From a Trickle to a Flood: Active Attacks on Several Mix Types

Andrei Serjantov[1] and Roger Dingledine[2] and Paul Syverson[3]

[1] University of Cambridge Computer Laboratory
(Andrei.Serjantov@cl.cam.ac.uk)
[2] The Free Haven Project (arma@mit.edu)
[3] Naval Research Laboratory (syverson@itd.nrl.navy.mil)

Abstract. The literature contains a variety of different mixes, some of which have been used in deployed anonymity systems. We explore their anonymity and message delay properties, and show how to mount active attacks against them by altering the traffic between the mixes. We show that if certain mixes are used, such attacks cannot destroy the anonymity of a particular message completely. We work out the cost of these attacks in terms of the number of messages the attacker must insert into the network and the time he must spend. We discuss advantages and disadvantages of these mixes and the settings in which their use is appropriate. Finally, we look at dummy traffic and SG mixes as other promising ways of protecting against the attacks, point out potential weaknesses in existing designs, and suggest improvements.

1 Introduction

Many modern anonymity systems are based on mixes. Chaum first introduced the concept in 1981 [Cha81], and since then researchers and developers have described many mix variations, eg. [Jak99,GT96,KEB98]. These have different aims and approaches, yet we still fail to understand the performance and anonymity tradeoffs between them.

In fact, some of the mixes used in well-known fielded systems such as Mixmaster [Cot94,MC00] are mentioned only briefly or not at all in the literature. We aim to start closing this gap by enumerating and exploring a variety of mix architectures. In particular, we consider the extent to which the mixes are vulnerable to active attacks such as the $n-1$ attack.

More specifically, an attacker targeting a specific message going into a mix can manipulate the batch of messages entering that mix so the only message unknown to him in the batch is the target message [Cot94,GT96]. This manipulation may involve delaying or dropping most or all other incoming messages (a *trickle* attack), or flooding the batch with attacker messages (a *flooding* attack). We call these attacks or combinations of them *blending* attacks.

We provide a rigorous analysis and comparison of several properties of each mix variant, including anonymity, latency, and resistance to blending attacks. We also give intuition and guidelines about which environments and circumstances are most suitable for each mix variant.

2 Blending Attack Taxonomy

In the past, many anonymity systems have been concerned with protecting their users against passive adversaries, either global or local, usually citing the $n-1$ or the blending attack as a vulnerability, with (quoting [BPS00]) "no general applicable method to prevent this attack". In this paper we discuss ways of reducing this vulnerability.

Here we consider a global *active* adversary who is not only able to see the traffic on all the links, but also to delay (remove) and insert arbitrarily many messages into the system in a short (constant) time. These are reasonable assumptions — methods of logging per-packet data on high bandwidth links exist, as does the possibility of building fast hardware to insert or delay messages. We further assume our attacker can send messages from many different source addresses; indeed, infrastructures to ensure sender authentication have proved difficult to build. More importantly, source authentication on links would trivially defeat the very anonymity free-route mix networks are intended to protect. The active attacker's threat comes from his power to log and manipulate messages on the links.

Note that the global active attacker can be viewed as a combination of two separate attackers: one who can only insert messages (global inserting attacker) and one who can only delay messages (global delaying attacker).

It is well known that the ability to insert or delay messages may allow the attacker to determine, for instance, the recipient of a particular message originating from a sender. We illustrate this in the case of a single threshold n mix[1]. The attack, commonly referred to as the $n-1$ attack, proceeds as follows: The attacker observes the target message leaving the sender heading towards the mix and delays it. He now starts sending messages into the mix until it fires. As soon as the mix fires, he stops all other messages from entering the mix and sends in the target message along with $n-1$ of his own messages. After the mix fires, he can recognize each of the $n-1$ messages when they leave the mix and can therefore determine the destination of the target message.

We now consider the properties of this attack before going on to examine how a similar arrangement could work on other mixes.

First of all, we note that the attack is *exact* — that is, it provides the adversary with the ability to determine the receiver of a particular message with certainty 1 (the anonymity of a message passing through the mix is 0)[2]. We also note that this attack does not depend on the rest of the mix network; that is, the attacker has enough power to always isolate and trace a particular message. We call such an attack *certain*.

We classify mixes into the following categories of vulnerability to blending attacks.

[1] The same attack can be used against a mix cascade by mounting it against the first mix, or against a mix network by repeating it ℓ (the number of mixes in the path) times.

[2] In this case, we can use either the information theoretic definition of [SD02] or the usual anonymity set definition.

– If no blending attack can reduce the anonymity of any message at all, the mix is strongly resistant to active attacks.
– If no blending attack can reduce the anonymity of any message below a constant k, then the mix has blending attack anonymity k.
– If the attacker can always reduce the anonymity of a message arbitrarily, but never to 0, the mix is vulnerable to non-exact, uncertain blending attacks.
– If the attacker can always reduce the anonymity of a message to 0, but may need to spend an arbitrary amount of resources (time/messages) to do so, the mix is vulnerable to exact, uncertain attacks.
– If the attacker is always able to reduce the anonymity of a message to 0 in a finite amount of resources, it is vulnerable to exact certain attacks.

Although it may appear that the "vulnerability" of the mixes goes up as we go down the list, this is not necessarily the case – the cost of the attack is important to consider as well. For example, suppose the anonymity of a message going through Mix 1 under active attack is proportional to the inverse of the cube of the number of messages expended in the attack. This mix can be seen as "more vulnerable" than Mix 2 which is always compromised by 10^6 attacker messages. Note that Mix 1 is vulnerable only to non-exact blending attacks, while the Mix 2 is vulnerable to exact certain attacks.

We now proceed to analyze and categorize several mixes. We suggest their blending attack cost functions (both of time and number of messages) where necessary.

3 Simple Mixes

In this section we describe various mixes divided according to their flushing algorithms. In particular we set out for each mix type: mix parameters, what the flushing algorithm is, the delay on messages in normal operations (i.e., when not under attack), the minimum anonymity provided against a passive adversary, analysis of the blending attacks on the mix, and finally a discussion of the adversaries capable of performing blending attacks. This section describes simple mixes, in which all of the messages in the mix are sent each time it fires. The more complex pool mixes are discussed in the next section.

In describing the different mix flushing algorithms and working out their properties, we make several assumptions:

– The mixes take a constant time to fire (send messages out).
– The mixes have limited physical memory and so can only contain a finite number of messages. Further, they have finite bandwidth, so can only receive a certain number of messages in a given amount of time.
– Mixes prevent message replays.[3]

[3] Actually, flooding to overflow replay caches is a closely related problem — for example, Mixmaster 2.0 [MC00] expired old entries in the replay cache when it had too many, providing a window of attack for a flooding adversary. But since [DDHM02] shows the feasibility of a free-route network that securely protects against replays until a periodic key rotation event (after which the history can be forgotten), we think our assumption is reasonable; we will ignore replays and related issues for the rest of this paper.

– Messages may or may not arrive at a uniform rate.
– In calculating the minimum anonymity we assume the global passive adversary. The vulnerability to the active attacker is described separately.
– When we talk about anonymity, we mean sender anonymity. Similar ideas can be applied to receiver anonymity as well.

3.1 Threshold Mix

Parameters: n, threshold.

Flushing Algorithm: When the mix collects n messages, it fires (delivers all n).

Message Delay: The minimum delay is ϵ (the target message arrives when there are $n-1$ messages already in the mix). The maximum delay can be infinite (the target arrives at a mix and no more ever arrive). Assuming a constant rate of arrival of messages r, we can calculate the mean delay: $\frac{n}{2r}$.

Anonymity: The minimum anonymity set is n. We assume that all the messages in the batch are from different senders (and go to different receivers).

Blending Attack Behaviour: The attack proceeds as outlined in Section 2 and is usually referred to as the $n-1$ or the flooding attack. It takes ϵ time as inserting the required messages and the (usually two) firings of the mix take a constant time. The attack takes a minimum of $n-1$ (the attacker waits until the mix is empty, forwards the target message to the mix along with $n-1$ attacker messages to flush it out) and a maximum of $2n-2$ messages (the attacker does not wait for the mix to become empty).

Adversaries: The above attack seemingly requires both the global delaying attacker capabilities and the global inserting attacker capabilities. However, this is not necessarily so. If the global inserting attacker is able to insert $n-1$ of his own messages between each of the "good" messages going into a mix, he effectively mounts an $n-1$ attack on each of them.

Another possible attack scenario is when an attacker owns a mix in a free route mix network. He has the capability to delay all the messages going through this mix until the conditions are "just right", i.e. the next mix contains $n-1$ of the attacker's messages and will fire as soon as the target message reaches it. Thus, he has the capability to attack the next mix in the route of each of the messages going through the mix he controls. (As a result, if the attacker own all the mixes but one, he is able to compromise anonymity of any message.)

3.2 Timed Mix

Parameters: t, period.

Flushing Algorithm: The mix fires (flushes all messages) every t seconds.

Message Delay: The minimum delay is ϵ, which occurs if the message arrives just before the mix is due to fire. The maximum delay is $t - \epsilon$, which is the case when the message arrives just after. The mean delay is $\frac{t}{2}$.

Anonymity: The minimum anonymity set is 0 — no messages arrive during the entire time period. The maximum anonymity set is theoretically infinite, but in practice is limited by the capacity of the mix. The mean anonymity set (assuming a rate of arrival of r msgs/s) is rt.

Note that the threshold and timed mixes are in some sense dual. If the goal of the anonymity system is to guarantee anonymity at the expense of fast message delivery, threshold mixes are good. (This is the scenario of a spy in a hostile country — if the anonymity is compromised, the spy is caught.) On the other hand, if timeliness of message delivery is crucial and anonymity is a bonus, the timed mix is ideal. Notice that if the messages are assumed to arrive at a constant rate, the properties of these mixes are exactly equivalent.

Blending Attack Behaviour: The attack is exact and certain and proceeds as follows: The adversary delays the target message for a maximum of time t until the mix fires. He then delivers the target message and blocks all other incoming messages. After another t seconds the mix fires again producing the target message on its own. This takes a maximum of $2t - \epsilon$ and a minimum of ϵ seconds (when the mix was empty and about to be fire), and 0 messages. The attack is usually referred to as the trickle attack.

Adversaries: This attack does not require any insertion of messages, so the global delaying attacker is sufficient. We also note that this "attack" can happen naturally in low traffic conditions, so a dishonest mix may be able to attack the next hop of a message simply by holding it until the next hop is empty and about to fire.

3.3 Threshold Or Timed Mix

Parameters: n, threshold; t, period.

Flushing Algorithm: The mix fires (flushes all messages) every t seconds or when n messages accumulate in the mix.

Message Delay: The maximum message delay is $t - \epsilon$, the minimum is ϵ.

Anonymity: The minimum anonymity set is 0 – no messages arrive during the entire time period. The maximum anonymity is n.

Blending Attack Behaviour: This design gives the worst case of threshold and timed mixes (and thus still exact and certain). The adversary can choose to perform a trickle attack, a flood attack, or a mixture of the two depending on whether he prefers to wait or send messages. It can be performed in a minimum of 0 messages in somewhere between 0 and $2t - \epsilon$ seconds or with a maximum of $2(n - 1)$ messages, in ϵ seconds.

Adversaries: This attack can be performed by either the global inserting or the global delaying attacker.

3.4 Threshold And Timed Mix

Parameters: n, threshold; t, period.

Flushing Algorithm: A mix fires (flushes all messages) every t seconds but only when at least n messages have accumulated in the mix.

Message Delay: The minimum delay is ϵ, and there is no maximum delay.

Anonymity: The minimum anonymity of this mix is n. The maximum anonymity is in theory infinite, but is limited in practice by the number of messages the mix can hold.

Blending Attack Behaviour: The $n-1$ attack is still exact and uses a combination of the attacks on the threshold and the timed mixes. It takes a maximum of $2t-\epsilon$ and a minimum of ϵ seconds; and a maximum of $2(n-1)$ and a minimum of $n-1$ messages.

Adversaries: It is clear that to mount an attack on this mix, the attacker has to have the capability both to delay and to insert messages.

4 Pool Mixes

Not only are all the attacks described in the previous section exact and certain, but they are also low-cost. We now examine pool mixes, which give the adversary only an uncertain attack and substantially increase the cost.

4.1 Threshold Pool Mix

Parameters: n, threshold; f, pool.

Flushing Algorithm: The mix fires when $n+f$ messages accumulate in the mix. A pool of f messages, chosen uniformly at random from all the messages, is retained in the mix. (Consider these messages as feedback into the mix.) The other n are forwarded on. Note that the threshold is the threshold of messages that must be received to fire again during ongoing operation. For pool mixes this is distinct from the 'threshold' of messages to fire when the mix is completely empty, e.g., $n+f$ for the current mix type.

Message Delay: The minimum delay is ϵ, the maximum delay is infinite — until n new messages arrive, the mix does not fire. Note, however, that even if there is a constant flow of messages and the mix is firing periodically, there is still a small but non-zero probability of the message remaining in the mix for an arbitrarily long time. The mean delay is $1+\frac{f}{n+f}$ rounds. If the messages arrive at a rate of r per second, then the average delay is $(1+\frac{f}{n+f})\frac{n}{r}$ seconds.

Anonymity: Here we have to resort to the information theoretic definition of anonymity described in [SD02] rather than the standard set-based definition, since the probabilities of the senders (receivers) sending (receiving) the message are unequal. Note also that the anonymity of the message going through a mix depends on the entire history of events that have happened in this mix. Thus, we achieve the maximum anonymity A_{max} when all of the messages that have ever passed through the mix come from different senders. Serjantov and Danezis carried out this analysis in [SD02].

$$A_{max} = -\left(1 - \frac{f}{n}\right) \log{(n+f)} + \frac{f}{n} \log f$$

The concept of minimum anonymity in pool mixes is slightly more elusive. In the threshold mix case we assumed that all the messages in the batch come from different senders. So, regardless of previous senders, the minimum anonymity of a threshold pool mix is at least n, and therefore no worse than that of a corresponding threshold mix. We could assume that all the other messages may have come from the same sender and thus provide no anonymity, but this would be overly pessimistic – the entire history of the mix is unlikely to consist of messages from just one sender.

Blending Attack Behaviour: In general, the blending attack has two phases: flushing the mix so that no good messages remain inside it, then forwarding in the target message and flushing it out onto the network. With simple mixes, one flush was sufficient for each phase. With pool mixes, this is no longer the case. Furthermore, there is now a small but non-zero probability that a given message (e.g. the target message) remains in the mix for an arbitrary length of time. Intuitively, the attack ceases to be certain. It proceeds as follows:

The attacker delays the target message, fills up the pool mix and flushes it. If j of the attacker messages don't come out, he knows that $f - j$ good messages remain inside the pool mix. He now delays all the other incoming good messages and tries to flush the remaining good messages out of the mix (he can distinguish them as they come out). Of course, this takes more messages than to flush out a threshold mix (see below for details), but the attack is still exact (if/when the attacker succeeds in getting all the messages to leave the mix, he knows it). When all good messages have been flushed, he just sends in the target message, and flushes the mix as before until the target message comes out. Because the attacker is not guaranteed to flush out the mix completely or to flush out the target message, the attack is uncertain.

Analysis: Flushing out the mix: after r rounds, the probability that a particular message that was in the mix at round 0 is still in the mix by round r is

$$\left(\frac{f}{n+f}\right)^r$$

If there were N good messages in the mix initially, the expected number left after r flushes is N times the above.

Alternately, we might consider the chance that all N good messages have left the mix by round r:

$$\left(1 - \left(\frac{f}{n+f}\right)^r\right)^N$$

The number of good messages in the mix at the beginning of the attack can vary between f and $n+f-1$.

Example 1 *Consider a pool mix with a threshold of 100 and a pool of 60 messages. The average delay of a message through this mix is 1.6 rounds. Assuming the attacker targets a mix with 60 messages, the expected number of good messages remaining in the mix falls below 1 after 5 rounds or 500 attacker messages. Similarly, the chance that all 60 messages have left the pool goes above 50% after 5 rounds, and reaches 99% after 9 rounds.*

Thus, the use of a pool mix in an anonymity system not only makes the $n-1$ attack uncertain, but also more expensive in terms of attacker messages at the cost of increasing the average delay of a message passing through the mix.

4.2 Timed Pool Mix

Parameters: t, period; f, pool; (0, threshold).

Flushing Algorithm: The mix fires every t seconds. A pool of f messages chosen uniformly at random is retained in the mix. The others are forwarded on. If f or fewer messages are in the mix, it does not fire. (Thus, strictly speaking, this is a threshold and timed pool mix, for which the threshold is 0.)

Message Delay: The minimum message delay is ϵ, and there is no maximum delay (if no messages arrive, the messages that are in the pool will never leave the mix). Like in the case of the threshold pool mix, there is again a small but non-zero probability that any given message could remain in the mix for an arbitrarily long time even if there are messages flowing through the mix.

Anonymity: The timed nature of this mix allows an arbitrarily large number (only limited by the memory and bandwidth capacity of the mix) of messages to be mixed together. If we assume a constant rate r of message arrival, the anonymity provided by this mix can be calculated in just the same way as for the threshold case; but we leave this calculation for future work.

Alternatively, the anonymity of a message going through this mix can be calculated from the history of the mix — a record of the operation of the mix ever since it was started. Like in the threshold case, a mix's history includes the senders of each message and also the number of messages that got mixed together in each batch (and, potentially, a whole host of other features). Of course, in practice, a record of only the last few rounds gives a good approximation.

The minimum anonymity of a timed pool mix is clearly smaller than that of a threshold pool mix (unless the threshold is 1). If the pool is small relative to the batch size, then the bulk of the anonymity comes from mixing the target message with the batch of incoming messages, not from mixing it with the messages in the pool. Because the timed mix does not always have this batch of messages for mixing the target message, its minimum anonymity should be considered to be very much worse than that of the threshold pool mix unless a large pool is maintained. As was shown in [SD02], the anonymity contribution of pool messages is quantifiably greater than those of batch messages. So a larger pool is always preferable from this perspective. Of course this would have an impact on message delay, reducing one of the potential advantages of having a timed mix.

Blending Attack Behaviour: Two flavours of blending attack are possible on this mix. The adversary has no control over when the mix fires, but he can choose to add many or just a few messages to each batch. (He prevents all other messages from reaching the mix).

By adding as many messages as possible in one round, he maximizes the probability that after one flush only the attacker messages will be left in the pool. He knows that he has succeeded if f of his messages do not come out with the flush. This approach is very inefficient in terms of the number of messages added by the attacker (b) since the probability of flushing the mix by this method is $\frac{f^2}{b+f}$ (supposing there were f good messages in the mix initially).[4] However, this aims to flush the good messages out of the mix in one round, and therefore a maximum of t seconds. Thus the entire attack can be executed in less than $2t$ seconds with an arbitrarily high probability of success.

Alternatively, the attacker can add just one message to each round for many rounds. This is very efficient in the number of messages b (indeed, this attack is much more efficient than the one on the threshold pool mix in terms of messages, but clearly not in terms of time). The probability of flushing the mix is $f\left(\frac{f}{f+1}\right)^b$, but this approach delays all the messages to the mix by tb seconds, which is highly observable to the users of the anonymity system.

The attacker can choose either or a combination of these approaches, based on whether he can send out many messages or delay messages to a mix for a long time.

4.3 Timed Dynamic-Pool Mix (Cottrell Mix)

Parameters: t, period; f_{min} minimum pool; *frac*, fraction of messages to be sent; (n, threshold)

[4] As previously mentioned, in practice there is an upper limit on b due to the finite memory capacity and/or bandwidth of the mix.

Flushing Algorithm: The mix fires every t seconds, provided there are $n + f_{min}$ messages in the mix; however, instead of sending n messages (as in a timed-and-threshold constant-pool mix), the mix sends the greater of 1 and $\lfloor m * \mathit{frac} \rfloor$ messages, and retains the rest in the pool, where $m + f_{min}$ is the number of messages in the mix ($m \geq n$). If $n = 1$, this is the mix that has been used in the Mixmaster remailer system for years. We use the term 'Cottrell mix' for $n = 1$ and 'timed dynamic-pool mix' for the more general case when n might be greater than one.

We have called constant-pool mixes simply 'pool mix' up to now for consistency with the previous literature. Hereafter, 'pool mix' will refer to either constant or dynamic pool mixes.

When messages arrive at a constant rate of 1 per period, Cottrell mixes are equivalent to both timed mixes and threshold-1 constant-pool mixes. Specifically, if the rate r of message arrival is $1/t$, the mix will forward 1 message in every period and retain f_{min} in the pool. For a general Cottrell mix, if the messages arrive at a constant rate of $n * \mathit{frac}/t$ and $\lceil n * \mathit{frac} \rceil = n * \mathit{frac}$, then this is equivalent to a constant-pool mix with threshold of $f_{min} + n(1 - \mathit{frac})$.

Message Delay: Like the other pool mixes, the minimum delay is ϵ, and there is no upper limit on the delay. The mean delay depends on the future rate of arrival of messages into the mix; it is at least as high as that of a timed constant-pool mix and typically higher.

Anonymity: The dynamic-pool mix has identical minimum and maximum anonymity properties compared to a timed constant-pool mix. We could similarly use a log of the mix's activity to calculate the anonymity of a message passing through it (although the calculation would be slightly different). Also, we note that the anonymity provided by the mix would be higher than that provided by either a timed or threshold constant-pool mix: As the number of messages in the mix goes up, *frac* keeps the chance of the message remaining in the mix constant, whereas it decreases in the case of the timed constant-pool mix. While the chance of a message remaining in a threshold constant-pool mix is also constant for each flush, increased message rate means more frequent flushing, thus reducing the chance of a message remaining in the constant-pool threshold mix per unit time.

Blending Attack Behaviour: The introduction of the dynamic parameter *frac* has several new consequences compared to the timed constant-pool mix.

Firstly, the maximum probability of flushing the mix of good messages in one flush is *frac*. Therefore there is no possibility of flushing the mix with high probability in one flush: the first of the two blending attacks on timed constant-pool mixes is blocked. As already noted, it is similarly more resistant to flooding than a constant-pool threshold mix.

Secondly, the attacker has to find out how many messages are in the mix. Of course, the number of good messages in the mix is easy to calculate from the number of messages that come out.

Finally, the number of messages inside the pool mix may be arbitrarily high and cannot be reduced to below f_{min} in one round. Therefore, if we wish to send a message that is harder to track, we should send it at a time of higher traffic — thereby increasing the cost (in terms of messages or time) of attempted attacks on it.[5]

Thus timed dynamic-pool mixes require the attacker to delay all the traffic to the mix for a substantial number of rounds, and therefore for a substantial time.

Example 2 *A Cottrell mix with a pool of 60 messages and fraction of $\frac{6}{10}$ requires 5 rounds for the expected number of good messages remaining in the mix to fall below 1. If the period is 5 minutes, the overall flushing time is 25 mins. 500 messages will be used in the first part of the attack.*

This example shows that the protection this mix provides against blending attacks is still weak. Indeed, the probability of a successful attack is proportional to $\frac{1}{frac^k}$ where k is the number of rounds. Ideally we would like a mix for which the anonymity does not go to 0 even as the cost of the attack goes up.

We would also be happier with a mix where the probability of a successful attack is proportional to $\frac{1}{P(k)}$ where P is a polynomial of a small degree.

We note that the minimum anonymity of the Cottrell mix is still very much worse than that of the threshold constant-pool mix (where the threshold size is much larger than the pool size). Therefore, we propose a larger threshold for the timed dynamic-pool mix, to improve its minimum anonymity properties. We believe this modification will not have adverse effects on the blending attack properties, although will slightly decrease the efficiency of this mix.

5 More on Resisting Blending Attacks

5.1 Resisting Exact Attacks

All of the mixes presented above are vulnerable to exact attacks. In other words, the attacker is always able to find out if his attack is successful, and possibly expend more resources to try again if not.

However, we could do better. If we make it hard to determine the number of good messages in the mix, it will be harder for the attacker to flush the mix as he will not know how many messages he needs to get rid of. This is easily possible by choosing the number of messages to flush from a binomial probability distribution (flipping a biased coin once for each message to decide whether it is to be flushed or not). The weight of the coin is chosen by an appropriate function of the number of messages in the mix. Thus, it would take the attacker several rounds to establish with any confidence the number of good messages in the mix initially, and he can never be exactly sure of that figure (unless he has the whole history).

[5] Unfortunately, an attacker capable of arbitrary message insertions, as we have been assuming, will make it hard to determine times of higher legitimate traffic.

While introducing randomness appears to be a promising direction, note that it would not increase the cost of the attack significantly (if at all). Thus, we shall not pursue it further.

5.2 Resisting Blending vs Verification

The tools that we give honest mixes to protect against these blending attacks are precisely the tools that dishonest mixes use to launch them. *Uncertainty* comes from not letting observers know when a given message might leave the mix; *inexactness* comes from not letting observers know how many messages are currently inside the mix. This flexibility allows a mix to prevent the adversary from learning the details of message flow, but at the same time it allows a compromised mix to manipulate message flow, for example by delaying a message until it's ready to launch a trickle attack on the downstream hop.

Many basic mix verification schemes, such as that proposed in [Cha81] where senders observe the batches exiting a mix to confirm it processed their messages, require schemes where the messages come out with the next mix flush. More complex robustness schemes [Jak99] enforce timely message processing if the adversary does not own a threshold of participating mixes. The receipt and witness scheme of [DFHM01] works because a mix loses reputation if it hasn't obtained a receipt from the next hop within the allowed timeframe. All of these systems are designed to ensure that messages will leave each mix predictably. Yet more verification schemes that work in a system with pool mixes are described in [Jer00,FGJP98]. There the authors propose a commitment scheme which ensures that the mix commits to a decision before he receives the target message, and his actions are verifiable by the sender of the target message.

In some topologies, such as free route mix networks, a widespread delaying attack is difficult to perform because the adversary must control all mixes that are sending messages to the target mix. In this case we may gain more by allowing mixes to protect against blending attacks than we lose by permitting compromised nodes to manipulate messages. At the other extreme, in a cascade topology, a single compromised mix has absolute control over the downstream message batch. In this case verification of correct processing is critical, to prevent wholescale (unobservable!) anonymity compromise. Methods to detect an attack afterwards, e.g. by checking for successful delivery of test messages at the end of the cascade [DS02], can help detect misbehaving mixes — but this technique only works for cascade topologies, and it assumes that the cascade head cannot distinguish test messages. As we design new protocols and batching approaches, we must consider this tradeoff between protection from blending attacks and verifiability.

5.3 Cover Traffic

Another possible protection against the blending attacks is cover traffic. Indeed, in designing the Mixmaster system [Cot94], Cottrell realized that even the timed dynamic-pool mixes still leave the system vulnerable to attacks. He decided to introduce cover traffic to improve attack resistance.

At the moment, Mixmaster has the following cover traffic policy: at each flush one dummy is put out onto the network. The dummy message generated by the mix looks like a normal message, but has a constant length of 5 hops and ends at a mix rather than at a receiver. Note that this dummy policy still allows an attacker to flush the mix free of good messages and be certain about it. However, once the good message is inserted into the mix, at every round at least one good message comes out. Naturally, when two messages come out, the attacker knows that the target message was one of those, but he does not know which one. If the attacker finishes here, he has reduced the anonymity of the message to 1 (that is, two possible receivers). However, he can do better. Because Mixmaster is a mix network, he can keep tracking both messages (and all the messages that result when these pass through more mixes) until he can detect that one message has gone to a recipient. That is the target message since all the dummy messages end in a mix. This attack is exact, but much more expensive than anything we have described previously. We can make the adversary's job harder by making the mix choose its route length for the dummies from a uniform distribution rather than always choosing 5. We can get further protection by allowing mixes to send cover traffic to the users. However, providing complete protection with this approach is very hard. Each mix must know all the users in the system: if a mix only delivers dummies to a subset of the users, an adversary can distinguish with better than even probability between a dummy and a legitimate message.

Note that constant rate dummy policies do not affect the blending attack properties provided by the mixes themselves. Constant dummies simply magnify the scale of the attack without changing the properties. For example, even with the Mixmaster cover traffic policy, a threshold mix network can be attacked in ϵ time (but a very large number of messages).

5.4 Making Messages Unrecognizable

Link encryption offers some limited protection against traffic analysis. Encrypted links between honest nodes prevent an adversary from recognizing even his own messages, thus making it harder to flood mixes and watch where outgoing messages go. However, since the adversary performing the flooding attack can choose the next destination of each message, he may still be able to recognize his messages. At the extreme, he can direct all his chaff messages at the same next mix, so any other message will still stand out. A good link padding scheme may be able to further frustrate him, but much more research remains.

Babel [GT96] introduces *inter-mix detours*, a scheme where mix nodes can choose to rewrap a message and send it through a few randomly chosen new hops — so even the sender cannot be sure of recognizing his message as it leaves the mix. This approach could help complicate blending attacks, but also introduces reliability risks; we leave its analysis for future work.

6 Limitations of Stop-and-Go Mixes

While active attacks have been widely cited in the literature and methods of protection against them have been suggested informally, most of these have not been rigorously analysed or evaluated [Cot94,GT96,Jer00,KEB98].

We now concentrate our attention on one particular proposal — Stop-and-Go Mixes [KEB98]. The authors outline several techniques that could be used to protect mix systems against active attacks like the ones considered in this work.

The first is a scheme for an anonymity system based on a mix cascade that requires authentication. This is based on a stronger assumption — that the attacker cannot mount a distributed $n-1$ attack where the hostile messages come from different users. Unfortunately, it seems very hard to reuse this idea in free route networks, since there is no centralized input location at which we can do authentication; compromised mixes could claim to have done authentication on traffic they instead generate themselves.

The second scheme, Stop-and-Go mixes (SG mixes), involves the sender estimating a time window during which the message is allowed to arrive at each of the mixes making up the route of the message. If the message arrives during that time period it will be forwarded on; otherwise it will be discarded. Therefore the attacker's ability to delay messages is much more limited, so active attacks are harder.

The attack is uncertain, and the authors argue that the probability of executing the attack successfully is very low.

However, the scheme relies on the users being able to accurately calculate a security parameter that depends on their estimate of the rate of arrival of messages to the mixes that the target message will go through *at the time it will travel through them.* In other words, the users need to be able to predict the traffic levels in the system in the future to remain anonymous. This weakness is likely to be exploited by the attacker (the exact details depend, of course, on how the security parameter is calculated). Further, an attacker able to insert arbitrary messages into the system will still be able to arbitrarily affect the input batch with which a target message enters any mix, regardless of the timing requirements of that message. (It is conceivable that combining SG mixes with reputation systems as in [DFHM01] or [DS02] might help.)

Thus, we defer the evaluation of SG mixes to future work as the precise details of parts of the protocol crucial to the security of the system have not yet been worked out.

7 Conclusion

We present a set of mixes and examine their anonymity, message delay and blending attack properties. In particular, we suggest that it is useful to partition the mixes into certain categories of vulnerability but emphasize that the cost and worst case attack scenario are important qualities to consider.

Simple timed mixes seem best for anonymity infrastructures that require low latency. On the other hand, if we do not need guaranteed low latency, adding a pool to the mix can significantly improve anonymity. Allowing the pool to process a fraction of waiting messages (above a certain threshold) each round further improves robustness against flooding attacks designed to flush out a target message.

Although we show the mixes to be rather vulnerable to active attacks, some avenues still have hope. The first of these is verification schemes. We also touch on cover traffic, a more widely used solution. We assess the cover traffic policy used in Mixmaster, point out weaknesses, and discuss some approaches to strengthening its dummy policy.

The paper can also be treated as a tutorial on the different styles of mixes and as a recommendation to the Mixmaster implementors to alter the cover traffic policy and introduce thresholds into the mixes.

		Delay			Anonymity		
		Avg.	Min.	Max.	Avg.	Min.	Max
Simple	Threshold	$\frac{n}{2r}$	ϵ	∞	n	n	n
	Timed	$\frac{t}{2}$	ϵ	$t-\epsilon$	rt	0	mix capacity
	Thresh. or Time		ϵ	$t-\epsilon$		0	n
	Thresh. & Time		ϵ	∞		n	mix capacity
Constant Pool	Threshold	$(1+\frac{f}{n+f})\frac{n}{r}$	ϵ	∞		$\geq n$	$-\left(1-\frac{f}{n}\right)\log(n+f)+\frac{f}{n}\log f$
	Timed		ϵ	∞		≥ 1	total # of senders
Dynamic Pool	Cottrell		ϵ	∞		≥ 1	total # of senders
	Thresh. & Time		ϵ	∞		$\geq n$	total # of senders

References

[BPS00] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on the Design Issues in Anonymity and Observability*, pages 10–29. 2000.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cot94] L. Cottrell. Mixmaster and remailer attacks, 1994. `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[DDHM02] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. Manuscript, 2002. `http://mixminion.net/`.

[DFHM01] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A Reputation System to Increase MIX-net Reliability. In I. Moskowitz, editor, *Information Hiding, 4th International Workshop (IH 2001)*, pages 126–141.

Springer-Verlag, LNCS 2137, 2001.
`http://www.freehaven.net/papers.html`.

[DS02] R. Dingledine and P. Syverson. Reliable MIX Cascade Networks through Reputation. In M. Blaze, editor, *Financial Cryptography (FC '02)*. Springer Verlag, LNCS (forthocoming), 2002.
`http://www.freehaven.net/papers.html`.

[FGJP98] E. Franz, A. Graubner, A. Jerichow, and A. Pfitzmann. Comparison of Commitment Schemes Used in Mix-Mediated Anonymous Communication for Preventing Pool-Mode Attacks. In C. Boyd and E. Dawson, editors, *3rd Australasian Conference on Information Security and Privacy (ACISP'98*, number 1438 in LNCS. Springer-Verlag, 1998.

[GT96] C. Gülcü and G. Tsudik. Mixing Email with *Babel*. In *Internet Society Symposium on Network and Distributed Sytem Security (NDSS'96)*, pages 2–16. San Diego, CA, 1996.

[Jak99] M. Jakobsson. Flash Mixing. In *Principles of Distributed Computing - PODC '99*. ACM, 1999.
`http://citeseer.nj.nec.com/jakobsson99flash.html`.

[Jer00] A. Jerichow. *Generalisation and Security Improvement of Mix-mediated Anonymous Communication*. Ph.D. thesis, Technischen Universitat Dresden, 2000.

[KEB98] D. Kesdogan, J. Egner, and R. Buschkes. Stop-and-go-MIXes providing probabilistic anonymity in an open system. In *Proceedings of the International Information Hiding Workshop*. 1998.

[MC00] U. Möller and L. Cottrell. Mixmaster Protocol — Version 2. Unfinished draft, 2000. `http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt`.

[SD02] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In P. Syverson and R. Dingledine, editors, *Privacy Enhancing Technologies*, LNCS. San Francisco, CA, 2002.
`http://petworkshop.org/2002/program.html`.