# Practical Traffic Analysis: Extending and Resisting Statistical Disclosure

Nick Mathewson and Roger Dingledine

The Free Haven Project
{nickm,arma}@freehaven.net

**Abstract.** We extend earlier research on mounting and resisting passive long-term end-to-end traffic analysis attacks against anonymous message systems, by describing how an eavesdropper can learn sender-receiver connections even when the substrate is a network of pool mixes, the attacker is non-global, and senders have complex behavior including generating padding messages. Additionally, we describe how an attacker can use extra information about message distinguishability to speed the attack. Finally, we simulate our attacks for a variety of scenarios, focusing on the amount of information needed to link senders to their recipients. In each scenario, we show that the intersection attack can still succeed, albeit more slowly—in some cases, so slowly as to be impractical.

## 1 Introduction

Mix networks aim to allow senders to anonymously deliver messages to recipients. One of the strongest attacks against current deployable designs is the *long-term intersection attack*. In this attack, a passive eavesdropper observes a large volume of network traffic and notices that certain recipients are more likely to receive messages after particular senders have transmitted messages. That is, if a sender (call her Alice) maintains a fairly consistent pattern of recipients over time, the attacker can deduce Alice's recipients.

Researchers have theorized that these attacks should be extremely effective in many real-world contexts, but so far it has been difficult to reason about when they would be successful and how long the attacks would take.

Here we extend a version of the long-term intersection attack called the statistical disclosure attack [12] to work in real-world circumstances. Specifically, whereas the original model for this attack makes strong assumptions about sender behavior and only works against a single batch mix, we show how an attacker can learn Alice's regular recipients even when:

- Alice chooses non-uniformly among her communication partners, and can send multiple messages in a single batch.
- The attacker lacks *a priori* knowledge of the network's average behavior when Alice is not sending messages.
- Mixes use a different batching algorithm, such as Mixmaster's dynamic-pool algorithm [22, 27] or its generalization [14].

- Alice uses a mix network (of any topology, with synchronous or asynchronous batching) to relay her messages through a succession of mixes, instead of using just a single mix.
- Alice disguises when she is sending real messages by sending traffic padding to mix nodes in the network.
- The attacker can only view a subset of the messages entering and leaving the network (so long as this subset includes some messages from Alice and some messages to Alice's recipients).
- The cover traffic generated by other senders changes slowly over time. (We do not address this case completely).

Each deviation from the original model reduces the rate at which the attacker learns Alice's recipients, and increases the amount of traffic he must observe.

Additionally, we show how an attacker can exploit additional knowledge, such as distinguishability between messages, to speed these attacks. For example, the attacker can take into account whether messages are written in the same language or signed by the same pseudonym, to partition them into different classes and analyze the classes independently.

The attacks in this paper fail to work when:

- Alice's behavior is not consistent over time. If Alice does not produce enough traffic with the same group of regular recipients, the attacker cannot learn Alice's behavior.
- The attacker cannot observe how the network behaves in Alice's absence. If Alice always sends the same number of messages, in every round, forever, the attacker may not be able to learn who receives messages in Alice's absence.
- The attacker cannot tell when the sender is originating messages.

We begin in Section 2 by presenting a brief background overview on mix networks, traffic analysis, the disclosure attack, and the statistical disclosure attack. In Section 3 we present our enhancements to the statistical disclosure attack. We present simulated experimental results in Section 4, and close in Section 5 with recommendations for resisting this class of attacks, implications for mix network design, and a set of open questions for future work.

## 2 Previous work

Chaum [9] proposed hiding the correspondence between sender and recipient by wrapping messages in layers of public-key cryptography, and relaying them through a path composed of *mixes*. Each mix in turn decrypts, delays, and re-orders messages, before relaying them toward their destinations. Because some mixes might be controlled by an adversary, Alice may direct her messages through a sequence or 'chain' of mixes in a network, so that no single mix can link her to her recipient.

Many subsequent designs have been proposed, including Babel [18], Mixmaster [22], and Mixminion [13]. We will not address the differences between these systems in any detail: from the point of view of a long-term intersection attack, the internals of the network are irrelevant so long as the attacker can observe messages entering and leaving the network, and can guess when a message entering the network is likely to leave.

Another class of anonymity designs aims to provide low-latency connections for web browsing and other interactive activities [5, 8, 15, 25], but we do not address them in this paper because short-term timing and packet counting attacks seem sufficient against them [28].

Attacks against mix networks aim to reduce the anonymity of users by linking anonymous senders with the messages they send, by linking anonymous recipients with the messages they receive, or by linking anonymous messages with one another. For a detailed list of attacks, consult [1, 24]. Attackers can trace messages through the network by observing network traffic, compromising mixes, compromising keys, delaying messages so they stand out from other traffic, or altering messages in transit. They can learn a given message's destination by flooding the network with messages, replaying multiple copies of a message, or shaping traffic to isolate a target message from other unknown traffic [27]. Attackers can discourage users from using honest mixes by making them unreliable [1, 16]. They can analyze intercepted message text to look for commonalities between otherwise unlinked senders [23].

### 2.1 The intersection attack

Even if all the above attacks are foiled, an adversary can mount a *long-term intersection attack* by correlating the times at which senders and receivers are active [7].

A variety of countermeasures make intersection attacks harder. Kesdogan's stop-and-go mixes [20] provide probabilistic anonymity by letting users specify message latencies, thereby broadening the range of times messages might emerge from the mix network. Similarly, batching strategies [27] as in Mixmaster and Mixminion use message pools to spread out the possible exit times for messages.

Rather than expanding the set of messages that might have been sent by a suspect sender, other designs expand the set of senders that might have sent a target message. A sender who also runs a node in the mix network can conceal whether a given message originated at her node or was relayed from another node [4, 17, 26]. But even with this approach, the adversary can observe whether certain traffic patterns are present when a user is online (sending) and absent when a user is offline (not sending) [30, 31].

A sender can also conceal whether she is currently active by consistently sending decoy (dummy) traffic. Pipenet [10] conceals traffic patterns by constant padding on every link. Unfortunately, a single user can shut down the network simply by not sending. Berthold and Langos aim to increase the difficulty of intersection attacks with a scheme for preparing plausible dummy traffic and having other nodes send it on Alice's behalf when she is offline [6], but their design has many practical problems.

Finally, note that while the adversary can perform this long-term intersection attack entirely passively, active attacks (such as blending attacks [27] against a suspected sender) can help him reduce the set of suspects at each round.

### 2.2 The disclosure attack

In 2002, Kesdogan, Agrawal, and Penz presented the disclosure attack [19], an intersection attack against a single sender on a single batch mix.

The disclosure attack assumes a global passive eavesdropper interested in learning the recipients of a single sender Alice. It assumes that Alice sends messages to $m$ recipients; that Alice sends a single message (recipient chosen at random from $m$) in each batch of $b$ messages; and that the recipients of the other $b-1$ messages are chosen at random from the set of $N$ possible recipients.

The attacker observes the messages leaving the mix and constructs sets $R_i$ of recipients receiving messages in batch $i$. The attacker then performs an NP-complete computation to identify $m$ mutually disjoint recipient sets $R_i$, so that each of Alice's recipients is necessarily contained in exactly one of the sets. Intersecting these sets with subsequent recipient sets reveals Alice's recipients.

### 2.3 The statistical disclosure attack

In 2003, Danezis presented the statistical disclosure attack[12], which makes the same operational assumptions as the original disclosure attack but is far easier to implement in terms of storage, speed, and algorithmic complexity.

In the statistical disclosure attack, we model Alice's behavior as an unknown vector $\overrightarrow{v}$ whose elements correspond to the probability of Alice sending a message to each of the $N$ recipients in the system. The elements of $\overrightarrow{v}$ corresponding to Alice's $m$ recipients will be $1/m$; the other $N-m$ elements of $\overrightarrow{v}$ will be 0. We model the behavior of the cover traffic sent by other users as a known vector $\overrightarrow{u}$ each of whose $N$ elements is $1/N$.

The attacker derives from each output round $i$ an observation vector $\overrightarrow{o_i}$, each of whose elements corresponds to the probability of Alice's having sent a message to each particular recipient in that round. That is, in a round $i$ where Alice has sent a message, each element of $\overrightarrow{o_i}$ is $1/b$ if it corresponds to a recipient who has received a message, and 0 if it does not. Taking the arithmetic mean $\overline{O}$ of a large set of these observation vectors gives (by the law of large numbers):

$$\overline{O} = \frac{1}{t}\sum_{i=i}^{t}\overrightarrow{o_i} \approx \frac{\overrightarrow{v} + (b-1)\overrightarrow{u}}{b}$$

From this, the attacker estimates Alice's behavior:

$$\overrightarrow{v} \approx b\frac{\sum_{i=1}^{t}\overrightarrow{o_i}}{t} - (b-i)\overrightarrow{u}$$

Danezis also derives a precondition that the attack will only succeed when $m < \frac{N}{b-1}$, and calculates the expected number of rounds to succeed (with 95% confidence for security parameter $l = 2$ and 99% confidence for $l = 3$) [11]:

$$t > \left[ m \cdot l \left( \sqrt{\frac{N-1}{N}(b-1)} + \sqrt{\frac{N-1}{N^2}(b-1) + \frac{m-1}{m}} \right) \right]^2$$

## 3 Extending the statistical disclosure attack

### 3.1 Broadening the attack

Here we examine ways to extend Danezis's statistical disclosure attack to systems more closely resembling real-world mix networks. We will examine the time and information requirements for several of these attacks in Section 4 below, by running them against simulated networks.

**Complex senders, unknown background traffic:** First, we relax the requirements related to sender behavior. We allow Alice to choose among her recipients with non-uniform probability, and to send multiple messages in a single batch. We also remove the assumption that the attacker has full knowledge of the distribution $\overrightarrow{u}$ of cover traffic sent by users other than Alice.

To model Alice's varying number of messages, we use a probability function $P_m$ such that in every round Alice sends $n$ messages with probability $P_m(n)$. We still use a behavior vector $\overrightarrow{v}$ to represent the probability of Alice sending to each recipient, but we no longer require Alice's recipients to have a uniform $1/m$ probability. Alice's expected contribution to each round is thus $\overrightarrow{v} \sum_{n=0}^{\infty} n P_m(n)$.

To mount the attack, the attacker first obtains an estimate of the background distribution $\overrightarrow{u}$ by observing a large number $t'$ of batches to which Alice has *not* contributed any messages.[1] For each such batch $i$, the attacker constructs a vector $\overrightarrow{u_i}$, whose elements are $1/b$ for recipients that have received a message in that batch, and 0 for recipients that have not. The attacker then estimates the background distribution $\overrightarrow{u}$ as:

$$\overrightarrow{u} \approx \overline{U} = \frac{1}{t'} \sum_{i=1}^{t'} \overrightarrow{u_i}$$

The attacker then observes, for each round $i$ in which Alice *does* send a message, the number of messages $m_i$ sent by Alice, and computes observations $\overrightarrow{o_i}$ as before. Taking the arithmetic mean of these $\overrightarrow{o_i}$ gives us

$$\overline{O} = \frac{1}{t} \sum_{i=1}^{t} \overrightarrow{o_i} \approx \frac{\overline{m} \cdot \overrightarrow{v} + (b - \overline{m})\overline{U}}{b} \quad \text{where } \overline{m} = \frac{1}{t} \sum m_i$$

[1] The attack can still proceed if few such Alice-free batches exist, so long as Alice contributes more to some batches than to others. Specifically, the approach described below (against pool mixes and mix networks) can exploit differences between low-Alice and high-Alice batches to infer background behavior.

From this, the attacker estimates Alice's behavior as

$$\overrightarrow{v} \approx \frac{1}{\overline{m}}\left[b \cdot \overline{O} - (b - \overline{m})\overline{U}\right]$$

**Attacking pool mixes and mix networks:** Most designs have already abandoned fixed-batch mixes in favor of other algorithms that better hide the relation between senders and recipients. Such improved algorithms include timed dynamic-pool mixes, generalized mixes, and randomized versions of each [14, 27]. Rather than reordering and relaying all the messages whenever a fixed number $b$ have arrived, these algorithms store received messages in a *pool*, and at fixed intervals relay a *fraction* of the pooled messages based on the pool's current size.

When attacking such a mix, the attacker no longer knows for certain which batches of recipients contain a message from Alice. Instead, the attacker can only estimate, for each batch of output messages, the probability that the batch includes one of Alice's messages.

Following Díaz and Serjantov's approach in [14], we treat these mixing algorithms generically as follows: a mix relays a number of messages at the end of each round, depending on how many messages it is currently storing. All messages in the mix's pool at the end of a round have an equal probability of being included in that round's batch. Thus, we can characterize the mix's batching algorithm as a probability function $P_{\mathrm{MIX}}(b|s)$—the probability that the mix relays $b$ messages when it has $s$ messages in the pool. Clearly, $\forall s, \sum_{b=0}^{s} P_{\mathrm{MIX}}(b|s) = 1$: the mix will always output between 0 and $s$ messages.

We denote by $P_R^i(r)$ the probability that a message arriving in round $i$ leaves the mix $r$ rounds later. We assume that the attacker has a fair estimate of $P_R$.[2] Now, when Alice sends a message in round $i$, the attacker observes rounds $i$ through some later round $i + k$, choosing $k$ so that $\sum_{j=k+1}^{\infty} P_R^i(j)$ is negligible. The attacker then uses $P_R$ to compute $\overline{O_w}$, the mean of the observations from all of these rounds, weighted by the expected number of messages from Alice exiting in each round:

$$\overline{O_w} = \sum_i \sum_{r=0}^{k} P_R^i(r) \cdot m_i \cdot \overrightarrow{o_{i+r}} \approx \frac{\overline{m} \cdot \overrightarrow{v} + (\overline{n} - \overline{m})\overrightarrow{u}}{\overline{n}}$$

To solve for Alice's behavior $\overrightarrow{v}$, the attacker now needs a good estimate for the background $\overrightarrow{u}$. The attacker gets this by averaging observations $\overrightarrow{u_i}$ from batches with a negligible probability of including messages from Alice. Such batches, however, are not essential: If the attacker chooses a set of $\overrightarrow{u_i}$ such that each round contains (on average) a small number $\delta_a > 0$ of messages from Alice, averaging them gives:

$$\overline{U'} \approx \frac{\delta_a}{\overline{n}}\overrightarrow{v} + \frac{1 - \delta_a}{\overline{n}}\overrightarrow{u}$$

[2] The attacker can estimate $P_R$ by sending test messages through the mix, or by counting the messages entering and leaving the mix and deducing the pool size.

and the attacker can thus solve again for $\overrightarrow{v}$ in the earlier equation for $\overline{O_w}$, now using

$$\overrightarrow{u} \approx \frac{1}{1-\delta_a}\left[\overline{n}\cdot\overline{U'} - \delta_a \cdot \overrightarrow{v}\right]$$

Senders can also deviate from the original model by directing their messages through multi-hop paths in a network of mixes. While using a mix network increases the effort an attacker must spend to observe all messages leaving the system, it has no additional effect on intersection attacks beyond changing the delaying characteristics $P_R$ of the anonymity system.

Assume for the sake of simplicity that all mixes have the same delay distribution $P_R$, and that Alice chooses a path of length $\ell_0$. The chance of the message being delayed by a further $d$ rounds is now

$$P'_R(\ell_0 + d) = \binom{\ell_0 + d - 1}{d}(1 - P_D)^{\ell_0} P_D^d$$

If Alice chooses her path length probabilistically according to $P_L(\ell)$, we have

$$P'_R(r) = \sum_{\ell=1}^{r} P_L(\ell)\binom{r-1}{r-\ell}(1-P_D)^{\ell} P_d^{r-\ell}$$

Danezis has independently extended statistical disclosure to pool mixes [11].

**Dummy traffic:** Alice can also reduce the impact of traffic analysis by periodically sending messages into the network that are dropped inside the network.

Although these methods can slow or stop the attacker (as discussed below in Section 4), the change in the attack itself is trivial: Alice's behavior vector $\overrightarrow{v}$ no longer adds to 1, since there is now a chance that a message from Alice will not reach any recipient. Aside from this, the attacker can proceed as before, so long as Alice sends more messages (including dummies) in some rounds than in others.

**Partial observation:** Until now, we have required that the attacker, as a global passive adversary, observe all the messages entering and leaving the system (at least, all the messages sent by Alice, and all the messages reaching Alice's recipients). This requirement is not so difficult as it might seem: to be a "global" adversary against Alice, an attacker need only eavesdrop upon Alice, and upon the mixes that deliver messages to recipients. (Typically, not all mixes do so.)

A non-global attacker's characteristics depending on which parts of the network he can observe. If the attacker eavesdrops on a fraction of the *mixes* in the system, he receives a sample[3] of the messages entering or leaving the system. If such an attacker can see some messages from Alice and some messages to her recipients, he can still converge on the same $\overline{O}$ and thus the same estimation of Alice's behavior, but the attack will require more rounds of observation.

[3] But possibly a biased sample, depending on Alice's path selection algorithm.

Alternatively, an attacker who eavesdrops on a fraction of the *users* in the system receives *all* of the messages sent to or from those users but no messages sent to or from other users. So long as one of these users is Alice, the network (to such an attacker) is as if the messages sent by Alice to unobserved recipients were dummy messages. Now the attack converges as before, but with only information concerning the observed recipients: the attacker learns which of the observed recipients receive messages from Alice, and which do not.

**Time-variant background traffic:** If Alice's behavior changes completely and radically over time, long-term intersection attacks cannot proceed: the attacker cannot make enough observations of any version or subset of Alice's behavior to converge on a $\overline{v}$ for any of them.

On the other hand, if Alice's behavior $\overrightarrow{v}$ remains consistent while the behavior of the background traffic $\overrightarrow{u}$ changes slowly, the attacker still has some hope. Rather than estimating a single $\overline{U}$ from observations to which Alice does not contribute, the attacker estimates a series of successive $\overline{U_i}$ values based on the average behavior of the network during comparatively shorter durations of time. Now the attacker observes $\overrightarrow{o_i}$ as before and computes the average of $\overrightarrow{o_i} - \overline{U_i}$, as before. Now,

$$\overrightarrow{v} \propto \frac{1}{t} \sum_{i=1}^{t} \overrightarrow{o_i} - \overline{U_i}$$

So if an attacker can get good local estimates to $\overrightarrow{u}$, the intersection attack proceeds as before.

**Attacking recipients:** Finally, we note that an attacker can find recipients as well as senders by using slightly more storage and the same computational cost.

Suppose the attacker wishes to know which senders are sending anonymous messages to a given recipient Bob. The analysis remains the same: the attacker compares sender behavior in rounds from which Bob probably receives messages with behavior in rounds from which Bob probably doesn't receive messages. The only complication is that the attacker cannot tell in advance when Bob will receive a message. Therefore, the attacker must remember a window of recent observations at all times, such that if Bob later receives a message, the chance is negligible that the message was sent before the first round in the window.

## 3.2 Strengthening the attack

Section 3.1 showed how to extend the original statistical disclosure attack to reveal sender–recipient links in a broader range of circumstances.

In this section, rather than broadening the attack to work in new situations (at the expensive of needing increased traffic), we discuss ways to reduce the attack's required amount of traffic by incorporating additional information.

**Exploiting message partitioning:** The attacker's work is simplified if some output messages are *linkable.* Two messages are linkable if they are likelier to originate from the same sender than are two randomly chosen messages. We consider a special case of linkability, in which we discover linkage by *partitioning* messages into separate classes such that messages in the same class are likelier to come from the same sender than two messages chosen at random.

The easiest scenario for partitioning is pseudonymity: in a typical pseudonym service, each sender has one or more pseudonyms and all delivered messages are associated with a pseudonym. An eavesdropper who can connect senders to their pseudonyms could trivially use this information to connect senders and recipients. To do so, he might treat pseudonyms as virtual message destinations: instead of collecting observations $\overrightarrow{o_i}$ of recipients who receive messages in round $i$, the attacker now collects observations $\overrightarrow{o_i}$ of linkable classes (e.g. pseudonyms) that receive in round $i$. Since two distinct senders don't produce messages in the same linkability class, the elements of Alice's $\overrightarrow{v}$ and the background $\overrightarrow{u}$ are now disjoint, and thus easier for the attacker to separate.

It's also possible that the partitioning may not be so perfect: sometimes many senders will send messages in the same class. For example, two binary documents written in the same version of MS Word are more likely to be written by the same sender than two messages selected at random.

To exploit these scenarios, the attacker chooses a set of $c$ partitioning classes (such as languages or patterns of usage), and assigns to each observed output message a probability of belonging to each class. The attacker then proceeds as before, but instead of collecting observation vectors with elements corresponding the recipients, the attacker now collects observation vectors whose elements correspond to number of messages received in each round by each $\langle \text{recipient}, \text{class} \rangle$ tuple. (If a message might belong to multiple classes, the attacker sets the corresponding element of each possible class to the probability of the message's being in that class.) The statistical disclosure attack proceeds as before, but messages that fall in different classes no longer provide cover for one another.

**Exploiting *a priori* suspicion:** Finally, the attacker may have reason to believe that some messages are more likely to have been sent by the target user than others. For example, if we believe that Alice speaks Urdu but not Arabic, or that Alice knows psychology but not astrophysics, then we will naturally suspect that an Urdu-language message about psychology is more likely to come from Alice than is an Arabic-language message about astrophysics.

To exploit this knowledge, an attacker can (as suggested in the original statistical disclosure paper) modify the estimated probabilities in $\overrightarrow{o_i}$ of Alice having sent each delivered message.

## 4 Simulation results

In Section 3.1, we repeatedly claim that each complication of the sender or the network forces the attacker to gather more information. But how much?

To find out, we ran a series of simulations of our attacks, first against the model of the original statistical disclosure attack, then against more sophisticated models. We describe our simulations and present results below.

**The original statistical disclosure attack:** Our simulation varied the parameters $N$ (the number of recipients), $m$ (the number of Alice's recipients), and $b$ (the batch size). The simulated "Alice" sends a single message every round to one of her recipients, chosen uniformly at random. The simulated background sends to $b-1$ additional recipients per round, also chosen uniformly at random. We ran 100 trial attacks for each chosen $\langle N, m, b\rangle$ tuple. Each attack was set to halt when the attacker has correctly identified Alice's recipients, or when 1,000,000 rounds have passed. (We imposed this cap on run time to keep our simulator from getting stuck on hopeless cases.)

We present the results of our simulations in Figure 1 (the low-$m$ curves are at the bottom). As expected, the attack becomes more effective when Alice sends messages to only a few recipients (small $m$); when there are fewer recipients for Alice to hide hers among (small $N$); or when batch sizes are small (small $b$).

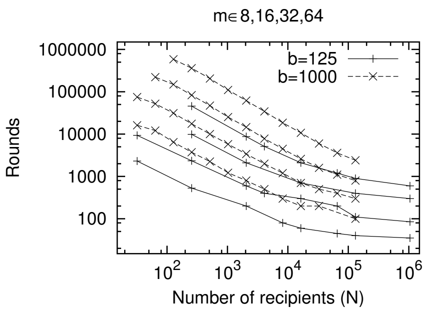

**Fig. 1.** Statistical disclosure model: Median rounds to guess all recipients

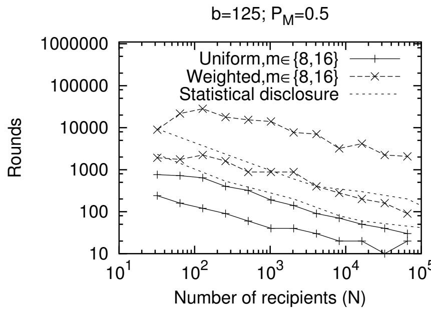

**Fig. 2.** Unknown background: Median rounds to guess all recipients

**Complex sender behavior and unknown background traffic:** The next simulation examines the consequences of a more complex model for background traffic, and of several related models for Alice's behavior.

We model the background as a graph of $N$ communicating parties, each of whom communicates with some of the others. We build this graph according to the "scale-free" model proposed in [2] and analyzed in [3], which shares desirable properties with "small-world" networks [29]. Scale-free networks share the "six degrees of separation property" (for arbitrary values of six) with small-world networks, but also mimic the clustering and 'organic' growth of real social networks, including citations in journals, co-stars in IMDB, and links in the WWW. For these trial attacks, the background messages were generated by choosing nodes from the graph with probability proportional to their popularity (connectedness). This simulates a case where users send messages with equal frequency and choose recipients uniformly from among the people they know.

Again, we simulated trial attacks for different values of $N$ (the number of recipients) and $m$ (the number of Alice's recipients). Instead of contributing one message per batch, however, Alice now contributes messages according to a geometric distribution with parameter $P_M$ (such that Alice sends $n$ messages with probability $P_m(n) = (1 - P_M)P_M^n$). We also tried two methods for assigning Alice's recipients: In the 'uniform' model, Alice's recipients are chosen according to their connectedness (so that Alice, like everyone else, is likelier to know people who are already well-known), but Alice still sends to each chosen recipient with equal probability. In the 'weighted' model, not only are Alice's recipients chosen according to their connectedness, but Alice also sends to them proportional to their connectedness. We selected these models to examine the attack's effectiveness against users whose behavior is generated with the same model as other users' (U), and against users who mimic the background distribution (W).

The results are in Figure 2, along with the results for the original statistical disclosure attack as reference. As expected, the attack succeeds easily, and finishes faster against uniform senders than weighted senders for equivalent values of $\langle N, m, b\rangle$. Interestingly, the attack against uniform senders is *faster* than the original statistical disclosure attack—because the background traffic is now clustered about popular recipients, Alice's recipients stand out more.

**Attacking pool mixes and mix network:** Pooling slows an attacker by increasing the number of output messages that can correspond to each input message. To simulate an attack against pool mixes and mix networks, we abstract away the actual pooling rule used by the network, and instead assume that the network has reached a steady state, so that each mix retains the messages in its pool with the same probability ($P_{\text{delay}}$) every round. We also assume that all senders choose paths of exactly the same length.

Unlike before, 'rounds' are now determined not by a batch mix receiving a fixed number $b$ of messages, but by the passage of a fixed interval of time. Thus, the number of messages sent by the background is no longer a fixed $b - n_a$ (where $n_a$ is the number of messages Alice sends), but now follows a normal distribution with mean $BG = 125$ and standard deviation set arbitrarily to $BG/10$.[4]

To examine the effect of pool parameters, we fixed $m$ at 32 and $N$ at $2^{16}$, and had Alice use the 'uniform' model discussed above. The results of these simulations are presented in Figure 3. Lines running off the top of the graph represent cases in which fewer than half of the attacks converged upon Alice's recipients within 1,000,000 rounds, and so no median could be found.

[4] It's hard to determine standard deviations for actual message volumes on the deployed remailer network: automatic reliability checkers that send messages to themselves ("pingers") contribute to a false sense of uniformity, while some users generate volume spikes by sending enormous fragmented files, or maliciously flooding discussion groups and remailer nodes. Neither group blends well with the other senders.

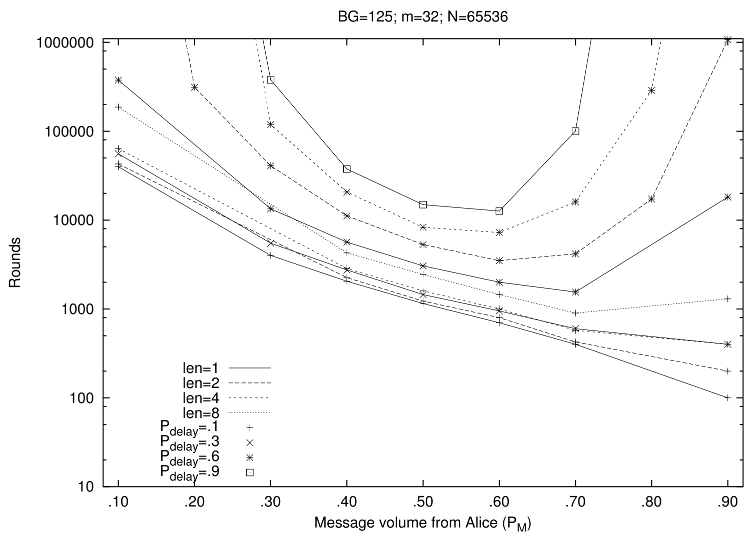

**Fig. 3.** Pool mixes and mix networks: Median rounds to guess all recipients

From these results, we see that increased variability in message delay slows the attack by increasing the number of output messages that may correspond to any input message from Alice, effectively 'spreading' each message across several output rounds. More interestingly, pooling is most effective at especially high or especially low volumes of traffic from Alice: the 'spreading' effect here makes it especially hard for the attacker to discern rounds that contain messages from Alice when she sends few messages, or to discern rounds that don't contain Alice's messages when she sends many messages.

**The impact of dummy traffic:** Several proposals exist for using dummy messages to frustrate traffic analysis. Although several of them have been examined in the context of low-latency systems [21], little work has been done to examine their effectiveness against long-term intersection attacks.

First, we choose to restrict our examination (due to time constraints) to the effects of dummy messages in several cases of the pool-mix/mix network simulation above. Because we are interested in learning how well dummies thwart analysis, we choose cases where, in the absence of dummies, the attacker had little trouble in learning Alice's recipients.

Our first padding strategy ("independent geometric padding") is based on the link padding strategy from the Mixminion design [13]: Alice generates a random number of dummy messages in each round according to a geometric distribution with parameter $P_{\text{junk}}$, independent of her number of real messages.

Padding slows the attack, but does not necessarily stop it. As shown in Figure 4, geometric padding is most helpful when the underlying mix network has a higher variability in message delay to 'spread' the padding between rounds. Otherwise, Alice must send far more padding messages to confuse the attacker.

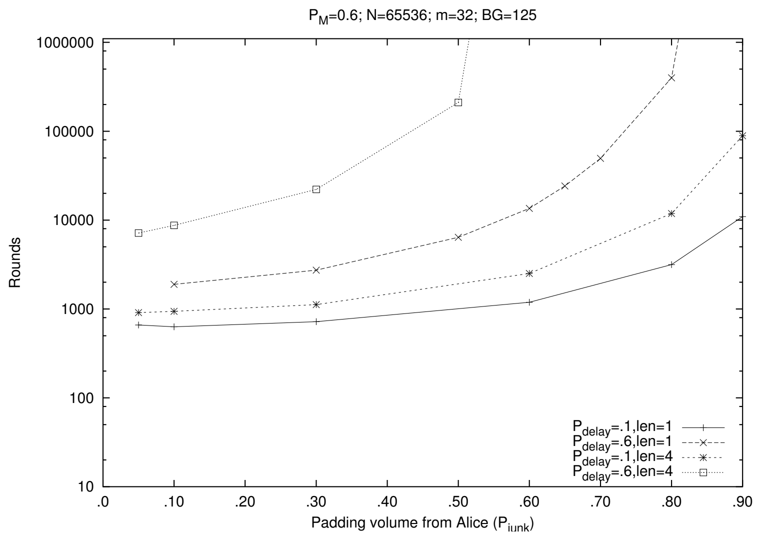

**Fig. 4.** Independent geometric dummy messages: Median rounds to guess all recipients

We are currently running our simulations on other padding models, including "imperfect threshold padding" (Alice always tries to pad up to a threshold of $M$ messages per round, but is sometimes offline).

**The impact of partial observation:** Finally, we examine the degree to which a non-global passive adversary can mount the statistical disclosure attack. Again, we base our simulation on the mix network simulation used as the basis for the padding trials above.

Clearly, if Alice chooses only from a fixed set of entry and exit mixes as suggested by [31], and the attacker is watching none of her chosen mixes, the attack will fail—and conversely, if the attacker is watching all of her chosen mixes, the attack proceeds as before. For our simulation, therefore, we assume that all senders (including Alice) choose all mixes as entry and exit points with equal probability for each message, and that the attacker is watching some fraction $f$ of the mixes. We simulate this by revealing each message entering or leaving the network to the attacker with probability $P_{\text{observe}} = f$. The attacker sees a message when it enters *and* when it exits with probability $(P_{\text{observe}})^2$.

The results in Figure 5 show that the attacker can still implement a long-term intersection attack even when he is only observing part of the network. When most of the network is observed ($P_{\text{observe}} > 70\%$ in our results), the attack is hardly impaired at all. As more of the network is concealed ($.4 < P_{\text{observe}} < .7$) the attack becomes progressively harder. Finally, as $P_{\text{observe}}$ approaches 0, the required number of rounds approaches infinity.

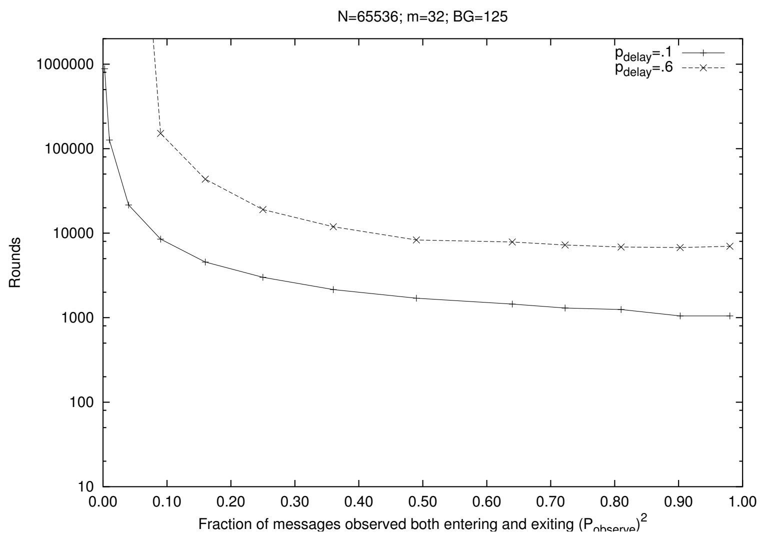

**Fig. 5.** Partial observation: Median rounds to guess all recipients

## 5 Conclusions

Our results demonstrate that long-term end-to-end intersection attacks can succeed in the presence of a variety of complicating factors. In closing, we suggest several open questions for future work, and offer recommendations for mix network designs.

**Questions for future work:** Many questions remain before the effectiveness of long-term intersection attacks can be considered a closed problem.

It would be beneficial to find closed-form equations for expected number of rounds required to mount these attacks, as Danezis does for statistical disclosure.

The attacks we have discussed here assume a purely passive adversary, but they can easily be generalized to incorporate information gained by an active attacker. Past work on avoiding blending attacks [27] has concentrated on preventing an attacker from being certain of Alice's recipients—but in fact, an active attack that only reveals slight probabilities about Alice's recipients can provide information to speed up the intersection attacks in this paper.

It seems clear that pseudonymous services will fall to intersection attacks far faster than anonymizing services. How strong is this effect, and can it be prevented? (We are currently simulating scenarios related to pseudonyms.)

Our analysis has focused on the impact of Alice's actions on Alice alone. How do Alice's actions (for example, choice of padding method) affect other users in the system? Are there incentive-compatible strategies that provide good security for all users?

There are other possible approaches to thwarting traffic analysis, including alternative padding regimes (as mentioned above in the discussion for Figure 4). These should be investigated.

Although real social networks behave more like scale-free networks than like the original disclosure attack's model, our models for user behavior still have room for improvement. For example, real users probably do not send messages with a time-invariant geometric distribution: most people's email habits are based on a 24-hour day, and a 7-day week. The effects of this variation may be significant.

Many of our simulations found "sweet spots" for settings such as mix pool delay, message volume, padding volume, and so on. Identifying those points of optimality in the wild would be of great practical help for users.

**Implications for mix network design:** If we were to design a mix network based on our findings here, what steps should we take to frustrate intersection attack?

The first lesson is this: **high variability** in message delays is essential. By 'spreading' the effects of each incoming message over several output rounds, variability in delay increases each message's anonymity set, and amplifies the effect of padding.

**Padding** seems to slow traffic analysis, especially as the volume of padding approaches the volume of the sender's actual messages, drowning out the signal. On the other hand, significant padding may be too cumbersome for most users.

Users should be educated about the effects of their chosen **message volume**: sending infrequently is safe, especially if the user doesn't repeat the same traffic pattern long enough for the attacker to identify it. Conversely, sending "almost always" is comparatively safe. But users in between appear vulnerable to intersection attacks.

Mix networks should take steps to **minimize the number of messages** that a limited attacker can see entering and exiting the network. Possible approaches include encouraging users to run their own mixes; choosing messages' entry and exit points to cross geographical and organization boundaries; and (of course) increasing the number of mixes in the network.

Much threat analysis for high-latency mix networks has aimed to provide perfect protection against an eavesdropper watching the entire network. But unless we adopt an unacceptable level of resource demands, it seems that some highly distinguishable senders will fall quickly, and many ordinary senders will fall more slowly, to long-term intersection attacks. We must stop asking whether our anonymity designs can forever defend every conceivable sender. Instead, we should attempt to quantify *how long* our designs can defend *which senders* against an adversary who sees *how much*. This paper helps move anonymity system threat analysis from inflexible security proofs to quantification of risk for given parameters of adversaries, senders, and mixes.

## Acknowledgments

Thanks go to Gerald Britton, Geoffrey Goodell, Novalis, Pete St. Onge, Peter Palfrader, Alistair Riddoch, and Mike Taylor for letting us run our simulations on their computers.

## References

1. Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Information Hiding (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, 2001.
2. Albert-Lázló Barabási and Réka Albert. Emergence of scaling in random networkds. *Science*, 286:509–512, October 1999.
3. Albert-Lázló Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 2000.
4. Krista Bennett and Christian Grothoff. GAP – practical anonymous networking. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. Springer-Verlag, LNCS 2760, March 2003.
5. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.
6. 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.
7. 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.
8. Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.
9. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1982. `<http://www.eskimo.com/~weidai/mix-net.txt>`.
10. Wei Dai. Pipenet 1.1. Usenet post, August 1996. `<http://www.eskimo.com/~weidai/pipenet.txt>` First mentioned in a post to the cypherpunks list, Feb. 1995.
11. George Danezis. *Better Anonymous Communications*. PhD thesis, University of Cambridge, December 2003.
12. George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
13. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15. IEEE CS, May 2003.

14. Claudia Díaz and Andrei Serjantov. Generalising mixes. In Roger Dingledine, editor, *Proceedings of the Privacy Enhancing Technologies workshop (PET 2003)*. Springer-Verlag, LNCS 2760, March 2003.
15. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. Manuscript, January 2004.
16. Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2357, 2002.
17. 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.
18. Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium (NDSS 96)*, pages 2–16. IEEE, February 1996.
19. Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.
20. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.
21. Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, *Financial Cryptography*. Springer-Verlag, LNCS (forthcoming), 2004.
22. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.
23. Josyula R. Rao and Pankaj Rohatgi. Can pseudonymity really guarantee privacy? In *Proceedings of the 9th USENIX Security Symposium*, pages 85–96. USENIX, August 2000.
24. J. F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.
25. Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.
26. Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.
27. Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.
28. Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *Computer Security – ESORICS 2003*. Springer-Verlag, LNCS (forthcoming), October 2003.
29. Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.
30. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Network and Distributed Security Symposium (NDSS 02)*. IEEE, February 2002.
31. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *IEEE Symposium on Security and Privacy*, pages 28–41. IEEE CS, May 2003.