Message Splitting Against the Partial Adversary

Andrei Serjantov[1] and Steven J. Murdoch[2]

[1] The Free Haven Project, UK
schnur@gmail.com
[2] University of Cambridge Computer Laboratory,
15 JJ Thomson Ave,
Cambridge, CB3 0FD, UK
http://www.cl.cam.ac.uk/users/sjm217/

Abstract. We review threat models used in the evaluation of anonymity systems' vulnerability to traffic analysis. We then suggest that, under the partial adversary model, if multiple packets have to be sent through these systems, more anonymity can be achieved if senders route the packets via different paths. This is in contrast to the normal technique of using the same path for them all. We comment on the implications of this for message-based and connection-based anonymity systems. We then proceed to examine the only remaining traffic analysis attack – one which considers the entire system as a black box. We show that it is more difficult to execute than the literature suggests, and attempt to empirically estimate the parameters of the Mixmaster and the Mixminion systems needed in order to successfully execute the attack.

1 Introduction

Traffic analysis is a procedure for inferring relationships between individuals from their communication patterns. In this paper we examine traffic analysis in the context of anonymous communication systems which are designed to hide those very relationships.

The anonymity properties provided by different anonymity systems vary. Some, e.g. DC-nets, provide sender and/or receiver untraceability by hiding the very existence of traffic between their users, even against the most powerful adversaries[3]. However, users of such systems typically incur overwhelming communication costs, so we do not consider them here.

Instead, we focus on systems that provide unlinkability, e.g. Mixminion [3] and Tor [4], which only aim to prevent an adversary from linking together the sender and receiver of messages. Recent literature contains a number of exact [5,6] and statistical [7,8,9,10] traffic analysis attacks against anonymous communication systems, all based around intersection attacks. Based on these results, it can be argued that low latency communication is impossible to perform anonymously. Although given *enough* traffic, the attacker can compromise the users' anonymity, we argue that attention should be focused on finding the

[3] see [1] and [2, Chapter 2] for a more precise definition.

cases where anonymity is provided, and ensuring that practical cases fall into this category.

2 Threat Models for Traffic Analysis

Message-based systems [11,3] are believed to provide good anonymity properties against a global passive adversary, as long as the sender only transmits one message through the system [3]. This is, however, unrealistic, both in the context of email and with web browsing. Hence, we consider the case where each sender transmits at least one message and possibly many more, which introduces the potential for intersection attacks [5,6,7,8,9,10]. Connection-based anonymity systems, such as Tor [4], are thus vulnerable, as are remailers, such as Mixmaster [11] and Mixminion [3], when large files are sent.

In this section, we also assume that each sender communicates with exactly one receiver and that no receiver is communicated to by more than one sender. Initially, we assume that all packets from a sender to a receiver are sent via the same route (i.e. via the same sequence of nodes), but by relaxing this restriction later, we show that greater anonymity can be provided. We now proceed to review some threat models which are appropriate in these settings and the attacks which are made possible.

2.1 Global Passive Adversary

The global passive adversary is perhaps the most popular threat model used to evaluate anonymity properties of anonymity systems, see for instance [2,6,10]. While it can be argued that this threat model is stronger than realistically needed, a system that withstands this adversary is necessarily secure against a weaker attacker. In the global passive model, the adversary logs all traffic, both to and from all mixes and all users. The attacker's goal is to link incoming connections to outgoing connections.

Perhaps the simplest traffic analysis attack is packet counting. The situation shown in Figure 1 can be considered as an example of large files being sent through Mixminion. Here, the adversary can deduce that the messages from C were sent to F. It is interesting to note that even if we remove the restriction that each user must have exactly one communication partner, the adversary can still show that at least one message is sent from C to F. However, by splitting traffic, C's anonymity can be improved, at the cost of delaying when the reassembled message will arrive. For illustration, we have used routes of length 2, but this can be extended, with the constraint that the last node of all routes must be the same, so as to allow the message to be reassembled.

If C splits his traffic over two different routes, as is shown in Figure 2, the adversary can still link C to F. However, if we now remove the restriction of one communication partner per user, then from the attacker's perspective, C may also be communicating with D and E, and it is A and B who are communicating with F. While this seems to make a difference in theory, in practice, by observing

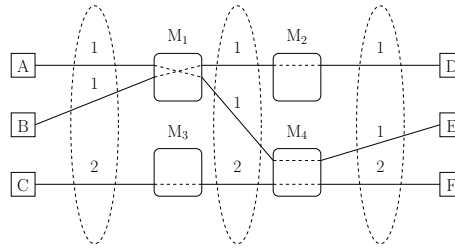

Fig. 1. Example network topology with attacker's scope of monitoring shown in dashed ellipses. From the attacker's perspective, there are two possibilities: {A–D, B–E, C–F} and {A–E, B–D, C–F}, where each user has exactly one communication partner. If this restriction is lifted, then there are two more possibilities: {A–D, B–F, C–E, C–F} and {A–F, B–D, C–E, C–F}. However, in all these cases, C must have sent at least one message to F, and so is offered no anonymity

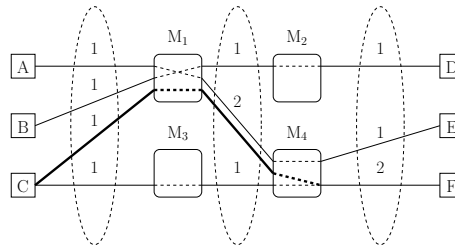

Fig. 2. Unlike Figure 1, C is provided with anonymity as there is another possibility: {A–F, B–F, C–D, C–E}. The thicker line is the new link, and we assume the attacker does not know whether there are one or two connections running over each link, only the number of packets

the sending patterns of C, the attacker is likely to be in a good position to deduce whether the two messages C sends are destined for the same receiver or not, hence our assumption above.

2.2 Adaptive Adversary

A non-global adversary can achieve the same results as a global adversary by being able to move points of monitoring, *taps*, fast enough. We call such an adversary *adaptive*. In Figure 3, Attack 1 consists of first monitoring M_1 then after establishing that the target stream, from A, goes to M_4, moving the tap there to establish the destination. This is more powerful than only monitoring inputs and outputs to the mix network, shown in Attack 2, as will be discussed further below.

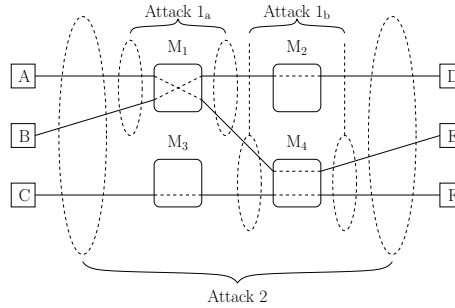

Fig. 3. Adaptive partial adversary

In the global passive and adaptive adversary scenarios, intersection attacks are extremely difficult to defend against, so instead we consider more realistic, weaker threat models and examine how their consideration might affect the design of anonymity systems.

2.3 Partial Adversary

We have shown above that a global passive adversary can compromise anonymity through intersection attacks. As this threat model is usually considered stronger than what most realistic adversaries are capable of, this does not necessarily mean that anonymity cannot still be provided in practice. By relaxing the threat model, we can show what users can do to avoid their anonymity being compromised by a realistic adversary. In the next section we consider one particular type of partial adversary, the circumstances in which they may be encountered and how to defend against them.

The partial adversary does not monitor all links. He has a limited number of taps and may put these at some, but not all, points on the network. For example,

some links may be outside his jurisdiction. Where he will place these depends on his goals. If he is interested in a particular user, he would put a tap near this user. However, if he does not know which users he is interested in or is interested in all users, he would be better advised to put the taps near as many mixes as possible, as usually there are fewer mixes than users. The key property that distinguishes the partial and the global adversary is that the partial adversary is not able to monitor all mixes or all users. Alternatively, there is at least one mix which some of the users can send traffic to, without it being observed by the attacker.

An additional restriction on the threat model would be to impose a similar restriction on the receiver side: there is at least one mix which can send messages to at least some receivers without them being observed.

As shown in Figure 4, even if an intersection attack can separate two links, if the attacker does not have a full view of all inputs and output, he cannot be sure that there is not another "2-message" connection which is unobserved. This scenario could occur if the taps are placed near M_1, D and E, but the other mixes are far away from the attacker's control.

However, if the links B to M_1 and M_4 to E both have, for example, exactly 1561 messages, then the probability of there being another connection which is unobserved, happening to have exactly the same number of packets, is low and hence the attacker can link B to E. This is, of course, a typical *traffic confirmation attack*, where the attacker confirms a previously held suspicion on sender/receiver linkage. We investigate this further in Section 3.

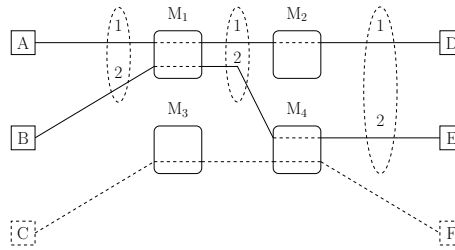

Fig. 4. Partial adversary view

3 Defeating the Partial Adversary

Our method of defending against the packet-counting attack presented in the previous section is based on sending the packets via different routes through the network. First, we present the intuition, and then a concrete scheme together with an evaluation of anonymity. Finally, we note that not only does this

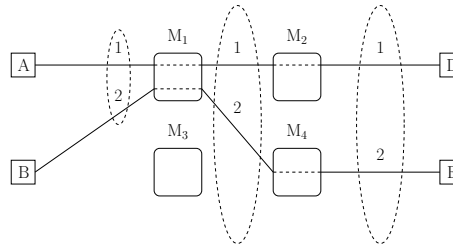

Fig. 5. Here, if B chooses M_1 rather than M_3 as his entry node, the attacker can establish that B communicates with E, and that A communicates with D

scheme protect against the packet counting attack, but also makes various flow correlation [10] and timing attacks [12] harder.

Our definition of the attacker ensures that they will not be monitoring all the mixes to which messages will arrive from users. If the users find out which of the mixes these are, they can simply send all their traffic through them. However, this is unlikely to be the case. Hence, if a sender forwards all his traffic to a receiver via the same sequence of mixes (chosen at random), there is some probability that the traffic from him to the first mix will not be observed and hence he will remain undetected. This probability is $1 - a/n$ where a is the number of entry mixes monitored by the attacker and n is the total number of mixes. However, with probability a/n, the sender's traffic is observed, allowing the attacker to mount the simple traffic confirmation packet-counting attack and compromise the anonymity of the sender. For an exposition of this and related issues see [13,14].

Our aim here is to present a scheme which significantly increases the probability of a sender not being traced by the attacker. To see why sending the traffic via different routes (and, crucially, via many different first nodes) helps, consider the following example.

In Figure 5, B has chosen a first mix which is being monitored. Since he is sending more data than A, and the output is monitored, the adversary knows that B is communicating with E. Of course, if B chose M_3 then he would be protected, but rather than relying on chance, if he splits the data between M_1 and M_3, as shown in Figure 6, the attacker only observes B sending one packet. Thus, the attacker cannot deduce that deduce that B really sent 2 packets and hence must have communicated with E.

Of course, not everything is as simple as might seem from the trivial examples above, but the intuition is clear – we want a scheme which hides, from a partial adversary, the number of messages sent. Several questions arise. Firstly, how should the sender choose routes for his packets, and more specifically, how many packets should he send through each of the entry nodes. Secondly, how much anonymity is really gained from this? We attempt to answer these questions in the next section.

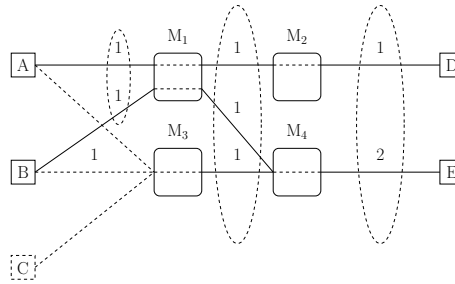

Fig. 6. By splitting the data, B hides the fact that he communicates with E, despite one of the entry nodes being monitored. The attacker is unsure whether B is communicating with E or whether A or an unmonitored C are

3.1 Choosing Routes

We have made no assumptions about how the attacker monitors the messages inside the mix network, yet we want messages to be mixed to best defeat the attacker's monitoring. It is intuitively clear that after having chosen the first mix, the routes for the packets should be chosen uniformly at random, and independently from each other. The remaining problem is then how to pick the first mix on each of these routes.

Suppose the sender has s packets to send and the attacker is monitoring a out of n mixes. Sending s/n packets to each mix is not a good approach, as the attacker will observe $a \times s/n$ packets and, knowing the fraction of the mixes he is monitoring, can deduce s. A better scheme is to send c_i packets, to each mix i, where c_i is chosen from an exponential distribution. Hence, the probability of sending x packets to a particular mix is:

$$P(x) = n/se^{-n/sx}$$

Note that both the mean and the variance of this distribution is s/n. Let $\mathbf{C}$ be the set of c_i, for i in $1 \ldots n$. Call $s' = \sum_i c_i$. Now if $s' > s$, the sender can simply can add dummy packets to make up the number of packets to s'. If, however, $s < s'$, the algorithm is rerun. Note that the probability of $\sum_i c_i \geq s = 0.5$.

Suppose the attacker observes some (s^-) messages, from which he calculates the mean and the standard deviation of the above exponential distribution in the usual fashion, call these $\hat{s_j}$. Hence the total number of messages observed by the attacker is $a\hat{s_j}$. His goal is to compute an estimate for s, the total number of messages sent. Ideally, the attacker would like to calculate the probability distribution over $\hat{s}$ and then try to mount a probabilistic version of the packet counting attack. An easier way for him to proceed is to observe the upper and lower bounds on s, based on the his observation. The lower bound of the attacker's estimate is s^- as all the mixes the attacker was not able to observe could have received 0 packets and the upper bound is infinity. The latter is not

helpful, so we provide an alternative estimate based on a rough approximation of the 3 standard deviation event.

Based on his observations, the attacker's estimate of the total number of messages sent by the user is $n\hat{s_j}$. The attacker's estimate of the standard deviation of the exponential distribution used by the sender is also $n\hat{s_j}$. Hence, the 3 standard deviation upper bound on the number of packets sent is $n\hat{s_j} + 3\hat{s_j}\sqrt{n-a}$. Note that this estimate is based on the assumption that the numbers of packets sent to each mix were generated *independently*. Of course, they were not – if the total number of packets to be sent to the mixes was lower than the size of the actual message, the algorithm is rerun.

The attacker might try to make use of this by performing a Bayesian estimation. This requires taking a known distribution of file sizes sent through anonymous communication systems as a prior and using the observations to come up with a refined distribution. This is left for future work, mainly due to the lack of a good prior.

We have assumed that the attacker observes all the receivers, so the anonymity set of our target sender is the set of receivers who have received more than $n\hat{s_j}$ packets in the interval from when the sender sent the first message, to the attackers estimate of when all the sent messages have arrived. The length of this period varies according to the system in question; for example, in Mixminion this would be a few days after the packets were sent; see Section 5.1.

However, splitting messages over several routes is not always the best option. If the sender can *a priori* determine that, even after splitting his message, the mix network will not provide him with anonymity, he should send all the messages through a single first mix. For example, this would be the case if there are no other users of the system. This will maxmise the probability that none of his messages will pass through an observed mix. The tradeoff to be made here is beyond the scope of this paper. The route selection algorithm could also take into account the administrative domain that nodes are in, as discussed in [15], however we do not consider this option here.

We claim that we have now reduced the effectiveness of the packet counting attack. However, there are more attacks to deal with. We examine one particular attack which has recently been presented by Danezis [12,16]. Essentially, the attacker takes a signal which represents a stream of input traffic, and a model of the mix, and convolves the two together. The result defines the expected output. The attacker can compare this with the output traffic streams of the mix and determine which of these matches. The attack assumes that the incoming traffic is distributed according to a Poisson distribution and Danezis has shown the attack to be effective in a simulated environment under these assumptions.

Our scheme above prevents this attack being mounted on a per-mix basis: a stream of packets incoming to the first mix will scatter randomly between the outgoing links as the routes of the packet were chosen independently. Hence, rather than analysing each mix individually, the attacker has to view the entire mix network as one supermix. Even if the attacker can see all inputs and outputs to the supermix, this is weaker than the global passive adversary threat model,

as shown in Figure 7. In Attack 1, the attacker can clearly follow the data from A to D as no mixing is taking place. However in Attack 2, the supermix formed by $M_1 - M_4$ does mix the two streams.

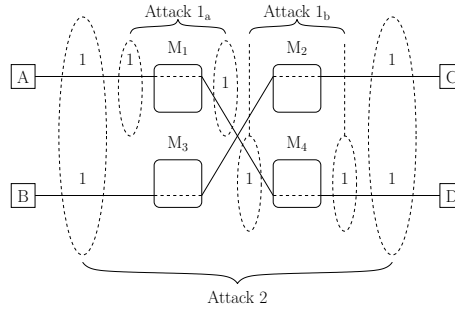

Fig. 7. Adaptive partial adversary versus supermix analysis

The mix analysed in [12] is expressed via a delay characteristic. The delay characteristic is the probability distribution over the possible delays experienced by incoming messages. To mount this attack, we therefore need to know the characteristic delay function of our anonymity system abstracted as a mix.

There are two possible approaches: either to combine models of mixes into a model of a *complex mix* or to empirically measure the characteristics of the system as a whole. We present both approaches, and show that they run into difficulties. While this does not prove that such attacks are impossible to mount, they do bring into question their effectiveness in realistic environments.

4 Deriving Delay Properties of "Complex Mixes"

As Serjantov et al. observed in [17], there are two main types of mix delay strategies: timed and threshold. Here we attempt to suggest characteristic delay functions for combinations of each of these types of mixes.

4.1 Timed Mixes

We assume that message arrival is uniformly distributed over the time period t, in other words, when the user sends his message, he is not aware of when the mix flushes, or he does not take that knowledge into account. In this case, the delay characteristic of a timed mix is rather simple – it is the uniform distribution between 0 and t. Note that the delay characteristic of a cascade of i timed mixes is a uniform distribution between 0 and a point in the interval between t and ti (depending on how the mixes are synchronised).

The characteristic delay function for a timed pool mix, however, depends on traffic levels which, as shown in [18] do not follow a well known probability distribution. This is because the probability of a message remaining in the mix depends on how many messages arrive at the mix during the same round. The only easy case here is if the attacker assumes that the traffic arrives at a constant rate. In this case, the characteristic delay function is a discretized geometric distribution. If the delay is measured in rounds

$$P(\text{delay} = i) = p^{i-1}q$$

where $p = n/n + f$, n being the constant number of messages in the mix at each round and f the constant size of the pool. Of course, the above is also the probability of ti seconds delay.

The characteristic function of a cascade of m timed pool mixes is:

$$P(\text{delay} = i) = \binom{m}{i-m} p^{i-m} q^m$$

If a timed pool mix works on the basis of having a constant probability, p, of forwarding a message (a variant of the Timed Dynamic Pool Mix [11]), then its characteristic delay function is exactly the one given above. Nevertheless, these examples are slightly contrived. A more sensible mix would have a minimum pool or alter the probability of forwarding each message, based on how many messages are inside the pool [19]. From this we can safely conclude that deriving accurate characteristic delay functions for realistic timed mixes is infeasible without knowledge of the distribution of message arrivals.

4.2 Threshold Mixes

The situation is not any better with threshold mixes. Calculating the characteristic delay function for a simple threshold mix already requires assumptions on traffic and hence has the problems detailed above and in [18].

As one might expect, under the constant traffic assumption, the characteristic delay of the threshold pool mix, in rounds, is the same as that of the corresponding timed pool mix.

The characteristic delay function of a network of threshold mixes requires not only assumptions about traffic arrivals to the network, but also about the choice of routes that users take for their messages. Indeed [2, Chapter 6] suggests that calculating the delay of a mix network of threshold mixes is as hard as calculating the anonymity of it.

5 Estimating Delay Properties of Complex Mixes

One possible approach which can be used in practice is simply to sample the network under appropriate traffic conditions (i.e. those similar to the time when the target message is sent) and hence obtain the characteristic delay function. In

order to test our assumptions and evaluate the consequences of our proposals, we attempted to measure the effective delay function of the Mixmaster and Mixminion "supermix" over 26 days. The client software was run in its default configuration, except for Mixminion, where we forced the path length to 4 (by default it varies). To avoid our probe messages interfering which each other, we kept no more of our messages in the system than there are mixes. Latency data was collected by a Python [20] script and graphed in GNU R [21], based on a design described in [22]. This data is available for download[4].

We would, of course, have liked to evaluate the characteristic delay function of a more real-time anonymity system than Mixmaster or Mixminion, but the obvious candidate, Tor, optimises for efficiency and does not aim to protect itself from this attack. Hence, in order to demonstrate the difficulty of calculating the characteristic delay function, we have resorted to attempting to estimate it for the above systems.

5.1 Results

The distribution of measured latencies is shown in Figure 8, and the change of latency over time is shown in Figure 9, along with the distribution of latencies in two selected intervals. A statistical summary of the full data set, as well as the selected intervals, is shown in Table 1.

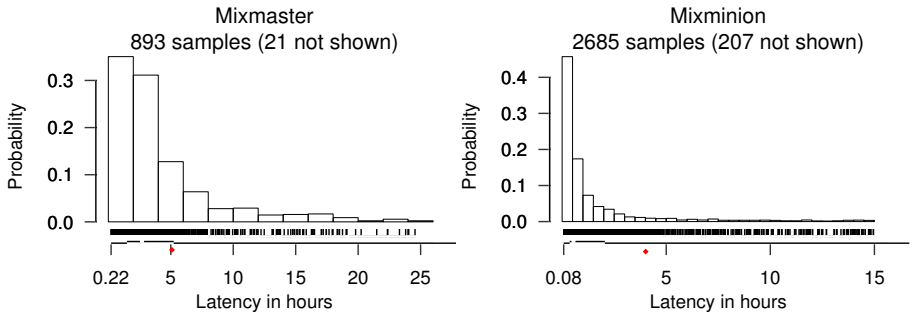

Fig. 8. Latency measurements of Mixmaster and Mixminion

As can be seen, Mixmaster has a larger median latency than Mixminion. This is because Mixminion is currently in alpha so, by default, nodes use a 10-minute timed mix. The algorithms used by Mixminion nodes at time of writing is shown in Table 2. Although the latency for most messages is below the 4×10 min limit expected for a path length of 4, 44% are above. Some of these are explained by the nodes using non-default mixing algorithms, but others are due to nodes which fail and later recover.

[4] `http://www.cl.cam.ac.uk/users/sjm217/projects/anon/`

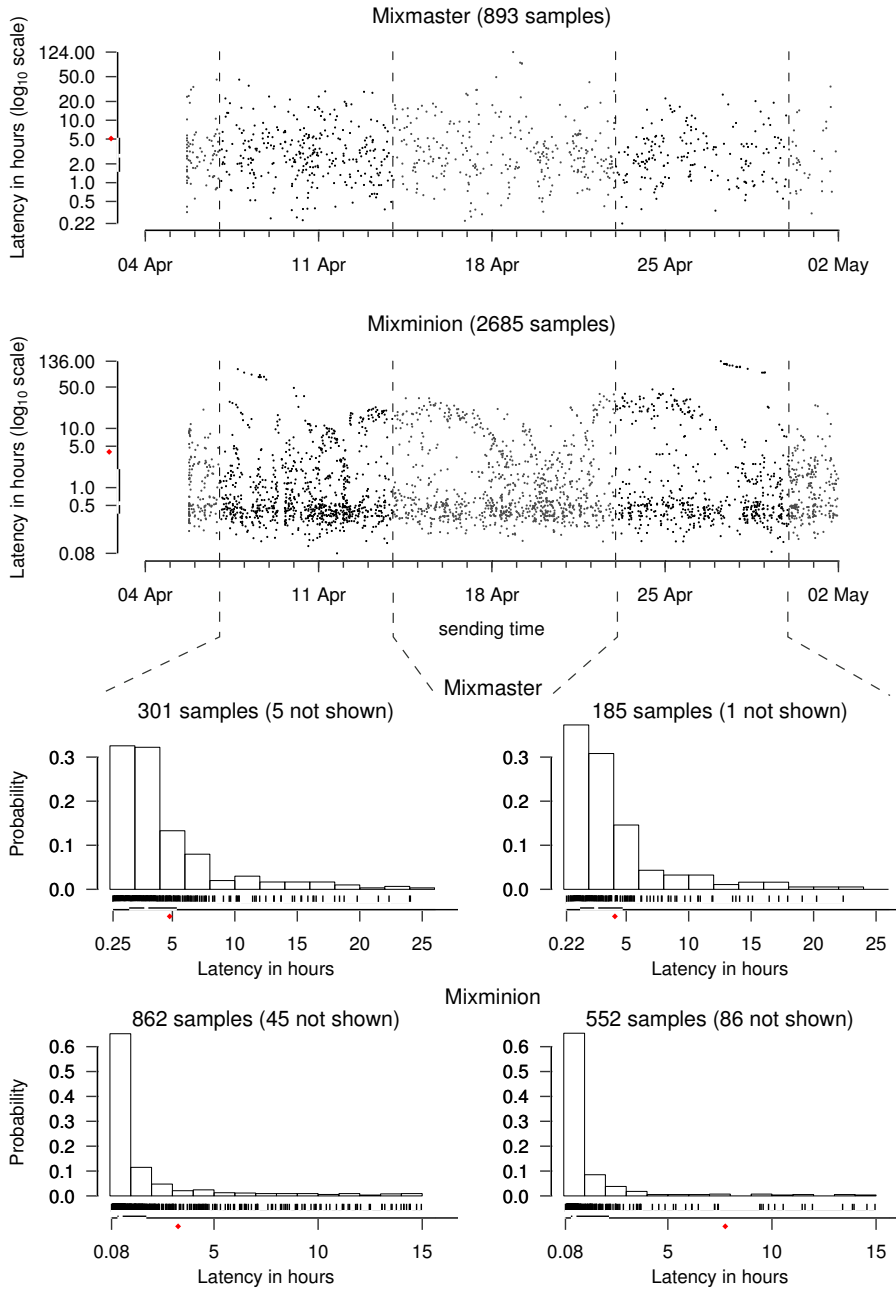

Fig. 9. Latency measurements over time

Table 1. Summary of data collected

	Mixmaster latency (hours)			Mixminion latency (hours)		
	Overall	Range 1	Range 2	Overall	Range 1	Range 2
Min.	0.22	0.25	0.22	0.08	0.08	0.08
Q.1	1.49	1.54	1.30	0.36	0.36	0.34
Med.	2.70	2.91	2.60	0.55	0.51	0.51
Q.3	5.23	5.36	4.72	2.05	1.75	2.17
Max.	123.70	44.78	25.79	136.40	100.10	136.40
Mean	5.10	4.78	4.09	4.01	3.27	7.76

Table 2. Mixminion node mixing algorithms. A timed mix flushes all messages from the pool after every mix interval. A dynamic pool, as is used in Mixmaster, flushes a randomly selected set of messages after every mix interval, such that the pool size never falls below the pool minimum size, and the percentage of the pool sent out is no more than the pool rate. A binomial dynamic pool flushes a randomly chosen number of messages, based on the number that would be flushed using the dynamic pool algorithm

Number of nodes	Mixing algorithm
25	Timed. Mix interval: 10 min (*default configuration*)
2	Timed. Mix interval: 15 min
1	Timed. Mix interval: 20 min
1	Timed. Mix interval: 30 min
1	Dynamic pool. Mix interval: 30 min, Pool Rate: 50%, Pool Minimum Size: 5
1	Binomial dynamic pool. Mix interval: 10 min, Pool Rate: 70%, Pool Minimum Size: 3
1	Binomial dynamic pool. Mix interval: 30 min, Pool Rate: 50%, Pool Minimum Size: 5

6 Conclusion

In this paper we examined the problem of sending a large message (more precisely, a sequence of many messages) through mix networks. Firstly, we presented a well-defined notion of a partial adversary threat model. We then show that in the presence of this adversary, sending packets via different routes through a mix network yields increased anonymity. To perform a packet counting attack, the attacker must now know the total number of fragments a message was split into, but by distributing the fragments over entry mixes using a non-uniform distribution, a partial adversary is uncertain as to the total. Finally, we demonstrate that intersection attacks relying on knowledge of the characteristic delay function, while powerful, are more difficult to perform on deployed mix designs than previously thought. This is due to the dependency of message latency on input traffic, which is not known and difficult to estimate. This supports previous observations on the unpredictability of traffic in anonymity systems.

7 Acknowledgements

We thank Roger Dingledine and the anonymous reviewers for their insightful comments, George Danezis for his honest feedback, and Len Sassaman for helpful suggestions on improving this paper.

References

1. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.17 (2000)
2. Serjantov, A.: On the Anonymity of Anonymity Systems. PhD thesis, University of Cambridge (2004)
3. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a Type III Anonymous Remailer Protocol. In: Proceedings of the 2003 IEEE Symposium on Security and Privacy. (2003)
4. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. (2004)
5. Kesdogan, D., Pimenidis, L.: The hitting set attack on anonymity protocols. In: Proceedings of 6th Information Hiding Workshop (IH 2004). LNCS, Toronto (2004)
6. Kesdogan, D., Agrawal, D., Penz, S.: Limits of anonymity in open environments. In Petitcolas, F., ed.: Proceedings of Information Hiding Workshop (IH 2002), Springer-Verlag, LNCS 2578 (2002)
7. Agrawal, D., Kesdogan, D., Penz, S.: Probabilistic Treatment of MIXes to Hamper Traffic Analysis. In: Proceedings of the 2003 IEEE Symposium on Security and Privacy. (2003)
8. Danezis, G.: Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, Katsikas, eds.: Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003), Athens, IFIP TC11, Kluwer (2003) 421–426

9. Danezis, G., Serjantov, A.: Statistical disclosure or intersection attacks on anonymity systems. In: Proceedings of 6th Information Hiding Workshop (IH 2004). LNCS, Toronto (2004)
10. Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On flow correlation attacks and countermeasures in mix networks. In: Proceedings of Privacy Enhancing Technologies workshop (PET 2004). LNCS (2004)
11. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster Protocol – Version 2. Draft (2003)
12. Danezis, G.: The traffic analysis of continuous-time mixes. In: Proceedings of Privacy Enhancing Technologies workshop (PET 2004). LNCS (2004)
13. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an Analysis of Onion Routing Security. In Federrath, H., ed.: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, Springer-Verlag, LNCS 2009 (2000) 96–114
14. Wright, M., Adler, M., Levine, B.N., Shields, C.: An analysis of the degradation of anonymous protocols. In: Proceedings of the Network and Distributed Security Symposium – NDSS '02, IEEE (2002)
15. Feamster, N., Dingledine, R.: Location diversity in anonymity networks. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004), Washington, DC, USA (2004)
16. Danezis, G.: Better Anonymous Communications. PhD thesis, University of Cambridge (2004)
17. Serjantov, A., Dingledine, R., Syverson, P.: From a trickle to a flood: Active attacks on several mix types. In Petitcolas, F., ed.: Proceedings of Information Hiding Workshop (IH 2002), Springer-Verlag, LNCS 2578 (2002)
18. Díaz, C., Sassaman, L., Dewitte, E.: Comparison between two practical mix designs. In: Proceedings of 9th European Symposium Research in Computer Security (ESORICS). LNCS, France (2004)
19. Díaz, C., Serjantov, A.: Generalising mixes. In Dingledine, R., ed.: Proceedings of Privacy Enhancing Technologies workshop (PET 2003), Springer-Verlag, LNCS 2760 (2003)
20. Python Software Foundation: Python. `http://www.python.org/` (2003)
21. R Development Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. (2004) ISBN 3-900051-07-0 `http://www.R-project.org/`.
22. Tufte, E.R.: The Visual Display of Quantitative Information. 2nd edn. Graphics Press (1992) ISBN 0-961392-10-X.