# Timing analysis in low-latency mix networks: attacks and defenses [⋆]

Vitaly Shmatikov and Ming-Hsiu Wang

The University of Texas at Austin

**Abstract.** Mix networks are a popular mechanism for anonymous Internet communications. By routing IP traffic through an overlay chain of mixes, they aim to hide the relationship between its origin and destination. Using a realistic model of interactive Internet traffic, we study the problem of defending low-latency mix networks against attacks based on correlating inter-packet intervals on two or more links of the mix chain. We investigate several attack models, including an active attack which involves adversarial modification of packet flows in order to "fingerprint" them, and analyze the tradeoffs between the amount of cover traffic, extra latency, and anonymity properties of the mix network. We demonstrate that previously proposed defenses are either ineffective, or impose a prohibitively large latency and/or bandwidth overhead on communicating applications. We propose a new defense based on adaptive padding.

## 1 Introduction

Mix networks are a practical way to enable anonymous communications on public networks. The goal is to hide the relationship between the origin of the traffic (*e.g.*, a Web browser) and the destination (*e.g.*, a website). A *mix*, first proposed by Chaum [6], can be thought of as a server that accepts incoming connections and forwards them in such a way that an eavesdropper cannot easily determine which outgoing connection corresponds to which incoming connection.

Because any given mix may be compromised, traffic is usually routed through a chain of mixes. Intuitively, even if some mixes in the chain are malicious, the other ones provide some anonymity for connections routed through them. Many different architectures for mix networks have been proposed in literature [3, 13, 9, 18, 11]. We focus on *low-latency* mix networks, whose main purpose is to protect privacy of *interactive* Internet communications, including popular applications such as Web browsing. Empirical evidence indicates that low-latency anonymity systems attract many more users than high-latency ones [12].

Like any anonymity system, a mix network can be attacked in a variety of ways. Some of the mix routers may be compromised by the attacker; endpoints of a repeatedly used chain may be linked by statistical analysis of message distribution within the network [7, 10]; statistical properties of randomly constructed routes may be exploited to determine the likely route origin [32, 26], and so on.

[⋆] This work was partially supported by NSF grants CNS-0509033 and IIS-0534198, and the Department of Defense under Contract No. H98230-05-C-1650.

In this paper, we assume that the attacker has direct access to packet streams on some of the network links. Many mix networks are specifically intended to provide anonymity against attackers who control the communication medium. Traffic analysis is an especially serious threat for low-latency mix networks because it is very difficult to hide statistical characteristics of the packet stream *and* satisfy the stringent latency requirements imposed by interactive applications.

Details of the route establishment protocol are not important for our analysis. Once a route through the mix network has been constructed, all packets are usually encrypted and padded to hide payload size. Inter-packet time intervals are usually *not* hidden because low latency requires that each packet be dispatched as soon as it has been generated by the application. This can be exploited by the attacker. By correlating inter-packet intervals on two links, he may be able to determine with high probability that the links belong to the same route.

We will refer to this as the *timing analysis* attack. This attack is probabilistic, and may suffer from false positives and false negatives. The standard measure of success is the *crossover error rate*, at which the attacker's false positive rate is equal to his false negative rate. The lower the crossover error rate, the more successful the attack. The conventional defense is to send *cover traffic* on each link in order to hide actual packets in the stream of padding (dummy) packets.

**Our contributions.** We analyze resilience of low-latency mix networks to inter-packet interval correlation attacks using a *realistic* traffic model based on HTTP traces from National Laboratory for Applied Network Research (NLANR) [21].

We propose *adaptive padding*, a new defense against timing analysis. In our scheme, intermediate mixes inject dummy packets into statistically unlikely gaps in the packet flow, destroying timing "fingerprints" without adding any latency to application traffic. We also present a version of our scheme which defends against active attackers at the cost of relatively small extra latency.

The purpose of adaptive padding is to prevent the attacker from determining which of the multiple simultaneous connections is being carried on a given network link, not to hide whether a certain user or connection is currently active. Constant-rate padding may provide better protection for the latter, although variants such as defensive dropping [17] are trivially defeated by measuring packet density because real and dummy packets are dropped at different rates.

We focus on short-lived connections, and do *not* aim to provide a comprehensive defense against long-term statistical disclosure attacks. It is not clear whether this is at all achievable — in any real-world network, there will inevitably exist small statistical differences between packet flows which cannot be completely hidden unless dummy traffic generators are perfectly synchronized.

Using simulated traffic, we quantify the basic tradeoff between the padding ratio (average number of dummy packets per real packet) and protection against timing analysis. We show that adaptive padding can defeat timing analysis with relatively low padding ratios, *e.g.*, 0.4 attacker's crossover error rate can be achieved with the $2:3$ average ratio between dummy and real packets. (Maximum crossover error rate is 0.5, which corresponds to random guessing.) We also investigate *active* attacks, in which artificial gaps and bursts are introduced into packet flows in order to "fingerprint" them. Adaptive padding provides significant protection against active attacks at a relatively low extra latency cost.

We compare adaptive padding with defenses based on sender-originated, constant-rate cover traffic and variants such as defensive dropping, showing that they are not feasible for traffic exhibiting realistic statistical characteristics. To defeat passive timing analysis with reasonable padding ratios, they require prohibitively high extra latency. They also fail completely against an active attack.

**Organization of the paper.** We survey related work in section 2. Our model and metrics are described in section 3, the adaptive padding scheme in section 4. Simulation methodology and results are presented in section 5. In section 6, we discuss active attacks; in section 7, we argue that constant-rate defenses are not feasible. Deployment issues are in section 8, future directions in section 9.

## 2 Related work

Traffic analysis attacks based on packet flow correlation [28, 2, 1, 25] and statistical characteristics of individual mixes [8] have been recognized as a serious threat to low-latency mix networks, but few defenses have been proposed to date.

Venkatraman and Newman-Wolfe [22, 31] presented a mathematical model for passive traffic analysis attacks and proposed a defense that requires complete knowledge of traffic on all links by a trustworthy entity.

Timmerman [29, 30] proposed an adaptive "traffic masking" technique for latency-insensitive applications such as email. Her S-DATM algorithm uses cover traffic and artificial delays to ensure that traffic emerging from each user conforms to a certain profile. By contrast, we move the responsibility for traffic shaping *inside* the mix network, and do not aim to precisely reproduce a particular traffic shape (as discussed below, this requires prohibitive latencies).

Berthold and Langos [4] also focus on high-latency networks, proposing that intermediate mixes inject dummy traffic to hide whether a connection is active or not. By contrast, our goal is to prevent the attacker from using fine-grained timing characteristics of the packet stream to determine which of several simultaneous connections is carried by a given network link.

Rennhard *et al.* [24] present an adaptive technique for artificially delaying packets from multiple connections at intermediate mixes in order to reduce the amount of cover traffic. A similar technique without any cover traffic was proposed by Zhu *et al.* [33]. With a realistic model of actual traffic, however, the timing "fingerprints" of two flows are likely to be sufficiently different so that the only effective defense is to actively reshape the flows at intermediate routers. Unfortunately, the cost of this defense is prohibitive latency.

Fu *et al.* [14, 15], followed by Levine *et al.* [17], demonstrated that even packet flows protected by constant-rate cover traffic are vulnerable to statistical analysis of inter-packet intervals. Fu *et al.* propose to generate cover traffic with variable inter-packet intervals, which is achieved by our adaptive padding scheme.

In the "defensive dropping" scheme [17], route initiators generate dummy packets, marking each one so that it is dropped by one of the intermediate mixes. The initiator must send *all* traffic at the same constant rate, delaying real packets so as not to exceed the chosen rate. Bursty traffic generated by interactive applications suffers vast extra latency in this case (see section 7). The possibility of an active attack is mentioned in [17], but it is not included in the simulations, and no defenses are proposed.

Devastating timing attacks have been successfully demonstrated in real-world mix networks [19, 23]. To prevent a particular type of timing analysis performed by a malicious client, Øverlier and Syverson recommend using a trusted entry node [23], which is complementary to the defenses proposed in this paper.

## 3 Model and metrics

**Network.** We use a simplified model of a mix network. A single connection consists of an *initiator* (or *sender*), a sequence (*path*) of $N$ mixes, and a destination server. In a "short-path" network, $N$ is set randomly to 2 or 3; in a "long-path" one, $N$ is selected randomly from between 5 and 8. We ignore the system-specific details of the path establishment protocol, and assume that all packets from the initiator to the server follow the same path through the (overlay) mix network.

We make the standard assumption that, following the path establishment protocol, all intermediate mixes share pairwise symmetric keys with the initiator, and that each consecutive pair of mixes on a path shares a pairwise symmetric key. We assume an end-to-end TCP connection between the initiator and the server, *i.e.*, there is no separate TCP connection between each consecutive pair of intermediate mixes. An example of such a system is Tarzan [13]. We further discuss feasibility of our techniques in various types of mix networks in section 8.

**Timing analysis.** We consider an attacker who measures *inter-packet intervals*, *i.e.*, time differences between observations of consecutive packets, on two network links in order to infer whether these links carry the same connection. Even when packets are padded and encrypted, inter-packet intervals tend to remain correlated within the same IP flow. Moreover, traffic associated with interactive applications such as Web browsing tends to occur in bursts. Sequences of inter-packet intervals vary widely between different packet flows, and can thus be used to "fingerprint" a connection. Following Levine *et al.* [17], we assume that the attacker divides time into fixed-size windows, counts the number of packets observed during each window, and correlates the sequences of packet counts.

An *active* attacker can also impose his own unique timing signature (by dropping packets or introducing artificial bursts) onto the flow he is interested in, and then attempt to identify this signature on other network links.

Our attack model is deliberately simple. We ignore the effects of packet drops and bursts on higher-level TCP behavior. In a real attack, the attacker may also look at timing characteristics other than inter-packet intervals, actively modify packet streams in order to cause observable changes in network behavior, corrupt packets to cause TCP retransmission, and so on. Our model is sufficient for demonstrating serious problems with previously proposed solutions, and enables us to directly compare adaptive padding with defensive dropping [17].

**Defense metric.** Our attacker correlates packet counts on two network links within a certain time window. If the correlation coefficient exceeds some threshold, the attacker decides that the links carry the same flow. This analysis can suffer from *false positives* (the attacker erroneously determines that unrelated links carry the same flow) and *false negatives* (the attacker erroneously determines that the links are unrelated even though they do carry the same flow).

High correlation thresholds increase the false negative rate and decrease the false positive rate, while low thresholds do the opposite. The standard metric in this situation is the *crossover error rate* (called *equal error rate* in [17]), at which the false positive rate is equal to the false negative rate. A low crossover error rate means that the attacker achieves *both* a low false positive rate and a low false negative rate, *i.e.*, the defense is ineffective. On the other hand, *high crossover rates mean that the defense is good.* The highest crossover rate is 0.5. If the error rate is greater than 0.5, the attacker can simply flip all the answers.

**Negative impact on network performance.** Defenses against timing analysis use dummy traffic to hide gaps between real packets and/or alter timing patterns of flows by delaying packets or dropping them completely. Both techniques have negative consequences for network performance as observed by the end users. Adding dummy traffic consumes bandwidth, while delaying or dropping packets increases latency. The ideal defense should minimize both effects.

Our metrics are the average *padding ratio* of dummy packets to real packets across all links of a single connection, and the maximum and average extra *delay* per real packet in the defended viz. undefended network.

## 4 Adaptive padding

After a mix receives a packet, our adaptive padding algorithm samples from the statistical distribution of inter-packet intervals. If the next packet arrives before the chosen interval expires, it is forwarded and a new value is sampled. To avoid skewing resulting intervals towards short values, the distribution is modified slightly to increase the probability of drawing a longer interval next time. If the chosen interval expires before a packet arrives, the gap is "repaired" by sending a dummy packet. Each mix is assumed to have a store of properly encrypted dummy packets, ready for injection into the packet stream (see section 8).

We assume that each mix knows a rough statistical distribution of inter-packet intervals for a "normal" flow. This distribution can be pre-computed from traffic repositories such as NLANR [21], or from the mix's own observations. Our defense is fundamentally probabilistic, and may provide relatively poor protection for flows whose distribution of inter-packet intervals is very different from the assumed distribution. Nevertheless, our method is a significant improvement over alternatives that simply ignore statistical characteristics of the protected flows by assuming that all senders emit traffic at the same constant rate.

**Data structures.** For each connection that passes through it, a mix maintains a data structure consisting of several *bins.* The bins are mutually exclusive and

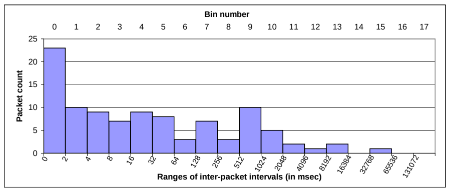

**Fig. 1.** Data structure representing distribution of inter-packet intervals at each mix.

jointly cover all interval values from 0 to infinity. Each bin except the last corresponds to a finite range of inter-packet intervals; the last bin represents all intervals longer than a certain value. We will numbers bins as $b_0$ (corresponding to the shortest inter-packet intervals), $b_1$, and so on. For interactive Internet traffic, the distribution of inter-packet intervals tends to be wide but heavily biased to short values. Therefore, we found that increasing the range represented by each bin exponentially with the bin index works well. Intuitively, short inter-packet intervals, which occur frequently, are split into several bins, while long intervals, which are rare, are allocated to a single bin (see example in fig. 1).

**Adaptive padding algorithm.** We assume that a route through the network has been established prior to actual communication. Upon receiving the first packet of a connection, the mix forwards it and initializes the bins by drawing a sample of $N$ values from the statistical distribution of inter-packet intervals, where $N$ is a parameter of the system. Each sampled value falls into some range represented by exactly one bin, and a *token* is placed into that bin. For simplicity, a token counter is associated with each bin, and is incremented appropriately.

Upon receiving a packet, the mix randomly selects a token and removes it from its bin. An *expected inter-packet interval* (EIPI) is chosen randomly from the range represented by that bin. If another packet does *not* arrive before EIPI expires, the mix sends a dummy packet to the next mix, and selects a new token (the old token is *not* replaced). If a packet arrives before EIPI expires, this means that the mix has chosen a "wrong" interval. It places the token back into its bin, calculates the actual interval between the new packet and its predecessor, removes a token from the bin corresponding to that interval, and forwards the packet. This is done to avoid skewing the actual inter-packet interval distribution towards short values. A new token is then randomly chosen, and so on.

The number of tokens decreases as more packets arrive on the connection. Rarely, on an unusually long-lived connection, a packet may arrive after an interval that falls into an empty bin $b_i$. In this case, to avoid significantly affecting statistical characteristics of the resulting flow, the token is removed from the next non-empty bin $b_j$ such that $i < j$. Alternatively, all bins can be re-filled from the distribution; the difference between the two methods is negligible in practice.

If all bins are empty, they are re-filled using a new sample from the distribution of inter-packet intervals, and the distribution itself updated, if necessary.

This adaptive padding algorithm repairs gaps that may have been caused by intentional or unintentional packet drops. The algorithm is probabilistic, and may randomly select a very short EIPI when there is a "legitimate" gap in the packet flow (*e.g.*, the sender's application is silent). This may result in tremendous amounts of cover traffic. One solution is to ignore the bins corresponding to low intervals in the token selection algorithm, *i.e.*, to only insert dummy packets into relatively large gaps. We investigate the tradeoff between padding ratios and effectiveness of adaptive padding against traffic analysis in section 5.2.

**Destroying natural fingerprints.** Real packet flows tend to be bursty, and each flow naturally has a unique pattern of inter-packet intervals which can be used as its "fingerprint" by the attacker in order to distinguish two or more flows. This fingerprint can be eliminated by dispatching all packets at the same constant rate, but this imposes prohibitive extra latency (see section 7).

The basic algorithm described above selects a new EIPI after receiving a packet or after the previous EIPI expired. To destroy natural patterns, however, gaps should be filled without adding packets where traffic is already dense. We thus propose a more sophisticated *dual-mode* algorithm.

**(Burst)** After a packet has been received, a new expected inter-packet interval is selected only from *higher* bins, *i.e.*, those associated with long intervals. This collection of bins will be referred to as the *High-Bins Set* (HBS).

**(Gap)** After the previously chosen expected inter-packet interval expired without receiving a packet and a dummy packet has been sent, the new interval is selected only from *lower* bins, *i.e.*, those associated with short intervals. This collection of bins will be referred to as the *Low-Bins Set* (LBS).

Intuitively, when the flow contains a natural burst, the next expected interval is chosen to be long to decrease the chance of introducing a dummy packet where packet density is already high. When the flow contains a natural gap, the next interval is short, to increase the chance of introducing a dummy packet. HBS and LBS can be configured based on observed traffic characteristics (see below).

Basic adaptive padding does not impose any extra latency on real packets. (A small delay is needed to defeat active attacks, as described in section 6.) Extra processing at each mix consists of generating a random number for each received packet, and, if necessary, drawing a dummy packet from its store. This is negligible compared to decryption and re-encryption that must be performed on each packet. The exact amount of dummy traffic is a parameter of the system, and depends on the desired effectiveness against timing analysis (see section 5.2).

## 5 Experimental evaluation

We evaluate four schemes: undefended, defensive dropping [17], a variant of defensive dropping with constant-rate *cover* traffic (*i.e.*, real traffic is not delayed, only dummy packets are sent at constant rate), and adaptive padding. For each scheme, attacker's crossover error rate is computed on 3000 simulated flows.

To simplify comparisons with [17], we use the same model for network links. The packet drop rate is drawn from an exponential distribution with the mean of either 5% (between the sender and the first mix), or 1% (mix-to-mix links). The average delay $d$ is uniformly random between 0 and 100 ms. The actual delay for each packet is drawn from an exponential distribution with mean $d$.

The traffic model in our simulations is based on actual TCP traffic traces from Lawrence Berkeley Laboratory [16] and NLANR archives [21]. We found that our results are consistent across all traces. The data presented in the rest of the paper are based on the NLANR Auckland-VIII data set [20]. The distribution of inter-packet intervals within a single TCP connection, used by the adaptive padding algorithm in our simulations, was also extracted from these traces.

To simulate application traffic, we select an inter-packet interval at random from the distribution, generate a "real" packet after the interval expires, and repeat. For adaptive padding and undefended simulations, this is exactly the traffic emitted by the sender. For the variant of defensive dropping with constant-rate cover traffic, a real packet is sent as soon as it is generated, and a dummy packet is sent at some constant rate. For standard defensive dropping, both real and dummy packets are sent at the same constant rate, *i.e.*, real packets are placed in a queue until it's time to send them. We do *not* model TCP acknowledgements, re-transmissions, exponential backoff in response to dropped packets, and other TCP features that may be exploited by a sophisticated attacker. Our simple model is sufficient to demonstrate the power of inter-packet interval correlation.

In the undefended simulation, each mix simply forwards packets to the next mix. With defensive dropping, the first mix drops a dummy packet with probability 0.6. With adaptive padding, mixes inject dummy packets using the dual-mode algorithm of section 4 and the pre-set distribution of inter-packet intervals.

## 5.1 Attack model

The attacker observes a set of *entry* links and a set of *exit* links with the goal to determine which exit link corresponds to which entry link.

Attacker's observation time is set to 60 seconds, enough to defeat previously proposed defenses. Increasing observation time reduces effectiveness of *any* defense, including ours. Even small statistical differences between packet flows can be detected if the observation time is long enough. We conjecture that long-term attacks cannot be prevented without assuming that some senders or mixes emit cover traffic in perfect synchrony, which cannot be achieved by any real system.

The attacker divides time into windows of $W$ seconds [17]. Empirically, $W = 1$ gives the most accurate results. For each observed link, the attacker counts the number of packets $x_k$ during the $k$th window, producing a sequence of packet counts for each link. For every possible entry-exit pair, he computes the cross-correlation of the two sequences as $r(d) = \frac{\sum_i((x_i-\mu)(x'_{i+d}-\mu'))}{\sqrt{\sum_i(x_i-\mu)^2}\sqrt{\sum_i(x'_{i+d}-\mu')^2}}$, where delay $d = 0$ and $\mu$, $\mu'$ are the means of the two sequences.

If the correlation $r(d)$ for a pair of links exceeds some threshold $t$, the attacker determines that the links carry the same flow. Otherwise, he determines that they carry different flows. For a given $t$, the false positive rate is the fraction of pairs that carry different flows but were erroneously determined to carry the same flow;

the false negative rate is the fraction of same-flow pairs that were erroneously determined to carry different flows. The attacker chooses $t$ so that the false positive and false negative rates are equal. This is the attacker's crossover error rate. High crossover rate means that the defense is effective.

## 5.2 Evaluation results

An overview of our results is in fig. 2. The crossover rates for defensive dropping and adaptive padding are from configurations that produce, on average, one dummy packet for each real packet (1:1 padding ratio). The constant rate for both types of defensive dropping is 6.67 packets per second.

**Undefended.** Timing analysis is extremely effective against unmodified network flows. (Recall that only inter-packet intervals are unmodified; all other defenses, including encryption, are deployed, and no mixes are corrupted.) The crossover rate is close to 0, *i.e.*, there exists a correlation threshold that results in negligible false positive and false negative rates. Inter-packet intervals on two links of the same path have correlation close to 0.9 vs. less than 0.3 for unrelated links.

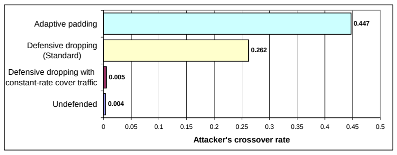

**Fig. 2.** Comparison of four defenses against timing analysis.

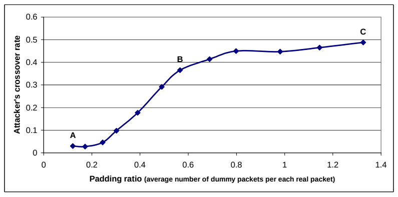

**Fig. 3.** Padding ratio vs. effectiveness against timing analysis.

**Defensive dropping with constant-rate cover.** The variant of defensive dropping in which dummy packets are sent at constant rate while real packets are sent as soon as they are produced by the application does not provide much protection, with the crossover rate close to 0. Constant-rate cover traffic may hide periods of inactivity, but does not eliminate patterns of inter-packet intervals.

**Adaptive padding and defensive dropping.** Defensive dropping and adaptive padding are the two defenses that offer some protection against timing analysis, increasing the attacker's crossover rate to above 0.25 in our simulations.

The two defenses work for different reasons. Defensive dropping decreases correlations within the same flow to $0.4 - 0.6$, while raising (erroneous) correlations of different flows to $0.3 - 0.5$. This is due to constant-rate cover traffic, which causes all flows to look similar initially. For defensive dropping to work, other flows must be sent at the *same constant rate* as the protected flow.

Adaptive padding, on the other hand, raises correlation between different flows only to $0.1 - 0.3$, and most of the defensive effect is due to lowering correlation within the same flow to $0.2-0.4$. Adaptive padding thus provides *standalone* defense for a flow even if other flows do not use any defenses.

Fig. 3 displays the fundamental tradeoff of adaptive padding between the padding ratio and the attacker's crossover error rate. As the padding ratio increases, the attacker's error rate goes up at the cost of increased network congestion, as more dummy packets must be generated for each real packet.

At point A in fig. 3, LBS is bins $b_{4-7}$, and HBS is bins above $b_{12}$. The error rate is raised only to .03, but only 1 dummy packet is needed per 9 real packets. Defensive dropping achieves the same 1:9 padding ratio if the constant rate is set to 3 packets per second. The resulting error rate of 0.27 is better than adaptive padding, but the average *extra latency* per packet exceeds 3.5 seconds (see fig. 6).

At point B in fig. 3, LBS is set to bins $b_{3-6}$, and HBS to bins above $b_{11}$. On average, this requires 0.56 dummy packets per each real packet and achieves 0.37 error rate with zero extra latency. By contrast, the constant rate that achieves a comparable padding ratio for defensive dropping results in (significantly worse) 0.2 error rate, with average extra latency of 1.1 seconds per packet.

At point C, LBS is bins $b_{0-8}$, and HBS is bins above $b_{10}$. The resulting padding ratio is around 1.3:1, and the attacker's error rate is 0.48, close to theoretically optimal. In our simulations, defensive dropping was unable to achieve similar error rates with padding ratios under 50:1.

**Short paths.** When paths are short (2 or 3 mixes) and defensive dropping is used, attacker's error rates decrease slightly. Fewer variations due to natural network delays and drops are accumulated by the flows, and distinguishing features of entry links are still pronounced on exit links, leading to higher correlations.

With adaptive padding, crossover rates decrease, too. Padding ratios decrease as well, because fewer mixes inject dummy packets, *e.g.*, at point B in fig. 3, the error rate is 0.19 with the padding ratio of 1:3. Decreasing padding ratios tend to outweigh decreasing error rates, *i.e.*, for a given padding ratio, the error rate for shorter paths is comparable or better than that for longer paths.

## 6 Active attacks

In addition to passively observing network links, an *active* attacker may impose his own timing signature onto the target flow and attempt to recognize this signature on other network links. We assume that he cannot create new packets (this requires knowledge of the symmetric keys of all subsequent mixes), nor replay packets (this is easy to prevent with caching and hash checking).

**Artificial gaps.** The attacker may drop several consecutive packets in the target flow to create a large gap. Fig. 4 shows the results of simulating 3000 normal flows and and 3000 target flows in which the attacker drops several consecutive packets on the entry link 1 second after the flow has started.

Dropping even a small number of consecutive packets drastically decreases the effectiveness of defensive dropping. With constant rate, all flows look very similar initially, so a noticeable change to one of the flows, such as introduction of a large gap, is almost guaranteed to create a recognizable feature.

With adaptive padding, artificial drops do not decrease the attacker's error rate. Intermediate mixes are likely to reduce the gap by injecting dummy packets. Moreover, other flows may have similar gaps due to natural variations.

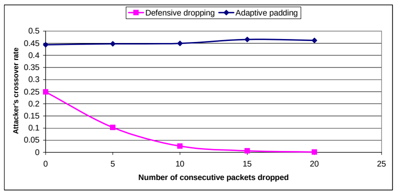

**Fig. 4.** Active dropping attack.

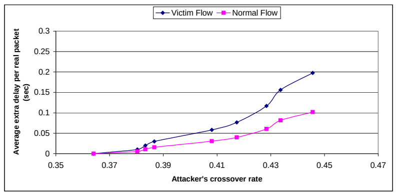

**Fig. 5.** Latency penalty vs. attacker's error rate for 5-sec artificial burst attack.

**Artificial bursts.** The attacker can create a signature from artificial packets bursts by holding up packets on a link and then releasing all of them at once.

We simulated this attack with a 5-second and 15-second attacker's queue (while very effective, the latter is somewhat unrealistic, as it is likely to result in the higher-level TCP connection simply being dropped). Defensive dropping provides no defense: the crossover rate is 0 in both cases, *i.e.*, the attacker can perfectly identify the target flow. With adaptive padding, the crossover rate drops from .45 to .36 with a 5-second queue, and to .21 with a 15-second queue.

Our modified adaptive padding algorithm breaks up bursts by queueing all packets whose inter-arrival time is in the first bin (*i.e.*, very short). Each such packet is delayed for a short random interval. Fig. 5 shows the tradeoff between the crossover rate and the extra latency imposed on real packets. As expected, the longer the delay, the better the defense. This penalty is paid only by packets with extremely short inter-arrival times; the impact on normal flows is small.

## 7 Comparison with constant-rate defenses

We further compare adaptive padding with constant-rate defenses, including variations such as the defensive dropping scheme of Levine *et al.* [17].

**Latency is prohibitive.** Constant-rate defenses fundamentally assume that senders emit traffic at a constant rate. Low-latency mix networks, however, are intended to provide anonymity for *interactive* applications such as Web browsing, and it is well-known (and borne out by real-world traces such as [21]) that Web traffic is bursty. Therefore, the sender must delay packets when the rate of actual traffic generated by the application exceeds the pre-set constant rate. Furthermore, this delay propagates to *all* subsequent packets. If the rate of real traffic exceeds the pre-set constant rate, packet delays increase to infinity.

Fig. 6 quantifies the latency penalty. For example, if the constant rate is 5 packets per second, real packets are delayed by 1.1 seconds on average. This delay

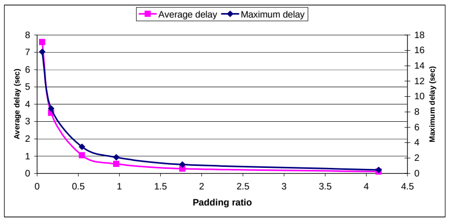

**Fig. 6.** Latency penalty of constant-rate defenses.

may be acceptable for relatively one-directional and non-interactive applications. Constant-rate defenses may thus be a good choice for users who are willing to tolerate increased latencies and are not worried about active attacks.

For interactive applications, however, extra latency is likely to be prohibitive. Moreover, there is evidence that low latency is essential for user adoption of anonymity systems [12]. A non-trivial increase in latency may cause fewer users to participate in the system, resulting in lower baseline anonymity.

The average *maximum* delay for each flow in our simulations is approximately 3.5 seconds, with many flows delayed by almost 10 seconds. Delays like these are likely to result in dropped TCP connections, disabling the network for all purposes (such effects are beyond the scope of our simplified model).

By contrast, adaptive padding does not impose any extra latency against a passive attacker, and only a small latency against an active attacker.

**Everybody must send at the same constant rate.** As observed in section 5.2, constant-rate defenses are effective only if *most* flows in the network are emitted at the same constant rate, which is clearly unrealistic. On the other hand, adaptive padding is effective in protecting a single flow even if other flows in the network do not use any defense against timing analysis.

**There is no "right" constant rate.** It may appear that the latency problem may be solved by setting a high constant rate which matches the shortest inter-packet interval(s) of actual traffic. Unfortunately, this is not feasible. Inter-packet intervals associated with traffic bursts are so short that the constant rate must be exceedingly high, resulting in vast amounts of dummy traffic when bursts are *not* occurring. From the network perspective, this "solution" is equivalent to taking the most congested time slice and expanding it to the entire connection.

**Defense fails against active attacks.** As shown in section 6, constant-rate defenses, including defensive dropping, do not provide any defense against artificial gaps or bursts introduced by an active attacker.

**Defense fails at high traffic rates.** With defensive dropping, only dummy packets may be dropped by intermediate mixes. If the application generates real packets at a higher rate than the chosen constant rate, most packets on each link are real and cannot be dropped. This becomes simple constant-rate traffic, which is vulnerable to basic timing analysis [14, 17].

**Defense reveals presence of real traffic.** Even if real traffic is sparse (but bursty), the constant-rate blend of real and cover traffic produced by defensive dropping will likely consist of alternating sequences of real and dummy packets. Because only dummy packets may be dropped by intermediate mixes, the attacker can conclude that periods of sparse packets are padding and periods of dense packets are real traffic. If constant-rate cover traffic with defensive dropping is used to hide whether the connection is active or not, the attacker can break the defense simply by observing packet density.

## 8 Creation and management of dummy packets

The main feature of our approach is that dummy packets are injected into the flow by intermediate mixes (as opposed to the route initiator), in order to "smooth out" statistical anomalies in inter-packet intervals. In the simplest case, the mix creates dummy packets itself, encrypting them with the next mix's key. The next mix decrypts the packet, recognizes it as a dummy, re-encrypts it and forwards it on. A passive attacker cannot feasibly distinguish encrypted dummy packets from encrypted real packets by observing the network. An active attacker who can compromise a mix, however, will be able to recognize dummy packets generated by the preceding mixes, negating the effects of cover traffic.

**Pre-computation of dummy packets.** For security in the presence of compromised mixes, injected dummy packets should be indistinguishable from real packets by all subsequent mixes. If layered encryption is used, dummy packets should be encrypted with the the same keys in the same order as real packets.

An intermediate mix does not know its successors in the mix chain, and thus cannot properly encrypt a dummy packet itself. One solution is to have the initiator *pre-compute* large batches of dummy packets for all mixes in the chain. This is done *offline*, *e.g.*, during route setup, and thus has no impact on the bandwidth and latency of actual communication. During route setup, each mix receives from the initiator a batch of dummy packets. The batch is encrypted with their shared pairwise key. Each dummy packet is properly encrypted with the keys of all successor mixes (this does not leak their identities). Whenever the mix needs to inject a dummy packet, it simply gets it from the batch. None of its successors on the route can tell a real packet from an injected dummy packet.

These batches can be replenished periodically, or, depending on the implementation, a mix may signal to the route initiator that it needs a new batch when the connection is quiet. This approach is secure against compromised mixes, and trades off storage at the intermediate mixes against online computation (since mixes no longer generate dummy packets on the fly), resulting in faster performance. It also prevents malicious mixes from flooding the connection with bogus dummy packets, because they will not decrypt properly at the successor mixes.

Injection of pre-computed packets into a stream of encrypted packets assumes that encryption is block cipher-based, as in, *e.g.*, the Freedom system [5]. If a stream cipher is used, as in onion routing [27, 11], the state of the cipher used for the $i$th layer of encryption must be synchronized between the sender and the

$i$th mix in the path. Because the sender cannot predict when an intermediate mix may need to inject a dummy packet, pre-computation is infeasible, and an alternative mechanism such as reverse paths (see below) must be used.

**Malicious clients and servers.** Vulnerability of real-world mix networks to timing analysis performed by malicious clients and servers has been shown by, respectively, Øverlier and Syverson [23], and Murdoch and Danezis [19]. In our case, we assume that dummy packets are sent beyond the last mix to the destination server, and that the latter can recognize and discard them.

This can sometimes be achieved without server cooperation. For example, if the sender knows that the server discards all packets with an invalid message authentication code (MAC), he can append invalid MACs to all dummy packets. Even if the attacker compromises the last mix, he does not learn the key shared between the sender and the destination, and thus cannot check MAC validity.

In general, adaptive padding requires server cooperation, and thus does not protect from malicious servers. This may appear to be a serious disadvantage viz. sender-originated cover traffic, but, in reality, many low-latency applications such as Web browsing are bidirectional, and thus also require server cooperation. For instance, if defensive dropping is used to protect HTTP connections, a malicious server can easily track propagation of its responses back to the client.

Protection offered by adaptive padding (or, in general, by any mix-injected cover traffic) is a mirror image of protection offered by sender-originated cover traffic. With the former, the last link of a mix chain is padded, but the first link is not. With the latter, the first link is padded, but the last link is not. Therefore, the former requires server cooperation, while the latter requires client cooperation (and is insecure against a malicious client).

**Reverse routes.** Another solution is to reverse the conventional route setup process so that the server (rather than the initiator) ends up sharing a key with each mix in the chain. The sender encrypts packets only with the server's key and sends them to the first mix on the path. Each succeeding mix encrypts the packet with the key it shares with the server. The server unwraps all onion layers of encryption. (A similar mechanism is used by location-hidden servers in Tor.) With reverse routes, intermediate mixes can easily inject dummy packets into the flow — all they need to do is simply encrypt them with the key they share with the next mix, and send them with the proper route identification.

The Øverlier-Syverson attack on hidden servers demonstrates the importance of protecting reverse paths from malicious clients. Since this cannot be done with sender-based defenses, mix-injected cover traffic is a better solution in this case.

## 9 Challenges and future directions

Attacks considered in this paper by no means exhaust the list of possible threats against low-latency mix networks. We made the standard, unrealistic assumption that all connections start at the same time. In reality, attacks based on correlating start and end times of connections may prove very successful. In general, it is very difficult to hide connection start and end times using dummy traffic because the mix network handles dummy and real packets differently (*e.g.*, dummies can or even should be dropped, while real packets are never dropped).

With adaptive padding, any intermediate mix may inject a dummy packet. The more mixes a flow has traversed, the denser it tends to become. The attacker may be able to estimate the hop count of a flow by measuring its density. This cannot always be used to attack anonymity, however. Furthermore, padding ratios are higher on the links closer to the destination. Even when the average padding ratio is low, links toward the end of the path may experience more congestion. On the flip side, the client-side links, which tend to be slower and less reliable, are free of padding. In two-way communication, where the same route with adaptive padding is used in both directions, the padding ratios balance out.

## References

1. A. Back, I. Goldberg, and A. Shostack. Freedom Systems 2.1 security issues and analysis. `http://www.freehaven.net/anonbib/cache/freedom21-security.pdf`, May 2001.
2. A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proc. 4th International Workshop on Information Hiding*, volume 2137 of *LNCS*, pages 245–257, 2001.
3. O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: a system for anonymous and unobservable Internet access. In *Proc. Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 115–129, 2000.
4. O. Berthold and H. Langos. Dummy traffic against long-term intersection attacks. In *Proc. 2nd International Workshop on Privacy-Enhancing Technologies*, volume 2482 of *LNCS*, pages 110–128, 2002.
5. P. Boucher, A. Shostack, and I. Goldberg. Freedom Systems 2.0 architecture. `http://www.freehaven.net/anonbib/cache/freedom2-arch.pdf`, December 2000.
6. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
7. G. Danezis. Statistical disclosure attacks. In *Proc. Security and Privacy in the Age of Uncertainty*, volume 250 of *IFIP Conference Proceedings*, pages 421–426, 2003.
8. G. Danezis. The traffic analysis of continuous-time mixes. In *Proc. 4th International Workshop on Privacy-Enhancing Technologies*, volume 3424 of *LNCS*, pages 35–50, 2004.
9. G. Danezis, R. Dingledine, and N. Mathewson. Mixmixion: Design of a type III anonymous remailer protocol. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–15, 2003.
10. G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proc. 6th International Workshop on Information Hiding*, volume 3200 of *LNCS*, pages 293–308, 2004.
11. R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *Proc. 13th USENIX Security Symposium*, pages 303–320, 2004.
12. R. Dingledine, N. Mathewson, and P. Syverson. Challenges in deploying low-latency anonymity. NRL CHACS Report 5540-265, 2005.
13. M. Freedman and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proc. 9th ACM Conference on Computer and Communications Security*, pages 193–206, 2002.

14. X. Fu, B. Graham, R. Bettati, and W. Zhao. On effectiveness of link padding for statistical traffic analysis attacks. In *Proc. 23rd IEEE Conference on Distributed Computing Systems*, pages 340–349, 2003.
15. X. Fu, B. Graham, R. Bettati, W. Zhao, and D. Xuan. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proc. 32nd International Conference on Parallel Processing*, pages 483–492, 2003.
16. Internet Traffic Archive. Two hours of wide-area TCP traffic. `http://ita.ee.lbl.gov/html/contrib/LBL-TCP-3.html`, January 1994.
17. B. Levine, M. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems. In *Proc. 8th International Conference on Financial Cryptography*, volume 3110 of *LNCS*, pages 251–265, 2004.
18. U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. `http://www.abditum.com/mixmaster-spec.txt`, July 2003.
19. S. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proc. IEEE Symposium on Security and Privacy*, pages 183–195, 2005.
20. National Laboratory for Applied Network Research. Auckland-VIII data set. `http://pma.nlanr.net/Special/auck8.html`, December 2003.
21. National Laboratory for Applied Network Research. PMA special traces archive. `http://pma.nlanr.net/Special/`, 2005.
22. R. Newman-Wolfe and B. Venkatraman. High level prevention of traffic analysis. In *Proc. IEEE/ACM 7th Annual Computer Security Applications Conference*, pages 102–109, 1991.
23. L. Øverlier and P. Syverson. Locating hidden servers. In *Proc. IEEE Symposium on Security and Privacy*, 2006.
24. M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, and D. Hutchison. Analysis of an anonymity network for Web browsing. In *Proc. IEEE WETICE*, pages 49–54, 2002.
25. A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *Proc. 8th European Symposium on Research in Computer Security*, volume 2808 of *LNCS*, pages 116–131, 2003.
26. V. Shmatikov. Probabilistic analysis of an anonymity system. *J. Computer Security*, 12(3-4):355–377, 2004.
27. P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. IEEE Symposium on Security and Privacy*, pages 44–54, 1997.
28. P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In *Proc. Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 96–114, 2000.
29. B. Timmerman. A security model for dynamic adaptable traffic masking. In *Proc. New Security Paradigms Workshop*, pages 107–116, 1997.
30. B. Timmerman. Secure adaptive traffic masking. In *Proc. New Security Paradigms Workshop*, pages 13–24, 1999.
31. B. Venkatraman and R. Newman-Wolfe. Performance analysis of a method for high level prevention of traffic analysis using measurements from a campus network. In *Proc. IEEE/ACM 10th Annual Computer Security Applications Conference*, pages 288–297, 1994.
32. M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proc. ISOC Network and Distributed System Security Symposium*, 2002.
33. Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proc. 4th International Workshop on Privacy-Enhancing Technologies*, volume 3424 of *LNCS*, pages 207–225, 2004.