On Flow Correlation Attacks and Countermeasures in Mix Networks

Ye Zhu*, Xinwen Fu, Bryan Graham*, Riccardo Bettati and Wei Zhao
Department of Computer Science
Texas A&M University
College Station TX 77843-3112, USA
E-mail: {zhuye, bgraham}@tamu.edu*, {xinwenfu, bettati, zhao}@cs.tamu.edu

Abstract

In this paper, we address issues related to flow correlation attacks and the corresponding countermeasures in mix networks. Mixes have been used in many anonymous communication systems and are supposed to provide countermeasures that can defeat various traffic analysis attacks. In this paper, we focus on a particular class of traffic analysis attack, flow correlation attacks*, by which an adversary attempts to analyze the network traffic and correlate the traffic of a flow over an input link at a mix with that over an output link of the same mix. 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 a mix with any known batching strategy may fail against flow correlation attacks in the sense that for a given flow over an input link, the adversary can correctly determine which output link is used by the same flow. We also investigated methods that can effectively counter the flow correlation attack and other timing attacks. The empirical results provided in this paper give an indication to designers of Mix networks about appropriate configurations and alternative mechanisms to be used to counter flow correlation attacks.*

1 Introduction

This paper studies flow correlation attacks and the corresponding countermeasures in mix networks. With the rapid growth and public acceptance of the Internet as a means of communication and information dissemination, concerns about privacy and security on the Internet have grown. Although it can potentially be used for malicious purposes, *Anonymity* is legitimate in many scenarios such as anonymous web browsing, E-Voting, E-Banking, E-Commerce, and E-Auctions. In each of these scenarios, encryption alone cannot achieve the anonymity required by participants [30, 31].

Since Chaum [6] proposed the mix network, researchers have developed various anonymity systems for different applications. Although a significant amount of effort has been put forth in researching anonymous communications, there has not been much systematic study of the performance of mix networks in terms of anonymity degree provided and quality-of-services maintained. This paper focuses on the quantitative evaluation of mix performance. We are particularly interested in flow-based communication, which is widely used in voice over IP, web browsing, FTP, etc. These applications may have anonymity requirements, and the mixes are supposed to provide countermeasures that can defeat traffic analysis attacks.

We focus our analysis on a particular type of attack, which we call a *flow correlation attack*. In this type of attack, 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 mission critical information related to the flow (e.g., sender and receiver). Our major contributions are summarized as follows:

- We formally model the behavior of an adversary who launches flow correlation attacks. In order to successfully identify the output port of an incoming flow, the flow correlation attack must accurately measure the similarity of traffic flows into and out of a mix. Two classes of correlation methods are considered, namely *time-domain* methods and *frequency-domain* methods. In the time domain, *mutual information* is used to measure the traffic similarity. In the frequency domain, a matched filter based on the *Fourier spectrum* and the *Wavelet spectrum* is utilized.

- 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* as the measure of success for the attack, where Detection rate is defined as the probability that the adversary correctly correlates flows into and out of a mix. We will show that, given a sufficient amount of data, known mix strategies fail, that is, the attack achieves close to 100% detection rate. This remains true, even in batching strategies that sacrifice QoS concerns (such as a significant TCP goodput reduction) in favor of security.

- While many mix strategies rely on other mechanisms in addition to batching alone, it is important to understand the vulnerability of batching. In our experiments, we illustrates the dependency between attack effectiveness for various batching strategies and the amount of data at hand for the attacks. These results should guide mix designers in the educated choice of strategy parameters, such as for striping or for path rerouting.

To counter flow correlation attacks, we investigate countermeasures based on our theoretical analysis. In our method, we purposely synchronize the sending time of packets along a set of output links. The proposed approach is more efficient than similar methods.

The remainder of this paper is organized as follows: Section 2 reviews the related work. In Section 3, we outline our Mix network model, the adversary 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. In particular, we consider both time-domain and frequency-domain traffic analysis methods. In Section 5 we evaluate the performance of mix networks in terms of detection rate and FTP goodput. Serious failure of mix networks in terms of providing flow anonymity is observed from the data we collect. Consequently, in Section 6, we present an effective and efficient method that can provide a guaranteed detection rate with high FTP goodput. We conclude this paper and discuss the future work in Section 7.

2 Related Work

Chaum [6] pioneered the idea of anonymity in 1981. Since then, researchers have applied the idea to different applications such as message-based email and flow-based low-latency communications, and they have invented new defense techniques as more attacks have been proposed.

For anonymous email applications, Chaum [6] proposed to use relay servers, i.e. *mixes*, rerouting messages, which are encrypted by public keys of the mixes. An encrypted message is analogous to an onion constructed by the sender, who sends the onion to the first mix. Using its private key, the first mix peels off the first layer, which is encrypted using the public key of the first mix. 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. After getting the second mix's address, the first mix sends the peeled onion. This process proceeds in this recursive way. 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 also proposed the return address and digital pseudonyms for users to communicate with each other in an anonymous way.

Helsingius [13] implemented the first Internet anonymous *remailer*, which is a single application proxy that just replaces the original email's source address with the remailer's address. It has no reply function and is subject to all the attacks mentioned below. Eric Hughes and Hal Finney [23] built the *cypherpunk remailer*, a real distributed mix network with reply functions that uses PGP to encrypt and decrypt messages. The system is subject to a global passive attack and replay attack to its reply mechanism. Gülcü and Tsudik [12] developed a relatively full-fledged anonymous email system, *Babel*. Their reply technique does not need the sender to remember the secret seed to decrypt the reply message, but it is subject to replay attack. They studied the threat from the trickle attack, a powerful active attack. Another defect of Babel is that a mix itself can differentiate the forwarding and replying messages. Cottrell [19] developed *Mixmaster* which counters a global passive attack by using message padding and also counters trickle and flood attacks [12, 28] by using a pool batching strategy. Mixmaster does not have a reply function. Danezis, Dingledine and Mathewson [7] developed *Mixminion*. Although Mixminion still has many problems, its design considers a relatively complete set of attacks that researchers have found [2, 3, 4, 17, 24, 28]. The authors suggest a list of research topics for future study.

Low-latency anonymous communication can be further divided into systems using core mix networks and peer-to-peer networks. In a system using a core mix network, users connect to a pool of mixes, which provides anonymous communication, and users select a forwarding path through this core network to the receiver. *Onion routing* [32] and *Freedom* [5] 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* [25], *Tarzan* [8] and P^5 [29] 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.* [31] gave a quantitative analysis for identifying a web page even if encryption and anonymizing proxies are used. They took 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 [27] 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. The attacks we will present later in this paper are very effective even when a large amount of noise exists. Other analyses focus on the anonymity degradation when some mixes are compromised, e.g. [25]. We understand that the attacks used against message-based email mix networks can also threaten low-latency flow-based mix networks; however, we feel that traffic analysis attacks are also a serious problem for low-latency mix networks because of its QoS requirements. Our reasoning will be explained in detail in the following sections of this paper.

3 Models

3.1 Mix and Mix Network

A mix is a relay device for anonymous communication. Figure 1 shows the communication between users using one mix. 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 considered to be necessary techniques for a mix to prevent timing-based attacks.

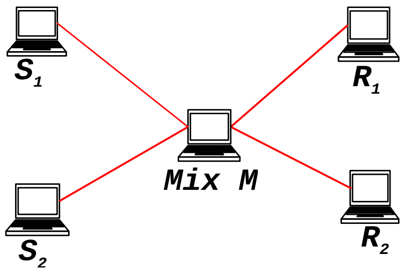


Figure 1. A Single Mix

The main objective of this paper is to analyze the effectiveness of mixes against a class of timing-based attacks.

A mix network consists of multiple mixes that are inter-connected by a network. A mix network may provide enhanced anonymity, as payload packets may go through multiple mixes. Even in such a mix network, it is important that each individual mix provides sufficient security and QoS so that the end-to-end performance can be guaranteed. Thus, our analysis on a 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 [36]. In fact, if we view a mix network (for example Onion routing [32]) 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, flood attacks, and many other forms of attacks ([7, 28]). Serjantov [28] summarizes seven batching strategies that have been proposed. We will evaluate each kind 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 mix*, while batching strategies from S_5 to S_7 are denoted as *pool mix*.

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 pre-defined 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 change packet interarrival times much for mixes using batching, these attacks (and our analysis) are unaffected by reordering. Thus, our results are applicable to systems that use any kind of reordering methods. 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 it 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.

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 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

Table 1. Batching Strategies

3.3 Threat Model

In this paper, we assume that the adversary uses a classical timing analysis attack ([10, 30]), which we summarize as follows:

1. The adversary observes input and output links of a mix, 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 often difficult to detect. This type of attack can be easily staged on wired and wireless links [14] by a variety of agents, such as malicious ISPs or governments ([21, 35]).

2. To maximize the power of the adversary, we assume that she makes observations on all the links of the mix network.

3. The mix's infrastructure and strategies are known to the adversary. This is a typical assumption in the study of security systems. The above two assumptions create the worst case in terms of security analysis.

4. 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.

5. To simplify the following discussion, we assume that dummy traffic is not used in the mix network. Some of the modern anonymous communication systems such as Onion routing ([1]) 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.

6. Finally, 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 [27], 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 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 kind of flow correlation attack is harmful in many scenarios. For example, in Figure 1, the adversary can discover the communication relationship between senders (S_1 and S_2) and receivers (R_1 and R_2) by matching senders' output flows and receivers' input flows. Using the flow correlation attack techniques, the adversary can find out a flow's sender and receiver if she catches a fragment of the flow in the mix network, thus breaking the anonymity despite the mix network. In a peer-to-peer mix network, the adversary can even reconstruct the path of this TCP connection by using these flow correlation techniques. In this subsection, we discuss the attack in more detail.

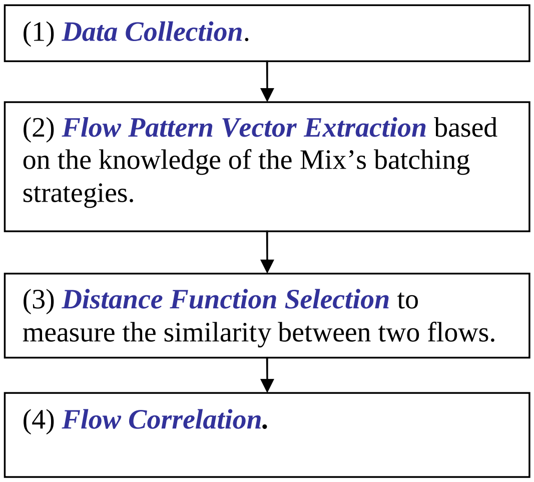

(1) Data Collection.
(2) Flow Pattern Vector Extraction based on the knowledge of the Mix's batching strategies.
(3) Distance Function Selection to measure the similarity between two flows.
(4) Flow Correlation.

Figure 2. Typical Flowchart for Flow Correlation

Figure 2 shows a flowchart of the typical procedure which the adversary may use to perform flow correlation. We now describe each step in detail.

(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 (for example, using tcpdump [33], Cisco's NetFlow [15], or others). We assume that all the packets are encrypted and padded to the same size, and hence only arrival time is of interest. The arrival times of packets at input link *i* form a time series

$$A_i = (a_{i,1}, \cdots, a_{i,n}) \tag{1}$$

where $a_{i,k}$ is the k^{th} packet's arrival time at input link i, and n 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

$$B_j = (b_{j,1}, \cdots, b_{j,m}) \tag{2}$$

where $b_{j,k}$ is the k^{th} packet's arrival time at output link j, and m is the size of the sample collected during a given sampling interval. The packets come out from mixes in batches. The length of sampling interval usually is 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 because the flow traffic characteristic is indeed observable at the input or because it was observable at the input of the mix network.

(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 "similarity" 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 4.3, we will discuss specific definitions of transformations for different batching strategies. Currently, for the convenience of discussion, let us assume that A_i is transformed into pattern vector $X_i = (x_{i,1}, \cdots, x_{i,q})$. And time series B_j is transformed into $Y_j = (y_{j,1}, \cdots, y_{j,q})$. Note, here the two pattern vectors have the same length.

(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 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. In Section 4.2, we will discuss two effective distance functions: one is based on mutual information and the other is based on the frequency-spectrum-based matched filter.

(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.

This approach can be easily extended to cases when multiple flows are aggregated over an input link [36]. The conclusions we obtained in this paper, however, are consistent with those obtained in [36]. The key idea is that by properly calculating the distance, we can find a correlation between one input flow and a set of output flows.

4.2 Flow Pattern Vector Extraction

In this subsection, we discuss how to choose pattern vectors X_is and Y_js. We will start with pattern vectors for the output link traffic first. 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). We will see that different classes should have different transformation methods.

For threshold triggered batching strategies, packets come out from the mix in batches. Hence, the inter-arrival time of packets in a batch is determined by the transmission latency, 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 elapses between two

[1] 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.

batches. Normalizing this relationship, we define the elements in pattern vector Y_j as follows:

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

In the calculation, we may need to truncate the original time series $B_j = (b_{j,1}, b_{j,2}, \cdots, b_{j,n})$ so that only complete batches are used.

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 pre-defined constant. Thus, following a similar argument made for the threshold triggered batching strategies, we define the elements in pattern vector Y_j as follows:

$$\begin{aligned} Y_{j,k} &= \frac{\text{Number of packets in the } k^{th}\text{time out interval}}{\text{(time of } k^{th} \text{ time-out) - (time of } (k-1)^{st} \text{ time-out)}} & (4) \\ &= \frac{\text{Number of packets in the } k^{th} \text{ time out interval}}{\text{Pre-defined inter-time-out length}} & (5) \end{aligned}$$

Again, in the calculation, we may need to truncate the original time series B_j so that only complete batches are used.

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 sub-intervals and calculate the average traffic rate in each sub-interval 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 sub-interval partition depends on how the mix is batching the packets. Due to the space limitation, we will not further discuss the details of the methods developed. Readers are referred to [36] for details.

4.3 Distance Functions

In this paper, 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. 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 $\{(X_{i,1}, Y_{j,1}), \cdots, (X_{i,q}, Y_{j,q})\}$ correspond to a sample of the joint random variable $(\mathcal{X}_i, \mathcal{Y}_j)$. With these definitions, the distance function $d(X_i, Y_j)$ 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 $\mathcal{Y}_j$,

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

Here, we need to estimate marginal distributions ($p(x_i)$ and $p(y_j)$) and their joint distribution $p(x_i, y_j)$. In this paper, we use histogram-based estimation of mutual information $\hat{I}(\mathcal{X}_i, \mathcal{Y}_j)$ of continuous distributions [18], which is given as follows.

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

where q is the sample size. 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 cell (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 therefore use FFT 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 flow 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 $M(X_i^F, Y_j^F)$,

$$d(X_i, Y_j) = \frac{1}{M(X_i^F, Y_j^F)} = \frac{1}{\frac{<X_i^F, Y_j^F>}{||Y_j^F||}} \tag{8}$$

where $< X_i^F, Y_j^F >$ is the inner product of X_i^F and Y_j^F, and $||Y_j^F|| = \sqrt{< Y_j^F, Y_j^F >}$. Please refer to [16] for details about the calculation of FFT and Wavelet over a vector. Due to the space limit, please refer to [36] for detailed results of the Wavelet-based method, which has similar results to the FFT method reported in this paper.

5 Empirical Evaluation

In this section, we evaluate the effectiveness of a selection of batching strategies (listed in Table 1) for a mix under our flow correlation attacks. We will see the failure of a mix under our traffic flow correlation attacks and batching strategies' influence on TCP flow performance.

5.1 Experiment Network Setup

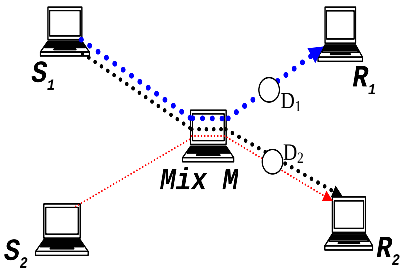


Figure 3. Experiment Setup

Figure 3 shows our experimental network setup. Our mix is implemented on Timesys/Real Time Linux operating system for its timer accuracy [34]. The Mix control module that performs the batching and reordering functions is integrated into Linux's firewall system [20] using *Netfilter*; we use the corresponding firewall rules to specify what traffic should be protected. 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 dominance in the Internet. However, the results are generally applicable to other kinds of flows. 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. 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 from the perspective of the FTP flow. We maintain the traffic rate on both output links of the mix at approximately 500 packets per second (*pps*). The objective of the adversary in this experiment is to identify the output link that carries the FTP flow.

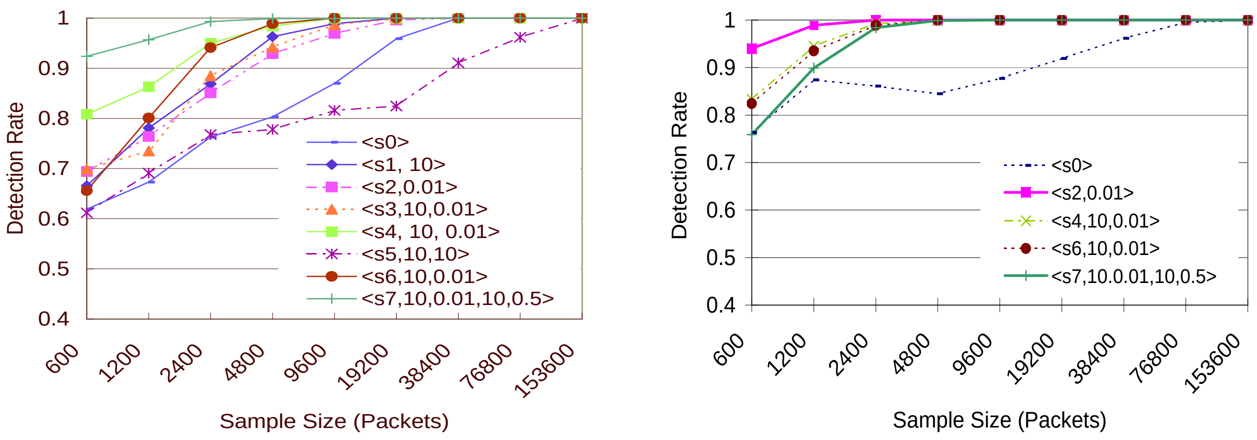
5.2 Metrics

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, we measure its *efficiency* in terms of quality of service (QoS) perceived by the applications. We use *FTP goodput* as an indication of FTP quality of service (*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

Figure 4 shows the detection rate for systems using a link-based batching strategy. Figure 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% of the number -sample size- of packets. Parameters in the legends of these figures are listed in the same order as in Table 1.



(a) Mutual Information

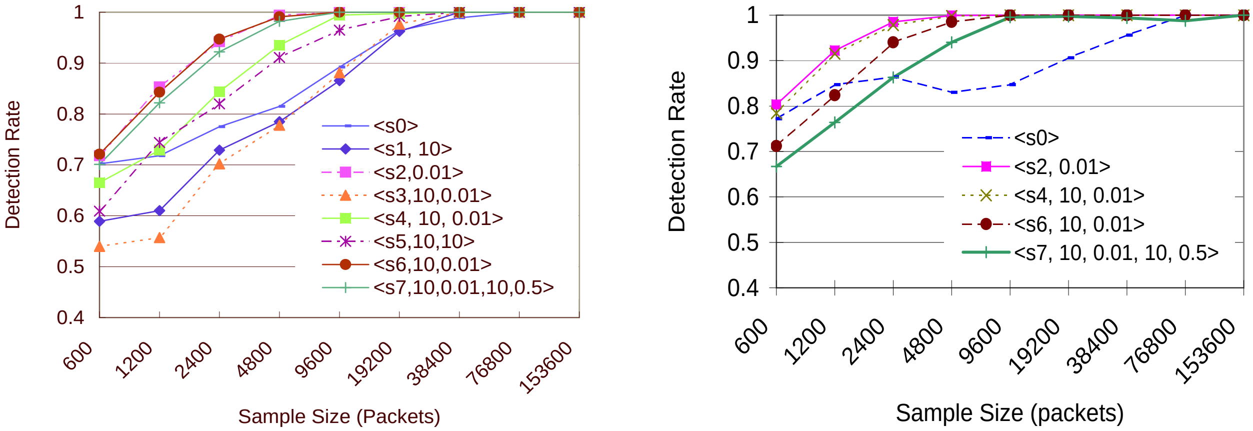
(b) Matched Filter

Figure 4. Detection Rate for Link-based Batching

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% when the sample size is sufficiently large. This is consistent with intuition, as more data implies that there is more information about the input flow, which in turn improves the detection rate.

2. Different strategies display different resistances to flow correlation attacks. Here there are some phenomena that contradict intuition: (a) the strategy without any batching, i.e., strategy S_0 in Table 1, is not always the



(a) Mutual Information

(b) Matched Filter

Figure 5. Detection Rate for Mix-based Batching

worst one in terms of countering the attack. (b) Some researchers in previous studies argued that pool mixes (strategies S_5 to S_7) perform better than simple mixes (strategies S_1 to S_4) in message-based mix networks. Our figures 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.

3. Frequency-analysis-based distance functions typically outperforms mutual-information-based distance functions in terms of detection rate. For many batching strategies, the former performs significantly better. This is because there are phasing issues in frequency-analysis-based attacks. Therefore, lack of synchronization between data collected at input and output port has a minor effect on the effectiveness of the attack.

4. To compare mix-based batching strategy with link-based batching strategy, we find that no one dominates the other.

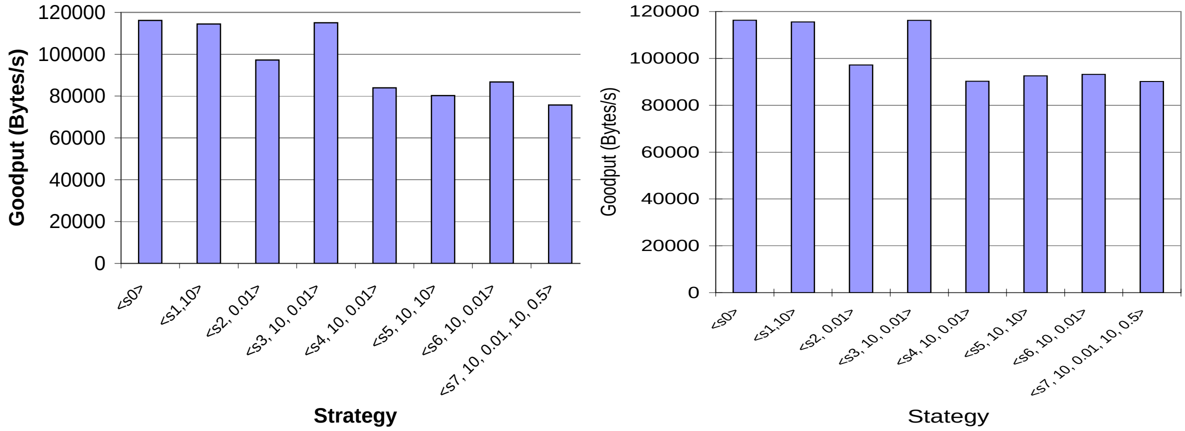
Overall, our data shows that the mix using any of batching strategies S_1, S_2, $\cdots$, S_7 fails under the flow correlation attacks. One of the reasons is that TCP flows often demonstrate interesting patterns such as periodicity of rate change and burstiness in particular when the TCP loop-control mechanism is triggered by excessive traffic perturbation in the mixes. Figure 4 and 5 show that flow correlation attacks can well explore the 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. Figure 6 quantitatively shows the degradation of FTP goodput for a mix using different batching strategies.

In Figure 6, we compare FTP goodput between a strategy without any batching (S_0) and other batching strategies ($S_1, S_2, \cdots, S_7$). We still use the network setup in Figure 3. The traffic other than FTP is configured as follows: 400pps from S_1 to R_1 and 500pps from S_2 to R_2. Based on these experiments and the results illustrated in Figure 6, we make the following observations:

1. FTP goodput is decreased because of the use of batching.



(a) Link Based Batching

(b) Mix Based Batching

Figure 6. FTP Goodput

2. Different batching strategies have different impact on the FTP goodput. In general, pool batching strategies (strategy S_5 to S_7) cause a worse FTP goodput than simple batching strategies (strategy S_1 to 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.

   Chaum mentioned this problem in [6]. He proposed to use dummy traffic to reduce the possible long delay of payload packets on a mix. Thus, FTP's performance can actually be limited by other traffic flows.

6 A Countermeasure and its Performance

From the discussion above, it is apparent that traditional batching strategies and reordering are not sufficient for mixes to effectively counter flow correlation attacks. Additional measures are needed. In this section, we introduce a relatively efficient and effective countermeasure and evaluate its performance in terms of FTP goodput.

6.1 Overview

A class of possible countermeasures can be developed based on the lessons learned in the previous sections. If a flow correlation attack relies on comparisons of pattern vectors of outgoing traffic, it will be ineffective when all packet vectors are identical. Thus, this type of flow correlation attacks can be effectively countered if a mix can make all the output flows look identical. As a result, assuming that we have the input flow vector X_i and l output flow vectors $Y_1, \cdots, Y_l$,

$$d(X_i, Y_1) = \cdots = d(X_i, Y_j) = \cdots = d(X_i, Y_l), \tag{9}$$

and the only analysis strategy for an adversary would be to randomly guess which output flow is correlated to an input flow. This results in a detection rate of $\frac{1}{l}$.

Because naturally the rates of traffic along all the output links of a mix are different, we have to appropriately insert dummy packets to make all the output flows behave in the same way. A challenge here is to insert a minimum number of dummy packets.

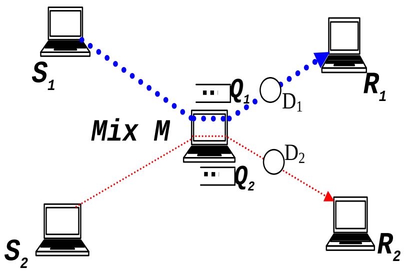
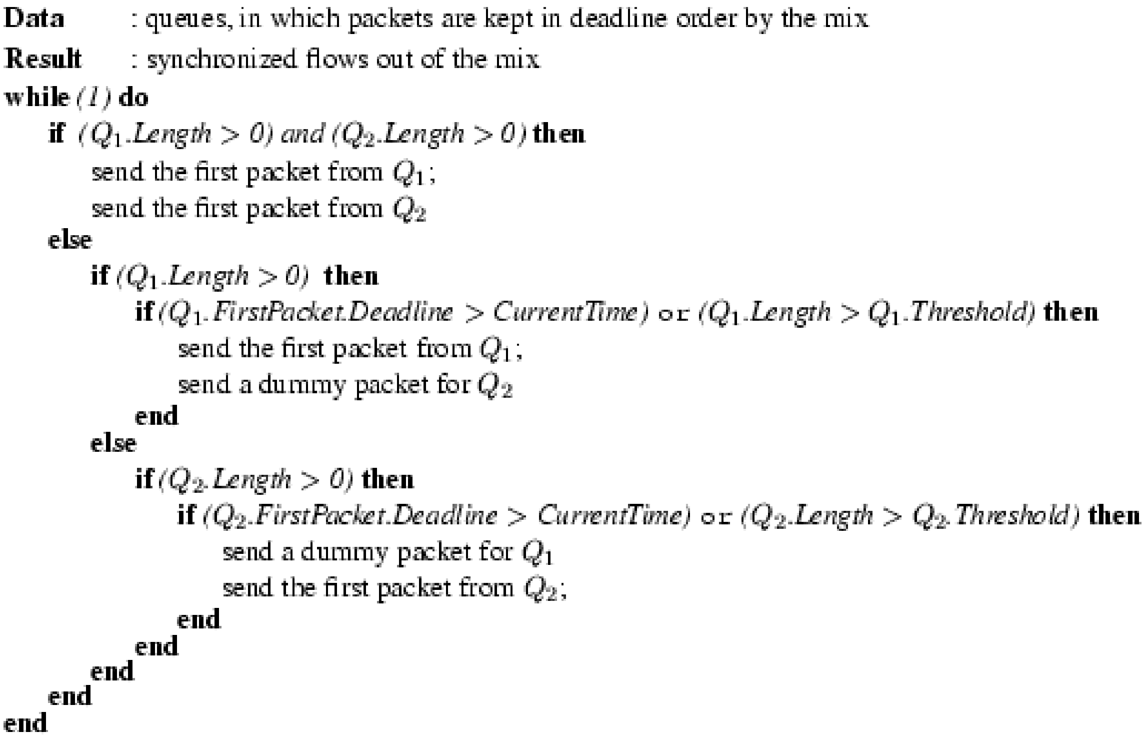


Figure 7. Network Setup for the New Countermeasure

Such an output-control algorithm is illustrated in Figure 7. Mix M maintains two output queues, Q_1 for the link between Mix M and node R_1, and Q_2 for the link between Mix M and node R_2. At any time, if each queue has a packet, they are sent out in some pre-defined order, e.g., the packet in Q_1 first and the packet in Q_2 second. By doing so, one of the two queues will be always empty. Let us say, for the moment, that Q_2 is empty. A deadline is assigned to each packet waiting in Q_1. If a packet in Q_1 reaches its deadline, a dummy packet will be generated for Q_2. Then, the payload packet from Q_1 and the dummy packet from Q_2 are sent out in the predefined order. A dummy packet will also be generated for Q_2 if the queue length of Q_1 goes beyond a preset threshold. In this way, we can ensure a maximum delay on each packet, and we also guarantee that neither queue will overflow.



```
Data      : queues, in which packets are kept in deadline order by the mix
Result    : synchronized flows out of the mix
while (1) do
    if ($Q_1$.Length > 0) and ($Q_2$.Length > 0) then
        send the first packet from $Q_1$;
        send the first packet from $Q_2$
    else
        if ($Q_1$.Length > 0)  then
            if ($Q_1$.FirstPacket.Deadline > CurrentTime) or ($Q_1$.Length > $Q_1$.Threshold) then
                send the first packet from $Q_1$;
                send a dummy packet for $Q_2$
            end
        else
            if ($Q_2$.Length > 0) then
                if ($Q_2$.FirstPacket.Deadline > CurrentTime) or ($Q_2$.Length > $Q_2$.Threshold) then
                    send a dummy packet for $Q_1$
                    send the first packet from $Q_2$;
                end
            end
        end
    end
end
```

Figure 8. Algorithm for Output Traffic Control

Figure 8 gives the new countermeasure algorithm on Mix M for the anonymity system in Figure 7. We can see that the output traffic of the Mix is now synchronized, and the adversary cannot observe any difference among the output flows.

This method can be easily extended and optimized for more complicated cases. The number of virtual output links of a mix can be very large since we assume a peer-to-peer mix network. Since we only maintain virtual queues, the overhead is limited. In the case of a large network with a small number of flows, there still needs to be a lower bound LB_Q of the number of virtual queues required for each mix to maintain anonymity. In other words, we do not necessarily need to synchronize every output link when traffic is slow, but we will synchronize a minimum number LB_Q of links. For example, if there is one virtual queue with a packet whose deadline is reached, we have to send out dummy packets to the other $LB_Q - 1$ virtual links.

Output traffic control is not new and has been proposed for example in [26], where messages at the output ports are forwarded periodically[2] The algorithm in Figure 8 is more efficient and probably more effective than the approach described in [26]. It is more efficient because packets are forwarded based on each queue's status: once each queue has payload packets, the first packet in each queue is sent out and packets suffer smaller delay at Mixes. It is likely more effective because periodic traffic patterns are very difficult to generate with sufficient accuracy. We showed in NetCamo [10, 11], for example, how high-accuracy traffic analysis can easily break periodic link padding schemes.

6.2 Performance Evaluation of Output Traffic Control

We are interested in how traffic flows traversing a mix affect each other. In particular, we evaluate the TCP performance. Again FTP is used as an example in the evaluation.

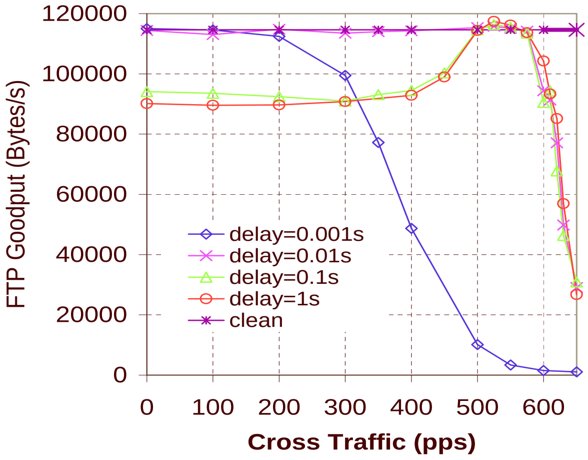


Figure 9. FTP Goodput Using Output Traffic Control ("clean" means no output traffic control)

Figure 9 gives the FTP goodput measurement for our new scheme for the network setup in Figure 7. We set the threshold of each queue at *50* packets. The path from S_2 to R_2 has FTP traffic and UDP traffic of 400pps. Cross traffic in Figure 9 refers to the UDP traffic along the path S_1 to R_1. Both paths have a propagation delay of *0.3* second. We have the following observations from these experiments:

1. While not evident from Figure 9, the observed detection rate of the correlation attack is 50% in all the cases when the new countermeasure is used. This is expected, as the new method can guarantee a detection rate of $1/LB_Q$ where $LB_Q = 2$ in this case.

2. The goodput for the clean FTP is 114,628.83 bytes/s. When the delay parameter is set to 0.01s, the same goodput is achieved as long as the cross traffic is less than 525 pps. This is very significant. It indicates that,

[2] The paper is too vaguely written for us to figure out exactly what forwarding mechanism is used.

once the delay parameter is properly selected, our new method can achieve high throughput (as high as the case without mix) while guaranteeing a low detection rate.

3. For the cases of delay equal to 0.01s, 0.10s, and 1.00s, right after the cross traffic goes beyond 525 pps, all have their goodput drop rapidly. This is due to the fact that the cross traffic is so heavy that the FTP's TCP protocol detects congestion and adapts accordingly.

4. It is also interesting to note, that when the cross traffic is low and the value of delay parameter is large (say, the cross traffic is less than 500 pps and delay is equal to 0.10s or 1.00s), the goodput is low (about 93,000 bytes/s). This is consistent with intuition: if the cross traffic is low and delay is large, then the traffic of our FTP flow may have to wait longer than in other cases, resulting in a reduction of goodput.

5. Finally, in the case when the value of delay parameter is small, say, equal to 0.001s, the curve of goodput is monotonically decreasing. In this case, it is likely that a packet from the FTP flow will be transmitted due to the deadline expiration, rather than the arrival of a packet from the cross traffic. Thus, the cross traffic always contributes negatively to the goodput performance here by creating dummy packets.

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 proper statistical analysis, an adversary can always 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% as long as enough data is available. This is true even if heavy cross traffic exists. The experimental 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 formation of a new packet control method for mixes in order to overcome their vulnerability to flow correlation attacks. Our new method 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. The new method is flexible in controlling the overhead by adjusting the maximum packet delay.

Our study is the first that systematically models and analyzes flow correlation attacks and their countermeasures. The work presented in this paper is largely empirical. We are currently developing an analysis framework that allows quick, back-of-the-envelope calculations to assess the effectiveness of batching strategies in countering flow correlation attacks. It is an open question what statistical analysis methods an adversary may use. Performance bounds and estimates in terms of detection rate and throughput may be developed by following the approaches taken in [9] and [22], respectively.

References

[1] Onion Routing Development Achives. Link padding and the intersection attack. `http://archives.seul.org/or/dev`, 2002.

[2] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[3] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[4] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[5] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. `http://osiris.978.org/˜brianr/crypto-research/anon/www.freedom.net/prod%ucts/whitepapers/Freedom_System_2_Architecture.pdf`, Dec. 2000.

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

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

[8] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[9] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On effectiveness of link padding for statistical traffic analysis attacks. *ICDCS*, 2003.

[10] Xinwen Fu, Bryan Graham, Dong Xuan, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. *ICPP*, 2003.

[11] Y. Guan, X. Fu, D. Xuan, P. U. Shenoy, R. Bettati, and W. Zhao. Netcamo: Camouflaging network traffic for qos-guaranteed critical allplications. In *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, Special Issue on Information Assurance*, volume 31 of *4*, pages 253–265, July 2001.

[12] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.

[13] Johan Helsingius. Press release: Johan helsingius closes his internet remailer. `http://www.penet.fi/press-english.html`, 1996.

[14] John D. Howard. An analysis of security incidents on the internet 1989 - 1995. Technical report, CMU dissertation, 1997.

[15] Cisco Systems Inc. Netflow services solutions guide. `http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwh%ite.htm`, 2003.

[16] MathWorks. Documentation for mathworks products (release 13). `http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml`, 2003.

[17] M. Mitomo and K. Kurosawa. Attack for Flash MIX. In *Proceedings of ASIACRYPT 2000*. Springer-Verlag, LNCS 1976, 2000.

[18] R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16(3):233–246, 1989.

[19] Ulf Möller and Lance Cottrell. Mixmaster Protocol — Version 2. `http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-mixmaster2-proto%col-00.txt`, January 2000.

[20] The netfilter/iptables project. Netfilter. `http://netfilter.samba.org/`, 2003.

[21] Federal Bureau of Investigations. Carnivore diagnostic tool. `http://www.fbi.gov/hq/lab/carnivore/carnivore2.htm`, 2003.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[23] Sameer Parekh. Prospects for remailers - where is anonymity heading on the internet. `http://www.firstmonday.dk/issues/issue2/remailers/`, 1996.

[24] J. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 10–29. Springer-Verlag, 2001.

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

[26] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, and D. Hutchison. Analysis of an anonymity network for web browsing. *WET ICE*, 2002.

[27] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *European Symposium on Research in Computer Security (ESORICS)*, 2003.

[28] A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: active attacks on several mix types. `citeseer.nj.nec.com/serjantov02from.html`, 2002.

[29] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. p^5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[30] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. *10th USENIX Security Symposium*, 2001.

[31] Qixiang Sun, Daniel R. Simon, Yimin Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. *IEEE Symposium on Security and Privacy*, 2002.

[32] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.

[33] tcpdump.org. tcpdump. `http://www.tcpdump.org/`, 2003.

[34] TimeSys. Timesys linux docs. `http://www.timesys.com/index.cfm?hdr=home\header.cfm&bdy=home\bdy\library.cfm`, 2003.

[35] Greg Walton. China's golden shield: Corporations and the development of surveillance technology in china. `http://www.totse.com/en/privacy/privacy/pucc.html`, 2003.

[36] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Correlation attacks in a mix network. *Texas A&M University Computer Science Technical Report TR2003-8-9*, 2003.