Blending different latency traffic with alpha-mixing

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

[1] The Free Haven Project (arma@freehaven.net)
[2] The Free Haven Project (aas23@freehaven.net)
[3] Naval Research Laboratory (syverson@itd.nrl.navy.mil)

Abstract. Currently fielded anonymous communication systems either introduce too much delay and thus have few users and little security, or have many users but too little delay to provide protection against large attackers. By combining the user bases into the same network, and ensuring that all traffic is mixed together, we hope to lower delay and improve anonymity for both sets of users.
Alpha-mixing is an approach that can be added to traditional batching strategies to let senders specify for each message whether they prefer security or speed. Here we describe how to add alpha-mixing to various mix designs, and show that mix networks with this feature can provide increased anonymity for all senders in the network. Along the way we encounter subtle issues to do with the attacker's knowledge of the security parameters of the users.

1 Introduction

Anonymous communication systems today don't provide much protection against a large attacker. Tor [11] and JAP [3] have hundreds of thousands of concurrent users, but their low latency and low overhead mean they do not defend against an adversary who observes most of the network. At the other end of the spectrum, Mixminion's design [5] theoretically provides strong security against a global attacker by adding high variance in latency, but this latency has crippled adoption — which in turn decreases the security that the network can provide, discouraging even the users who need high security [2, 10].

Here we design a hybrid mix batching strategy that combines users with different anonymity and performance goals into the same network.

In our scheme each sender communicates an α – a security parameter – to each mix along the route of her message. The time the message spends inside each mix (and hence the anonymity it accumulates) then depends on the size of this security parameter. The message's α value at each mix decrements based on certain events, and when it reaches zero it is reintegrated back into the mix network. Our scheme can be combined with any of the standard mix types such as timed mixes, pool mixes, etc. [14] to give each sender more control over the anonymity/performance tradeoff of her message.

Users that desire better anonymity then have the opportunity to obtain it by increasing α for their messages. More importantly, there is a network effect: when the attacker knows little about the security parameters chosen by individual users, all senders will benefit because of the mere *possibility* that they chose a higher α.

In this paper we start by outlining some simple alpha-mix designs and analysing the anonymity properties they can provide to users with different security preferences. Next we look at the strategies users should follow when picking the security parameter for each mix in the message's path. In Section 5, we look at the incentives users have for choosing a high security parameter themselves rather than expecting others to take the latency penalty (and thus provide more anonymity to everyone). Lastly we consider more sophisticated alpha-mixing strategies which should provide better properties but are hard to analyse.

2 Deterministic-alpha mix

While *threshold mixes* fire only when a sufficient number of messages have arrived, *timed mixes* simply fire at regular intervals. Timed mixes may be appropriate for traffic for which timeliness matters, since with threshold mixes the time until the next firing is unpredictable without assumptions about the rate of incoming messages. On the other hand, threshold mixes can provide minimum anonymity properties.

"Deterministic" here refers to the algorithm by which messages change α after each mix firing. Later we will consider algorithms that will change alpha probabilistically, for example based on the number of messages with certain alpha values in the mix. In this section, all messages simply drop one alpha level after each mix firing.

Timed deterministic-alpha mix: The mix fires every t seconds. All messages for which $\alpha = 0$ are sent out.[1] All remaining messages have their α decremented by one. New messages that arrive before the next firing are buffered based on their initial α and are placed at the according α level.

Threshold deterministic-alpha mix: This is the same as the timed version, except that the mix fires when at least a threshold n of messages with $\alpha = 0$ are in the mix. Note that since the number of messages with $\alpha = 1$ may be above the firing threshold, some batches may include more than n messages. (When many messages with $\alpha > 0$ are waiting in a mix before a threshold number of $\alpha = 0$ accumulate, this is analogous to the situation where many mixes in a free-route threshold-mix net are waiting and nearly full while messages are being accumulated at relatively empty mixes.)

[1] We do not describe the reordering of messages or changing of their appearance in this paper. We assume that messages emerging from a mix have an appearance that cannot be correlated with their appearance upon entering the mix and that the order of all $\alpha = 0$ messages is randomly permuted before they are sent.

As we will see, one of the virtues of alpha mixing is that the timed/threshold distinction for mixes can blur, and it becomes more a distinction for firing strategies of individual messages than of mixes. For our initial analysis we will assume a steady-state network with constant rate of incoming messages, which means against a passive adversary the anonymity properties are equivalent.

It is also possible to have a threshold-or-timed alpha mix in which all messages are decremented in the alpha stack if either t seconds have passed or n messages have arrived. Similarly, one can have a threshold-and-timed alpha mix to reduce the effective rate of flooding attacks [14]. Even more complex variants of these designs are discussed in Section 6.

2.1 Deterministic-alpha mix: anonymity against a local passive adversary

Here we describe the anonymity for a threshold alpha mix during steady-state (i.e., messages arrive with various alphas at a regular rate, and the mix fires at regular intervals).

We assume the adversary does not know the specific alpha of any message entering the mix, e.g., that this is provided to the mix encrypted together with the message. However, we do allow that the adversary might know the strategy by which alpha was chosen; we examine this issue further in Section 2.2. What should that strategy be? It would seem that choosing higher alphas would correspond to greater anonymity for messages. We now make this more precise.

Claim. Given any set of other messages in a threshold deterministic-alpha mix, a message has greater anonymity if it is assigned an alpha from a broader range (chosen uniformly) than from a narrower range.

Proof. Suppose messages occur with some distribution of alphas in a mix with firing threshold n. A sender will assign to message M an initial α_M for a particular mix in a given position in the message's path. Suppose the adversary knows the strategy chosen by the sender. Assume the choice of strategies are between choosing α_M from either a range of 0 to j or a range of 0 to $k > j$. The anonymity set size increases by $n(k - j)$ if the broader range is chosen. (In information-theoretic terms, the entropy has increased by $\log(n(k - j)$.) If the adversary does not know the strategy, then we cannot put a precise number on his uncertainty. However, the less predictable the range is to the adversary, the greater the uncertainty is, even if we cannot say how much. She can either guess too small a range and risk not seeing the output message at all, or guess too large and include many additional batches in the anonymity set for the message. (These points carry over mutatis mutandis when we reason probabilistically rather than just possibilistically.)

If the adversary does know the strategy (although still not the actual α) for each incoming message, then the anonymity of M is less affected by the strategy that other messages use for choosing α in a steady-state network. However, if

the strategies are not known, then choosing α from a broader range increases the anonymity for other messages in the mix as well, although it is difficult to say by how much. If the distribution of strategies across all messages in the mix at any time is known to the adversary, however, then it is clear that increasing the range from which α is chosen for any unknown one of those messages increases the uncertainty about the future batch in which any of the messages still in the mix will emerge. Thus,

Claim. Assume a set of messages in a steady-state deterministic-alpha mix. Assume the α_M for any message M is chosen uniformly at random from the range given by $0 \leq \alpha_M \leq k_M$. Then anonymity increases for every message M in the mix if any $k_{M'}$ increases.

The key is not that a high α necessarily provides better security, but rather that when the variance of our α is high, its value within the range is hard for the attacker to predict.

In summary, for threshold mixes or steady-state timed mixes, choosing α from a broader range improves the anonymity for that message whether the adversary knows one's strategy or not. Further, if the adversary knows nothing about the strategies of choosing alphas or knows simply the distribution of strategies, then increasing the α-range for any message improves anonymity for all messages.

2.2 Attacker knowledge

In the previous section we noted that the anonymity properties provided by alpha mixes depend on what the attacker knows about the security parameters of the users. Specifically, while choosing from a wider range of alphas improves anonymity, an attacker can reduce anonymity if he has information about which alphas are chosen. We illustrate this on a simple example.

Consider sender anonymity in the setting of just one mix, illustrated on two rounds only (equivalently, suppose maximum alpha is 1):

Round 1: $I_1 = i_{1,1} \dots i_{m,1}$ entered the mix, messages $o_{1,1} \dots o_{x,1}$ came out.

Round 2: $I_2 = i_{1,2} \dots i_{n,2}$ entered, messages $o_{1,2} \dots o_{y,2}$ came out.

Let $\alpha(x)$ be the set of possible alphas of message x as known by the attacker. Note that if the attacker knows nothing, then $\forall x,\ \alpha(x) = \{0, 1\}$.

Our target message is $o_{1,2}$. The sender anonymity set (in messages) is:

$$\{x | x \in I_1 \wedge 1 \in \alpha(x)\} \cup \{y | y \in I_2 \wedge 0 \in \alpha(y)\}$$

Hence (almost) any knowledge of alphas by the attacker degrades anonymity. Note that complete knowledge of alphas by the attacker *may* leave the message with no anonymity; however, this is extremely unlikely (or amounts to a rather expensive variant of the trickle attack).

Indeed, when analysing alpha mixes we need not constrain ourselves to reasoning about anonymity sets. We now compute the anonymity probability distribution, but first we need a little more formalization of the assumptions. Essentially, where we allowed the attacker possibilistic knowledge about the alphas of the messages, we now allow him (better) probabilistic knowledge.

Notation: call α_M the alpha in message M. Hence the attacker knows the probability distributions $P(\alpha_M = a)$ for every message M with a ranging from 0 to a_{max}.

Now, the anonymity probability distribution:

$$\text{Normalise}(\{p|M \in I_1 \wedge p = P(\alpha_M = 0)\} \cup \{p|M \in I_2 \wedge p = P(\alpha_M = 1)\})$$

and the anonymity is the entropy of this distribution. Clearly, the more the attacker knows about alpha, the lower the anonymity.

2.3 Correlating message content with requested security

Now let us study an interesting example which has long been known intuitively... Suppose the attacker knows that sender S only sends with a high security parameter (let's say alpha of 5). He now sees a message from sender S at round 0, and a message detailing Enron's finances emerges at round 5. Suppose further that all other messages have an alpha of 0. Our above definitions give the target message the anonymity set of all the senders of round 5 union S. Nevertheless, we conjecture the attacker will tend to suspect that S sent the message. How can we reconcile the intuition of the attacker with our formalism above and how can we design the system to avoid such a judgement?

The attacker is likely to be correct — what we ignore here is the fact that the choice of the security parameter is likely *conditional* on the importance of the message and the attacker has used this fact to form his judgement. In order to avoid this, we must (paradoxically!) ignore this fact completely and pick alphas from a distribution which is independent of the receiver and the message's content. Of course, we cannot defeat this attack entirely because the sender's distribution will still be conditional on her utility function: messages from users with higher security needs will in fact still behave differently.

There are still external factors to consider. We'd like to go a step further and make the sender's software enforce that she doesn't vary alpha based on each message's receiver or content. This approach would best convince the attacker that the sender *could not* have changed it. Also, if a given user is the only sender with extremely high alpha values, then intersection attacks over time (watching the high-value messages and what senders were active before each) will reveal her [4, 13]. But we will ignore these black-box network attacks since they are not the focus of this paper.

Below we will see that some strategies for choosing the alpha values are more effective than others at preventing the attacker from learning the security preferences of senders.

3 Allocating $\sum \alpha$ against a distributed adversary

In the previous section we discussed the fact that an adversary who can learn about the sender's alphas can weaken her anonymity. For example, sending only

high value messages and picking high security parameters for them can actually decrease anonymity.

In this section we examine an attack that a compromised mix can perform to deduce the sender's alphas, and we deal with the problem of allocating the overall message's security parameter $\Sigma\alpha$ over the mixes in the message's path. There are two problems to solve. Firstly, if a bad mix observes one of the alphas, it should get as little information as possible about the other alphas of this message.[2] Secondly, it should be hard for the bad mixes to link any alpha parameter to a particular sender, i.e. figure out how much any sender is concerned about security.

One possible solution for picking a sequence of $\alpha^{(i)}$ (where the "(i)" represents the i^{th} mix in the route) is simply to pick from a uniform distribution over the partitions of $\Sigma\alpha$ into ℓ buckets where the buckets themselves are indistinguishable. The number of such partitions are given by

$$\sum_{k=1}^{\ell} Q(\Sigma\alpha, k)$$

where Q denotes the number of ways of partitioning $\Sigma\alpha$ into exactly k distinct parts. Generating values from such a distribution is possible, for instance, using the algorithm described in [7]. This seems to deal with the first problem (the analysis to show this is beyond the scope of this paper). For the second part, it depends what the sender wants to protect: does she care about having an estimate of the security parameter associated with just herself, with herself and the recipient, or just the recipient? Note that if the first and the last mixes are bad and can observe a "higher security" message passing through each of them, they can conjecture that it is one of a relatively small set of sensitive messages. There are a variety of properties to explore in this area; we merely observe that by reordering the value that we obtain from the uniform distribution over partitions, we can make sure that the minimum values in that partition are sent to the first and the last mix. For example, if $\Sigma\alpha = 5$, then the distribution is uniform over: $\{5,0,0,0\}, \{4,1,0,0\}, \{3,2,0,0\}, \{3,1,1,0\}, \{2,1,1,1\}$. Supposing we draw the partition $\{3,1,1,0\}$, we reorder it into $\{0,3,1,1\}$ and hence obtain a sequence of alphas to insert into the message.

If we wish to guarantee that neither the first nor the last mix can locally know anything about the sensitivity level of a message, we can simply stipulate for message M that $\alpha_M^{(0)} = \alpha_M^{(n)} = 0$ (for a path length of $n+1$). Similarly we could stipulate that $\alpha_M^{(1)} = \alpha_M^{(n-1)} \leq 1$, etc. The tradeoff is that with each such move we are reducing what an adversary observing just the endpoints can learn about sensitivity of messages, but a more concentrated set of nodes in the center learn more about the sensitivity of messages. Against an adversary who controls the central node(s) combined with, e.g., a global passive observer, our protection is diminished. We can gain advantage against both types of adversaries by

[2] Note the similarity between picking an alpha and message splitting [15] — in both cases they are distributions over partitions.

increasing path length, with the usual concomitant risk to robustness of delivery that comes with increased path length.

4 Dummies

Our focus so far has been on steady-state networks with passive adversaries. However, we want to provide uncertainty even in edge cases where there is a momentarily lull in traffic [8, 9, 14]. An active attacker can arrange an edge case via blending attacks, but a passive attacker can also simply wait for an edge case to occur. For timed mixes there will be occasions when only a single message enters and leaves the mix in a given round. Alpha mixes have a clear advantage here since there is no guarantee that the message that exited the mix is the same message that entered. The attack is never exact (guaranteed to recognize a target message as it exits the mix) unless the adversary can bound the range of α with certainty for all messages he observes.

We provide a very lightweight dummy policy that guarantees that no exact attack is possible against an alpha mix, even for active attackers: simply initialize the mix with a single dummy message set at an arbitrary alpha. Before firing, always check the mix for the presence of a dummy somewhere in the alpha-stack. If none is present, add one.

But what do we mean by "arbitrary alpha"? Obviously it must occur within some finite range. It could be uniformly chosen between 0 and the maximum expected α. If a message is ever received with a higher α, then the maximum should be raised to this level. Such a strategy will prevent any exact attack, but it will still allow most practical attacks that dummies are intended to counter (active or passive) because most traffic will not have high alpha. Thus, a single message entering and a single message exiting a timed mix in a single firing interval are much more likely to be the same message than a dummy.

A strategy that should maximize uncertainty at least in the edge cases would be to insert dummies according to the expected distribution of α_M for messages M entering the mix. The expected distribution can be determined by observation. Mixes joining the network can be initialized with a default expected distribution averaged from one or more mixes already in the network. If the network is uninitialized, individual mixes can be initialized with a uniform strategy (as above), or better a geometric one, e.g., add a dummy at level α with probability $2^{-(\alpha+1)}$. Dummy policy can then be periodically shifted to reflect the distribution of alphas for actual traffic through the mix. More research remains here to make this dummy approach resistant to an adversary who sends lots of messages with non-standard alphas into a particular mix to influence its view of a typical value for alpha.

If active attacks are suspected, the amount of dummy traffic added to the alpha stack can be increased according to the expected duration of and strength of the blocking (assuming timed deterministic-alpha mixes, for which there is no point in flooding) and the anonymity one intends to maintain for messages so attacked.

The easiest way to disguise dummies from others in the network is to route them in a circuit leading back to the mix that generates them [6]. The length of the path should be randomly chosen as suggested in [14]. Obviously the alphas chosen for the dummy message at other mixes in the path should be distributed to minimize recognition of the message as a dummy; hence some dummies should follow an alpha pattern as if they had entered the network at that mix and others should appear to be in mid path as they emerge from the mix (cf. Section 3).

5 Strategic Choice of Alpha

As observed in Section 2.1, the anonymity of any message can be improved by greater uncertainty about the alpha level of *other* messages. Since Alice benefits from the fact that other people might choose non-zero α for their messages, she has an incentive to take advantage of this by choosing a lower α to get better performance but still have good security. This can be viewed as a commons: everybody will hope that somebody else takes the latency hit.

There are two ways to resolve this risk. First, note that not all users have the same sensitivity level: some users favor performance and others favor anonymity. Three factors are most important in characterizing the utility function for our users: their need for anonymity, their willingness to accept delay, and their guess at (expectation of) the current alpha levels in the network. In [2] it was shown that there can be optimal levels of free riding: more-sensitive users have incentive to provide "free" communications service for less-sensitive users by running network nodes because this will still provide additional value in the form of better anonymity protection for the more-sensitive users. This can provide adequate incentive even if there are many others running nodes. Similarly, while the existence of higher α traffic may reduce Alice's incentive to set higher α levels for her own traffic, it does not eliminate that incentive.

Second, when Alice chooses her alphas' range based on her sensitivity and timeliness constraints for her own messages, she gets increased autonomy and control over her own security and utility. Indeed, if an adversary can make reasonable guesses about a choice of alpha range for a message, then much higher or much lower alphas for other messages in a mix might actually decrease the anonymity set for a target message. For example, consider a mix containing a target message with low alpha and an ancillary message that is either from about the same alpha range or from a much larger alpha range than the target message. If the adversary learns that the second message has a larger range, then his uncertainty about the target message decreases.

Even more significantly, however, security is hard to get right when it doesn't depend on the strategic behavior of others. Users of the system are not likely to have such fine-tuned knowledge of the system, the behavior of others, and their own needs. Thus if we can prescribe recommendations for choice of alpha, for example based on analysis and observed patterns within the network, we can expect most people to heed them. (On the other hand, they may not — we

can also expect hyperbolic discounting of risk, disregard of risk for expedience, etc. [1].)

Alpha mixing itself is likely to affect the applications that can be securely used and how, so recommendations are likely to evolve. Initial recommendations can be guided by existing anonymity networks. Traffic that must arrive in real-time obviously must have $\sum \alpha = 0$. For more sensitive traffic, we might initially try to follow networks such as Mixminion and Mixmaster. But how can we do that? These use a dynamic batching strategy in which messages are chosen for the current batch randomly by the mix from a collective pool, while alpha mixing is based on individual choices made by the sender. We now turn to various generalizations on the basic deterministic-alpha mix design, including ways to combine these features.

6 Beta Alpha: Variations on Alpha Mixing

In the previous sections, we investigated and analysed some basic alpha mixing designs and the incentive questions and attacks that arise from them. In this section we introduce and briefly discuss some more complex designs that are harder to analyse fully but may provide better protection against stronger attacks.

6.1 Preventing end-to-end timing attacks on alpha mixnets

The prior work that is probably most similar to alpha mixing is stop-and-go mixing [12]. In stop-and-go mixing, the sender gives to each mix in the path a time interval. If the message arrives within the interval, it is sent at the end of the interval, otherwise it is discarded. This approach is similar to the timed deterministic-alpha mix described above, but an important difference is that a stop-and-go mixnet must be entirely synchronized to prevent losing messages. Alpha mixes offer predictable delivery times, but will still mix and deliver messages even if some nodes in the path are not adequately synchronized. On the other hand, this flexibility is also a flaw: an adversary that is global-passive except for being able to delay messages from a single sender could batch up a victim's messages and send them through an alpha mixnet all at once. Unless all the messages have $\sum \alpha = 0$ the adversary will gain limited information from this attack, but he can still learn more than from a stop-and-go mixnet.

We could include timestamps along with the α that each mix receives, and require that the message be dropped if it arrives more than some delta from the timestamp. This would make timed alpha mixes essentially equivalent to stop-and-go mixes, which might prove useful against timing correlations by such an adversary. For example, Alice might send one hundred messages to Bob that are sensitive so each has $\sum \alpha^{(i)}$ chosen uniformly at random from a range of 0 to 10. An adversary that can block all messages from Alice during this period and send them into the network will see approximately ten messages delivered to Bob immediately followed by approximately ten messages in each of the next nine time intervals. However, we need not resort to assuming a synchronized

network. Instead of including any timestamps, Alice could choose $\sum \alpha^{(i)}$ from some private distribution on a private range (not necessarily including 0). This would (1) prevent such an attack if the adversary cannot predict her distribution, (2) still have as much predictability on delivery time as stop-and-go mixes, and (3) unlike stop-and-go, still allow eventual delivery of all messages (unless they're dropped by the attacker). We are not primarily focused in this paper on end-to-end timing attacks, and we will say no more about them.

6.2 Variations on deterministic-alpha mixing

In the basic threshold deterministic-alpha mix, if there are *threshold* $= n$ messages in each of alpha levels 0 through ℓ, all of the messages in levels 0 through ℓ will be sent at once; however, messages from the different levels will not be mixed together. The mix will send all messages with $\alpha = 0$, lower the stack, send the next batch of messages that now have $\alpha = 0$, etc. An adversary may not know exactly where level i ends and level $i + 1$ begins because there may be more than n messages in a given level, but if more than n messages emerge he can know that the last messages to emerge were considered more sensitive by their senders than the first, in a stepped linear order of sensitivity. And by sending in messages of his own at known alpha levels above 0 the adversary can learn the exact levels of the messages that emerge between his messages. Then, by flooding first $\alpha = \ell$, then $\alpha = \ell - 1$, ..., then $\alpha = 0$, the adversary can guarantee a flush of the mix all the way up to $\alpha = \ell$ while also learning the alpha level of most of the messages.

The simplest solution is simply to mix all messages that emerge at once. This will prevent an adversary from watching the order in which messages exit during a flush and thus learning about their sensitivity. The stronger attack we worry about is the blending attack: an adversary emptying the mix of all messages up to the highest reasonably expected level, trickling in a message, then flooding with $\alpha = 0$ messages repeatedly to learn the sensitivity of that message and its next destination. Batching all outgoing messages together, combined with the dummy schemes presented in Section 4, would substantially reduce the risk from blending.

We could also use a threshold-and-timed mix, which would prevent the adversary from triggering an alpha-stack dump because only messages of one alpha level will emerge in each time interval. It is unclear what the local advantage is of this vs. the above multilevel-batching threshold mix. In addition, having threshold-and-timed batching would preclude the predictability advantages of timed mixes while the multilevel-batching approach could potentially offer faster performance. The primary risk of not having timing limitations on mix firing is the end-to-end effects that the adversary could induce by flooding, which would not be countered by our dummy scheme. However, that assumes a powerful adversary that can flood and watch the entire network. The nice thing about alpha mixing is that we can still have both good realtime properties and threshold protections together.

There are various ways to have realtime and threshold properties together in one mix design. We note two of them next.

6.3 Dynamic-alpha mixing

In this design, alphas are assigned to messages as they have been all along, except instead of deterministically decreasing by one after each mix firing, there is a probabilistic function f that dictates how they decrement:
$\alpha_{M,i+1} = f(\alpha_{M,i}, Pool(\alpha_{M,i}))$ where
$Pool(\alpha_{M,i}) = |\{M' : 1 \leq \alpha_{M',i-1} \leq \alpha_{M,i-1}\}|$

We believe that f would typically be monotonically nonincreasing. The sender gives f_M to a mix along with α_M. We would expect that there be some small number of easy-to-compute fs that can be chosen. The idea is that alphas decrease but only as a function of the current alpha level of the message and how many messages are in the pool below it. We have also limited the input of f to messages that arrived with a non-zero alpha, although this is not necessary. This effectively puts each message in a dynamic pool, which could also be timed.

6.4 Tau mixing

We have been describing alpha all along as a level which determines a batch of messages that a given message will be sent with, after (or possibly also together with) the messages in the alpha levels that are below it in the stack. This lends itself naturally to the batching concept familiar in the mix literature. Intuitively, threshold batching implies unpredictable delays since we don't know how long it will take for a threshold number of messages to accumulate at $\alpha = 0$. Timed mixing on the other hand will allow a predictable delay by providing an upper bound on latency. But because timed mixing also provides a *lower* bound on latency, threshold batching can be faster because it can allow messages to be processed as quickly as they arrive, provided the batch size does not get in the way.

This is the idea behind tau mixing: a message M arrives at a mix with an associated threshold τ_M of how many other messages must be sent by the mix between the arrival and sending of M. Multiple messages that have the same tau can be sent together after mixing, e.g., three messages that arrive with $\tau = 2$ are sent together. Messages that are to be sent as quickly as possible are assigned $\tau_M = 0$. This can provide realtime properties limited only by the processing speed of the network components. For example, if a message with $\tau = 0$ arrives at a mix containing messages with current $\tau = 1$, $\tau = 2$, and $\tau = 3$, the latter three should be mixed and sent together after sending the former. (We assume messages with initial $\tau = 0$ should always be sent as quickly as they arrive without the delay associated with mixing.) Messages that are more sensitive should be assigned a $\sum \tau_M^{(i)}$ from a private distribution on a range that increases with sensitivity. Many of the same features of alpha mixing apply, including the dummy strategy discussion, the techniques for allocating $\sum \tau$ across the mixes in the path, and so on.

If taus are purely threshold values, then an adversary that is powerful enough to perform a sustained flush of the entire network will be able to conduct end-to-end timing correlations on more sensitive messages (assuming we stick to a purely internally routed dummy scheme). To address this attack, both a threshold and a random minimum delay at each mix can be given as security parameter. This will prevent effective flushing unless the adversary can also perform sustained blocking of all inputs to the mixnet, and even then the attack will be substantially slowed.

7 Conclusion

In this paper we have presented a mixing technique that works together with traditional batching strategies to allow senders with varying anonymity and performance goals to share the same network and have their traffic mixed. Aside from simply letting high-sensitivity users choose to get higher anonymity for their messages, the key property it provides is a network effect: when *some* users ask for higher anonymity, *all* users can benefit.

While we proved anonymity properties for the simplest versions of alpha mixing, we have only begun to explore the possibilities and analysis of this design. Future work includes:

Multiple messages and stream-based communication: This paper has assumed the *single-message model*, where each sender produces individual uncorrelated messages. We did describe countermeasures to end-to-end timing correlations in Section 6; however, we have not carefully examined the implications of stream-based communication. Much of the reason for the success of Tor and JAP is not just the low overhead, but rather their support for bidirectional streams. But the *stream model* introduces many end-to-end anonymity attacks that seem hard to resolve simply with better batching strategies.

A full analysis of the alpha mix design: In this paper we have added to mixes an additional user-defined security parameter and explored some scenarios of attacker's knowledge about it. However, the more complex dynamic-alpha mixes and tau mixes are yet to be analysed; this seems difficult as we need to make some assumptions both about how users choose their security parameters and what the attacker knows about them.

User behavior: However much we postulate about how users behave, there is no substitute for actually getting user profiles and learning how to create incentives for secure behavior. We expect that unless we protect our users, they will try to condition their security parameter on the threat level of the message; as we have seen above this reduces rather than increases anonymity.

References

1. Alessandro Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *ACM Conference on Electronic Commerce*, pages 21–29, 2004.
2. Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the economics of anonymity. In Rebecca N. Wright, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2742, Jan 2003.
3. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, 2000.
4. 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.
5. 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.
6. George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.
7. Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986. Available from: `<http://cgm.cs.mcgill.ca/~luc/rnbookindex.html>`.
8. Claudia Díaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of ESORICS 2004*, LNCS, France, September 2004.
9. 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.
10. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *Proceedings of Workshop on Economics and Information Security (WEIS06)*, June 2006.
11. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, Aug 2004.
12. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.
13. Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, May 2004.
14. Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In F. Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.
15. Andrei Serjantov and Steven J. Murdoch. Message splitting against the partial adversary. In George Danezis and David Martin, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2005)*. Springer-Verlag, May 2005.