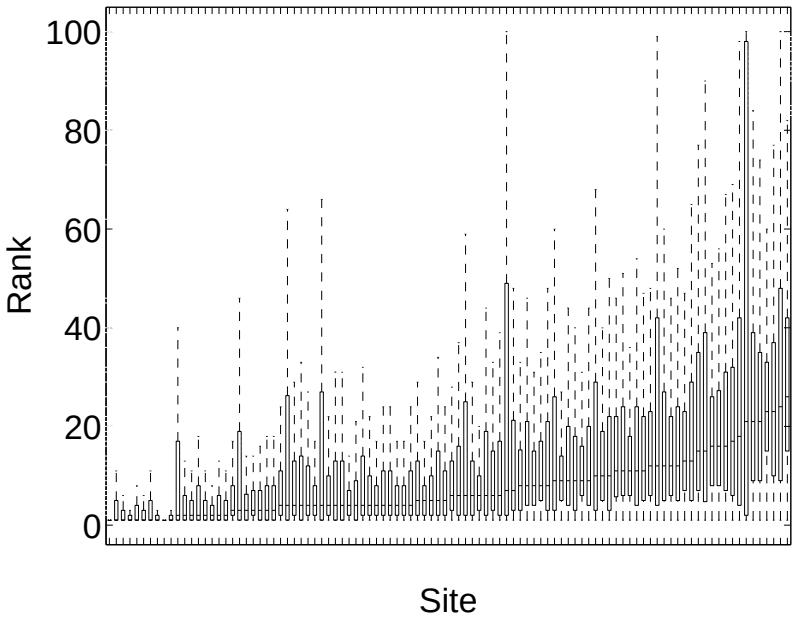

Figure 2.7. Per-site rank in the ordered list of guesses with a one hour gap.

interarrival times. For each of the 100 sizes for which we collected data, the mean variance of the normalized packet interarrival times was at least 12% greater than the mean variance of the normalized packet sizes.

From this result, we conclude that packet sizes appear to the most useful first-order characteristic in differentiating among traces. In the next chapter, we examine the performance of more robust methods of matching traces to profiles on the basis of packet size alone.

2.2.2 Predicting Identifiability

Some sites may be more identifiable than others. There is an obvious way to evaluate identifiability based on our classification methodology. By examining the *rank* of the correct site in the list our classifier returns, we have a metric describing how identifiable a site is. We show in Figure 2.6 this metric for each of the 100 sites and various gap sizes. As one would expect, smaller gap sizes result in high identifiability. Some sites are surprisingly and consistently identifiable, while others

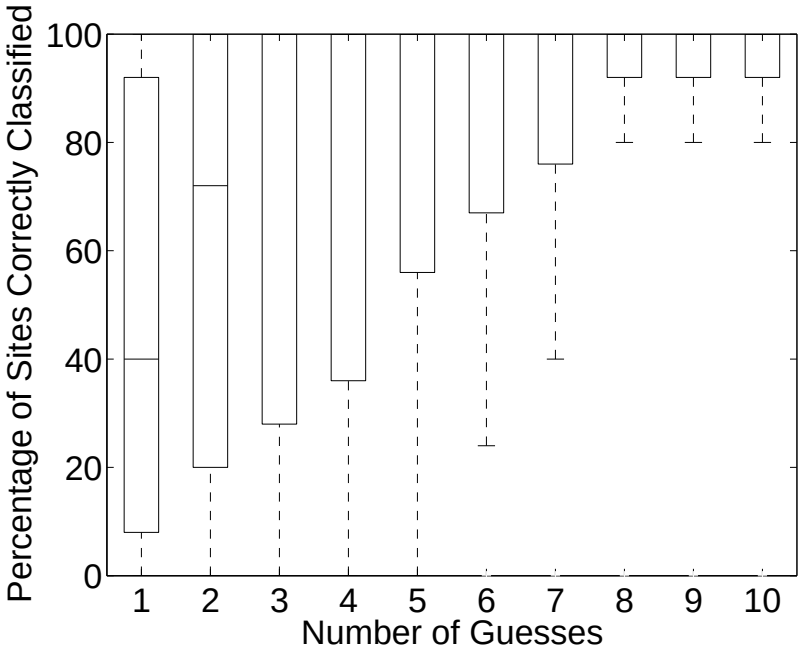

Figure 2.8. Accuracy per number of guesses for the 25 most identifiable sites with a one hour gap.

are quite difficult to differentiate. This trend is shown more explicitly in Figure 2.7. The accuracy of the most identifiable sites is much higher. In Figure 2.8, we show the accuracy for the top 25 sites. Accuracy with one guess is 40%, and increases to above 70% with two guesses.

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

2.3 Conclusion

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

CHAPTER 3

IDENTIFYING THE SOURCE OF ENCRYPTED HTTP TRAFFIC

As we have previously described, encrypted connections provide confidentiality at many different network layers. In combination with a proxy, this setup forms the basis of private communication over the Internet. Many such systems prevent observers from determining the true destination of IP traffic: multi-proxy tunnels using the Tor anonymous communication system [16]; simple SSH tunnels to a single proxy; and IPSec ESP mode tunnels to a remote VPN concentrator. WPA link layer connections also hide the destination IP address and content of a connection from an observer.

In this chapter, we further evaluate traffic analysis techniques that infer the source of a web page retrieved under the cover of an encrypted tunnel. In particular, we examine techniques that identify sources by comparing observed traffic to profiles of known sites created from packet lengths.

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

Despite this simplification, we found that under reasonable assumptions, traces were identifiable between 66–90% of the time. On the basis of our experiments, we

expect performance to scale well, and we strongly suspect that more sophisticated methods could do better.

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

Privacy is the dual problem of digital forensics — accordingly, the attack is also an important method of investigation that advances the field of digital forensics. This is because commonly used forensic techniques for gathering and analyzing network data are limited to traffic with overt IP headers and data [6, 7]. Encrypted tunnels thwart the legitimate gathering of evidence by authorized law enforcement. Advances in the field of forensics require moving beyond this limitation. The results presented in this chapter are one such advance.

The remainder of this chapter is organized as follows. In Section 3.1, we describe the attack model and in Section 3.2, the data collection methodology we utilize for the evaluation. In Section 3.3, we describe our experimental methodology and results. We discuss the implication of these results and describe future directions in Section 3.4.

3.1 Model

In this section, we describe our model of an observer that can execute traffic analysis attacks to profile web pages and identify encrypted traffic on the basis of these profiles. We first describe our assumptions about how data can be collected by an observer. Then, we describe two specific methods to create these profiles, one

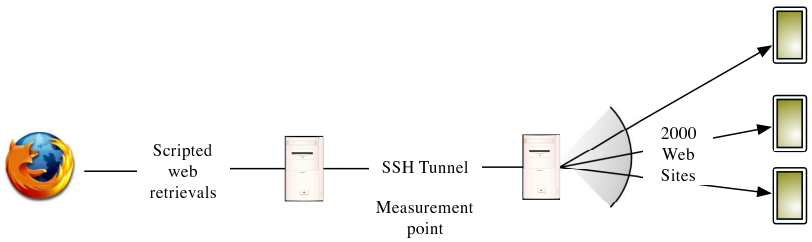

Figure 3.1. An illustration of the measurement setup. This figure also shows the position an observer may take to utilize profiling.

based upon a similarity metric (Jaccard's coefficient) and one based upon a supervised learning technique (the simple Bayes classifier).

3.1.1 Observer Model

Figure 3.1 illustrates the network setup under consideration throughout this chapter. In it, the client connects to a remote proxy over an encrypted transport layer. The proxy makes requests on the client's behalf, and returns the results over the encrypted connection. The observer is limited to examining the encrypted traffic and creates a log of packet lengths (and interarrival times, if desired) corresponding to each distinct page load. Our observer has unlimited storage for these logs. We assume that the observer is not able to discriminate among individual objects as in Sun, et al. We assume the client uses a modern browser for retrieval, as described in Section 3.2.2, which prevents the observer from obtaining this information. Because our techniques focus on packet lengths, it is not a requirement that the observer create profiles on the same link that she observes traffic. However, for simplicity, we assume this is the case.

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

The proxy we evaluate in this paper is the OpenSSH implementation of a one-hop SOCKS proxy. However, we expect our results hold for VPN proxies and WPA base stations. Neither of these systems significantly alter packet lengths since they perform no buffering and packet aggregation or fragmentation. The Tor and JAP[1] low-latency anonymity systems provide only very limited aggregation and fragmentation. We believe that these systems will also be vulnerable to a form of this attack.

To launch the class of traffic analysis attacks that we evaluate in this chapter, the observer requires a library of traffic traces between a client and a list of known web sites. We show in Section 3.3 that this library can be collected before or after the attack. Using the traces of connections to known destinations, the observer attempts to decide, in some fashion, which of the known traces most closely resembles the encrypted, unknown trace, as we explain below.

3.1.2 Profiling Methods

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

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

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

length, in bytes, of the packet. The value assigned to each attribute is the number of packets observed in that trace with the corresponding *direction* and *length*.

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

$$S(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{3.1}$$

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

Our second method for identifying unknown instances is the simple (or so-called naive) Bayes classifier. We do not give a full explanation of the classifier here: a recent text such as Witten and Frank [53] will provide the relevant details. In short, the naive Bayes classifier assumes independence between all attributes, and estimates the probability of a set of value $\overline{A} = A_1, \ldots, A_n$ belonging to a particular class C_i as:

$$p(C_i|\overline{A}) \propto p(C_i) \prod_{j=1}^{n} (p(A_j|C_i)) \tag{3.2}$$

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

3.2 Data Collection

To investigate the effectiveness of a profiling attack, we required a data set consisting of logs of encrypted traffic between a client and many servers.[2] We imposed several requirements on this data set. First, it had to be of sufficiently large size to make the problem non-trivial, while not so large to prevent multiple collections each day with our limited resources. Second, it had to reflect a real-world group of users. Finally, it had to be collected in a fashion analogous to the manner in which an actual observer would attempt. In this section, we present the details of these processes so that others can validate or recreate our collection process.

3.2.1 Initial Data Collection

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

The initial step was to log all requests to the department's DNS server from 01 December 2005 through 04 January 2006. This month of logs yielded 44,305,203 requests from 828 hosts. We removed all requests that were not for address (`A`) records, as HTTP traffic would not have generated them. We removed all requests that were from outside the department, as we were studying users within the department. We

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

removed all requests that were for names within our domain, as these tend to correlate with intra-department service accesses (such as secure shell access) and not HTTP requests.

We then removed requests we judged to be the result of automatic processes. Specifically, we removed all requests for a given name that were made from the same host with an average frequency of greater than once per five minutes in any eight hour period. We then heuristically removed most requests for PlanetLab[3] machines.

From these requests, we constructed a list of URLs of the form `http://ipaddress/`, along with the associated count for each. On 01 February 2006, we attempted to contact each of these sites with an HTTP GET request on port 80. We removed from the list all sites that were unreachable, refused the connection, or returned an HTTP error that was not a redirection. We also replaced the `ipaddress` with the actual hostname the remote machine used, and we resolved all redirections. Finally, we summed the counts of all sites that redirected to the same URL. This final list of *(count, URL)* pairs was 109,479 pairs in length. For our experiments, presented in the next section, we focused on the 2,000 most accessed sites, which accounted for 64% of all web requests. The relationship between rank and number of accesses is show in Figure 3.2.

3.2.2 Page Retrieval

To gather realistic traces, we set up a client host with a recent GNU/Linux distribution. We used Mozilla Firefox 1.5[4] to retrieve each URL via a SOCKS proxy. OpenSSH 4.2p1[5] was set up to perform application level dynamic port forwarding (the `-D` option) and act as a proxy. This created an encrypted channel over which the

[3] `http://www.planet-lab.org/`

[4] `http://www.mozilla.org/projects/firefox/`

[5] `http://www.openssh.org/`

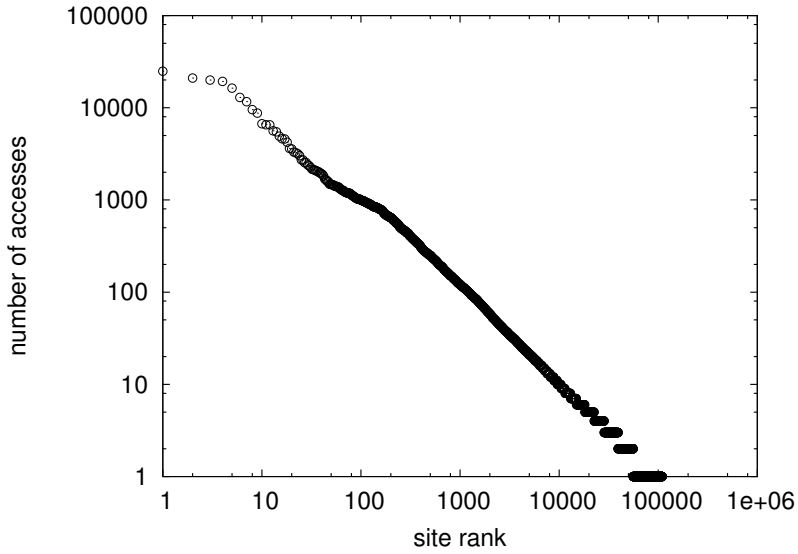

Figure 3.2. The relationship between site rank and number of accesses. Sites are ranked according to total number of accesses to each site observed.

HTTP requests and responses were forwarded, and it made distinguishing individual objects infeasible, as Firefox generally makes multiple simultaneous connections to load a web page that are multiplexed over the secure channel.

We configured Firefox to not cache data between retrievals, which allowed us to focus on the specific question of identifying encrypted streams by their profiles. We installed the latest Macromedia Flash plugin[6], as many of the URLs in our list contained content rendered by this plugin. We also configured Firefox to not attempt various extraneous connections, due to live bookmarks, automatic update checks, and the like. While disabling these features makes the resulting traffic somewhat less realistic, we believe it to be a reasonable trade-off to allow us to focus on the specific problem under investigation. As Firefox loaded each URL in our list, we used tcpdump 3.9.4[7], linked against libpcap 0.9.4, to log the first 68 bytes of each packet. This length is sufficient to capture IP and TCP headers of the packet and

[6] `http://www.macromedia.com/`

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

thus determine the total packet length. The information in these logs form the basis of the profiles that we detail in the next section.

3.3 Evaluation

In this section, we describe the experimental methodology we used to evaluate our proposed classification methods as well as the results of that evaluation, based upon two months of data we collected. We give evidence that the two methods, one based upon Jaccard's coefficient and the other upon the simple Bayes classifier, have several properties of interest: We show that the Jaccard-based classifier's accuracy, under reasonable assumptions, is over 60%. We give evidence that it will scale reasonable well to large data sets. We show that many profiles, once constructed, remain valid for long periods of time. We show that training data can be gathered before or after test data, with a negligible effect upon accuracy.

3.3.1 Experiment Setup

To evaluate the effectiveness of the profiling attack described above, we collected traces of encrypted traffic as described in Section 3.2. Specifically, each *sample* consists of the log of a retrieval of the front page from each of the 2,000 sites. We created a new sample once every six hours for a period of two months, for a total of nearly 480,000 samples.

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

- t describes the number of sequential samples that form the training set.
- s describes the number of sequential samples that form the test set.

- Δ describes the number of sequential samples between the training and test sets. $\Delta = 0$ indicates the test set starts with the sample immediately following the last sample in the training set.

- N describes how many of the 2,000 pages we considered in a particular experiment. If it is less than 2,000, then we reduced the number of traces in each sample to this number by removing the traces corresponding to the same pages from all samples. We removed the least-popular (as determined by the method in Section 3.2.1) pages first.

The result of each experiment is a table of probabilities of class membership for each of the instances in the test set. From this, we determine the *k-identifiability* of each instance. The k-identifiability for an instance is defined as 1 if the actual class of the instance is in the top k of the predicted class list, as ordered by probability estimate (predicted classes with the same estimate are ordered in an arbitrary but fixed manner) or 0 if the actual class is not in the top k.[8] The *k-accuracy* for an experiment is the average of the k-identifiability value for each instance in the test set.

3.3.2 Classifier Performance

We evaluated the effect of changing each of the independent variables listed above. Unless the variable was the isolated and changing variable, each of the following graphs assumes $k = 1$, a training set size of $t = 4$ (one day of data), a test set size of $s = 4$, $\Delta = 3$ so that the training and test sets are one day apart, and $N = 1000$ pages. We chose random initial sample numbers such that 10 individual experiments were run with all other variables equal — these otherwise identical experiments are

[8]During later analyses of the plaintext of the retrieved web pages, we noted that 43 of the 2,000 pages were essentially identical to the other 1,947 we retrieved. The graphs in the following sections control for this bias by considering a classification correct if it for a class corresponding to an identical web page.

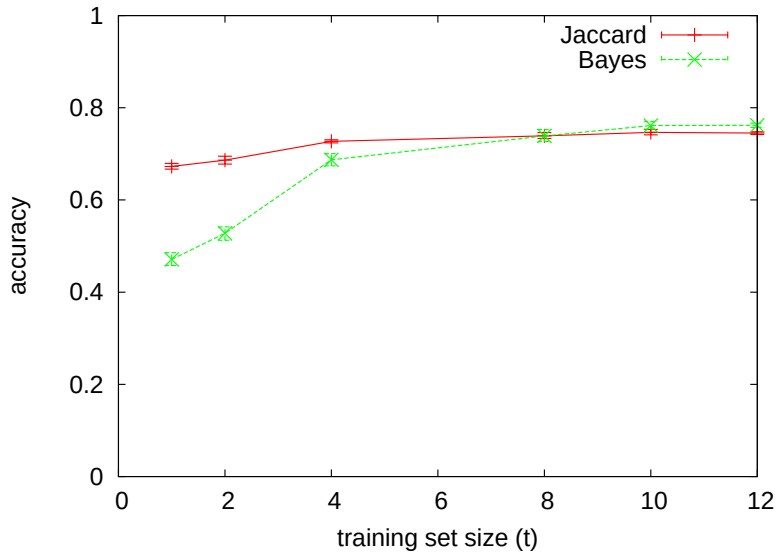

Figure 3.3. Effect upon accuracy of varying training set size

the source of the 95% confidence intervals in the graphs. Often the intervals are too small to be observed.

Figures 3.3, 3.4, 3.5, and 3.7 show the results of these experiments.

In Figure 3.3, we show the effect of varying the training set size. While accuracy increases as more samples are added to the training sets, the rate of increase slows when $t = 4$. Thus, even small training sets give good accuracy, and additional training data give diminishing returns on accuracy. In Figure 3.5, we show the effect of varying Δ. Larger delays between the training and test sets result in lower accuracy, but the decrease appears to be linear in the delay and tolerably small (a drop from 73% to 63%), even after four weeks of delay. Thus, training data remains useful even after a significant amount of time has passed. This implies that it need not be collected too frequently, increasing the utility of large sets of such data. In Figure 3.4, the effect of a larger k is shown. Unsurprisingly, allowing the observer more chances to identify a trace allows for higher accuracy. At $k = 10$, the observer can expect to have correctly identified the trace 90% of the time.

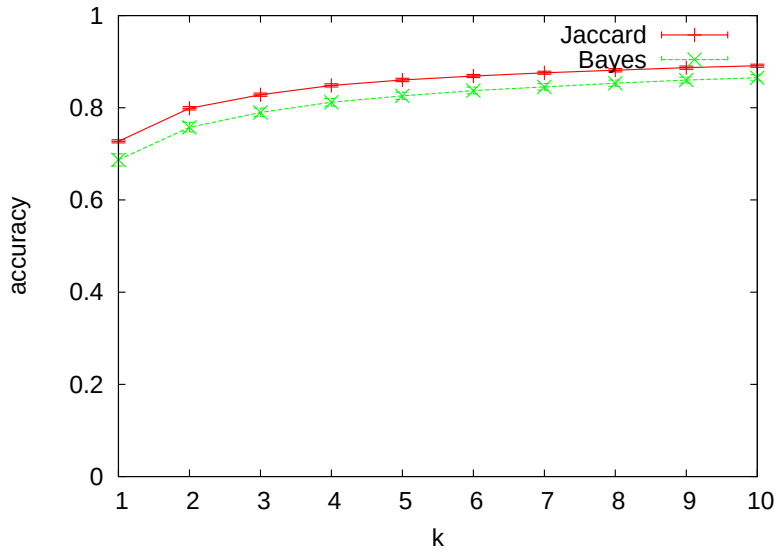

Figure 3.4. Effect upon accuracy of different values of k

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

Table 3.1. Regression analysis for $acc = A \log_2 N + B$

Figure 3.7 shows the effect of varying the number of pages in the training and test sets. The drop in accuracy appears to follow a relationship of the form:

$$acc = A \log_2 N + B \tag{3.3}$$

A linear regression analysis is presented in Table 3.1 which shows that a reasonably close log-linear relationship exists. This relationship implies that the profiling attack has good scalability properties, even as N grows toward the size of the Internet.

3.3.3 Forensics Feasibility

This method of identifying encrypted traffic does not require gathering profile data prior to observing the traffic being analyzed. Such a situation occurs when the

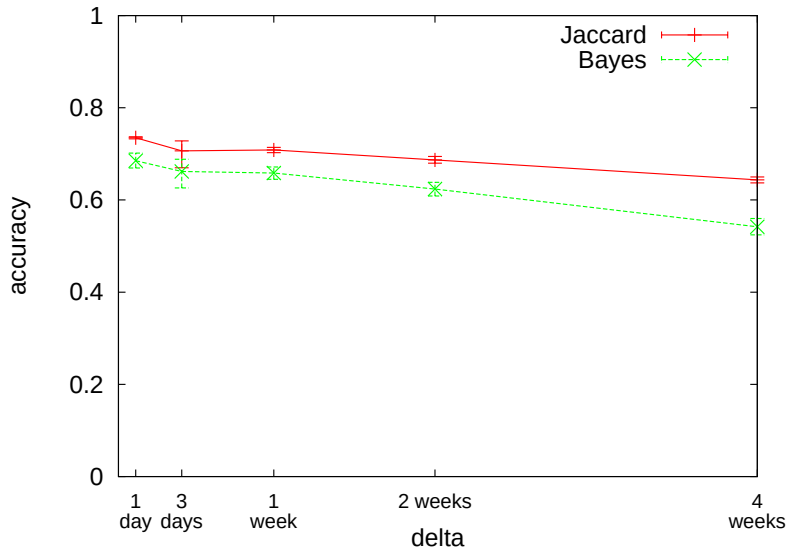

Figure 3.5. Effect upon accuracy of varying the delta (time between training and testing)

profiler is an investigator attempting to identify traces gathered in the past. Figure 3.6 shows the results of the experiments from Figure 3.5 with one important difference: the training set occurs after, rather than before, the test set. For otherwise identical parameters, the relative decrease in accuracy ($\frac{original - reversed}{original}$) is less than 3% ($p < 0.01$) across all of the experiments.

3.3.4 Explaining Performance

As the Jaccard-based classifier only sees occurrences of *(length, direction)* and not counts, some traces will be indistinguishable from others. If we assume no changes in the traces between the training and test sets, the accuracy of the Jaccard-based classifier is bounded by the number of unique traces within each sample. Since pages can change over time (and retrieval can differ due to cross-traffic), and thus training and test sets can differ, this upper bound is well above the accuracy we observed in practice. Figure 3.7 shows the fraction of unique samples in the training set and the accuracy of the Jaccard-based classifier on these sets.

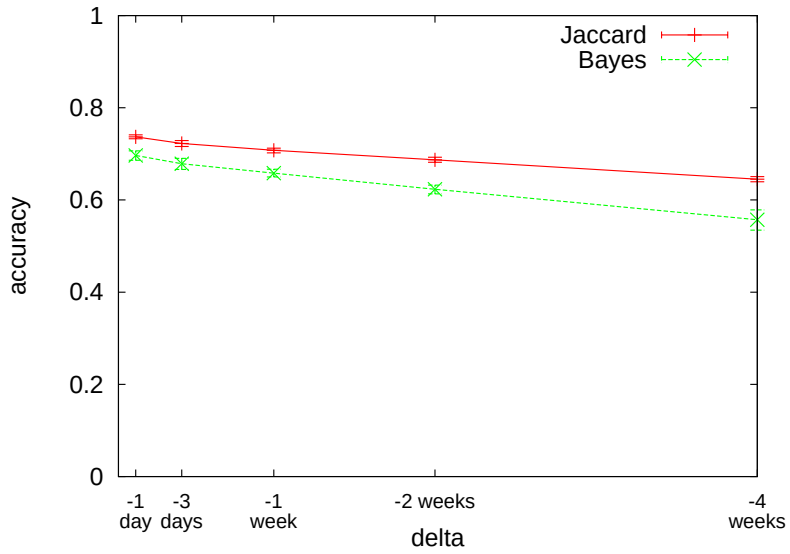

Figure 3.6. Accuracy after swapping the test and training sets

To model the source of this uniqueness, we examined the underlying distribution of (packet size, direction) tuples across all instances in the training sets for the 10 experiments corresponding to 2,000 sites. This distribution is shown in Figures 3.8 and 3.9. The distribution of per-trace occurrences has an entropy, as defined by:

$$H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{3.4}$$

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

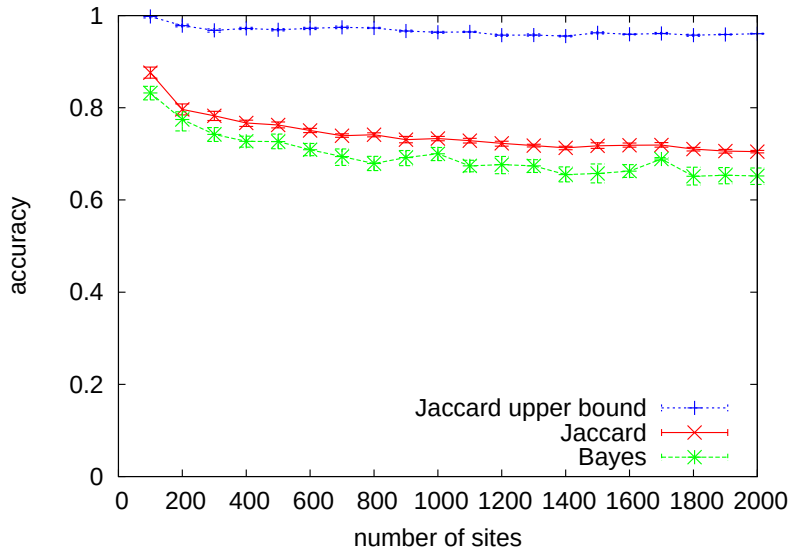

Figure 3.7. Effect upon accuracy of varying the number of sites considered. Also shown in the theoretical maximum accuracy that the Jaccard-based classifier can achieve.

identifiability, but as we show in Chapter 4, privacy-conscious system designers will be likely to utilize padding in their designs and render this analysis unnecessary.

3.4 Discussion

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

Building a library of profiles of encrypted HTTP traffic is a reasonable activity for both research and law enforcement purposes. Many forensic investigators will lack the time and tools to gather such data. We have shown the collecting such data is straightforward. Further, we have shown the such data can be used to build an identification system based solely upon packet size — such profiles can be collected

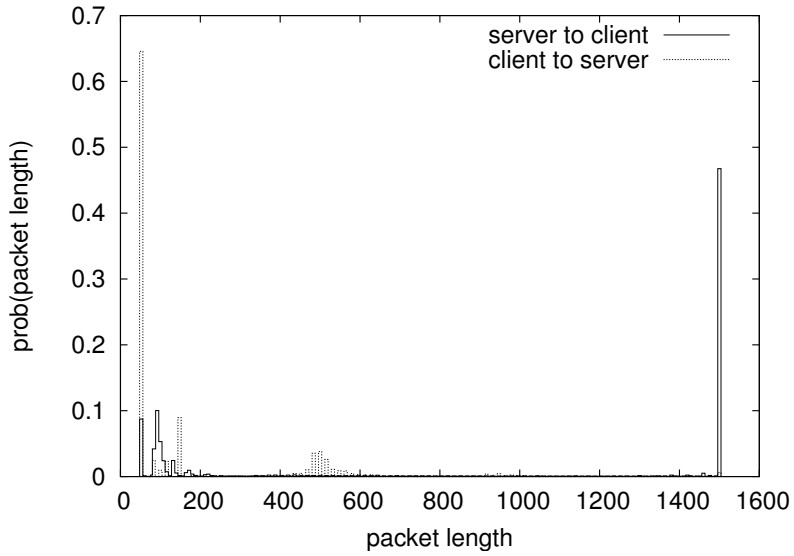

Figure 3.8. Overall packet length distribution. This distribution is based on every observed packet.

from anywhere on the Internet and will not differ for most pages. It is certainly feasible to build software that will enable researchers or investigators to collaborate and create a profile of the entire Internet in a distributed fashion. This library would be similar to a distributed version of the National Software Reference Library[9] which maintains forensic hashes of many commercial applications and operating systems. Given the log-linear scaling of our classification method, we expect that such a library

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

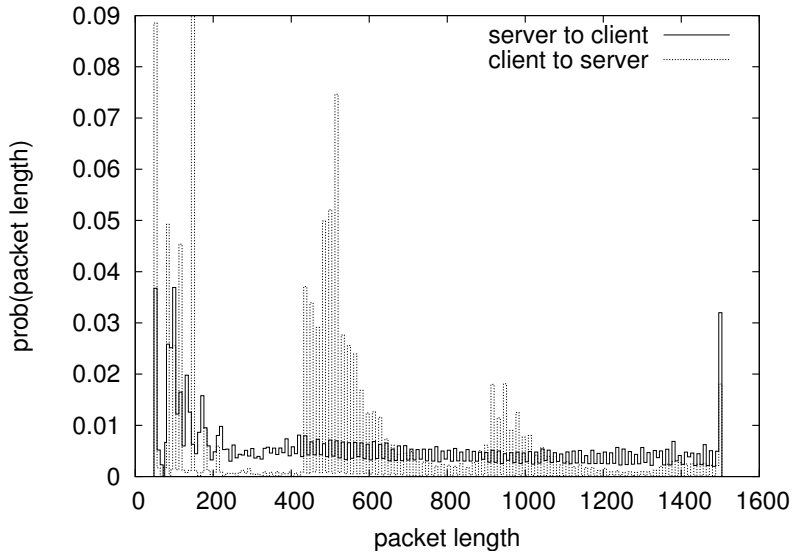

Figure 3.9. Per-trace occurrence distribution. Packet lengths are included in this distribution at most once per occurrence in each trace. This distribution more accurately reflects the Jaccard-based classifier's input than that of Figure 3.8.

would have signification forensic value.

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

The Netcraft survey[10] shows that are approximately 38 million active web sites on the Internet as of February 2006. If 400 volunteers profiled 600 sites an hour, then the entire Internet could be updated in a distributed library once a week. A greater number of volunteers would reduce the amount of work each has to do, or increase the accuracy of the profiles by updating the profiles more frequently. The mean size of an naive Bayes profile in our existing experiment is about 350 bytes; an archive of Internet top-level pages can thus be stored with less than 13GB. Of course, each Internet site consists of many pages. We conjecture that this problem

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

can be addressed because many pages are based on common templates. For example, all Google web searches result in the same layout. The web site of the NY Times looks the same for our observer even though the content changes quite frequently. In general, we expect that sites with many, dynamically generate pages will follow templates out of a necessity for manageable administration; sites that with static content that is different on every page are likely to be updated only infrequently, and therefore require infrequent profiling.

3.5 Conclusion

There are several obvious extensions to this work which we do not consider here. First is a more thorough examination of the effect of introducing timing information into the profiles, as we discussed in Chapter 2. While this addition has the potential to significantly improve accuracy, it comes at a potentially high cost: the loss of location-neutrality in gathering the profiles. Second, is an examine the efficacy of more complex classification methods. Both of the methods we utilized in this study assume independence among the attributes in the traces. We have given evidence that this independence assumption is flawed, and we expect that classifiers which model inter-attribute dependence will have improved accuracy. Third, an evaluation of this identification method on larger data sets and with a more realistic network substrate. While we expect performance to remain consistent, it is possible that real-world systems may have unanticipated effects upon our method's effectiveness.

Given that profiling is a highly effective method for identifying the sources of encrypted connections, and that building profiles can be distributed and performed asynchronously from the task of matching against them, it is clear that some defense against the attack is necessary. In the next chapter, we examine the sources of identifiability of packet streams, enumerate the various ways that such sources can be hidden, and evaluate several schemes to suppress identifiability.

CHAPTER 4

PREVENTING PROFILING ATTACKS

In the previous chapter, we saw that profiling attacks are a serious threat to user privacy. In this chapter, we examine in more detail the sources of traffic identifiability, with an eye toward hiding this information from an attacker.

We begin with a discussion of the underlying sources of identifiability of encrypted packet streams. Since the contents of the streams are encrypted, we are reduced to extracting features, based upon sizes and interarrival times of the packets that compose the streams. We discuss various models of these features in Section 4.1.

If these features can be obscured, then the efficacy of a profiling attack should fall. The most direct route to hiding these features is to modify the encrypted packet stream. We identify the direct changes to the packet stream that can be made to hide the feature in Section 4.2, and discuss the performance impacts that such changes are likely to have.

In Section 4.3, we implement in simulation a specific instance of the countermeasures described in Section 4.2, per-packet padding. We find that this approach is reasonably effective, lowering predictive accuracy to less than 8% while increasing traffic volume by 145%. We conclude and discuss proposed future work in Section 4.4.

4.1 Identifying Features

Consider the party attempting to identify the source or contents of encrypted streams with the profiling attack the attacker, and the party attempting to prevent this the defender. For an attacker to implement the profiling attack, he must start by

constructing profiles. To do so, he takes as input a set of packet traces corresponding to requests and responses sent over a link on the Internet. We refer to a single such trace as a log or a retrieval. From these logs, he identifies salient features and builds a model based upon these features.

The key question for both the attack and the defender is "What are the features that can be identified in a set of logs?" The attacker wishes to identify the salient feature set so as to build effective profiles that allow for discrimination between logs of differing origins. Ideally, these features will be consistent among logs representing identical underlying traffic, and distinct from logs of other traffic. The defender's curiosity is motivated by the opposing reason. By knowing which features an attacker will utilize, the defender can attempt to hide these features. In the following sections, we examine the features present in encrypted streams and their sources.

4.1.1 Features and Their Sources

As discussed in earlier chapters, two fundamental features can be extracted from a packet stream: packet size, and packet interarrival time. These features are essentially all that can be extracted from the stream if the stream is encrypted and the underlying encryption algorithm is assumed secure.[1] Any other feature or set of features must be inferred from these two.

In particular, we assumed in previous chapters that traces could be broken up by page load due to the relatively long pauses between user actions. For example, the time that it takes a page such as *www.google.com* to load is much less than time it takes one to enter a search term and click "Search." These long pauses, known as

[1]We do not consider information regarding the encryption algorithm and its state that can be inferred from the packet stream, though we note that recent work in this area indicates information leakage occurs here as well; Murdoch [34] showed that remote measurement of a computer's state is possible by inducing and measuring clock skew due to temperature variation!

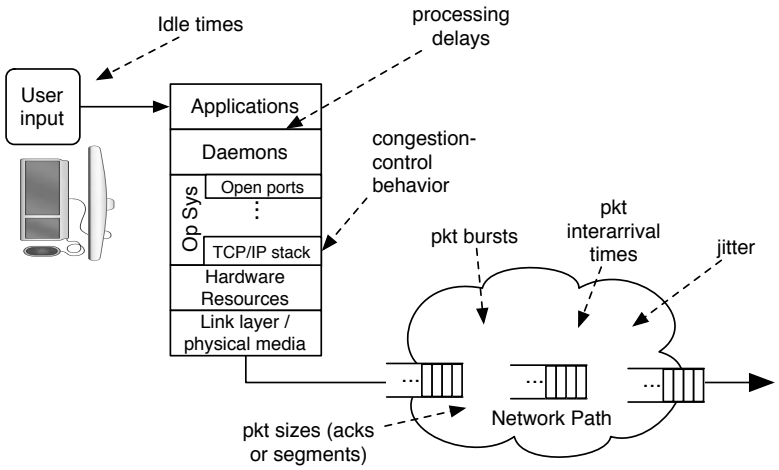

Figure 4.1. Sources of information and noise in encrypted packet traces.

think times, allow the attacker to identify distinct page loads — but note that such splits are performed on the basis of unusually long packet interarrival times.

We see in Figure 4.1 an explicit enumeration of the underlying sources of packet sizes and interarrival times. We surmise that profiling works because many of these sources are consistent over time:

- **User input**: Dynamic or on-line input, such as that of keyboard entry, actually exhibits far less variance than one might expect. Various prior work on patterns in user data entry [3] indicates that data input on keyboards is remarkably consistent among users and inputs. Stored data is in even more consistent, as there is no inter-keypress delay when it is sent out over the network.

- **Client-side considerations**: A host's behavior can be influenced by the input it receives from the network. In the case of an HTTP server, for example, the reply is directly contingent upon the request made to it — a request for a certain object will usually result in a transmission of that object. The behavior of a server is further affected by secondary characteristics of requests made to it. Again, using HTTP servers as an example, many servers will modify their

replies on the basis of headers other than the GET in the HTTP REQUEST or other information. For example, many web sites modify their content on the basis of a client's User-Agent (ie browser, version, and host OS).

- **Applications / Daemons**: The applications that a host is running will directly affect its output. Different implementations of network servers often have implementation-specific behavior that affect visible network output. Buffering and caching strategies, for example, will fundamentally influence packet size and the delays between packets. The CPU overhead that the application incurs may also affect packet timing.

- **Operating system**: As with individual applications, so with the operating system. In addition to the OS-level overhead, there is the further potential for delay, as the server process's progress will be affected by the choices that the process scheduler, disk scheduler, paging algorithm, etc.m̃ake.

  The network stack has fundamental effects on packet size and timing. Most relevant and widely-deployed is the TCP/IP suite. An OS's implementation of the TCP/IP stack is the source of a myriad of possible hidden variables. TCP/IP employs a variety of mechanisms to efficiently utilize available network bandwidth. These mechanisms include the use of a sliding window, the slow-start algorithm, the congestion avoidance algorithm, the fast retransmit and fast recovery algorithms, and more. The choice of algorithm and parameters for these algorithms each will influence observable behavior.

- **Host hardware**: Disk speed, CPU speed, bus bandwidth, RAM, network controllers – each can influence the output of the host. These effects are multiplied by the choices the drivers may make that modify, limit, or enhance the behavior of devices in the host.

- **Network path characteristics**: At each hop along the network path between end hosts, many of the above effects may apply – after all, a router is just another machine on the network, albeit a highly-specialized one. The network may introduce other effects, as well: Paths may vary over time (e.g. anything from round robin DNS to akamai-like anycast service). The interaction of router drop algorithms and external traffic may cause packet losses, delays, and retransmits. External traffic itself varies over time.

Each of the above sources of variance in the packet size and interarrival time is a source of information (signal) or randomness (noise). As an example of signal, consider the underlying data being sent when an HTTP request is sent. Any given server will respond differently to a GET request, but the same server will likely respond identically to an identical GET request, changing its response only if the underlying data is changed. As for noise, consider the timing variations induced by the network. Though cross traffic will introduce variance in the packet interarrival times for a given stream, such cross traffic is likely independent of the of the stream itself. Thus, repeated samplings of traces can help remove this source of noise from the profiles.

It is clear from the results presented in Chapter 3 that the features extracted from packet streams are not totally deterministic and predictable, and that there is underlying variation between and among the logs. Still, enough information exists to allow a high identification rate, on the basis of packet sizes alone.

4.1.2 Secondary Features

A host of secondary features can be inferred from the attacker-observable features of packet sizes and interarrival times. One such feature is the identity of the page itself, which is the goal of the profiling attack presented in previous chapters. Here

we note other such features, particularly those that might aid the attacker in refining the attack.

Any of the features described in Figure 4.1 might be discovered by a version of the profiling attack. Some of these might expand the usefulness of the profiling attack. For example, rather than fingerprinting the content of the stream and thus its source, a version of the profiling attack might be used to improve remote operating system fingerprinting, or to look for remote vulnerabilities during a security audit.

Another option is to attempt to profile user behavior more carefully. In particular, we have assumed in previous chapters that a user only accesses the front page of a particular site. In reality, users often navigate from page to page, both within and between sites. We conjecture that modeling the relationships between retrievals might improve the accuracy of the profiling attack. We have not attempted such modeling as of yet due to the difficulty in obtaining a representative sample of such user behavior.

4.2 Hiding Features

Given that the fundamental features are packet sizes and interarrival times and that we wish to hide these features, we are left with two options. First, we can attempt to manipulate the sources of these features. For example, using dynamic content, running different web servers for different requests, and using different operating systems in a server farm are all ways in which the underlying information for a given request might be changed. Practically, such approaches are flawed in that they require substantial changes on the part of the responder, who may not know or care about issues of user privacy. Thus we can pursue a second option: directly manipulating the packet stream itself at the point where it enters the anonymity system.

The option of manipulation of the stream within the anonymity system is attractive for several reasons. First, the manipulations will not require the consent of the initiator or responder, and are ideally transparent to them. Second, the initiator

might request the manipulations be deactivated, or dialed up to maximum, depending upon the content being accessed and the threat model. Finally, leaving the control of these manipulations within the system allows future modifications to the system and the effectiveness of such modifications becomes more well understood.

There are four basic manipulations that can be performed upon a packet stream:

- **Per-packet padding**: The system can add dummy bytes to individual packets. This padding increases the apparent size of individual packets, and if removed later in the system, decreases the apparent size of packets.

- **Fragment packets**: The system can fragment large packets into smaller packets. This fragmentation decreases the size of the packets that are fragmented and increases the total number of packets seen. It can also decreases packet interarrival time, as now more packets can be sent within a given time interval.

- **Delay packets**: The system can delay individual packets arbitrarily. This delay can change interarrival times in either direction — not because packets can be delayed negative amounts, but because the delay of a packet moves it temporally away from its predecessor and toward its successor. Delay may also decrease the packet count for a given span of time.

- **Insert dummy packets**: The system can insert dummy packets, that is, packets containing nonsense content. These additions will decrease packet interarrival time, and if removed later in the system, increase observed interarrival times.

All of the above options will incur a performance penalty, as they involve manipulations that are typically performed by the operating system, not an application-layer anonymous communication system. We observe that of the four, per-packet padding is likely to have the smallest level of negative interaction with TCP/IP. Thus, we ex-

plore a set of per-packet padding schemes in the next section as one way to obfuscate the features that an attacker trains upon.

4.3 Evaluating Per-Packet Padding

As discussed in the previous section, if an initiator suspects the presence of an observer he may attempt to obscure the features present in his traffic patterns. One method of obscuring these features is the use of per-packet padding. In this section, we evaluate in simulation four packet padding schemes in the context of the profiling attack from Chapter 3. We also evaluate a combination padding/fragmentation scheme corresponding to the scheme used by Tor, an implemented anonymous communication system.

Specifically, the methods of padding and fragmentation which we examine are:

- **linear:** Pad each packet to the nearest multiple of 50 bytes. This naive method adds a minimal number of bytes to each packet, and reduces the number of distinct sizes by up to a factor of 50. In practice, we saw packets of sizes ranging from 52 bytes to 1500 bytes (the Maximum Transmission Unit, or MTU), and thus reduced the number of distinct sizes under observation from about 1450 to 29.

- **exponential:** Pad each packet to the next largest power of two or the MTU bytes, whichever is smaller. This method significantly reduces the number of distinct sizes, to $\lceil \log_2 \mathrm{MTU} \rceil$, and to six in our experiments. This method can result in a doubling the size of each packet, in the worst case. In practice, we found that this increased the total data transmitted per log by less than 9%, because most data is carried in packets already of length equal to the MTU.

- **mice and elephants:** Pad each packet to either 100 or 1500 bytes. This method reduces the information available to the classifier even further than

exponential padding, by reducing the number of distinct packet sizes to two. All small packets (mostly ACKs) are padded to one size, and all other packets to another. Use of this method comes at a cost of nearly 50% growth in data transmitted.

- **MTU:** Pad each packet to the MTU. This method dramatically increases overall data transmitted (by nearly 150%) but renders all packets indistinguishable on the basis of packet length.

- **Tor-like:** Packets of length less than 512 bytes are padded to 512 bytes. Packets of length greater than 512 bytes are fragmented into packets of 512 bytes, and the last packet is padded if necessary. This method corresponds to Tor's "cells," though in practice Tor sends these cells over TLS-encrypted TCP links, where they may be aggregated into larger packets by the operating system. Our simulations do not attempt to emulate this behavior.

We assume the observer is able to determine the padding method being utilized and can adjust his training sets by padding them in the same fashion. Thus, we evaluate accuracy on the basis of training and test sets that have been padded in identical fashions. Note that our experimental setup still assumes the attacker can discriminate between packets sent by the initiator and packets sent by the responder, so the classifier being built actually sees twice as many classes of packets as described above.

In Figure 4.2, we show each method's effect upon accuracy, with a setup otherwise identical to that described in Chapter 3. Recall that k-accuracy is defined in Section 3.3.1. Table 4.1 lists each method's effect upon accuracy as well as the relative number of bytes transmitted. The Bayes-based classifier utilizes packet counts as well as packet size, and thus is better able to discriminate among instances with identical *(direction, length)* attributes but differing counts. The Bayes-based classifier retains

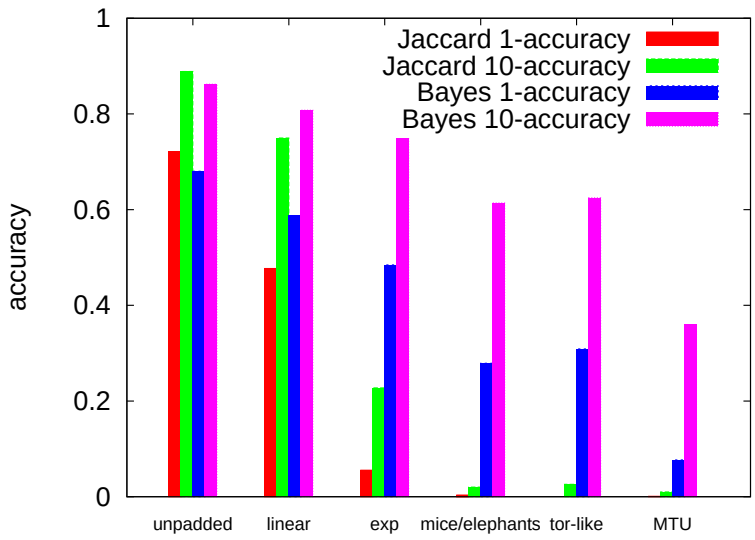

Figure 4.2. Effects of per-packet padding upon accuracy

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

Table 4.1. Per-packet padding and its effects upon accuracy and amount of data transmitted. The rightmost column shows the amount of data transmitted when using the specified padding method, relative to no padding.

enough accuracy, even under the MTU and Tor-like padding methods, to be of concern to a privacy-conscious user. Based upon this result, we believe that designers of anonymous communications systems must at a minimum utilize strong per-packet padding to preserve user privacy.

4.4 Discussion

In this chapter, we showed the resilience of profiling attacks to padding schemes. We saw that profiling attacks power is diminished only by aggressive padding, at a cost

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

Finally, we note that there are significant avenues for further exploration and understanding in preventing the profiling attack. We find three to be of particular interest:

First, there is the issue of non-deterministic padding schemes. The methods presented in this chapter are consistent from trace to trace, and thus leave more certainty among and between traces for an attacker to exploit. Introducing non-determinism, at the small cost of a random number generator, may significantly mitigate the threat of profiling attacks if the correct features can be randomized.

Second, an investigation of the cause of the efficacy of the profiling attack may yield insight into better means of counter it. While a cursory examination does not show that, for example, all easily identifiable sites have a single feature that makes them identifiable, there may be small combinations of features that do. If these features can be spread into all packet traces — even at the cost of using techniques other than padding — the profiling attack can be defeated. If only a small number of packets require modification to implement this defense, so much the better.

Third, an implementation of these defenses will allow us to measure their true performance cost. While the increase in data transmitted due to packet padding is straightforward, we suspect these techniques will interact with TCP/IP implementations is non-obvious ways. The designers of low-latency anonymity systems are reluctant to unnecessarily manipulate the traffic stream, for fear of introducing excess latency into their systems. We would like to verify that reasonable security properties can still be attained in the face of more advanced attacks, and to model the performance penalty that must be incurred to mitigate such attacks.

CHAPTER 5

ANONYMIZING TELEPHONY

In previous chapters, we have explored anonymity systems, the potential of profiling attacks against them, and the use of defensive padding to ameliorate such attacks. We now turn our attention to the applications of anonymous communication systems, in particular the application of telephony. Supporting Internet telephony (also knows as Voice over IP, or VoIP) with current anonymity systems is a hard but useful task. In this chapter, we examine the importance of this task and describe the potential difficulties. We then describe our specific contributions, which include a large measurement study, and performance and security analysis. We found that connections between four proxies located on the same continent could support aVoIP, with one-way end-to-end (i.e., initiator-to-responder) delays of under 150 ms over 90% of the time in some scenarios. However, intercontinental paths of proxies could not support aVoIP traffic so well: For paths with one such hop, 35% show acceptable performance, while two and three hops result in only 7% and 1% of the paths being of acceptable performance, respectively. We close with a discussion of the results of the study and its implications.

5.1 The Importance of Telephony

There are two interrelated reasons as to why we believe supporting anonymous VoIP (or aVoIP) is a useful and important goal. First, supporting real-time voice communication in an anonymous fashion is of critical importance. The citizens of free

and open societies require the ability to communicate without government, corporate, or criminal elements listening to and potentially chilling their speech.

Second, anonymous communication systems do not make communication invisible; they instead hide it among a crowd of users. Increasing the size of this crowd by adding more honest users can have only beneficial effects upon each user's anonymity. Increasing the diversity of the crowd, by incorporating users with different potential attackers or who live in different jurisdictions also increases the overall security of the system.

Telephony is a straightforward and intuitive communication technology. Its utility as a mechanism for speech is time-tested, though the assumption of privacy in its use is increasingly dubious, in the United States and abroad. Thus, if we can provide a strong technical foundation for anonymous telephony, we can help thaw chilled speech. Further, with such a "killer application," we expect to draw many more users to anonymity networks, which can only improve their overall utility.

5.2 Technical Challenges

Supporting telephony in current anonymity systems would clearly be beneficial to society and to the systems themselves. The obvious question then presents itself: Why don't current anonymity systems support telephony?

The answer to this question can be summarized in one word: latency. Latency is the delay between when a signal is sent, and when it arrives. When sending email, latencies of minutes are acceptable; when loading web pages, latencies of a few seconds are acceptable. When engaging in traditional telephone conversations, most people notice the adverse effects of latency if the latency exceeds the fairly tight bound of 50 ms. Cell phone, intercontinental calls, and Internet telephony such as Skype can have somewhat higher latency and lower quality than this tight bound, and have clearly accumulated large numbers of users over the past decade. The

International Telecommunication Union (ITU) suggests a one-way latency of 150 ms as the upper limit for Internet telephony, and a latency of 400 ms as an extreme upper bound for international telephone calls. While we expect that somewhat lower quality in telephony might be tolerated by privacy-conscious users, we observe that even single application-layer hops across the Internet can have performance poor enough to preclude tolerable telephony. The multiple hops that anonymity systems entail compound the probability of such bad hops appearing on the path. Packet loss is also a factor in perceived quality of Internet telephony, but our measurements showed that it was not typically a critical factor.

The challenge is two-fold. First, we must address the question of whether paths with usable performance characteristics exist. As we show in Section 5.4, paths of sufficiently high quality exist in a subset of the Internet. The problem of latency may be mitigated on larger portions of the Internet as bandwidth grows and routers get faster, but we note that physical constraints will always impose an ultimate upper limit on signal speed, and thus a lower limit on latency.

This latency limit, in combination with the underlying design of anonymous communication systems on the Internet, leads to the second challenge, and several related research questions. As we show, the existence of multihop paths of sufficient quality for telephony is a necessary but not sufficient condition for an anonymous telephony system.

Recall that such systems typically form paths randomly among a group of peers. As a given path grows, the tolerable average latency along each hop shrinks. Can end-points in a system predict the latency along the paths? If not, can the latency can be measured rather than predicted? How best to form paths, as choosing the lowest-latency path dramatically weakens the anonymity properties of the system? The last question in particular is of enduring value, as there will always be communi-

cation systems with varying classes of service that we may wish to develop anonymity systems atop. Addressing these questions is the subject of this chapter.

5.3 Architecture and Evaluation Methodology

In this section, we describe our architecture for anonymous VoIP in terms compatible with previous work. We also present our specific evaluation methodology, including performance metrics of interest.

5.3.1 Architecture

In order to evaluate the security and performance of aVoIP, we begin by describing the underlying architecture of an aVoIP system. Figure 5.1 illustrates the differences between aVoIP and related systems, namely plain VoIP, peer-to-peer VoIP (such as Skype), and anonymity systems, such as Tor. Each of the circles in this figure represents a host, and each of the links represents an application-layer hop between hosts. The VoIP illustration is a straightforward two-way connection; the peer-to-peer (or relayed) VoIP connection relies upon an intermediate host to relay traffic; this is typically required to circumvent traffic filtering at one or both end hosts.

In Tor, an *initiator* (the user) connects through a chain of *proxies* (or nodes), remote hosts that are running application-layer software that relays encrypted traffic and hides routing information from observers. These proxies relay the traffic to the user's intended recipient, called the *responder*; in Tor, this is often a Web site. The responder's traffic is returned over the same path. By default, Tor uses three intermediate proxies between the initiator and the responder. In this paper, we are concerned with an aVoIP system utilizing the same architecture as Tor, and as shorthand, we will refer to the system we evaluate as Tor or Tor-like. Thus, we assume that there are intermediate proxies between the initiator and responder.

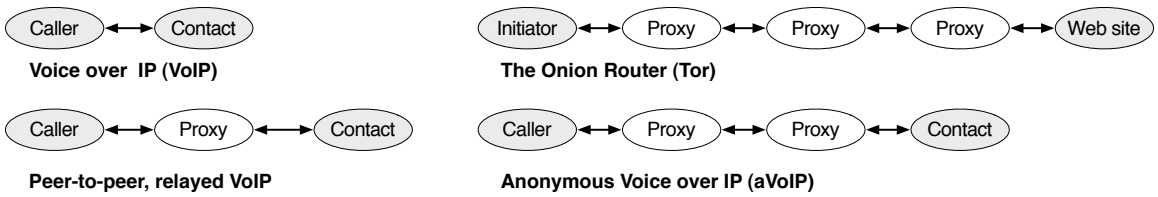

Figure 5.1. Path architecture comparison of VoIP, p2p VoIP, Tor, and aVoIP.

aVoIP's security assumptions differ from Tor in that both parties are interested in obtaining anonymity from external observers and the proxies in the aVoIP system (but not from each other). Thus we assume that the responder is running the final proxy on their own host, reducing the number of application-layer hops along the path from four to three.[1] We note that this reduction in the path length does not, by itself, lead to a significant loss of security when compared to Tor. Tor does not protect against timing analysis, by which two corrupt proxies in the first and third positions of the path could learn that they are on the same path. Levine et al. showed that this kind of analysis is very accurate when no countermeasures are in place [29]. Thus, in both Tor and aVoIP, the adversary needs to control two proxies to link the initiator with the responder.

5.3.2 Performance Metrics

While bulk-transfer and low-latency anonymous systems typically focus on whether *nodes* are simply up or down, deadline-oriented applications like VoIP require that the *path between pairs of nodes* be of sufficiently high quality. Three low-level metrics are typically used to characterize path quality for streaming multimedia applications such as VoIP: loss, latency, and jitter. Additionally, the ITU has developed a family of metrics for telephony. In this study, we use the E-model, described in ITU-T G.107. We briefly describe each of the metrics and their relevance below.

[1]The initiator and responder do not need to allow their Tor proxies to serve as proxies for other users, although doing so can increase their security.

- *Latency*: Latency from sender to receiver on the Internet is the sum of processing, queuing, propagation, and transmission delays. Each proxy and router adds each type of delay. Increasing latency increases the delay between when one user speaks and the other hears that speech — the negative correlation between latency and quality is obvious.

  Tor paths consist of four application-layer hops, and are thus typically of much greater logical and physical length than typical Internet paths. Further, when hops are intercontinental, latency increases dramatically. A one-way latency of over 150 ms is considered unacceptable for traditional telephony. EM radiation propagates at about $3\dot{1}0^8$ meters/sec in a vacuum, and closer to $2\dot{1}0^8$ m/s along copper wires. In 150ms, signals can travel about 30,000 km through copper. For comparison, the Earth is about 40,000 km in circumference. Propagation delay alone will not allow us to route a signal around the world and maintain latency acceptable for traditional telephony. Privacy-conscious users may accept a higher latency for more anonymity — they already do so when web browsing with Tor — but such a service clearly has limited appeal. We are not aware of experiments that demonstrate such tolerance.

- *Loss rate*: Packet loss is the result of either transmission errors resulting in a failed checksum or full queues in routers or at proxies. VoIP protocols can incorporate redundancy (e.g., forward error correction or FEC) into their audio codecs to compensate for arbitrary amounts of packet loss at the cost of extra bandwidth. Because VoIP streams are generally sent UDP rather than TCP, they do not reduce their rate during packet loss (a sign of congestion) unless designed to do so by the application programmer. The use of FEC and the human ear's tolerance for small disruptions in audio mean that the relationship between loss rate and perceived quality is not straightforward: small amounts of loss have no perceptible impact on quality, but once a codec-dependent threshold

is reached, quality drops off dramatically. We discuss this in more detail when discussing the E-model, below.

- *Jitter*: Jitter is an estimate of the statistical variance in packet interarrival time. VoIP decoding occurs in close to real time, and it requires that packets be available by a playout deadline, or they are essentially lost. The primary way to counter the effects of jitter is to buffer packets at the receiver ahead of playback, but this has the secondary effect of increasing latency. For interactive audio (and video) there is a limit to such delays that are tolerable to users. It is because of jitter requirements that VoIP cannot be supported with TCP.

- *E-model*: The E-model, described in ITU-T G.107, is a tool to assist in planning and evaluating telephony systems running on a computer network. Ideally, such systems are evaluated by a panel of experts who use the system and make judgments about its quality, resulting in a Mean Opinion Score (MOS) ranging from one to five. Unfortunately, such evaluations are expensive and time-consuming. The E-model is a quantitative model designed to approximate subjective judgments. A host of factors, ranging from loss and latency to loudness and room noise are measured and input into the model, which outputs a composite R score. This score can be mapped onto the one-to-five scale of the MOS and closely approximates the score a panel of experts would likely give. Table 5.1 provides a mapping from MOSs and R-scores to subjective terms.

  Cole and Rosenbluth [10] examined the E-model in the specific context of VoIP on the Internet. Their goal was to simplify the E-model's twenty terms, many whose values are irrelevant in network evaluation, to a more useful number. We omit the details of their simplification here, and show the final equation:

R-score	$90 < \mathrm{R} \leq 100$	$80 < \mathrm{R} \leq 90$	$70 < \mathrm{R} \leq 80$	$60 < \mathrm{R} \leq 70$	$50 < \mathrm{R} \leq 60$
MOS	4.34 — 4.5	4.03 — 4.34	3.60 — 4.03	3.10 — 3.60	2.58 — 3.10
Quality	Best	High	Medium	Low	Poor

Table 5.1. Mapping between quantitative and qualitative VoIP scores

$$\begin{aligned} R \;\sim\; & 94.2 - 0.024(d_{network} + 85) \\ & -0.11(d_{network} - 92.3)H(d_{network} - 92.3) \\ & -11 - 40\ln[1 + 10(e_{network} + (1 - e_{network})e_{de-jitter})] \end{aligned}$$

where $d_{network}$ is the end-to-end one-way network latency, $H(x)$ is the Heaviside step function (equal to 1 if $x >= 0$, and 0 otherwise), $e_{network}$ is the packet loss rate, and $e_{de-jitter}$ is the de-jittering delay. The constants are derived from measurements, and their derivation is described in detail in their work. We use this quantitative estimator of the R-score to evaluate paths in our measurements.

5.4 Evaluation

5.4.1 Measurement Setup

Supporting aVoIP requires forming multi-hop paths with sufficiently low latency, loss, and jitter. To see if such paths are available on the Internet we used the PlanetLab testbed and deployed to each host a tool capable of generating and measuring VoIP-like UDP traffic. As we focus here on network performance, we developed and used a small measurement tool that mimicked the behavior of a Tor-like anonymizing proxy capable of carrying UDP traffic. PlanetLab is composed of hosts of varying capability and varying background load, spread across the Internet and the world. In that sense, it is representative of hosts on the Internet in general. On the other hand, virtually all PlanetLab hosts are in academic autonomous systems (ASes), and this fact may bias the data in subtle ways — for example, many academic ASes in the US

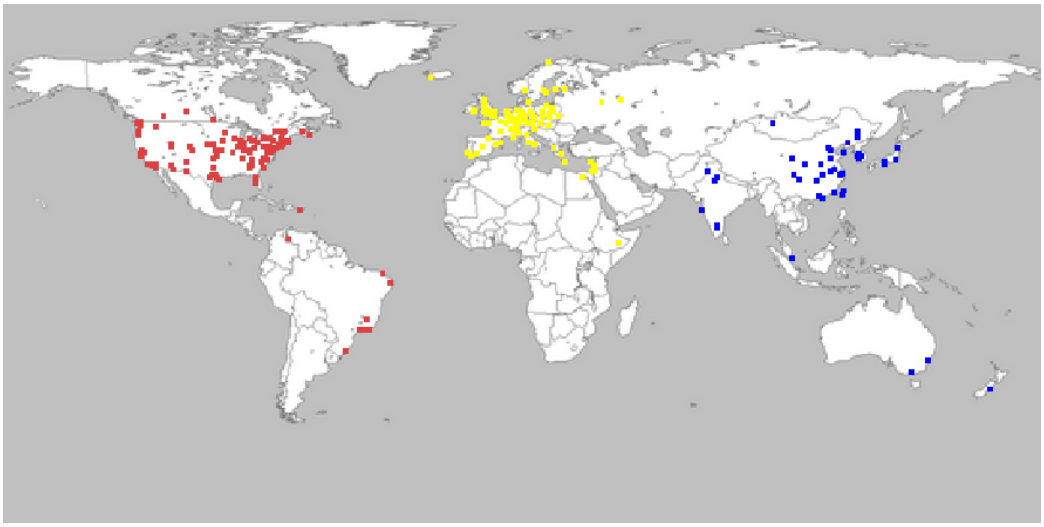

Figure 5.2. This map shows the diversity of the geographic locations of PlanetLab nodes.

Set	Hosts
Asia	40
Americas	49
Europe	121

Table 5.2. Average number of PlanetLab hosts up at any one time in each geographic area.

are connected through the high-speed Internet2 backbone. Despite this limitation, our finding of good paths on PlanetLab shows that good paths exist on the Internet, at least within PlanetLab and likely elsewhere.

The path we constructed in PlanetLab are analogous to Tor paths, consisting of an initiator, three intermediate nodes with associated application-layer hops, and a responder. For simplicity, we assume the responder is co-located with the last intermediate node. This assumption is reasonable, as an aVoIP user will want the cover traffic that running a Tor node generates to hide their own communication patterns. Our measurement protocol is as follows.

- **Preliminaries**: We are interested in performance within intracontinental and intercontinental geographic areas. We generated three sets of PlanetLab hosts,

corresponding to hosts located in Asia, Europe, and North America (see Figure 5.2). Each set typically had at least 40 active hosts, as shown in Table 5.2. As detailed below, we pruned inactive hosts from the lists dynamically. We refer to these sets as `asia`, `europe`, and `usa` in the remainder of this manuscript.[2] We define intracontinental hops as the hops that occur between two hosts in the same set, e.g., two hosts in `asia`. We define intercontinental hops as the hops that occur between two hosts in different sets.

- **Setup**: We performed measurements under four scenarios. For the first set, we constructed Tor-like paths by choosing four hosts uniformly at random without replacement from `asia`. The second and third scenario were formed analogously, but using hosts chosen from `europe` and `usa`. The final scenario, which we call `intercontinental`, was generated choosing a set uniformly at random for each of the four hosts, and then choosing that host from the corresponding set. The `intercontinental` scenario is thus a superset of the other three scenarios and encompasses varying numbers of intercontinental hops.

- **Measurement**: For each of the four scenarios, we performed measurements from February 22, 2007 to May 4, 2007. A master process performed the following steps continuously:

  1. Choose a new set of hosts according to the scenario.
  2. Connect to each host in the path, installing and running the measurement program.
  3. Perform a series of 10 UDP pings, measuring sending and arrival times at each hop along the path.

[2]The contents of these sets, our measurement code, and our data will be made available upon publication at `http://traces.cs.umass.edu/`.

4. Perform a test of one-way streaming data, sending UDP packets with an interarrival time of 20 ms, with an effective bitrate of 16 kbps. The actual bandwidth consumed was closer to 28.5 kbps due to IP and UDP headers. This bitrate is representative of the most common VoIP codecs and encapsulations, and corresponds to the assumptions in Cole and Rosenbluth [10]. Measure sending and arrival times at each node along the path.

5. Close down all measurement programs.

- **Additional considerations**: Hosts went up and down over the course of our measurements. We dynamically removed hosts from the set when connections to them failed, and we occasionally probed them in a separate process, returning them to the active set if they came back up. Several PlanetLab hosts had to be permanently removed from our sets due to apparent ingress or egress filtering on their Internet connections stopping our UDP measurements. In sum, our measurements are based on an average of 210 hosts.

From these measurements, we can compute round trip times, loss rates, and jitter. Unfortunately, we found the clocks on PlanetLab systems insufficiently synchronized to compute millisecond-granularity metrics with meaningful precision. Thus, we assume that one-way latency is half of the round trip time; this is a simplifying assumption and a limitation of this work, as some paths on the Internet are asymmetric in terms of delay.

5.4.2 Measurement Results

As described in Section 5.4.1, we set up multi-hop paths between proxies on PlanetLab and performed measurements on these paths. From the measurements, we can compute round trip times, loss rates, and jitter. We found that the clocks on PlanetLab systems are synchronized, but with an insufficient level of precision to

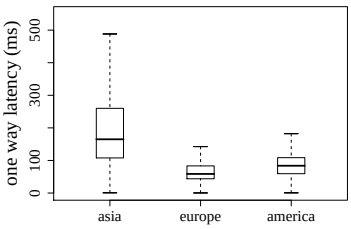

Figure 5.3. Intracontinental: Distributions of mean estimated one-way latency for paths within regions.

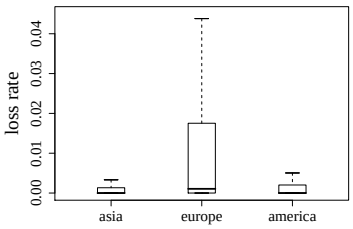

Figure 5.4. Intracontinental: Distributions of mean loss rate for paths within regions.

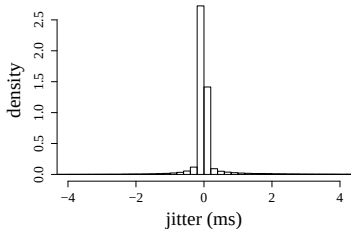

Figure 5.5. Intracontinental: Distributions of mean instantaneous jitter across all intracontinental measurements.

meaningfully compute millisecond-granularity metrics. Thus, we assume that one-way latency is half of the round trip time; this is a simplifying assumption, as paths on the Internet can be asymmetric in terms of delay. If the paths are asymmetric in latency, then our measurements of latency and R-scores under-estimate the variances but do not affect the averages. We base our conclusions about aVoIP largely on the averages we observed.

The box-and-whisker plot in Figure 5.3 shows the distribution of the means of the per-path one-way latency we observed for the three intracontinental scenarios. (The whiskers show the min and max values, the box is the semi-interquartile range, and the thick bar inside the box represents the median of the distribution.) Figure 5.4 shows the loss rate distributions for the three intracontinental scenarios. In all three scenarios, we observe that the majority of one-way latencies are within the ITU's limit for adequate quality, 150 ms. In the `europe` and `america` scenarios, virtually all paths we observed had this property. We conjecture that the higher latencies in the `asia` set were due to higher loads on the hosts and on the links between the hosts; unfortunately, the virtualization software that enables PlanetLab makes accurate measurement of these loads infeasible. Loss rates were also quite low; the median loss rate was zero in the `asia` and `america` scenarios, and 0.001 in the `europe` scenario.

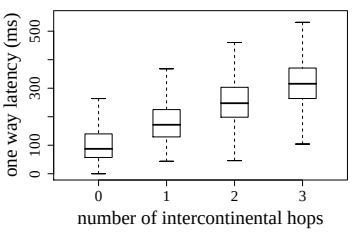

Figure 5.6. Intercontinental: Distributions of mean estimated one-way latency for paths with a given number of hops across regions.

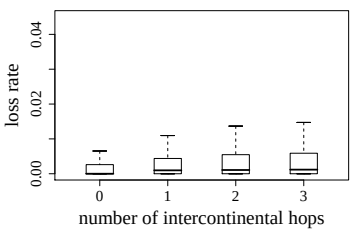

Figure 5.7. Intercontinental: Distributions of mean loss rate for paths with a given number of hops across regions.

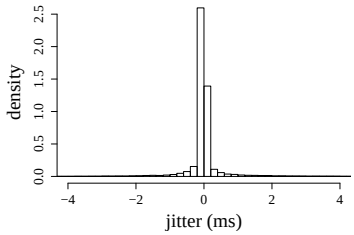

Figure 5.8. Intercontinental: Distributions of mean instantaneous jitter across all intercontinental measurements.

Figures 5.6 and 5.7 show the corresponding data for the intercontinental measurements, arranged by number of intercontinental hops. As expected, latency and loss rate increase with more hops between continents. Some of this effect can be attributed to the greater physical distance between hosts, and we conjecture some is due to load on links themselves.

Figures 5.5 and 5.8 show the distribution of the observed mean instantaneous jitter over intra- and intercontinental scenarios as a PDF. We found that the mean jitter was quite low in both scenarios: over 97% of all observed jitter was less than 3 ms, implying jitter was not a serious problem on the hosts.

As described in Section 5.3.2, the R-score is calculated on the basis of latency, loss, and the de-jitter buffer. To calculate the R-score for each measurement of a path, we assumed a static de-jitter buffer of 3 ms, on the basis of the jitter we observed (see Figs. 5.5 and 5.8). Any packets that were delayed longer than this value due to jitter were added to the loss rate for the measurement in question. We did not assume that loss was hidden by the transport layer, i.e., we assumed no packet-level forward error correction. As a result, the R-scores we show here are conservative estimates of the quality that could be supported on the paths we examined; non-experimental aVoIP implementations would likely implement adaptive jitter buffers.

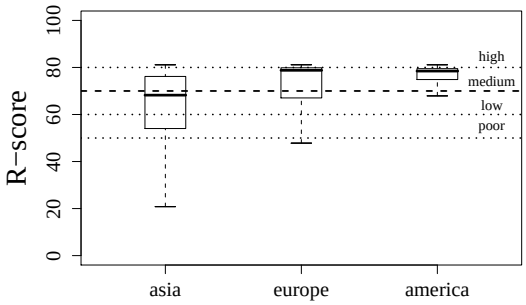

Figure 5.9. Intracontinental: R-score for paths in each region. Horizontal lines demarcate estimated quality levels.

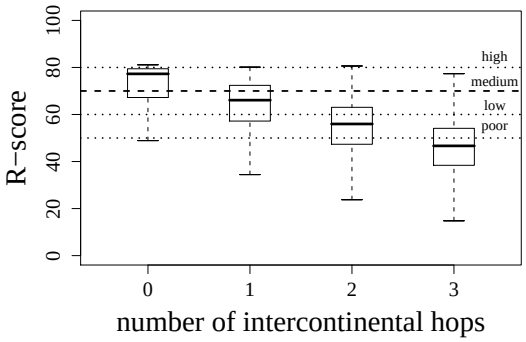

Figure 5.10. Intercontinental: R-score for paths with a given number of hops across regions. Horizontal lines demarcate estimated quality levels.

Figures 5.9 and 5.10 show the distributions of the calculated R-scores for paths formed according to intra- and intercontinental scenarios, respectively. The horizontal lines demarcate the varying quality levels, as described in Table 5.1. Of particular note is the line at $R = 70$, which is considered to be the cutoff between "medium" and "low" estimated quality, as well as the value we use to decide if connection quality is acceptable to support aVoIP.

5.4.3 Discussion

Our measurements suggest that aVoIP systems are feasible in most scenarios within continents and some scenarios on the global Internet. For intracontinental paths, in which proxies were co-located on the same continent, 71% of all observed paths had good enough quality of service to support aVoIP. More specifically, the percentage of measurements with acceptable quality was 46% in `asia`, 71% in `europe`, and 86% in `america`. This result is good news: Tor and similar low-latency anonymity systems are designed to protect against a local adversary, and if users can choose proxies spread around a continent, it is unlikely that each proxy will fall within the range of a sniffer on the local network, the local ISP, local law enforcement, etc.

The results for intercontinental quality were not as promising. As the number of intercontinental hops along the path increases, the percentage of our randomly formed paths with acceptable performance drops. For paths with one such hop, 35% show acceptable performance, while two and three hops result in only 7% and 1% of the paths being of acceptable performance. This decline is clearly due to the increased latency and loss along these paths.

These results have several implications. First, choosing proxies at random from among all such proxies in the world to form a path for aVoIP is unlikely to result in paths with acceptable performance. Thus, it will critical that Tor and similar systems allow paths to be formed on the basis of path performance. These systems do not currently permit such functionality, as it increases the vulnerability of the systems to certain forms of attack. The vulnerability arises because of the difficulty in measuring path performance in the presence of malicious nodes. Such nodes cannot be trusted to self-report performance accurately, implying that a trusted authority (such as the directory servers in Tor) must perform periodic measurements of performance. But malicious nodes can still attempt to corrupt the measurements; we explore this idea more fully in Section 5.5.

Second, there is a tradeoff present between path performance and the increased security that geographically dispersed proxies provide, subject to the concerns outlined by Feamster and Dingledine [18]. Murdoch and Zielinski show that this concern is real by experimentally determining that a single Internet exchange (IX) in the UK could observe as many as 27% of Tor connections [34]. Callers located on the same continent who wish to capitalize on location diversity must utilize an even number of intercontinental hops (i.e., zero or two), and thus limit the fraction of paths available to them. Whether this limit reduces their anonymity more than the increase afforded by location diversity depends upon the threat model.

Finally, we note that limitations on the fraction of paths with acceptable quality are not as constraining as they may appear. If the anonymity system has thousands of proxies, curtailing the paths to the best $x\%$ will be a problem only for small values of x, or in the case when an attacker can overly influence which proxies are in the best $x\%$.

5.5 Attacker Measurement Manipulation

Our proposed architecture for aVoIP, as described in Section 5.3, is subject to same threats as any similar p2p-based anonymous communication system, including Sybil, timing, and predecessor attacks. In this section[3], we focus on a vulnerability that is unique to aVoIP. To form paths with reasonable performance for aVoIP, users will have to rely on a quantitative metric of path quality, such as the R-score we described in Section 5.3.2. Note that a user cannot rely upon nodes to self-report performance, as malicious nodes can lie. Specifically, malicious nodes can falsely claim higher performance then they can actually deliver. Thus, we assume that users will test paths, or utilize the test results provided by trusted hosts, such as the directory servers in Tor.

Even with actual measurement, attackers can falsify performance test results so that other nodes appear to perform poorly. In doing so, they can artificially become a larger percentage of the peer group. As a result, attackers are more likely to compose the full path, lessening the time required to complete a successful predecessor attack. Our goal in this section is to quantify the effectiveness of this attack. In particular, we show that an attacker with limited resources is not very effective in making the predecessor attack practical, but that more powerful attackers can actively leverage their position to compromise user anonymity in about half the time a passive attack

[3]The work in this section was done in collaboration with Matthew Wright, who derived the analysis and performed the simulations described in this section.

would require. These results assume that the attacker can optimally degrade all mixed paths while attaining high performance for all attacker paths. A real attacker may have to adjust his attack to avoid detection and may not have high performance between all pairs of attacker-controlled nodes. We note that these attacks only affect users who form paths on the basis of path performance, and that the increased risks are a necessary cost of ensuring adequate performance for telephony.

Can the attack we describe in this section be prevented? The aVoIP measurement and path selection system is a distributed reputation system, and from one viewpoint, our attacker is a Sybil, that is, a single entity that controls a group of nodes. This view allows us to directly apply Cheng and Freidman's [9] result that states that this attack is impossible to defeat unless the reputation system is ineffective — nodes would form paths with only nodes they trust completely in the first place, which implies a very small set of anonymous peers and little anonymity. The Sybil attack cannot be defeated in an inexpensive fashion [17], though it can be detected in some scenarios [31]. However, our aVoIP design is only moderately more susceptible than Tor or other existing systems to the Sybil attack.

5.5.1 Attacker and Measurement Model

We are primarily concerned with the predecessor attack, as it is among the most powerful attacks that Tor-like path formation is subject to. Further, the use of non-random path formation for aVoIP affords the attacker an opportunity to optimize the attack by influencing user measurements. To conduct the predecessor attack, the attacker controls a subset of the nodes and simply waits for the user to randomly select attacker-controlled nodes for both ends of the path. The attacker, with c out of n Tor nodes, gets this position on $\frac{c^2}{n^2}$ of the user's paths [56], and he can use timing analysis on the user's packet stream to confirm that the initiator is communicating with the responder.

We model the attack on a system of n nodes with Tor-like path formation, where performance measurements are taken periodically by a trusted entity, such as the directory server, and provided to users of the system. We assume that there are h honest nodes and c attacker-controlled nodes, such that $n = h + c$. We consider paths with two intermediate nodes, as described in Figure 5.1. Let us divide the set of p valid paths among the n nodes into three groups: p_h *honest paths* consisting of two honest intermediate nodes, p_a *attacker paths* consisting of two attacker-controlled intermediate nodes, and p_m *mixed paths* consisting of one honest intermediate node and one attacker-controlled intermediate node. Following on the results of our performance measurements, let us say that $p^* \leq p$ paths provide acceptable performance (i.e. they have adequate R-scores) according to the user's requirements. Breaking this down further, $p_h^* \leq p_h$ honest paths, $p_a^* \leq p_a$ attacker paths, and $p_m^* \leq p_m$ mixed paths have adequate performance.

We assume that the user has the measured performance values from the directory server and selects only paths with adequate performance. The chance that the attacker controls a given path would normally be given by:

$$P_{normal} = \frac{p_a^*}{p^*} = \frac{p_a^*}{p_a^* + p_h^* + p_m^*} \tag{5.1}$$

On expectation, this attack would take $Calls_{normal} = 1/P_{normal}$ aVoIP calls between the same initiator and responder.

We now evaluate the attacker's attempts to manipulate measurements to increase his chances of controlling the user's paths. The attacker's ability to modify the system measurements is limited. When an attacker path is being measured, he can attempt to ensure that the path quality is adequate. Let us conservatively assume that he is always successful, e.g. that $p_a^* = p_a$. We assume that he cannot influence the measurement of honest paths. However, he can influence mixed paths. From Equation 5.1, we see that the attacker can increase his chance of controlling the

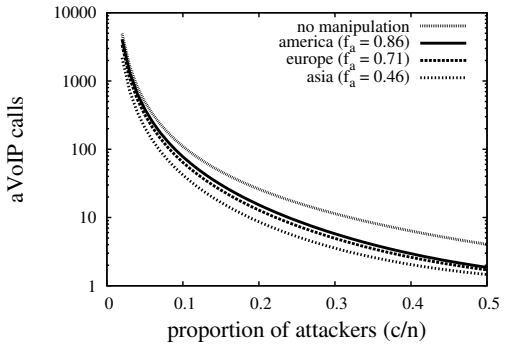

Figure 5.11. Manipulating Measurements: For different scenarios, the expected number of aVoIP calls between the same pair of users required for the attacker to control the path.

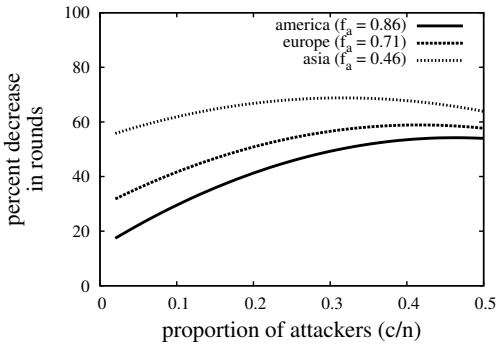

Figure 5.12. Attacker Improvement: For different scenarios, the percentage decrease of expected aVoIP calls between the same pair of users required for the attacker to control the path once.

path by lowering p_m^*. Let us assume that he is also always successful in this, e.g. that $p_m^* = 0$. The node performing the measurements could attempt to discover which nodes are manipulating measurements, but this task is difficult for several reasons. First, exhaustively measuring all paths may be infeasible, leaving the node with insufficient information to detect attackers. Second, the performance of paths on the Internet can fluctuate, increasing the uncertainty of detecting manipulation of the measurements. Finally, the attacker nodes can attempt to thwart detection by varying their performance over time.

Based on an attacker manipulating the measurements as described above, we see that the chance that he controls a given path is: $P_{manip} = \frac{p_a}{p_a + p_h^*}$. This gives us the expected number of aVoIP calls required for the attacker to succeed:

$$E[Calls_{manip}] = \frac{p_a + p_h^*}{p_a} \tag{5.2}$$

We apply Equation 5.2 to three scenarios based on our measurements of fraction of adequate paths, f_a: `americas` ($f_a = 0.86$), `europe` ($f_a = 0.71$), and `asia` ($f_a = 0.46$).

From Figure 5.11, we see that the attacker does better for lower fractions of adequate paths. This is expected, as we have assumed that the attacker paths are always adequate, leading to their higher proportion in the pool of adequately performing paths. We also notice that the attacker has limited gains when the attacker has few nodes in the system. For example, for 86% adequate paths and 5% attacker-controlled nodes, the attacker reduces the expected number of aVoIP calls needed for the attacker to control the path from 495 to 385, a 22% decrease. While this is a signficant decrease, either number represents a large number of phone calls.

When the attacker is stronger, however, manipulation of the measurements makes the predecessor attack much more effective. When the attacker controls 33% of the nodes, we get a drop from 9.4 calls to 4.6 calls — over 50%. For such an attacker, however, the number of calls is rather small in either case. Even without the attacker being able to manipulate the measurements, the system is insecure. Nevertheless, there are clearly intermediate fractions of attackers for which measurement manipulation is beneficial to accelerate the predecessor attack. A concerned user would need to limit the number of aVoIP calls she makes to the same party or to a group of parties to avoid exposure.

Overall, our proposed measurement and path selection framework retains most of the defenses of Tor against compromised node attacks. We see that when the fraction of attacker-controlled nodes is relatively small, and Tor is therefore somewhat resistant against attacks like the predecessor attack, measurement manipulation does not give attacker major gains in his effectiveness. However, when the fraction of attacker-controlled nodes is large, such that Tor is already quite vulnerable to the predecessor attack, measurement manipulation can significantly accelerate the attack. Given the potential benefits of such measurement, specifically the enabling of a new class of traffic in aVoIP, the security tradeoffs are worthwhile.

5.6 Conclusion

Overall, this study indicates that aVoIP with acceptable perceived quality is feasible for a useful subset of paths on the Internet. In particular, proxies that are geographically separated by large distances, but not globe-spanning distances, have sufficiently high performance to support aVoIP. These paths are similar to those currently in use for Tor, and so we believe implementing a UDP-aware Tor capable of carrying VoIP traffic is a reasonable goal. Supporting many intercontinental paths may be possible as routers improve, but will always have lower performance, as high latency imposed by speed-of-light constraints will take its toll. In sum, aVoIP paths cannot be constructed oblivious to the performance of the underlying network. Further, such paths have additional potential problems, as increasing location diversity often implies decreasing diversity in the underlying physical paths. Examining this tradeoff could be interesting future work.

On the basis of this study, we propose to have the Tor system perform path measurements and report the results using the trusted directory servers. This allows users to select paths with sufficient quality of service for VoIP. However, the requirement that performance between proxies be a part of path construction introduces a new attack. Our analysis demonstrates that attackers can leverage their position to make the predecessor attacker faster, although this optimization is most powerful only when the predecessor attack is already very efficient. Overall, we have shown that it is possible to make secure and well-performing aVoIP a reality while retaining most of the security benefits of Tor against compromised proxies. We believe that implementing aVoIP in Tor could provide an important privacy service to potentially many users.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter we offer some conclusions about our work in low-latency anonymity systems. We also offer several items of future work.

6.1 Conclusion

Users want private, anoymous communication on the Internet, and they want such capabilities to be transparent and easy-to-use. Unfortunately, their other desires, such as low latency, interact with the architecture of the Internet to make assuring anonymity difficult.

This dissertation presents a careful experimental evaluation of one way in which responder anonymity can be attacked in current, low-latency anonymity system on the Internet. We also describe and evaluate a set of defenses against such attacks. Finally, we examine the feasibility of extending current systems to a whole new class of services.

We present the profiling attack, a method for determining the content of encrypted network traffic, and thus breaking responder anonymity. This attack requires that an adversary collect a large set of network traffic corresponding to potential responders, extract salient features from this traffic, create a set of profiles based on these features, and be able to classify newly observed traffic on the basis of these profiles. We evaluate the feasibility of each of these steps, and show that methods exist for each that are effective and should scale to the size of the Internet.

We present and evaluate a defense to profiling attacks, based upon obfuscating the features used by such attacks. We show such defenses can work, at a non-trivial cost in bandwidth. Of particular interest, we show that the version of these defenses in use by current systems is inadequate to defend against the profiling attack.

We examine the feasibility of extending current systems to a new class of service, telephony. Such service is extremely sensitive to latency and loss. We show the results of a detailed measurement study indicating the possibility of extending low-latency anonymity systems to support such services. Extensions of this type are vital to increasing the number of potential users of such systems, which in turn improves the degree of anonymity they can be expected to provide.

6.2 Future Work

We have demonstrated a working system for user privacy and responder anonymity and a method by which such attacks can be mitigated. We have also provided evidence that current anonymity systems can be extended to support anonymous telephony. Still, several open questions remain to be addressed.

Identification of encrypted traffic through the use of classification techniques depends upon good feature extraction and a robust classification algorithm. In this dissertation, we argue that there are only two primary features that can be extracted from an encrypted packet stream: packet sizes and interarrival times. Regardless, there are many secondary features that might be inferred from the data stream that could aid in classification. Identifying these features and methods for extracting them is a topic of future research. Similarly, the choice of the classification algorithms in the work we present was based upon general performance parameters. Creating algorithms and models tailored to the specific task of classifying encrypted traffic could further improve the results we have presented here.

The mitigation of the profiling attack we presented is based upon traffic shaping. In this dissertation, we explored strictly deterministic traffic shaping; the use of packet padding and splitting to force packets to conform to specific distributions. While reasonably effective, these schemes resulted in rather large bandwidth overhead. Exploration of non-deterministic systems for modifying the primary features of an encrypted traffic stream would be of value to system designers seeking to incorporate such defenses.

We have given evidence that anonymous telephony is feasible on the Internet. Still, several key questions remain: How can path performance be measured in a distributed and trusted anonymous fashion? What other characteristics, such as the underlying physical components of the paths, must be factored into path formation? What is the optimal strategy for an attacker in control of nodes within such a system? Addressing questions is a rich area for future work.

In conclusion, we have prevented and evaluated an important class of attacks against real-world anonymity systems. We have shown the seriousness of such attacks, and the cost of mitigating them. Finally, we showed the feasibility of expanding the domain of real-world systems to include a new service, telephony, and examined some of the security tradeoffs such a system would imply.

BIBLIOGRAPHY

[1] Abe, Masayuki. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Proceedings of EUROCRYPT 1998* (1998), Springer-Verlag, LNCS 1403.

[2] Acquisti, Alessandro, Dingledine, Roger, and Syverson, Paul. On the Economics of Anonymity. In *Proceedings of Financial Cryptography (FC '03)* (January 2003), Rebecca N. Wright, Ed., Springer-Verlag, LNCS 2742.

[3] Berger, Yigael, Wool, Avishai, and Yeredor, Arie. Dictionary attacks using keyboard acoustic emanations. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security* (New York, NY, USA, 2006), ACM Press, pp. 245–254.

[4] Boneh, Dan, and Golle, Philippe. Almost entirely correct mixing with application to voting. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)* (Washington, DC, November 2002), Vijay Atluri, Ed., pp. 68–77.

[5] Brown, Zach. Cebolla: Pragmatic ip anonymity. In *Ottawa Linux Symposium* (June 2002).

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

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

[8] Chaum, David. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 4*, 2 (February 1981).

[9] Cheng, Alice, and Friedman, Eric. Sybilproof reputation mechanisms. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems (P2PEcon)* (August 2005), pp. 128–132.

[10] Cole, R. G., and Rosenbluth, J. H. Voice over ip performance monitoring. *SIGCOMM Comput. Commun. Rev. 31*, 2 (2001), 9–24.

[11] Cottrell, Lance. Mixmaster and remailer attacks. `http://riot.eu.org/anon/doc/remailer-essay.html`.

[12] Danezis, George. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)* (May 2004), vol. 3424 of *LNCS*.

[13] Danezis, George, Dingledine, Roger, and Mathewson, Nick. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (May 2003).

[14] Díaz, Claudia, Seys, Stefaan, Claessens, Joris, and Preneel, Bart. Towards measuring anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)* (April 2002), Roger Dingledine and Paul Syverson, Eds., Springer-Verlag, LNCS 2482.

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

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

[17] Douceur, John. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)* (March 2002).

[18] Feamster, Nick, and Dingledine, Roger. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)* (Washington, DC, USA, October 2004).

[19] Freedman, Michael J., and Morris, Robert. Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security* (New York, NY, USA, 2002), ACM Press, pp. 193–206.

[20] Furukawa, Jun, and Sako, Kazue. An efficient scheme for proving a shuffle. In *Proceedings of CRYPTO 2001* (2001), Joe Kilian, Ed., Springer-Verlag, LNCS 2139.

[21] Goldberg, Ian. On the security of the tor authentication protocol. In *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)* (Cambridge, UK, June 2006), Springer.

[22] Goldschlag, David M., Reed, Michael G., and Syverson, Paul F. Hiding Routing Information. In *Proceedings of Information Hiding: First International Workshop* (May 1996), R. Anderson, Ed., Springer-Verlag, LNCS 1174, pp. 137–150.

[23] Golle, Philippe, Jakobsson, Markus, Juels, Ari, and Syverson, Paul. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track* (San Francisco, USA, February 2004).

[24] Gulcu, Ceki, and Tsudik, Gene. Mixing email with babel. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)* (Washington, DC, USA, 1996), IEEE Computer Society, p. 2.

[25] Helmers, Sabine. A brief history of anon.penet.fi – the legendary anonymous remailer. *Computer Mediated Communication* (September 1997).

[26] Hintz, Andrew. Fingerprinting websites using traffic analysis. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)* (April 2002), Roger Dingledine and Paul Syverson, Eds., Springer-Verlag, LNCS 2482.

[27] Jakobsson, Markus. Flash Mixing. In *Proceedings of Principles of Distributed Computing - PODC '99* (1999), ACM Press.

[28] Kesdogan, Dogan, Egner, Jan, and Büschkes, Roland. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)* (1998), Springer-Verlag, LNCS 1525.

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

[30] Levine, Brian Neil, and Shields, Clay. Hordes — A Multicast Based Protocol for Anonymity. *Journal of Computer Security 10*, 3 (2002), 213–240.

[31] Margolin, N. Boris, and Levine, Brian Neil. Informant: Detecting sybils using incentives. In *Proceedings of Financial Cryptography (FC)* (February 2007).

[32] Mathewson, Nick, and Dingledine, Roger. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)* (May 2004), vol. 3424 of *LNCS*.

[33] Moeller, Ulf, and Cottrell, Lance. Mixmaster protocol version 3. `http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-v3-01.txt`.

[34] Murdoch, Steven. Hot or Not: Revealing Hidden Services by their Clock Skew. In *Proceedings of the ACM conference on Computer and Communications Security (CCS)* (October 2006).

[35] Murdoch, Steven J., and Danezis, George. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy* (May 2005), IEEE CS.

[36] Nambiar, Arjun, and Wright, Matthew. Salsa: a structured approach to large-scale anonymity. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security* (New York, NY, USA, 2006), ACM Press, pp. 17–26.

[37] Neff, C. Andrew. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)* (November 2001), P. Samarati, Ed., ACM Press, pp. 116–125.

[38] Ohkubo, Miyako, and Abe, Masayuki. A Length-Invariant Hybrid MIX. In *Proceedings of ASIACRYPT 2000* (2000), Springer-Verlag, LNCS 1976.

[39] Parekh, Sameer. Prospects for remailers. *First Monday 1*, 2 (1996). `http://www.firstmonday.org`.

[40] Pfitzmann, Andreas, and Hansen, Marit. Anonymity, unlinkability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. Tech. Rep. v0.28, Tu Dresden, Department of Computer Science, Institute for System Architecture, 2006.

[41] Pfitzmann, Andreas, Pfitzmann, Birgit, and Waidner, Michael. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems* (February 1991), pp. 451–463.

[42] Pfitzmann, Andreas, and Waidner, Michael. Networks without user observability – design options. In *Proceedings of EUROCRYPT 1985* (1985), Springer-Verlag, LNCS 219.

[43] Rackoff, Charles, and Simon, Daniel R. Cryptographic defense against traffic analysis. In *Proceedings of ACM Symposium on Theory of Computing* (1993), pp. 672–681.

[44] Reed, Michael G., Syverson, Paul F., and Goldschlag, David M. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication: Special Issue on Copyright and Privacy Protection 16*, 4 (May 1998).

[45] Reiter, Michael, and Rubin, Aviel. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security 1*, 1 (June 1998).

[46] Rennhard, Marc, and Plattner, Bernhard. Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society* (New York, NY, USA, 2002), ACM Press, pp. 91–102.

[47] Serjantov, Andrei, and Danezis, George. Towards an information theoretic metric for anonymity. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)* (April 2002), Roger Dingledine and Paul Syverson, Eds., Springer-Verlag, LNCS 2482.

[48] Serjantov, Andrei, and Murdoch, Steven J. Message splitting against the partial adversary. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2005)* (May 2005).

[49] Serjantov, Andrei, and Sewell, Peter. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003* (October 2003).

[50] Sun, Qixiang, Simon, Daniel R., Wang, Yi-Min, Russell, Wilf, Padmanabhan, Venkata N., and Qiu, Lili. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy* (Berkeley, California, May 2002), p. 19.

[51] Syverson, Paul, Tsudik, Gene, Reed, Michael, and Landwehr, Carl. Towards an Analysis of Onion Routing Security. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (July 2000), H. Federrath, Ed., Springer-Verlag, LNCS 2009, pp. 96–114.

[52] Teich, Al, Frankel, Mark S., Kling, Rob, and ching Lee, Ya. Anonymous communication policies for the internet: Results and recommendations of the AAAS conference. *The Information Society 15*, 2 (1999).

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

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

[55] Wright, Matthew, Adler, Micah, Levine, Brian Neil, and Shields, Clay. Defending Anonymous Communication Against Passive Logging Attacks. In *Proceedings IEEE Symposium on Security and Privacy (Oakland)* (May 2003), pp. 28–41.

[56] Wright, Matthew, Adler, Micah, Levine, Brian Neil, and Shields, Clay. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC) 4*, 7 (November 2004), 489–522.