7-2010

Correlation-Based Traffic Analysis Attacks on Anonymity Networks

Ye Zhu
Cleveland State University, y.zhu61@csuohio.edu

Xinwen Fu
Dakota State University, xinwen.fu@dsu.edu

Byran Gramham
Texas A & M International University, bgraham@tamu.edu

Riccardo Bettati
Texas A & M University - College Station, bettati@cs.tamu.edu

Wei Zhao

Publisher's Statement

Original Citation

Ye, Z., Xinwen, F., Graham, B., Bettati, R., & Wei, Z. (2010). Correlation-based Traffic Analysis Attacks on Anonymity Networks. Ieee Transactions on Parallel and Distributed Systems, 21, 7, 954-967.

Repository Citation

Correlation-Based Traffic Analysis Attacks on Anonymity Networks

Ye Zhu, *Member*, *IEEE*, Xinwen Fu, *Member*, *IEEE*, Bryan Gramham, *Member*, *IEEE*, Riccardo Bettati, *Member*, *IEEE*, and Wei Zhao, *Fellow*, *IEEE*

Abstract—In this paper, we address attacks that exploit the timing behavior of TCP and other protocols and applications in low-latency anonymity networks. Mixes have been used in many anonymous communication systems and are supposed to provide countermeasures to defeat traffic analysis attacks. In this paper, we focus on a particular class of traffic analysis attacks, *flow-correlation attacks*, by which an adversary attempts to analyze the network traffic and correlate the traffic of a flow over an input link with that over an output link. Two classes of correlation methods are considered, namely *time-domain* methods and *frequency-domain* methods. Based on our threat model and known strategies in existing mix networks, we perform extensive experiments to analyze the performance of mixes. We find that all but a few batching strategies fail against flow-correlation attacks, allowing the adversary to either identify ingress and egress points of a flow or to reconstruct the path used by the flow. Counterintuitively, some batching strategies are actually detrimental against attacks. The empirical results provided in this paper give an indication to designers of Mix networks about appropriate configurations and mechanisms to be used to counter flow-correlation attacks.

Index Terms—Privacy, mixes, anonymity, anonymous communication, flow-correlation attack.

1 Introduction

As the Internet is increasingly used in all aspects of daily life, the realization has emerged that privacy and confidentiality are important requirements for the success of many applications. It has been shown that, in many situations, encryption alone cannot provide the level of confidentiality required by users, since traffic analysis can easily uncover information about the participants in a distributed application.

User *anonymity* is one important confidentiality criterion for many applications, ranging from peer-to-peer file sharing and anonymous web browsing or e-mail, to various forms of electronic commerce, and finally to electronic voting. The nature of many such applications requires that the identity of either one or more of the participants remains confidential either from the other participant(s) or from third parties.

The anonymity of a system can be passively attacked by an observer in two ways, either through inspection of payload or headers of the exchanged data packets, or, when encryption is used, through *traffic analysis*. Sufficiently effective encryption can be used to prevent packet content inspection, giving prevalence to the second form of attack. Traffic analysis is typically countered by the use of intermediary nodes, whose role is to perturb the traffic flow and thus confuse an external observer. Such intermediaries (often called *mixes*) delay and reroute exchanged messages, reorder them, pad their size, or perform other operations. Chaum [1] proposed such a *mix network* to handle mail traffic.

The original Chaum mix network operates on entire mail messages at a time and therefore does not need to pay particular attention to latency added by the mixes. Increasingly, the data exchanged exceed by far the capacity of mixes, for example, in file-sharing applications. As a result, current mixes operate on individual packets of a flow rather than on entire messages. In conjunction with source routing at the sender, this allows for very efficient network-level implementations of mix networks.

Mixes are also being used in applications where low latency is relevant, for example, voice-over-IP or video streaming. Many other applications, such as traditional FTP or file-sharing applications, rely on delay-sensitive protocols, such as TCP, and are therefore in turn delay-sensitive as well. For such applications, it is well known that the level of traffic perturbation caused by the mix network must be carefully chosen in order to not unduly affect delay and throughput requirements of the applications. For the designer of the anonymity system, this results in a trade-off between the anonymity degree [2], [3], [4] and quality-of-service (QoS).

Although significant efforts have been put forth in researching anonymous communication since Chaum, only recently have systematic studies appeared to quantitatively capture the effect of traffic perturbation on the anonymity in realistic settings. It is, therefore, difficult to assess the improvement of anonymity that one attains for any given

- *Y. Zhu is with the Department of Electrical and Computer Engineering, Cleveland State University, Cleveland, OH 44120. E-mail: y.zhu61@csuohio.edu.*
- *X. Fu is with the College of Business and Information Systems, Dakota State University, Madison, SD 57042. E-mail: xinwen.fu@dsu.edu.*
- *B. Gramham and R. Bettati are with the Department of Computer Science, Texas A&M University, College Station, TX 77843. E-mail: bgraham@tamu.edu, bettati@cs.tamu.edu.*
- *W. Zhao is with the University of Macau, Taipa, Macau, China. E-mail: WeiZhao@umac.mo.*

cost in form of added latency and perturbation to traffic streams. Moreover, few quantitative guidelines exist on how different perturbation mechanisms perform.

This paper focuses on the quantitative evaluation of mix performance. We focus our analysis on a particular type of attack, which we call the *flow-correlation attack*. In general, flow-correlation attacks attempt to reduce the anonymity degree by estimating the path of flows through the mix network. Flow correlation analyzes the traffic on a set of links (observation points) inside the network and estimates the likelihood for each link to be on the path of the flow under consideration. An adversary analyzes the network traffic with the intention of identifying which of several output ports a flow at an input port of a mix is taking. Obviously, flow correlation helps the adversary identify the path of a flow and consequently reveal other critical information related to the flow (e.g., sender and receiver). Our major contributions are summarized as follows:

1. We formally model the behavior of an adversary who launches flow-correlation attacks. In order to successfully identify the path taken by a particular flow, the attacker measures the dependency of traffic flows. Two classes of correlation methods are considered, namely *time-domain* methods and *frequency-domain* methods. In the *time domain*, for example, statistical information about rate distributions is collected and used to identify the traffic dependency. Similarly, in the *frequency domain*, we identify traffic similarities by comparing the *Fourier spectra* of timing data. Our experiments indicate that mixes with many currently used batching strategies are weak against flow-correlation attacks, in the sense that attackers can easily determine the path taken by a protected flow.
2. We measure the effectiveness of a number of popular mix strategies in countering flow-correlation attacks. Mixes with any tested batching strategy may fail under flow-correlation attacks in the sense that, for a given flow over an input link, the adversary can effectively detect which output link is used by the same flow. We use *detection rate*, the probability that the adversary correctly correlates flows into and out of a mix, defined as the measure of success for the attack. We will show that, given a sufficient amount of data, known mix strategies fail; that is, the attack achieves close to 100 percent detection rate. This remains true even in batching strategies that sacrifice QoS (such as a significant TCP goodput reduction) in favor of security.
3. While many mix strategies rely on other mechanisms in addition to batching alone, it is important to understand the vulnerabilities of batching. In fact, for a given accuracy of the collected data, the effectiveness of such attacks depends primarily on the *amount* of collected data, i.e., on the length of the observation interval. In our experiments, we illustrate this dependency between attack effectiveness for various batching strategies and the amount of data at hand. These results should guide designers of anonymous communication systems in the informed choice of strategy parameters, such as for striping or for path rerouting [5].

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 outlines our Mix network model, threat model, and a formal definition of the problem. Batching strategies used by existing mix networks are also discussed in this section. Section 4 introduces traffic analysis methodologies that may be deployed by an adversary. We consider both time-domain and frequency-domain traffic analysis methods. In Section 5, we evaluate the performance of mix batching strategies in terms of detection rate and FTP goodput. We also examine the performance of large collections of mixes (so-called mix networks). The effectiveness of the described flow-correlation attacks depends on how packet rates are sampled. In Section 6, we empirically compare the effectiveness of the attacks for different sampling intervals and theoretically derive optimal intervals.

We conclude the paper and discuss the future work in Section 7.

2 Related Work

Chaum [1] pioneered the idea of anonymous communication in 1981. Since then, researchers have applied the idea to different applications such as message-based e-mail and flow-based low-latency communications, and they have developed new defense techniques as more attacks have been proposed.

For anonymous e-mail applications, Chaum [1] proposed using relay servers, called *mixes*, which encrypt and reroute messages. An encrypted message is analogous to an onion constructed by a sender, who sends the onion to the first mix:

1. Using its private key, the first mix peels off the first layer, which is encrypted using the public key of the first mix.
2. Inside the first layer is the second mix's address and the rest of the onion, which is encrypted with the second mix's public key.
3. After getting the second mix's address, the first mix forwards the peeled onion to the second mix. This process repeats all the way to the receiver.
4. The core part of the onion is the receiver's address and the real message to be sent to the receiver by the last mix.

Chaum proposed return address and digital pseudonyms for users to communicate with each other anonymously.

Low-latency anonymous communication can be further divided into systems using core mix networks and those using peer-to-peer networks. In a system using a core mix network, users connect to a pool of mixes, which provide anonymous communication, and users select a forwarding path through this core network to the receiver. *Onion routing* [6], *Freedom* [7], and—most prominently—TOR [8] belong to this category. In a system using a peer-to-peer network, every node in the network is a mix, but it can also be a sender and receiver. Obviously, a peer-to-peer mix network can be very large and may provide better anonymity in the case when many participants use the anonymity service and enough traffic is generated around the network. *Crowds* [9], *Tarzan* [10], *MorphMix* [11], and P^{5} [12] belong to this category.

This paper is interested in the study of passive traffic analysis attacks against low-latency anonymous communication systems. Sun et al. [13] give a quantitative analysis for identifying a web page despite the use of encryption and anonymizing proxies. The authors take advantage of the fact that a number of HTTP features, such as the number and size of objects, can be used as signatures to identify web pages with some accuracy. Unless the anonymizer addresses this, these signatures are visible to the adversary. Serjantov and Sewell [14] analyzed the possibility of a lone flow along an input link of a mix. If the rate of this lone input flow is roughly equal to the rate of a flow out of the mix, this pair of input flow and outflow flow are correlated. They also briefly discussed some of the possible traffic features used to trace a flow. In [15], Wright et al. analyze passive logging attacks on anonymous communication networks.

Attacks on low-latency anonymity networks can be classified into two categories, depending on whether they have access to the information about individual flows in the network or not. The attacks belonging to the first category assume that timing information about individual flows is available either from compromised mix nodes [16] or from a corrupt server [17]. The attacks proposed in the second category aim to match a known traffic flow with sets of *aggregate* flows, typically aggregate traffic through two or more outgoing ports. Danezis's attack [18] on continuous-time mix belongs to this category. Danezis proposed using likelihood ratios to detect a flow in aggregate traffic. To calculate the likelihood ratios, the adversaries need to know the information of cross traffic. The attacks we will present later in this paper belong to the second category. The proposed attack relies on the dependency between the flow of interest and aggregate flows containing the flow of interest. In comparison with Danezis's attack [18], the attacks proposed in this paper do not require information about cross traffic. In addition, we will show how these attacks remain effective in the presence of large amounts of noise.

Two recent attacks on large-scale anonymity networks illustrate the effectiveness of traffic analysis in practice. In [17], Murdoch and Danezis stage an active attack to trace back connections from a server to the victim client by modulating the traffic to the victim at the server and by remotely "sensing" the modulation by probind its interference on cross traffic that is generated by one or more corrupt Tor nodes. Similarly, Øverlier and Syverson [19], [20] describe how to locate hidden servers in the Tor network with the use of a corrupt Tor node and a client node. It is pointed out that all Tor nodes are volunteer peers; it is easy to add corrupt nodes to the network.

With the realization that attacks on anonymity networks are varied, easy to deploy, and effective, attention has started to focus on trying to understand the fundamental capabilities of anonymity infrastructures. Kesdogan et al. [21] use information-theoretic arguments to quantify limitations on the attack resistance of mix networks. Similarly, Zhu and Bettati [4] quantify the effect of imperfect mix implementations using information-theoretic means. Camenisch and Lysyanskaya [22] give a cryptographic definition of onion routing, which in turn allows the construction of secure onion routing schemes.

Correlation-based traffic analysis schemes are applicable beyond anonymity networks. For example, traffic analysis has been successfully applied to identify and locate stepping stones [23], [24], [25]. Most of these traffic analysis approaches are timing based. Similarly, Suh et al. use traffic correlation in the time domain to identify Skype relay nodes at the boundary of campus network settings [26]. Active approaches based on embedding watermark into traffic flows are proposed to detect stepping stones in [24], [25].

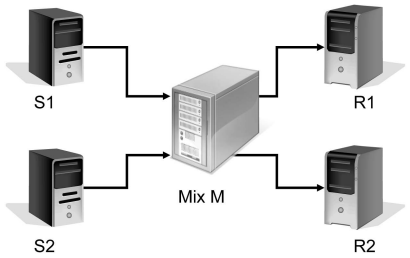


Fig. 1. A single mix.

3 Models

3.1 Mix and Mix Network

A mix is a relay device for anonymous communication. Fig. 1 shows a situation where users communicate using a single mix. such a single mix can achieve a certain level of communication anonymity: The sender of a message attaches the receiver address to a packet and encrypts it using the mix's public key. Upon receiving a packet, a mix decodes the packet. Different from an ordinary router, a mix usually will not relay the received packet immediately. Rather, it collects several packets and then sends them out in a *batch*. The order of packets may be altered as well. Techniques such as batching and reordering are simple means to perturb the timing behavior of packets across a mix, which in turn is considered necessary for mixes to prevent timing-based attacks. More sophisticated perturbation techniques, such as continuous mixes, have been recently described, but have been shown to be susceptible to flow-correlation attacks as well [27]. The main objective of this paper is to analyze the effectiveness of mixes against a special class of timing-based attacks. Some mix networks (most notably maybe Tor) do not explicitly batch packets at the mixes, but rather perturb traffic patterns implicitly by running TCP-style feedback-based protocols between mixes. In this paper, we limit our attention to mix networks with explicit batching.

A *mix network*, such as Onion Routing network or Tor Network, consists of multiple mixes that are interconnected by a network. Such a mix network may provide enhanced anonymity, as payload packets may go through multiple mixes. Since the end-to-end performance of any mix network eventually relies on the performance of its individual mixes, the analysis of the single mix provides a foundation for analyzing the end-to-end performance of mix networks. We discuss in detail how to extend our work to larger and complicated mix networks in [28]. In fact, if we view a mix network (for example, any portion of a Tor network [8]) as one *super mix*, the analytical techniques in this paper can be directly applied.

3.2 Batching Strategies for a Mix

Batching strategies are designed to prevent not only simple timing analysis attacks, but also powerful trickle attacks,

TABLE 1
Batching Strategies [30]

Glossary

n	queue size
m	threshold to control the packet sending
t	timer's period if a timer is used
f	the minimum number of packets left in the pool for pool Mixes
p	a fraction only used in Timed Dynamic-Pool Mix

Algorithms

Strategy Index	*Name*	*Adjustable Parameters*	*Algorithm*
s_0	Simple Proxy	$none$	no batching or reordering
s_1	Threshold Mix	$< m >$	if $n = m$, send n packets
s_2	Timed Mix	$< t >$	if timer times out, send n packets
s_3	Threshold Or Timed Mix	$< m, t >$	if timer times out, send n packets; elseif $n = m$ {send n packets; reset the timer}
s_4	Threshold and Timed Mix	$< m, t >$	if (timer times out) and ($n \geq m$), send n packets
s_5	Threshold Pool Mix	$< m, f >$	if $n = m + f$, send m randomly chosen packets
s_6	Timed Pool Mix	$< t, f >$	if (timer times out) and ($n > f$), send $n - f$ randomly chosen packets
s_7	Timed Dynamic-Pool Mix	$< m, t, f, p >$	if (timer times out) and ($n \geq m + f$), send $\max(1, \lfloor p(n - f) \rfloor)$ randomly chosen packets

flood attacks, and many other forms of attacks [29], [30]. Serjantov et al. [30] summarizes seven batching strategies that have been proposed. We will evaluate each of these strategies. Our results show that these strategies may not work under certain timing analysis attacks. These seven batching strategies are listed in Table 1, in which batching strategies from s_1 to s_4 are denoted as *simple mixes*, while batching strategies from s_5 to s_7 are denoted as *pool mixes*.

From Table 1, we can see that the sending of a batch of packets can be triggered by certain events, e.g., queue length reaching a predefined threshold, a timer having a time-out, or some combination of these two.

Batching is typically accompanied by reordering. In this paper, the attacks focus on the traffic characteristics. As reordering does not significantly change packet interarrival times for mixes that use batching, these attacks (and our analysis thereof) are unaffected by reordering. Thus, our results are applicable to systems that use any kind of reordering methods. More precisely, reorderings are in all cases caused by packets being delayed by the batcher, and can therefore be handled by modifying the batching algorithm accordingly. In this paper, we deal with a large class of batching strategies, and the rest have (such as the Continuous Mix [18]) been discussed by others. As such, in the rest of this paper, we will not discuss reordering techniques further.

Any of the batching strategies can be implemented in two ways:

Link-Based Batching. With this method, each output link has a separate queue. A newly arrived packet is put into a queue depending on its destination (and hence the link associated with the queue). Once a batch is ready from a particular queue (per the batching strategy), the packets are taken out of the queue and transmitted over the corresponding link.

Mix-Based Batching. In this way, the entire mix has only one queue. The selected batching strategy is applied to this queue. That is, once a batch is ready (per the batching strategy), the packets are taken out the queue and transmitted over links based on the packets' destination.

Each of these two methods has its own advantages and disadvantages. The control of link-based batching is distributed inside the mix and hence may have good efficiency. On the other hand, mix-based batching uses only one queue and hence is easier to manage. We consider both methods in this paper.

3.3 Threat Model

In this paper, we assume that the adversary uses a classical timing analysis attack [31], [32], which we summarize as follows:

We assume a *partially global* adversary, who can observe some (i.e., wherever he can place a monitoring point) but not necessarily all links of a mix network. We also assume that the mix nodes of interest are not compromised (as opposed to, for example, Levine et al.'s work [16]) but maybe other nodes are, and that the adversary is interested in breaking the anonymity provided by the mix network.

The adversary collects the packet interarrival times, and analyzes them. This type of attack is passive, since traffic is not actively altered (by, say, dropping, inserting, and/or modifying packets during a communication session), and is therefore very difficult to detect. This type of attack can be easily staged on wired and wireless links [33] by a variety of agents that have access to the network infrastructure, such as malicious ISPs or governments [34]. The inherently distributed nature of many anonymity networks makes it easy to access such information. For example, malicious Tor nodes can be used as traffic monitoring points inside the network. We assume that the traffic characteristic of the flow under consideration (the *input flow*) is known. This can be the case, for example, when the flow traffic characteristic is indeed observable on a link either inside or at the edge of the mix network. We assume that the input flow is a single flow that cannot be split further.

The Mix network topology and the general Mix strategies are known to the adversary. This is a natural assumption for many overlay mix networks. As we will point out in the following, there is no need for the attacker to know the detailed perturbation scheme used in the Mix, as our flow-correlation schemes work for just about any Mix, such as timed and batched mixes, and pooled and stop-and-go mixes. This is in contrast to Danezis' elegant attack to the Continuous Mix [18], which relies on the pairwise independence of packet timings across flows, which is not the case in most Mixes.

The adversary cannot correlate (based on packet timing, content, or size) a packet on a input link to another packet on the output link. Packet correlation based on packet timing is prevented by batching, and correlation based on content and packet size is prevented by encryption and packet padding, respectively.

To simplify the following discussion, we assume that the traffic in the mix network is not padded with any dummy traffic in addition to that naturally generated by the other users in the network. Some of the modern anonymous communication systems such as Tor [8], do not use dummy traffic because of its heavy consumption of bandwidth and the general lack of understanding of to what extent exactly dummy packets contribute to anonymity. Rather, they rely on naturally occurring cross traffic.

Given the threat model described above, we formulate the *Flow-Correlation Problem* as follows: Given a description of a flow of interest at the input of a Mix, and a number of flows of indistinguishable packets at the outputs of the Mix, which output link contains the flow of interest? In other words, we assume that the specific objective of the adversary is to identify the output link of a traffic flow that appears on an input link. Others have described similar attacks but under simplified circumstances. Serjantov and Sewell [14], for example, assume that the flow under attack is alone on a link thus making its traffic characteristics immediately visible to the attacker. In this paper, we consider flows inside (potentially large) aggregates, thus making the attack rather generally applicable.

4 Traffic Flow Correlation Techniques

This section discusses the traffic flow-correlation techniques that may be used by the adversary either to correlate senders and receivers directly or to greatly reduce the searching time for such a correlation in a mix network.

4.1 Overview

Recall that the adversary's objective is to correlate an incoming flow to an output link at a Mix. We call this *flow correlation*. This flow-correlation attack is harmful in a variety of situations. For example, in the single-mix scenario depicted in Fig. 1, the adversary can discover whom sender (say, S_1) is talking to (R_1 or R_2 in this case) by correlating the output traffic at the Mix to S_1's traffic despite cross traffic from S_2 or other senders. In a mix network, the adversary can easily reconstruct the path of the connection by combining measurements and results of flow correlation either at the network boundaries or within the network.

This section discusses the attack in more detail. Fig. 2 shows a flowchart of the typical procedure which the adversary may use to perform flow correlation. We now describe each step in detail.

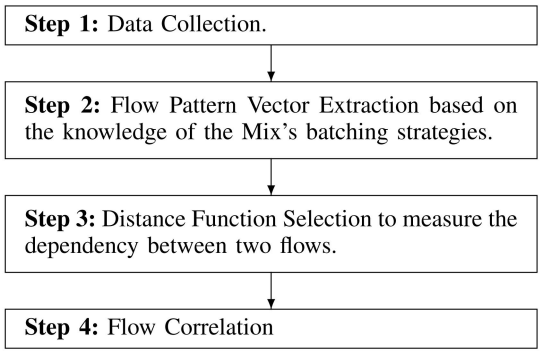


Fig. 2. Typical flowchart for flow correlation.

Step 1: Data Collection. We assume that the adversary is able to collect information about all the packets on both input and output links. For each collected packet, the arrival time is recorded using tools such as tcpdump [35] or Cisco's NetFlow [36]. We assume that all the packets are encrypted and padded to the same size, and hence, only the arrival time is of interest. The arrival times of packets at input link $i\psi$ form a time series

$$A_{i\psi} = (a_{i,1}, \ldots, a_{i,r}), \psi \quad (1)$$

where $a_{i,k\psi}$ is the kth packet's arrival time at input link i, and $r\psi$ is the size of the sample collected during a given sampling interval. Similarly, the arrival times of packets at output link j form a time series

$$B_{j\psi} = (b_{j,1}, \ldots, b_{j,s}), \psi \quad (2)$$

where $b_{j,k\psi}$ is the kth packet's arrival time at output link j, and $s\psi$ is the size of the sample collected during a given sampling interval. The packets come out from mixes in batches. We select sampling interval that is usually much longer than the duration of a batch. Hence, a sampling interval typically contains many batches. We make the simplifying assumption that the traffic characteristic of the flow under consideration (the *input flow*) is known. This can be the case, for example, when the flow traffic characteristic is indeed observable on a link either inside or at the edge of the mix network.

Step 2: Flow Pattern Vector Extraction. With the above notation, the strategy of the adversary is to analyze the time series A_is and B_js in order to determine if there is any dependency between an input flow and an output flow of the mix. However, a direct analysis over these time series will not be effective. They need to be transformed into so-called *pattern vectors* that can facilitate further analysis. We have found that effective transformation depends on batching strategies utilized by the mix. In Section 4.2, we will discuss specific definitions of transformations for different batching strategies. Currently, for the convenience of discussion, let us assume that $A_{i\psi}$ is transformed into pattern vector $X_{i\psi} = (x_{i,1}, \ldots, x_{i,q})$. And time series $B_{j\psi}$ is transformed into $Y_{j\psi} = (y_{j,1}, \ldots, y_{j,q})$. Note, here the two pattern vectors have the same length.

Step 3: Distance Function Selection. We define the distance function $d(X_i, Y_j)$, which measures the "distance" between an input flow at input link $i\psi$ and the traffic at output link j. The smaller the distance, the more likely the flow on an input link is correlated to the corresponding

flow on the output link. Clearly, the definition of the distance function is the key in the correlation analysis. Section 4.3 will discuss two effective distance functions: one is based on mutual information and the other is based on the frequency-spectrum-based matched filter.

Step 4: Flow Correlation. Once the distance function has been defined between an input flow and an output link, we can easily carry out the correlation analysis by selecting the output link whose traffic has the minimum distance to input flow pattern vector X_i.

4.2 Flow Pattern Vector Extraction

Once the data are collected, the relevant pattern vectors must be extracted. Recall that batching strategies in Table 1 can be classified into two classes: threshold-triggered batching (s_1, s_3, and s_5)[1] and timer-triggered batching (s_2, s_4, s_6, and s_7). The packet timing characteristics at the output link allows for targeted feature extraction for these different classes of batching.

For threshold-triggered batching strategies, packets leave the mix in batches. Hence, the interarrival time of packets in a batch is determined by the link bandwidth, which is independent of the input flow. Thus, the useful information to the adversary is the number of packets in a batch and the time that elapses between two batches. Normalizing this relationship, we define the elements in pattern vector Y_j as follows:

$$Y_{j,k} = \frac{\text{Number of packets in batch k in the sampling interval}}{(\text{Ending time of batch k}) - (\text{Ending time of batch k} - 1)}. \quad (3)$$

For timer-triggered batching strategies, a batch of packets is sent whenever a timer fires. The length of the time interval between two consecutive timer events is a predefined constant. Thus, following a similar argument made for the threshold-triggered batching strategies, we define the elements in pattern vector Y_j as follows:

$$Y_{j,k} = \frac{\text{Number of packets in the } k^{\text{th}} \text{ time-out interval}}{(\text{time of } k^{\text{th}} \text{ time-out}) - (\text{time of } (k-1)\text{st time-out})} \quad (4)$$

$$= \frac{\text{Number of packets in the } k^{\text{th}} \text{ time-out interval}}{\text{Predefined inter-time-out length}}. \quad (5)$$

For the traffic *without batching* (i.e., the baseline strategy s_0 defined in Table 1), we use similar methods defined for timer-triggered batching strategies as shown in (5).

The basic idea in the methods for extraction of pattern vectors is to partition a sampling interval into multiple subintervals and to calculate the average traffic rate in each subinterval as the values of the elements of traffic pattern vectors. The above two methods differ on how to partition the interval, depending on which batching strategy is used by the mix. We take a similar approach to extract pattern vectors X_is corresponding to Y_js. Again, the specific method of subinterval partition depends on how the mix is batching the packets.

1. Strategy s_3 could also be classified as timer-triggered. However, we treat it as threshold-triggered because it may send out a batch when the number of packets received by the mix has reached the threshold.

4.3 Distance Functions

The feature vectors are correlated using distance functions. In the following, we consider two kinds of distance functions: the first is based on a comparison of mutual information and the second on frequency analysis. The motivation and computation methods are given below.

4.3.1 Mutual Information

Mutual information is an information theoretical measure of the dependence of two random variables.[2] In our scenario, we can view the pattern vectors that represent the input and output flows as samples of random variables. If we consider the pattern vectors X_i and Y_j to be each a sample of the random variables $\mathcal{X}_i$ and $\mathcal{Y}_j$, respectively, then $\{(X_{i,1}, Y_{j,1}), \ldots, (X_{i,q}, Y_{j,q})\}$ correspond to a sample of the joint random variable $(\mathcal{X}_i, \mathcal{Y}_j)$. With these definitions, the distance $d(X_i, Y_j)$ between pattern vectors X_i and Y_j should be approximately inversely proportional to the mutual information $I(\mathcal{X}_i, \mathcal{Y}_j)$ between $\mathcal{X}_i$ and $\mathcal{Y}_j$,

$$d(X_i, Y_j) = \frac{1}{I(\mathcal{X}_i, \mathcal{Y}_i)} = -\frac{1}{\int\int p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}}. \quad (6)$$

In order to compute the mutual information $I(\mathcal{X}_i, \mathcal{Y}_i)$, we need to estimate the marginal distributions ($p(x_i)$ and $p(y_j)$) and their joint distribution $p(x_i, y_j)$. In the following, we use a histogram-based estimation of mutual information $\hat{I}(\mathcal{X}_i, \mathcal{Y}_j)$ of continuous distributions [37], which is given as follows:

$$\hat{I}(\mathcal{X}_i, \mathcal{Y}_j) \approx \sum_{u,v} \frac{K_{uv}}{q} \log \frac{K_{uv} q}{K_{u.} K_{.v}}, \quad (7)$$

where K represents the histogram, q is the sample size, i.e., the length of pattern vector X_i and Y_j. The sample space is a two-dimensional plane divided into $U \times V$ equally sized $\Delta X \times \Delta Y$ cells with coordinates (u, v). K_{uv} is the number of samples in the bin (u, v). ΔX and ΔY have to be carefully chosen for an optimal estimation.

4.3.2 Frequency Analysis

For timer-triggered batching strategies, we use FFT[3] or Wavelet on the sample X_i and Y_j to obtain the frequency spectrum X_i^F and Y_j^F. Then, we apply matched filter method over X_i^F and Y_j^F. We take advantage of the fact that frequency components of the input flow traffic carry on to the aggregate flows at the output link. Matched filter is an optimal filter to detect a signal buried in noise. It is optimal in the sense that it can provide the maximum signal-to-noise ratio at its output for a given signal. In particular, by directly applying the theory of matched filters, we can define the distance function $d(X_i, Y_j)$ as the inverse matched filter detector $M(X_i^F, Y_j^F)$,

2. Entropy, an information theoretical measure of uncertainty, was proposed to measure anonymity degree of anonymity networks in [2], [3]. In [4], we proposed to use mutual information as a measure of anonymity degree. Entropy is used to measure uncertainty of one random variable and mutual information is used to measure the dependency between two random variables.

3. Frequency analysis has been applied to traffic analysis before, e.g., in [38].

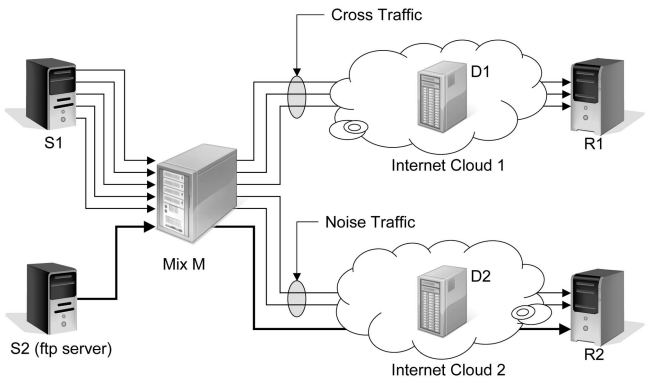


Fig. 3. Experiment setup.

$$d(X_i, Y_j) = \frac{1}{M(X_{i\psi}^{F\psi}, Y_j^{F\psi})} = \frac{1}{\frac{<X_i^F, Y_{j\psi}^F>}{\|Y_{j\psi}^F\|}}, \psi \quad (8)$$

where $<X_{i\psi}^F, Y_j^F>_\psi$ is the inner product of $X_i^{F\psi}$ and $Y_{j\psi}^F$, and

$$\|Y_j^{F\psi}\| = \sqrt{<Y_j^F, Y_j^{F\psi}>_\psi}$$

5 Empirical Evaluation

We evaluate in a testbed, the effectiveness of a selection of batching strategies (listed in Table 1) for a mix under our flow-correlation attacks. These experiments illustrate that all analyzed batching strategies largely fail under the attacks described here. In addition, they impose significant cost on applications, for example, by reducing TCP flow performance.

5.1 Experiment Network Setup

Fig. 3 shows our experimental testbed. The Mix control module that performs the batching and reordering functions is integrated into Linux's firewall system [39] using *Netfilter*; we use a set of firewall rules to specify what traffic should be protected. Traditional Linux kernels have a 10 msec timer granularity, which makes an high-fidelity implementation of timer-based batching strategies difficult. We use a version of Linux (Timesys/Real Time Linux [40]) that guarantees highly accurate timer behavior (maximum timer latency around 50-μsec). Two delay boxes D_1 and D_2 emulate the Internet propagation delay on different paths.

Our experiments reported here focus on TCP flows because of their prevalence in the Internet. However, the results are generally applicable to many other kinds of flows that either use TCP-like congestion control mechanisms or otherwise display strong timing "footprints" due to, for example, user dynamics. VoIP flows or sessions with "short" HTTP connections are instances of the latter. Given the same amount data, they are in general easier to correlate than the long-lasting TCP connections analyzed in this paper.

The traffic flows in our experiments are configured as follows: An FTP client on node R_2 downloads a file from the FTP server on S_2. We call this traffic flow the *flow of interest*. In our experiments, this flow carries packets at a rate of 100 packets per second (*pps*). The traffic from S_1 to R_2 serves as the random *noise traffic* to the FTP client. The traffic from node S_1 to node R_1 is the *cross traffic* through mix $M\psi$ from the perspective of the FTP flow. We adjust the rate of cross traffic and of the noise traffic so that the traffic rate on both output links of the mix are approximately 500 pps. The objective of the adversary in this experiment is to identify the output link that carries the FTP flow.

5.2 Metrics

We use *detection rate* as a measure of the ability of the mix to protect anonymity. Detection rate here is defined as the ratio of the number of correct detections to the number of attempts. While the detection rate measures the *effectiveness* of the mix (the lower the detection rate, the more effective the mix), we measure its *efficiency* in terms of QoS perceived by the applications. We use *FTP goodput* as an indication of FTP *QoS*. FTP goodput is defined as the rate at which the FTP client R_2 receives data from the FTP server S_2. Low levels of FTP goodput indicate that the mix in the given configuration is poorly applicable for low-latency flow-based mix networks.

5.3 Performance Evaluation

5.3.1 Effectiveness of Batching Strategies

Fig. 4 shows the detection rate for systems using a link-based batching strategy. Fig. 5 shows the detection rate for systems using a mix-based batching strategy as a function of the number of packets observed. A sample may include both FTP packets and cross traffic packets while FTP packets account for less than 20 percent of the number (sample size) of packets. Parameters in the legends of these

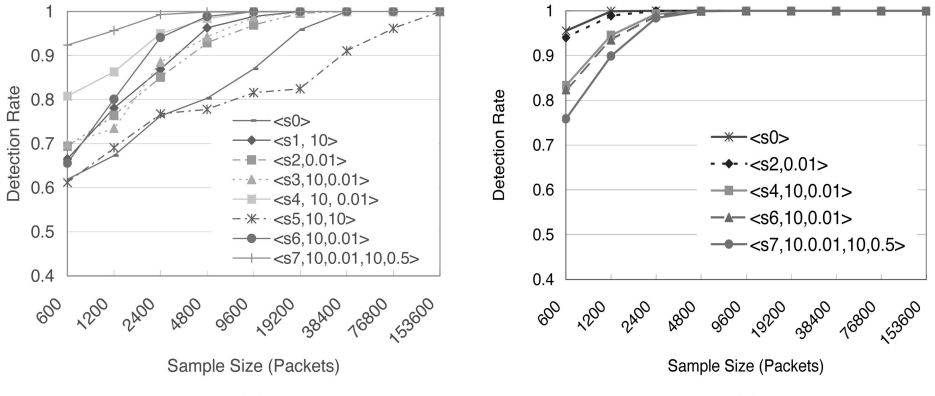
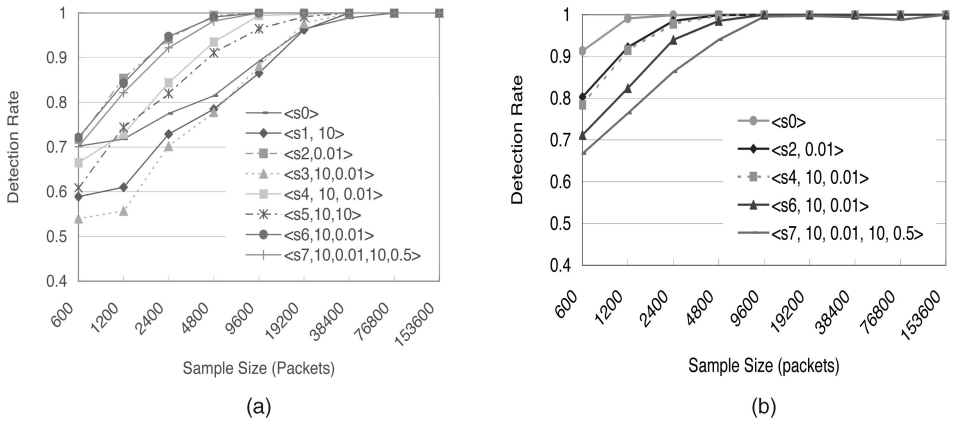
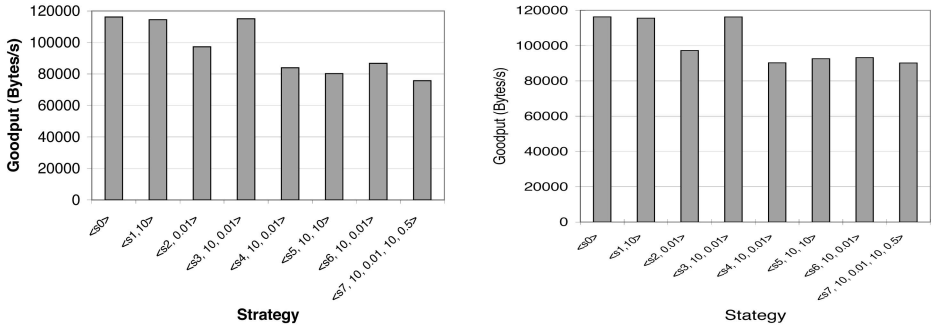


Fig. 4. Detection rate for link-based batching. (a) Mutual information. (b) Matched filter.



(a) (b)

Fig. 5. Detection rate for mix-based batching. (a) Mutual information. (b) Matched filter.



(a) (b)

Fig. 6. FTP goodput. (a) Link-based batching. (b) Mix-based batching.

figures are listed in the same order as in Table 1. Based on these results, we make the following observations:

1. For all the strategies, the detection rate monotonically increases with increasing amount of available data. The detection rate approaches 100 percent when the sample size is sufficiently large. This is consistent with intuition, as more data imply that there is more information about the input flow, which in turn improves the detection rate. In this set of experiments, the detection rate for random guesses is 0.5.
2. Different strategies display different resistances to flow-correlation attacks. A number of observations contradict intuition: a) The simple proxy strategy s_0 performs comparatively well in terms of countering the attack. b) Some researchers in previous studies argued that pool mixes (strategies s_5-s_7) perform better than simple mixes (strategies s_1-s_4) in message-based mix networks. Our results empirically show that this argument does not hold for low-latency flow-based mix networks. With our current parameter setting, the *best* pool batching strategy, timed dynamic-pool mix (strategy s_7) for message-based mix networks is almost the *worst* one for low-latency flow-based mix networks under the attack using mutual information. One of the reasons for these nonintuitive results is that batching introduces more dynamics into TCP flows and makes each TCP flow's features stand out compared with the background traffic.
3. Frequency-analysis-based distance functions typically outperform mutual-information-based distance functions in terms of detection rate. For many batching strategies, the former performs significantly better. This is because the frequency-based analysis is resilient to phasing. Therefore, lack of synchronization between data collected at input and output ports has some effect on the effectiveness of the attack.
4. We do not find a significant difference between link-based and mix-based batching.

Overall, our data show that the mix using any of batching strategies $s_1, s_2, \ldots, s_7$ fails under the flow-correlation attacks. One of the reasons is that TCP flows often demonstrate interesting patterns, such as periodicity of rate change and burstiness, in particular, when the TCP loop-control mechanism is triggered by excessive traffic perturbation in the mixes. Figs. 4 and 5 show that flow-correlation attacks can well explore this pattern difference between TCP flows.

5.3.2 Efficiency of Batching Strategies

As batching delays packets, one should expect that the overall performance (in terms of throughput) of TCP connections will be impacted by the mixes along their path. Fig. 6 quantitatively shows the degradation of FTP goodput for a mix using different batching strategies.

In Fig. 6, we compare FTP goodput between a simple proxy strategy (s_0) and other batching strategies ($s_1, s_2, \ldots, s_7$). We still use the network setup described in Fig. 3. Similar to the experiments above, we configure the noise traffic from S_1 to R_2 to carry 400 pps, and the cross traffic from S_1 to R_1 to carry 500 pps. Together with the 100 pps carried by the flow of interest from S_2 to R_2, both

TABLE 2
Nist-Net Parameters

Nist-Net	Mean Delay (ms)	Standard Deviation of Delay (ms)	Correlation between Successive Packet Delays
D_1	48.3343	1.9682	-0.0322
D_2	56.7815	4.1553	0.1231

output links of the mix carry 500 pps. Based on the experiments and the results illustrated in Fig. 6, we make the following observations:

1. FTP goodput is decreased because of the use of batching.
2. Different batching strategies have different impact on the FTP goodput. In general, pool batching strategies (strategies S_5-s_7) cause a worse FTP goodput than simple batching strategies (strategies s_1-s_4).
3. When the batching in the mixes is excessively aggressive, that is, when batching intervals are too long or threshold values too high, the batching interferes with the time-out behavior of TCP and FTP, and in some cases, FTP aborts. This is the case, in particular, for threshold-triggered mixes with no cross traffic.

5.4 Network Emulation through Nist-Net

In this set of experiments, the delay boxes in Fig. 3 are replaced by Nist-Net nodes [41] to emulate the effect of a real network situation. The Nist-Net parameters are listed in Table 2. The parameters for the Nist-Net nodes D_1 and D_2 were gathered from statistics of ping packet traces from Cleveland State University to ftp.linux.ncsu.edu and to mirror.linux.duke.edu, respectively. The flow of interest is limited to about 100 packet per second by Nist-Net.

Fig. 7 shows the detection rate for emulated networks using the detection method based on mutual information. We can observe that flow-correlation attacks approach a 100 percent detection rate when the sample size is sufficiently large.

5.5 Mixes in Networks with Packet Losses

In this set of experiments, we study the impact of dropped packets on flow-correlation attacks. Such packet loss can occur in overloaded networks or in wireless settings where environmental interference can cause packets to be dropped. The packet-drop behavior is defined by controlling the packet-drop probability in the Nist-Net nodes in Fig. 3.

Fig. 8 shows the detection rate when the network is dropping packets. The mix strategy used in this set of experiments is $<s_7, 10, 0.01, 10, 0.5>$, that is, a timed dynamic-pool Mix with pool size 10, batch size 10, batch interval 10 msec, and forwarding probability 0.5. Based on the results, we make the following observations:

1. The detection rate still approaches 100 percent when the sample size is sufficiently large.
2. The results for small drop rates (5 percent or less) appear to be no different than for no packet drops at all. As expected, for larger drop rates (more than 5 percent) the detection rate is higher than for lower drop rates. The reason for this is that a large number of packet drops makes the timing footprint of the TCP dynamics more obvious.

5.6 Mix Networks

Fig. 9 shows the network setup in this experiment. The center part of the topologies used in experiments is the mix cascade of different number of layers. Each sender on the left side has four flows traversing the mix network. We arrange paths of traffic flows so that each link in the cascade has some number of traffic flows. To simulate the cross traffic in the mix network, four larger aggregates of flows are added to the mix network. According to the self-similar nature of the network traffic [42], the high-volume cross traffic is Pareto distributed. The configuration of the flows through the six-layer cascade is shown in Table 3.

Fig. 10 shows the detection rate for topologies based on mix cascades of different layers. We can observe that when the sample size is sufficiently large, the detection rate approaches 100 percent. In this set of experiments, the detection rate for random guess is $\frac{1}{8}$ since there are eight potential receivers. So, even for small sample size such as

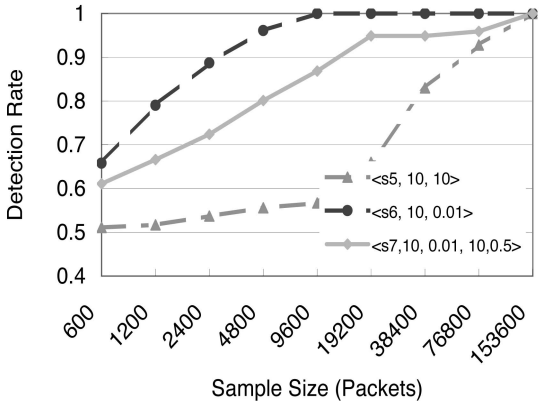


Fig. 7. Detection rate for emulated networks (mix-based batching) using mutual information.

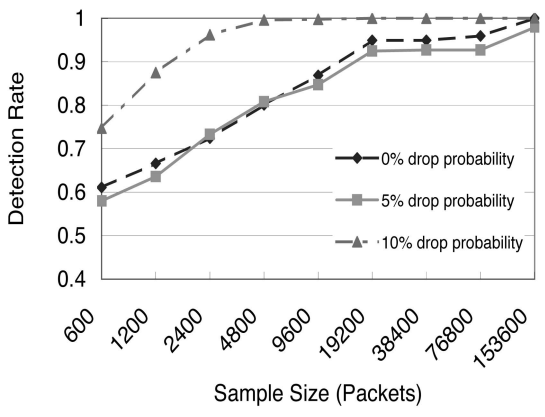


Fig. 8. Detection rate for network with packet loss.

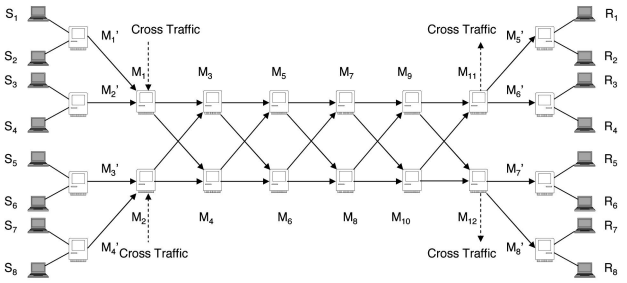


Fig. 9. Experiment setup of mix network (six layers).

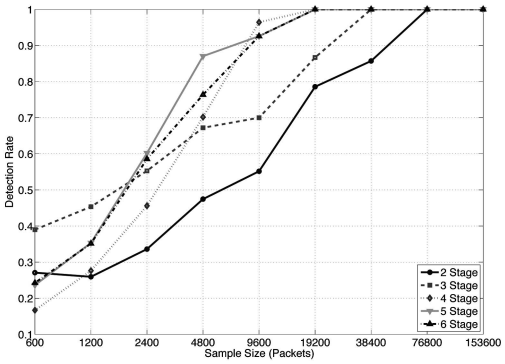


Fig. 10. Detection rate for topologies of different layers.

600 packets, flow-correlation attacks performs better than random guess. We can also observe that the detection rate of the topology of two-stage cascade is lower than other topologies. The reason is that more layers of mix cascade may make timing footprint of the TCP dynamics more obvious.

6 Sampling Interval Selection

6.1 Theoretical Proof and Empirical Validation

From the discussion in Section 4.3 and from the evaluation results above, we can see that frequency-analysis-based flow-correlation attacks are very effective. The effectiveness of the attack greatly depends on the careful selection of the sampling interval, since we calculate the Fourier spectrum over a set of packet average rates (i.e., the flow feature vector) in the sampling interval. In this section, we discuss how to select the sampling interval $\tau\psi$ in order to maximize the effectiveness of flow-correlation attacks. We still use FTP as an example for the discussion.

TABLE 3
Flow Configuration (Six Layers)

Flow ID	Path		Throughput (packets/s)
1	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	73.994
2	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_10 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	70.080
3	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	75.307
4	$S_1 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	74.503
5	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	67.970
6	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	76.708
7	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	76.879
8	$S_2 \rightarrow M_1' \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	75.806
9	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	77.511
10	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	73.741
11	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	73.204
12	$S_3 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	72.292
13	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	70.714
14	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	77.894
15	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	70.941
16	$S_4 \rightarrow M_2' \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	77.495
17	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	69.992
18	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	67.684
19	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	67.363
20	$S_5 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	69.285
21	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	67.262
22	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	70.279
23	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	67.905
24	$S_6 \rightarrow M_3' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	66.419
25	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_5' \rightarrow R_1$	FTP	72.785
26	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_6' \rightarrow R_3$	FTP	69.212
27	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_5$	FTP	73.296
28	$S_7 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_9 \rightarrow M_{12} \rightarrow M_8' \rightarrow R_7$	FTP	66.461
29	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{11} \rightarrow M_5' \rightarrow R_2$	FTP	68.461
30	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_9 \rightarrow M_{11} \rightarrow M_6' \rightarrow R_4$	FTP	71.645
31	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_7' \rightarrow R_6$	FTP	67.252
32	$S_8 \rightarrow M_4' \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_{10} \rightarrow M_{12} \rightarrow M_8' \rightarrow R_8$	FTP	73.207
33	$\rightarrow M_3 \rightarrow M_5 \rightarrow M_8 \rightarrow M_{10} \rightarrow$	Pareto	491.548
34	$\rightarrow M_3 \rightarrow M_6 \rightarrow M_8 \rightarrow M_9 \rightarrow$	Pareto	519.819
35	$\rightarrow M_4 \rightarrow M_5 \rightarrow M_7 \rightarrow M_{10} \rightarrow$	Pareto	497.629
36	$\rightarrow M_4 \rightarrow M_6 \rightarrow M_7 \rightarrow M_9 \rightarrow$	Pareto	502.180

Lemma 6.1. *An FTP flow with round trip time $RTT\psi$ has a frequency component with the maximum power density at $1/RTT\psi$ This frequency component is denoted as the* Feature Frequency *of the FTP flow. (Proof in the Appendix.)*

The basic idea is that FTP uses a loop-control mechanism. For most of the life time, an FTP flow acts on the information collected in each round trip time and thus demonstrates a strong periodicity at the round trip time RTT.

Based on Lemma 6.1, we have the following theory for the selection of sampling interval:

Theorem 6.2. *Assuming that a stable FTP flow on the input link of a mix has a round trip time RTT, to detect the output link of this FTP flow, we need to choose a sampling interval $\tau\psi$ smaller than or equal to $RTT/2$, i.e.,*

$$\tau\psi \leq \frac{RTT\psi}{2}.\psi \qquad (9)$$

Proof. When we do sampling and calculate the average rate of an FTP flow during the sampling interval, the process corresponds to a *zero-order hold* [43] sampling process. From Lemma 6.1, we know that an FTP flow's feature frequency is at $1/RTT\psi$ which we have to preserve for the best effectiveness of flow-correlation attack. Nyquist's sampling theorem [43] tells us that to preserve this feature frequency, the sampling rate $1/\tau\psi$ should be at least two times the feature frequency. That is,

$$\frac{1}{\tau\psi} \geq 2\frac{1}{RTT\psi}.\psi \qquad (10)$$

Thus,

$$\tau\psi \leq \frac{RTT\psi}{2}.\psi$$

□

Approximately, we can apply Theorem 6.2 to all the strategies. Figs. 11 and 12 show detection rate in terms of sampling interval. RTT of this FTP flow in question is around 300 msec. We can see that the maximum detection rate does happens at $RTT/2 = 150$ msec.

In practice, we cannot use any sampling interval smaller than half of RTT. There exists all kinds of interference from mixes and operating systems, which may introduce high-frequency noise in frequency domain. We, therefore, prefer to use a sample interval between $[RTT/4, RTT/2]$. In this

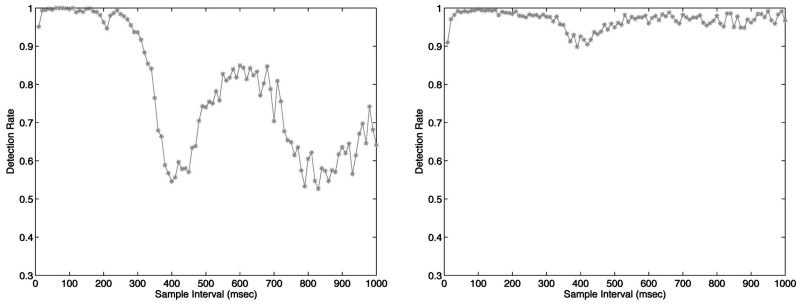


Fig. 11. Detection rate in terms of sampling interval based on Matched Filter. (a) $<s_0>$. (b) $<s_2, 0.01>$.

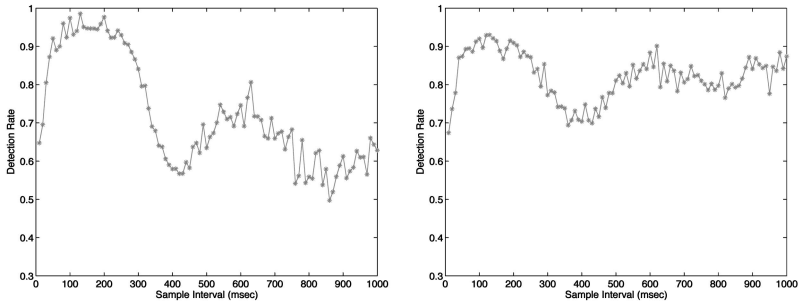


Fig. 12. Detection rate in terms of sampling interval based on Mutual Information. (a) $<s_0>$. (b) $<s_2, 0.01>$.

way, the zero-order hold operator acts as a low-pass filter with frequency response

$$H(f,\tau) = e^{-j\pi f\tau} \left[\frac{2\sin\left(\frac{2\pi f\tau}{2}\right)}{2\pi\psi} \right] \quad (11)$$

The main lobe of $|H(f,\tau)|$ is in the range $|f| < \frac{1}{\tau\psi}$. Thus, our sampling process will smooth the original instantaneous rate and remove a significant amount of noise. This in turn helps the flow-correlation attack. Figs. 11 and 12 show the influence of noise on detection rate: when $\tau\psi$ is much smaller than $RTT/2$, the detection rate deteriorates.

6.2 Discussion

In practice, the adversary does not know the exact RTT of the special flows in the system. Instead, she may need to a priori investigate the mix network and get a rough picture of possible TCP flow RTTs. The sampling interval can then be chosen to be half of the smallest of the possible RTTs, or simply the one that gives the best detection rate.

Figs. 11 and 12 also illustrate the complicated relationship between detection rate and sampling interval: In addition to the feature frequency component at $1/RTT\psi$, an FTP flow in reality contains minor feature frequencies as well, which are sufficient to differentiate an FTP flow from others. Fig. 13 gives an example of a power spectrum of an FTP flow.

For application-limited flows, such as ssh and much Web traffic, the feature frequency is not determined by RTT. Instead, it is determined by the application-level dynamics.

7 Summary and Future Work

We have analyzed mix networks in terms of their effectiveness in providing anonymity and quality-of-service. Various methods used in mix networks were considered: seven different packet batching strategies and two implementation schemes, namely the link-based batching scheme and mix-based batching scheme. We found that mix networks that use traditional batching strategies, regardless of the implementation scheme, are vulnerable under flow-correlation attacks. By using statistical analysis, an adversary can accurately determine the output link used by traffic that comes to an input flow of a mix. The detection rate can be as high as 100 percent as long as enough data are available. This is true even if heavy cross traffic exists. The data collected in this paper should give designers guidelines for the development and operation of mix networks.

The failure of traditional mix batching strategies directly leads us to the formulation of a new packet control method for mixes in order to overcome their vulnerability to flow-correlation attacks. Appropriate output control can achieve a guaranteed low detection rate while maintaining high throughput for normal payload traffic. Our claim is validated by extensive performance data collected from experiments.

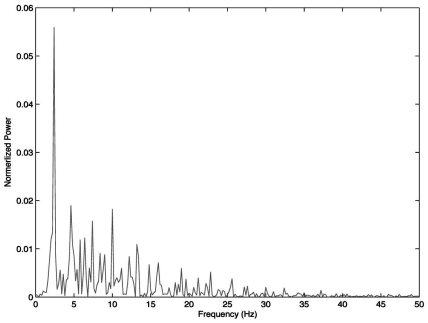


Fig. 13. Power spectrum of an FTP flow.

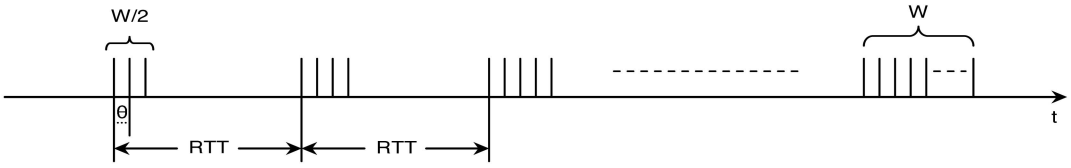


Fig. 14. TCP congestion window in congestion avoidance phase.

We have shown that output control is flexible in controlling the overhead by adjusting the maximum packet delay.

APPENDIX A

MAXIMUM FREQUENCY COMPONENT OF A TCP FLOW

Proof of Lemma 6.1. Based on [44], [45], the TCP flow dynamics can approximately be illustrated as in Fig. 14 if TCP-reno [46] version of congestion control algorithm is used. When a TCP flow is in additive increase phase, one more packet is sent each round trip time. While in multiplicative decrease phase, the packet number in one round trip time decreases by half from $W\psi$to $\frac{W}{2}$. The interdeparture time $\theta\psi$of two adjacent packets is determined by the smallest bandwidth along the flow path and jitter of queuing delay. Usually, $\theta\psi$is much smaller than RTT.

So, we can model the TCP packet train in congestion control phase as

$$x(t) = \sum_{k=0}^{\frac{W}{2}} \left(\sum_{l=0}^{\frac{W}{2}+k-1} \left(\delta(t\psi - l \cdot \theta - k \cdot RTT), \psi \right. \right. \quad (12)$$

where $\delta(t)$is the *unit impulse function*.

Its Fourier transformation is

$$X(\omega) = \sum_{k=0}^{\frac{W}{2}} \left(\sum_{l=0}^{\frac{W}{2}+k-1} \left(e^{-j\omega(k \cdot RTT + l \cdot \theta)} . \psi \right. \right. \quad (13)$$

Its energy-density spectrum is as shown in (14).

Since $\theta \ll RTT$, $|l-n| \leq W$, and $(l-n)\theta \rightarrow 0$, (14) can be approximated as follows:

$$\begin{aligned}
|X(w)|^2 &= \left[\sum_{k=0}^{\frac{W}{2}} \left(\sum_{l=0}^{\frac{W\psi}{2}+k-1} \left(\cos(k \cdot RTT\psi\,\omega + l \cdot \theta \cdot \omega) \right] \right.^2 \right. \\
&\quad + \left[\sum_{k=0}^{\frac{W}{2}} \left(\sum_{l=0}^{\frac{W}{2}+k-1} \left(\sin(k \cdot RTT\psi\,\omega + l \cdot \theta \cdot \omega) \right] \right.^2 \right. \\
&= \sum_{\substack{0 \leq k \leq \frac{W\psi}{2} 0 \leq l \leq \frac{W}{2}+k-1 \\ 0 \leq m \leq \frac{W\psi}{2} 0 \leq n \leq \frac{W}{2}+m-1}} \left(\cos(k \cdot RTT\psi\,\omega + l \cdot \theta \cdot \omega) \right. \\
&\quad \cos(m\psi RTT\psi\,\omega + n\psi\theta \cdot \omega) \\
&\quad + \sum_{\substack{0 \leq k \leq \frac{W\psi}{2} 0 \leq l \leq \frac{W}{2}+k-1 \\ 0 \leq m \leq \frac{W\psi}{2} 0 \leq n \leq \frac{W}{2}+m-1}} \left(\sin(k \cdot RTT\psi\,\omega + l \cdot \theta \cdot \omega) \right. \\
&\quad \sin(m\psi RTT\psi\,\omega + n\psi\theta \cdot \omega) \\
&= \sum_{\substack{0 \leq k \leq \frac{W\psi}{2} 0 \leq l \leq \frac{W}{2}+k-1 \\ 0 \leq m \leq \frac{W\psi}{2} 0 \leq n \leq \frac{W}{2}+m-1}} \left(\cos(((k-m)RTT\psi + (l-n)\theta)\omega) . \psi \right.
\end{aligned} \quad (14)$$

$$|X(w)|^2 \sum_{\substack{0 \leq k \leq \frac{W\psi}{2} 0 \leq l \leq \frac{W}{2}+k-1 \\ 0 \leq m \leq \frac{W\psi}{2} 0 \leq n \leq \frac{W}{2}+m-1}} \left(\cos((k-m) \cdot RTT\psi\,\omega) . \psi \right. \quad (15)$$

This corresponds to the case that all packets in one RTT are sent out at roughly the same time at the beginning of the RTT. When $\omega\psi = 2\pi \frac{h\psi}{RTT\psi}$ ($h\psi$is an integer), we get the maximum $|X(w)|^2$ since $\cos((k-m) \cdot RTT\psi\,\omega) = 1$.

Thus, the maximum frequency component of an FTP flow is around frequency $\frac{1}{RTT\psi}$.

REFERENCES

[1] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Comm. ACM*, vol. 24, no. 2, pp. 84-90, Feb. 1981.

[2] A. Serjantov and G. Danezis, "Towards an Information Theoretic Metric for Anonymity," *Proc. Privacy Enhancing Technologies Workshop (PET '02)*, R. Dingledine and P. Syverson, eds., pp. 41-53, Apr. 2002.

[3] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards Measuring Anonymity," *Proc. Privacy Enhancing Technologies Workshop (PET '02)*, R. Dingledine and P. Syverson, eds., pp. 54-68, Apr. 2002.

[4] Y. Zhu and R. Bettati, "Anonymity vs. Information Leakage in Anonymity Systems," *Proc. 25th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '05)*, pp. 514-524, 2005.

[5] O.R.D. Achives, "Link Padding and the Intersection Attack," http://archives.seul.org/or/dev, 2002.

[6] P.F. Syverson, D.M. Goldschlag, and M.G. Reed, "Anonymous Connections and Onion Routing," *Proc. IEEE Symp. Security and Privacy*, pp. 44-54, 1997.

[7] P. Boucher, A. Shostack, and I. Goldberg, "Freedom Systems 2.0 Architecture," http://www.freedom.net/products/whitepapers/Freedom_System_2_Architecture.pdf, Dec. 2000.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," *Proc. 13th USENIX Security Symp.*, pp. 303-320, Aug. 2004.

[9] M.K. Reiter and A.D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Trans. Information and System Security*, vol. 1, no. 1, pp. 66-92, 1998.

[10] M.J. Freedman and R. Morris, "Tarzan: A Peer-to-Peer Anonymizing Network Layer," *Proc. Ninth ACM Conf. Computer and Comm. Security*, pp. 193-206, 2002.

[11] M. Rennhard and B. Plattner, "Introducing MorphMix: Peer-to-Peer Based Anonymous Internet Usage with Collusion Detection," *Proc. ACM Workshop Privacy in the Electronic Soc. (WPES '02)*, pp. 91-102, 2002.

[12] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "p^5: A Protocol for Scalable Anonymous Communication," *Proc. IEEE Symp. Security and Privacy*, pp. 58-70, May 2002.

[13] Q. Sun, D.R. Simon, Y.-M. Wang, W. Russell, V.N. Padmanabhan, and L. Qiu, "Statistical Identification of Encrypted Web Browsing Traffic," *Proc. IEEE Symp. Security and Privacy*, pp. 19-30, 2002.

[14] A. Serjantov and P. Sewell, "Passive Attack Analysis for Connection-Based Anonymity Systems," *Proc. European Symp. Research in Computer Security (ESORICS '03)*, pp. 116-131, Oct. 2003.

[15] M. Wright, M. Adler, B.N. Levine, and C. Shields, "Defending Anonymous Communications against Passive Logging Attacks," *Proc. IEEE Symp. Security and Privacy (SP '03)*, pp. 28-41, May 2003.

[16] B.N. Levine, M.K. Reiter, C. Wang, and M.K. Wright, "Timing Attacks in Low-Latency Mix-Based Systems," *Proc. Financial Cryptography Conf. (FC '04)*, pp. 251-265, Feb. 2004.

[17] S.J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," *Proc. 2005 IEEE Symp. Security and Privacy,* pp. 183-195, May 2005.
[18] G. Danezis, "The Traffic Analysis of Continuous-Time Mixes," *Proc. Privacy Enhancing Technologies Workshop (PET '04),* pp. 35-50, May 2004.
[19] L. Øverlier and P. Syverson, "Locating Hidden Servers," *Proc. IEEE Symp. Security and Privacy,* May 2006.
[20] L. Øverlier and P. Syverson, "Valet Services: Improving Hidden Servers with a Personal Touch," *Proc. Sixth Workshop Privacy Enhancing Technologies (PET '06),* G. Danezis and P. Golle, eds., pp. 223-244, June 2006.
[21] D. Kesdogan, D. Agrawal, V. Pham, and D. Rautenbach, "Fundamental Limits on the Anonymity Provided by the Mix Technique," *Proc. IEEE Symp. Security and Privacy (SP),* pp. 86-99, 2006.
[22] J. Camenisch and A. Lysyanskaya, "A Formal Treatment of Onion Routing," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO '05),* V. Shoup, ed., pp. 169-187, Aug. 2005.
[23] Y. Zhang and V. Paxson, "Detecting Stepping Stones," *Proc. Ninth Conf. USENIX Security Symp. (SSYM '00),* pp. 13-13, 2000.
[24] X. Wang and D.S. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Interpacket Delays," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* pp. 20-29, 2003.
[25] Y.J. Pyun, Y.H. Park, X. Wang, D.S. Reeves, and P. Ning, "Tracing Traffic through Intermediate Hosts that Repacketize Flows," *Proc. 26th IEEE INFOCOM '07,* pp. 634-642, May 2007.
[26] K. Suh, D.R. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and Detecting Skype-Relayed Traffic," *Proc. 25th IEEE INFOCOM '06,* pp. 1-12, Apr. 2006.
[27] Y. Zhu, X. Fu, and R. Bettati, "On the Effectiveness of Continuous-Time Mixes under Flow Correlation Attacks," Technical Report TR2005-2-6, Texas A&M Univ. Computer Science, 2005.
[28] Y. Zhu, X. Fu, R. Bettati, and W. Zhao, "Anonymity Analysis of Mix Networks against Flow-Correlation Attacks," *Proc. IEEE GLOBECOM '05,* vol. 3, pp. 1801-1805, 2005.
[29] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," *Proc. 2003 IEEE Symp. Security and Privacy,* pp. 2-15, May 2003.
[30] A. Serjantov, R. Dingledine, and P. Syverson, "From a Trickle to a Flood: Active Attacks on Several Mix Types," *Proc. Information Hiding Workshop (IH '02),* F. Petitcolas, ed., pp. 36-52, Oct. 2002.
[31] X. Fu, B. Graham, R. Bettati, W. Zhao, and D. Xuan, "Analytical and Empirical Analysis of Countermeasures to Traffic Analysis Attacks," *Proc. 32nd Int'l Conf. Parallel Processing (ICPP '03),* pp. 483-492, Oct. 2003.
[32] D.X. Song, D. Wagner, and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH," *Proc. 10th USENIX Security Symp.,* pp. 337-352, 2001.
[33] J.D. Howard, "An Analysis of Security Incidents on the Internet 1989-1995," CMU dissertation, Technical Report, 1997.
[34] F.B.I, "Carnivore Diagnostic Tool," http://www.fbi.gov/hq/lab/carnivore/carnivore2.htm, 2003.
[35] tcpdump.org, "tcpdump," http://www.tcpdump.org/, 2003.
[36] cisco.inc., "Netflow Services Solutions Guide," http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm, 2003.
[37] R. Moddemeijer, "On Estimation of Entropy and Mutual Information of Continuous Distributions," *Signal Processing,* vol. 16, no. 3, pp. 233-246, 1989.
[38] S.-q. Li and J.D. Pruneski, "The Linearity of Low Frequency Traffic Flow: An Intrinsic I/O Property in Queueing Systems," *IEEE/ACM Trans. Networking,* vol. 5, no. 3, pp. 429-443, June 1997.
[39] netfilter.org, "Netfilter," http://netfilter.samba.org/, 2003.
[40] TimeSys, "Timesys Linux Docs," http://www.timesys.com/, 2003.
[41] M. Carson and D. Santay, "Nist Net: A Linux-Based Network Emulation Tool," *ACM SIGCOMM Computer Comm. Rev.,* vol. 33, no. 3, http://portal.acm.org/citation.cfm?id=956993.957007, pp. 111-126, July 2003.
[42] K. Park and W. Willinger, "Self-Similar Network Traffic: An Overview," citeseer.ist.psu.edu/park99selfsimilar.html, 1999.
[43] A.V. Oppenheim, A.S. Willsky, and S.H. Nawab, *Signals and Systems,* second ed. Prentice-Hall, 1997.
[44] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Networking,* vol. 7, no. 4, pp. 458-472, Aug. 1999.
[45] J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proc. ACM SIGCOMM '98,* pp. 303-314, 1998.
[46] K. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno and SACK TCP," *SIGCOMM Computer Comm. Rev.,* vol. 26, no. 3, pp. 5-21, Aug. 1996.