Sleeping dogs lie on a bed of onions but wake when mixed

Paul Syverson

Naval Research Laboratory, USA.

Abstract. We introduce and investigate *sleeper attacks* and explore them in the context of anonymous communication, especially mix networks. Sleeper attacks can make use of the interference inherent to mix protocols. Simply by leaving his own messages in a mix network an adversary can learn about the communication of network users. Sleeper attacks can be combined with epistemic attacks, statistical disclosure, or other attacks to be made even more effective. We use sleeper attacks to disprove the common belief that mix networks are necessarily more secure than onion routing networks. Finally we use our results to disprove another commonly held belief about computer security in general, that it is always conservative to prove security against the strongest possible adversary.

1 Introduction

Suppose Alice$_1$ and Alice$_2$ are known to be the two possible correspondents to Bob, and suppose each sends a message into two basic threshold mixes, mix$_1$ and mix$_2$ respectively. For purposes of this example it does not matter what the firing threshold of the mixes are.[1] Assume the adversary Dorm can see Alices send messages and can see Bob receive messages, but he cannot generally see any mixes send or receive messages or see the mixes internal workings. Under these circumstances, if Dorm later sees Bob receive a message from mix$_3$ he cannot tell which Alice sent it.

Suppose, however, that Dorm has previously left his own messages, S_1 and S_2 (each to and from himself) in mix$_1$ and mix$_2$, respectively. If he receives S_1 but not S_2, then, absent other considerations, he knows that mix$_1$ has fired while mix$_2$ has not. He thus knows that Alice$_1$ sent the message to Bob. The information leak in this toy example is not strictly speaking a passive channel

[1] For the unfamiliar reader, a *threshold mix* receives messages until it reaches a given *threshold* at which point it fires, forwarding all of the received messages to their next destination, which might be the ultimate receiver, a bulletin board, or another mix. Messages are transformed by the mix and the batch of messages permuted by the mix so that it is not feasible to match which honest messages going into the mix match which honest messages coming out, as long as there are at least two honest (not adversary controlled) messages. This paper assumes general familiarity with anonymous communications research. See [13, 6] for a background survey.

since Dorm had to place his own messages in the mixes in order for it to work. It is not, however, a typical active attack. As long as his messages are left in the mix, it does not matter when he put them there. His active component can be any time before, possibly long before, Alice sends her message. For this reason we call this a *sleeper attack*.

In this paper we will explore sleeper attacks on anonymous communication. We begin by describing sleeper attacks in the contexts of various types of mixes. We also note that a weaker adversary can be more effective using sleeper attacks in combination with other attacks than simply using those attacks by themselves.

Next we consider sleeper attacks on onion routing networks. A commonly held belief amongst anonymous communication researchers and practitioners is that mix networks are more secure than onion routing networks. On the other hand onion routing networks are far more practical and usable for most users and applications. Thus, these design alternatives are generally presented as making a trade-off between security and practicality. For example, the original Tor design paper [11] says that, "relay-based anonymity designs have diverged in two main directions. Systems like Babel [15], Mixmaster [23], and Mixminion [7] have tried to maximize anonymity at the cost of introducing comparatively large and variable latencies. Because of this decision, these high-latency networks resist strong global adversaries, but introduce too much lag for interactive tasks like web browsing, Internet chat, or SSH connections." We show that characterizing mixes versus onion routers as only a security versus practicality trade-off is misconstruing security: mix networks are in important ways less secure than onion routing networks, even if they are more secure in other ways.

In previous work [28], we examined the dependence of this characterization on unfounded trust-uniformity assumptions, on ignoring usability implications, and on unrealistic adversary models. Herein, we go a step further. There are certainly realistic configurations, environments, and adversaries for which mix networks are more secure than onion routing networks. We present examples where the opposite is true. Our examples have moderate and realistic adversaries. The networks consist of the same number of nodes in the same configuration, just one composed of mixes and the other composed of onion routers. Both networks have the same number of users. By their natures usage of the networks cannot be identical, but we will make them as comparable as possible. The capabilities of the adversaries and their deployment is the same in both networks. We will show that there exist such circumstances in which the onion routing network is more secure than the comparable mix network.

Another even more broadly held belief is that it is always conservative to assume the strongest possible adversary. This is a notion from many areas of computer security and cryptography not just anonymous communication. We show (with the same realistic systems and realistic adversaries) an example of two systems in which one system is more secure against the stronger adversary but the other is more secure against the weaker adversary.

2 Mixing sleepers awake

"It is nought good a sleping hound to wake."

Geoffrey Chaucer — Troilus and Criseyde

Mixes derive their security from altering the order of messages they receive to obscure the relation of inbound to outbound messages. This is true for mix designs from simple threshold mixes to timed dynamic pool (Cottrell) mixes to binomial mixes. Put differently, mixes create interference between messages. This interference puts bounds on the information leaked or rate of information leaked to an observer of the mix [24, 25, 29, 2] But it also puts a lower bound on information leaked to an observer. In a threshold mix with batch size n, an adversary observing a single input to and single output from the mix has uncertainty about whether they match that is bounded by n. A sleeper attack can take advantage of this.

Consider a layered network of threshold mixes with a sleeper in each mix, where there is one layer of mixes receiving inputs from senders forwarding to a second layer of mixes that forward messages to their ultimate recipients. Suppose the adversary observes just one message being sent into some mixes and sees just one message being received. Suppose he learns from his sleepers which layer1 mixes fires and which do not, and then learns which of the layer2 mixes fires and that the others do not. From this he knows the received message could not come from the sender into any layer1 mix that did not fire. Assuming the threshold and the distributions of messages are known, then for the observed input messages that could match the observed received message at all he also can attach a significantly higher probability to their matching the received message than he could without the sleepers. This could also be combined with knowledge about sending rates and thus the likely number of messages in a layer2 mix based on time since last firing to infer still more.

There are three basic categories of interference that mixes can have, based on the type of the mix. Mixes that require a number of messages to be received to fire have mandatory interference between a message sent by the mix and previous messages received by the mix. (This includes messages sent but not received by the mix, for example, dummy messages.) This also applies whether they are simple threshold mixes or use some sort of pool or other function that relates the probability of sending a message to previously received messages.

Mixes may also be purely timed: they randomly order the messages that they have received during a given interval and forward (some of) them (along with any messages from the mix itself) at the end of the interval regardless of what messages if any have been received in that interval. These mixes have contingent interference. Messages that are available for mixing will interfere, but if no messages are available, there is no interference with received messages. If the mix itself generates messages, then there is interference with those.

If a mix requires both a minimum interval of time and a minimum of received messages in order to fire, then it still has mandatory interference between the messages sent and previously received messages.

Stop-and-go (sg) mixes [19] forward a message that was sent at a time designated in the message regardless of other messages in the mix. Messages sent from stop-and-go (sg) mixes have no interference at all. (We restrict interference to that inherent in the protocol and treat as out of scope any interference from processing time for necessary computation or transmission of messages. This scope will also apply to onion routing networks, to which we will return below.) What this shows us is that stop-and-go mixes are mixes in name only. Any mixing they provide is virtual rather than inherent to their operation. For the remainder of the paper, we will restrict usage of 'mix' to systems that base the ordering of output messages at least partially on other messages the mix outputs, whether they were received or generated by the mix.

Sleeper attacks cannot reveal anything in an sg mix network or a purely timed mix network. For sg mixes, other messages can simply not interfere with a sleeper. For purely time mixes, there can be interference, but anything a sleeper attack could reveal is already known to the adversary from the mix protocol.

If a mix has any kind of pool or other function that makes the forwarding of a message held by a mix probabilistic when the mix fires, then a sleeper cannot determine with certainty when a message of interest was sent by the mix. Nonetheless, as long as the adversary can keep an adequate representative sample of sleepers in the mix, the he can learn from when sleepers are sent by the mix the same probabilistic information about received messages in sent batches as he could if he could observe the batches themselves emerging from the mix.

2.1 Combining the sleeper with epistemic and other attacks

"*He sees you when you're sleeping. He knows when you're awake.*"

John Frederick Coots and Haven Gillespie — "Santa Claus is coming to town"

Epistemic attacks on anonymity were first introduced by Danezis and Clayton [5]. They described a *route fingerprinting* attack in which an adversary knows which nodes in an anonymity network are known to which possible senders. Using this information, an adversary observing a message on even part of a route can use which senders would know how to construct that route to narrow down the set of possible senders. Danezis and Syverson [9] later described *route bridging*, which makes use of what senders do *not* know about the network nodes to determine which routes it would be impossible for some senders to construct and again narrow down the possible senders. These attacks use an adversary's observation of all messages entering or leaving a mix in a single batch. By knowing which senders could know about all the possible combinations of the three mixes involved in every possible route through the observed mix for that batch he can narrow down the number of senders. If the adversary can make use of sleepers and patterns of mix firings, then he can conduct such attacks without having to observe as much of the actual network. For example, if a message is received that would require going through a mix that did not fire if it were sent by one of the senders who otherwise match what is known about a route and received message, then that sender is eliminated without having to be able to directly observe the mix.

Disclosure attacks [18] and statistical disclosure attacks [4, 8] are long-term intersection attacks to determine who is talking to whom by observing when potential senders and receivers are present together and when they are not. The statistical version provides answers with high probability rather than with certainty, but it is also much more efficient. The original versions, both statistical and not, required a global passive observer. Later, Dingledine and Mathewson [20] showed how effective the attack could be when only part of the network was observed by the adversary. Sleeper attacks can be added to eliminate or support possible communication patterns by knowing which mixes fired and in which order, again making it possible to have an equally effective attack with a weaker adversary.

3 Sleepers and Onions

As already illustrated, there are settings where an adversary can learn information from a sleeper attack on a mix network. To make our initial toy example into something more real and concrete, suppose that an adversary (Dorm) is following a blog (Bob) and has some candidate posters to that blog (Alices) under observation. For simplicity we will assume two Alices. Suppose the Alices are known to use various anonymous communication systems, but for jurisdictional, legal, or resource reasons, none of these are observable by the adversary. All he can do is passively watch when either Alice sends or receives messages and he can see when the blog updates with new posts. Suppose the Alices are unknown to each other but both are relatively paranoid and relatively up on the anonymous communications literature. They thus each choose to use a high latency mix network for sensitive communications. Bob only updates twice a day. But given the high latency of the mix network, when Bob is observed to be updating with an item of interest, Dorm is able to discern from the sending activity of the Alices, and the pattern of mix firings he observed from his sleeper attacks that one of the Alices could not have sent the information of interest but the other could.

If we combine the pure sleeper attack with other information, Dorm may be able to conduct this attack even if he cannot directly observe the Alices. For example, if he is aware that they only know about different parts of the mix network, then he can use this for an epistemic attack together with a sleeper attack to make the same inference just described even if the only thing he can observe is Bob's public blog updates. Similarly if Dorm is aware that the Alices trust some parts of the network more than others and are thus inclined to prefer those parts [17], he can use this in conjunction with a sleeper attack and simply seeing Bob's updates to indicate which Alice is the likely poster, or to increase his confidence in previous suspicions.

Contrast this with onion routing networks. Assume the exact same situation as the above except that instead of using mix networks the Alices are using onion routing networks. In other words, assume the situation is exactly as above, except that the network nodes are onion routers rather than mixes. And assume that the Alices are communicating with Bob (or whoever) via an onion routing protocol rather than a mix protocol.

If Alice$_1$ did not use the onion routing network at all during the relevant period until after Bob's update appeared and Alice$_2$ did, then Dorm would be able to discern that Alice$_1$ could not have posted the information. But this is also true of a mix network for the same period, although the mix network can obscure the period that either Alice's messages could arrive at Bob. If we limit to situations where both Alices used the network in the period before Bob's update, then the sleeper attack will provide no information about which Alice posted the message in the case of an onion routing network and can determine which Alice made the post in the case of a mix network. And, this remains true if neither Alice is observed at all and a sleeper attack is combined with epistemic or trust-based attack.

In this section we have looked at sleeper adversaries that are relatively trivial to implement and deploy, either as capable of only sleeper attacks or in combination with other simple attacks. We have examined such adversaries applied to realistic communication settings involving anonymity networks. To compare mix networks to onion routing networks in these environments we kept the adversary capabilities exactly the same, and we kept the communicants, their use of the networks and the network configurations virtually the same. The only change was to have the network nodes run either an onion routing protocol or a mixing protocol. In these identical settings the mix protocol can leak significant information but the onion routing protocol leaks no information. We have thus shown unequivocally that it wrong to say mix networks are more secure than onion routing networks.

There are interference attacks that have been run against Tor [26, 14]. They are not actually attacks on onion routing at the protocol level. Rather they are attacks on implementations taking advantage of the time it takes to actually process and send communication through the Tor network. They are thus outside the scope of this paper. But we will consider them briefly. The attacks found by Danezis and Murdoch [26] have been shown by Evans et al. [14] to simply not work against the current Tor network, which is much much larger than the Tor network at the time of [26]. Evans et al. went on to explore extensions of Danezis and Murdoch's attacks that were feasible. They are feasible however, only in combination with several other attacks that, while plausible, will work only with certain types of application communication used in a particular way rather than against Tor communication in general [14]. And they still require sending a significant amount of traffic into the network at a constant rate, good clocks relatively well synchronized, and numerous other assumptions. Thus even if we considered implementation rather than just protocol-level attacks, compared with sleepers they require far more resources and assumptions, as well as specialized settings and specific types of application communications.

4 Anonymity networks and their adversaries

Onion routing networks are similar to mix networks in some respects, but they primarily derive their security from the unpredictability of routes and the inability of an adversary to watch enough of the network to be likely to see both ends of a connection. Though they have been combined with mixes for research purposes (so that some form of mixing was done at onion routers) this is not typical and currently considered to serve no useful purpose.

Security for mix networks is typically evaluated assuming a global passive adversary (GPA). For a large distributed network a global adversary is very strong, perhaps even unrealistically so. On the other hand, for a publicly accessible network that does not require registered users, it is also unrealistic to assume the adversary is not able to generate his own messages. Similarly, the original motivation for having mix networks rather than communicating via single mixes was that some mixes might be compromised [3]. It is also unrealistic to think that an adversary compromising a mix might not try to add, drop, or alter messages in his control if he can get away with doing so. For these reasons, many security analyses add to the GPA the ability to send messages into the network and the ability to create and/or manipulate messages at a compromised subset of the mixes.

This is the adversary model against which Mixminion was designed and evaluated and the one we initially adopt. Against this adversary onion routing is completely broken. Just the global passive element is enough to break onion routing. It has been long understood and experimentally verified on the Tor network, that a passive adversary can virtually always confirm the association of inbound connections to and outbound connections from the network by the timing and volume of traffic [27]. Indeed, it has been shown in simulation that simply creating connections is enough, the correlation can be confirmed even without sending any data [1]. For these reasons, we have said since inventing onion routing that it guards against traffic analysis, not traffic confirmation.

Observations such as we have just made are the reason that people have generally held that onion routing networks are less secure than mix networks. And against the above adversary they are. However, many have noted that when taking usability and performance into account, the size of both the network and user base for onion routing networks is much larger than for mix networks [10]. The public Tor network is orders of magnitude bigger and has orders of magnitude more users than the largest public mix networks that have existed. And this is one of several reasons that onion routing networks may be more secure than mix networks: it is much harder to have a realistic global adversary against the much larger Tor network than against a Mixmaster or Mixminion network. It is also easier for an adversary to apply all of its available resources to the few hundreds of Mixmaster users it detects than against the hundreds of thousands of Tor users, if it could even observe them all [28]. Still these are somewhat apples-and-oranges comparisons. Even if it is more realistic to do so, these observations are based on comparing very different network sizes and different sizes of user base.

In the previous section, however, we have shown that (for the same network configurations and size, with the same senders and receivers, and against the same adversary) mix networks can leak important information when onion routing networks do not. But this does not imply that onion routing networks are in general more secure than mix networks. As already noted, against an adversary that includes a global observer, onion routing is completely broken. The source and destination of every connection are exposed. A mix network is not completely broken against the above described adversary. Exactly what protection it provides is complicated and cannot be determined without at least parameterizing the number of senders, receivers, and network nodes—also what fraction of the network is compromised, what exactly the adversary can do at what rate, the rate and distribution of sending messages, whether observations are one-time or extended, and if so the dynamics of all the above, etc. Nonetheless, it is clear that, even with all this, for many reasonable choices of parameters mix networks provide some protection. One of the reasons a GPA is often chosen for analysis is that it simplifies such analysis to a tractable level. But as already noted, such a model is so unrealistic that it is not clear what we learn from using it. Though one could perhaps create believable settings where the GPA makes sense, for no application for anonymous communication yet published has it been plausible to assume such an adversary. This adversary would need to be able to watch the entire network, regardless of size, and yet cannot attempt to even slightly delay messages, send messages of its own, or corrupt some users to send (or not) at useful times.

The sleeper attack by itself requires a very weak adversary. He does not do anything to the messages of other users in any way that plays a role in the attack. (He cannot help affecting the firing rate and message ordering of mixes by placing messages in them, and he could conduct a 1 attack—the complement of an $n-1$ attack, but in a pure sleeper attack we ignore these.) He corrupts no nodes in the network. He can only observe sending and receiving behavior at a few points. He needs to generate messages at a relatively low rate. (Call a sleeper attack *complete* if the adversary always learns when a mix fires. For a complete sleeper attack in a network of threshold mixes, he must send at least one message per mix firing. For other types of mixes he can only achieve high likelihood of a complete attack.) He does not need a clock at all. He only needs to tell the ordering of mix firings relative to each other and any of the few transmissions he observes. This is thus an attack that even a quite low-resource adversary should be able to conduct easily. It is thus much more realistic than a global passive adversary. Like the GPA, however, once some other parameter settings are given, it should also prove tractable to analyze, although we do not explore that in this paper. We thus have two adversaries that are subadversaries of the most powerful adversary we have described above. One, however, refines the powerful adversary to something plausible, while the other refines it to something unrealistic.

We have uncovered something else in the above, however, besides a lesson about useful versus inherently impractical adversary models. A generally accepted truth of computer security is that it is conservative to assume the most powerful adversary. This has strong intuitive plausibility. If a system is secure against a more powerful adversary it should remain secure against an adversary with fewer capabilities or diminished capabilities. As has been shown in the multilevel security literature, however, intuitions can be deceptive.

There are theoretical examples of multilevel-secure systems that are effectively secure against a strong adversary but leak information against an adversary with fewer capabilities [21, 16, 22]. The examples we have shown are *monotonic*: they do not show better security of a given system in a given environment against a more powerful adversary than against a strictly weaker adversary. An adversary that is both globally observing and able to mount a sleeper attack can learn more against a mix network than an adversary that can only mount a sleeper attack. Similarly an adversary that is globally observing and can mount a sleeper attack is able to learn more against an onion routing network than one that can only mount a sleeper attack. Nonetheless, in the settings we have described, when the adversary is both globally observing and lays sleepers the mix network is stronger, whereas when the adversary is only able to lay sleepers and can only make the smaller set of observations described above, the onion routing network is stronger.

This result undermines a fundamentally held belief about computer security. We typically assume the strongest possible adversary for at least two reasons. One is that we assume that if the system is secure in that setting it will be secure in weaker settings. Previous literature has explored what is needed to make that assumption correct. The other reason for evaluating against the strongest possible adversary is that we assume that if one system is more secure than another against the stronger adversary, it will also be more secure than the other against the weaker adversary. Our example shows that this is not necessarily true. And this is not simply a point about differences in implementation that can create different vulnerabilities in the different systems. The crossover occurs with the protocols at the same level of abstraction; only the capabilities of the adversary are diminished. And, if what makes the strongest possible adversary stronger is something that is unrealistic, following the standard reasoning may lead us to choose the system that is less secure against a more realistic adversary.

In a fuller analysis we intend to explore this with more mathematical detail and rigor. We also intend to more fully explore the relationship between different types of networks and sleeper adversaries, both alone and combined with other adversaries. In [12] we discussed the security of combining messages with different latency and security needs in what we called alpha mixing. As we have seen, any kind of actual mixing (in other words, interference between received or generated messages in a mix) can be vulnerable to sleeper attacks. But one of the variants described was called timed alpha mixing, which was effectively a less restrictive variant of sg mixing. Like sg mixing, it is not actually mixing at all. Like basic onion routing, neither of these forms of ‘mixing’ is vulnerable to sleeper attacks. It will be interesting to explore the use of sg mixes or timed alpha mixes in combination with onion routing to examine the interplay of security it provides.

References

1. Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against Tor. In Ting Yu, editor, *WPES’07: Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society*, pages 11–20. ACM Press, October 2007.
2. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation/Information and Control*, 206(2–4):378–401, 2008.
3. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2):84–88, February 1981.
4. George Danezis. Statistical disclosure attacks. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
5. George Danezis and Richard Clayton. Route fingerprinting in anonymous communications. In *Sixth IEEE International Conference on Peer-to-Peer Computing, P2P 2006*, pages 69–72. IEEE Computer Society Press, 2006.
6. George Danezis, Claudia Diaz, and Paul Syverson. Anonymous communication. In Burton Rosenberg, editor, *Handbook of Financial Cryptography.* CRC Press, 2010.
7. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings, 2003 IEEE Symposium on Security and Privacy*, pages 2–15, Berkeley, CA, May 2003. IEEE Computer Society.
8. George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In Jessica Fridrich, editor, *Information Hiding: 6th International Workshop, IH 2004*, pages 293–308. Springer-Verlag, LNCS 3200, May 2004.
9. George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In Nikita Borisov and Ian Goldberg, editors, *Privacy Enhancing Technologies: Eighth International Symposium, PETS 2008*, pages 151–166. Springer-Verlag, LNCS 5134, July 2008.
10. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In Ross Anderson, editor, *Fifth Workshop on the Economics of Information Security (WEIS 2006)*, June 2006.
11. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–319. USENIX Association, August 2004.
12. Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending different latency traffic with alpha-mixing. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies: 6th International Workshop, PET 2006*, pages 245–257. Springer-Verlag, LNCS 4258, 2006.
13. Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys*, 42(1), 2010.

14. Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium*, pages 33–50, Montreal, Canada, August 2009. USENIX Association.
15. Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Symposium on Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.
16. Jeremy Jacob. On the derivation of secure components. In *IEEE Symposium on Research in Security and Privacy*. IEEE CS, May 1989.
17. Aaron Johnson and Paul Syverson. More anonymous onion routing through trust. In *22nd IEEE Computer Security Foundations Symposium, CSF 2009*, pages 3–12, Port Jefferson, New York, USA, July 2009. IEEE Computer Society.
18. Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien A. P. Petitcolas, editor, *Information Hiding: 5th International Workshop, IH 2002*, pages 53–69, Noordwijkerhout, The Netherlands, October 2002. Springer-Verlag, LNCS 2578.
19. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go MIXes: Providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding: Second International Workshop, IH 1998*, pages 83–98, Portland, Oregon, USA, April 1998. Springer-Verlag, LNCS 1525.
20. Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies: 4th International Workshop, PET 2004*. Springer-Verlag, LNCS 3424, 2005.
21. Daryl McCullough. Specification for multi-level security and a hook-up property. In *Proceedings, 1987 IEEE Symposium on Security and Privacy*. IEEE, May 1987.
22. John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *IEEE Symposium on Research in Security and Privacy*. IEEE CS, May 1994.
23. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol - version 3. IETF Internet Draft, 2003.
24. Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In Pierangela Samarati and Paul Syverson, editors, *WPES'03: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 79–88, October, 2003. ACM Press.
25. Ira S. Moskowitz, Richard E. Newman, and Paul Syverson. Quasi-anonymous channels. In *IASTED International Conference on Communication, Network, and Information Security (CNIS 2003)*, pages 126–131. ACTA Press, December 2003.
26. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy,(IEEE S&P 2005) Proceedings*, pages 183–195. IEEE CS, May 2005.
27. Lasse Øverlier and Paul Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (S& P 2006), Proceedings*, pages 100–114. IEEE CS, May 2006.
28. Paul Syverson. Why I'm not an entropist. In *Seventeenth International Workshop on Security Protocols*. Springer-Verlag, LNCS, 2009. Forthcoming.
29. Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 514–524. IEEE CS, June 2005.