Low Latency Anonymity with Mix Rings

Matthew Burnside and Angelos D. Keromytis

Department of Computer Science, Columbia University
{mb,angelos}@cs.columbia.edu

Abstract. We introduce *mix rings*, a novel peer-to-peer mixnet architecture for anonymity that yields low-latency networking compared to existing mixnet architectures. A mix ring is a cycle of continuous-time mixes that uses carefully coordinated cover traffic and a simple fan-out mechanism to protect the initiator from timing analysis attacks. Key features of the mix ring architecture include decoupling path creation from data transfer, and a mechanism to vary the cover traffic rate over time to prevent bandwidth overuse. We analyze the architecture with respect to other peer-to-peer anonymity systems – onion routing and batching mixnets – and we use simulation to demonstrate performance advantages of nearly 40% over batching mixnets while protecting against a wider variety of adversaries than onion routing.

1 Introduction

In 1981, Chaum proposed a method for anonymous communication called a mixnet[3]. Chaum envisioned a network of proxies, called mixes, each with a well-known public key. In this system, when an *initiator* wishes to send a message anonymously to a *responder*, the initiator chooses a route through the mixes and wraps the message in a series of encrypted layers, one layer per mix in the route. Each layer contains next-hop information and the encrypted next-innermost layer; the layers are peeled off one by one as the message traverses the route to the responder. Chaum's original system has descendants in a wide variety of anonymity-providing systems, including systems for anonymous web surfing[18, 2], peer-to-peer anonymity[11], and network level anonymity in the form of onion routing[12, 10].

Onion routing, equivalent to Chaum's system as described thus far, yields low-latency connectivity but is vulnerable to timing analysis. An adversary observing all links in an onion routing network can record arrival and departure times for all messages in the network and use statistical methods to determine exactly who is communicating with whom[19, 17]. Mixnets go further and attempt to address this vulnerability by processing messages in batches to disguise timing correlations. However, each mix along a route must delay processing a batch until the batch is full, so, in practice, mixnet communication is generally slower than onion routing. The result is, broadly, two designs: one provides strong anonymity at the expense of performance, the other provides weaker anonymity with performance.

We introduce an intermediate solution, *mix rings*, that give better performance than mixnets while maintaining anonymity that is stronger than onion routing. The goal of a mix or onion router is to disguise the correspondence between the input and output messages that each relays. This goal is achieved through cryptographically transforming each message so an observer cannot link the messages and then, in the case of mixes, obscuring timing information either through batching or individually delaying the messages. The work presented here introduces a third mechanism for obscuring timing information that eliminates the need for batching (and the corresponding performance degredation that comes with batching) while maintaining the anonymity properties of the standard mixnet.

The intuition is that the batching process in a continuous-time mixnet inherently reduces performance, since a message has to be delayed at each mix until the batch is full before it is forwarded. Batching is used to protect against timing analysis, and we show that the mix ring architecture protects against the same attacks, while paying a reduced performance penalty.

We route cover messages in a ring of mixes that contains the initiator. Using mechanisms described later, the source (*i.e.*, the initiator) of these cover messages is hidden, so an outside observer only sees an unbroken stream of messages traversing the ring, with no obvious source or destination. When the initiator wants to send a message to the responder, the initiator replaces a cover message with a data message that "forks" (we will use the term *fan-out*) into two messages when it reaches an arbitrary point on the ring. One of the messages contains the data destined for the responder, the other is a cover message that continues around the ring. Since all traffic movement in the ring is coordinated, timing analysis can determine no more than that the message originated somewhere in the ring. We demonstrate that this system gives performance advantages of nearly 40% over batching mixnets, while providing a stronger form of anonymity than onion routing.

Note that there exists a form of mixnet, called a mix cascade, that similarly provides low latency traffic. A mix cascade consists of a dedicated set of servers that redirect traffic from a large set of users on a predefined route. There is considerable discussion in the field of anonymity research on the merits of mix cascades vs. peer-to-peer style anonymity systems; [9] contains an excellent summary of both sides of the discussion. Mix rings are a peer-to-peer network, and thus we compare them only with other peer-to-peer anonymity systems.

The rest of this paper is organized as follows: in Section 2, we discuss related work. In Section 3, we define our threat model and give a detailed explanation of our system. In Sections 4 and 5, we examine the anonymity and performance of mix rings. We conclude the paper in Section 6.

2 Related work

Systems based on onion routing include Tarzan[11], a peer-to-peer, IP-layer anonymizing network and Tor[10]. Tor extends onion routing by adding support for integrity protection, congestion control, and location-hidden services through rendezvous points. Batching mixes add protection from timing analysis attacks by pooling messages in batches. Mixes have been used in a wide variety of applications such as ISDN service[15], IP-layer infrastructure[14], and email[6].

Stop-and-go mixes[13], or *sg-mixes*, come from a subtype of mixes called *continuous-time* mixes. Messages in a continuous-time mixnet are delayed individually, rather than in batches. In *sg-mix* networks, message delays are randomly chosen from an exponential distribution within a window of estimated arrival time for that mix. Of course, this requires the initiator to predict traffic levels throughout the mixnet. Mix rings are closely related to *sg-mix*es but use constant rather than random delays so a principal does not have to predict network traffic levels. (There exist attacks on connection based continuous-time mixes that are based on observation of injected traffic[5], but these attacks do not apply to mix rings, as mix rings are not connection-based.)

Pipenet[4] is effectively a mix network with the addition of pairwise cover traffic between all mixes. This network provides strong theoretical anonymity but is impractical due to its considerable bandwidth requirements. Mix rings are similar to Pipenet but we reduce bandwidth requirements through mix ring's ability to increase and decrease cover traffic on demand.

Pfitzmann, *et al.*, consider anonymity in a ring topology in the context of local area networks [16] and describe such a system as being both fault tolerant and efficient. Mix rings may be considered an extension of this work into the inter-network realm. Other ring-related work includes [1] which uses tokens to carry messages on routes that they compare to public transportation bus routes.

Danezis, *et al.* route dummy messages in a ring through an anonymity system in what they call heartbeats [8]. Using heartbeats, an anonymity system can detect some forms of tampering on the network, including an adversaries who might speed up or slow down network links, or block them entirely. Mix rings inherently provide the same protections, since every cover traffic message in the ring can be viewed as a heartbeat.

3 Design and operation

We begin our discussion by describing our threat model and the design goals for the system, then move on to the design of the system.

3.1 Threat model

We consider the broad categories of adversaries first described in [18]: a local eavesdropper (*e.g.*, a malicious first-hop router), one or more malicious mixes, a malicious responder, and finally a global adversary who can observe and modify all network links. We further assume that the global adversary can compromise some, but not all, mixes in the network. We assume no adversary can reverse cryptographic transformations.

3.2 Design goals

Our broad goal is to design a peer-to-peer system that finds a compromise between the (relatively) weak anonymity/low latency of onion routing and the strong anonymity/high latency of mixnets. Thus, the specific goals of the architecture proposed in this paper are to provide the flexibility of a peer-to-peer anonymity system, while demonstrating anonymity comparable to that of a mixnet, but with lower latency than a mixnet. We specifically consider latency rather than bandwidth due to the batching nature of mixnets; the batching process affects latency but does not inherently affect bandwidth.

3.3 System model

The basic mix ring infrastructure is a set of continuous-time mixes, each with a well-known public key. (Note that in continuous-time mixes, messages are delayed individually, rather than in batches.) The initiator must be a mix, but the responder need not be.

When a mix receives a message, it decrypts the message to obtain next-hop information, a set of options, and a recursively defined body. (See Section 3.4 for more details on the structure of a message.) The mix performs any operations specified by the options, pads the body out to the original length of the message, and then forwards it on to the specified next hop.

Consider an initiator who wants anonymous communication with a particular responder. This initiator chooses a random set of mixes, logically organizes them into a ring, and negotiates session keys with each. The initiator then uses the method described in Section 3.5 (below) to anonymously start routing cover traffic around the ring.

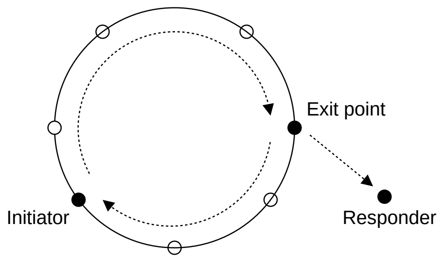

Fig. 1. The initiator routes both cover traffic and data traffic around the ring. The data traffic leaves the ring through a randomly chosen exit point.

The initiator sends a message to the responder, outside the ring, by constructing a special message that fans out when it reaches a randomly chosen exit point on the ring. That is, when the message is processed by the exit point, the processing generates two messages: one message destined for the responder and one message that continues along the ring, indistinguishable from other cover traffic. The result is that the total number of cover messages circling the ring is unchanged, and from an adversary's point of view, each member of the ring is equally likely to have been the source of the message. (See Figure 1.)

3.4 Mix ring messages

Messages in mix rings are similar to other forms of onion-style messaging, *e.g.*, Minx[7]. Each message has the following form:

$$\{T, H_1, B_1, [H_2, B_2], P\}$$

A message consists of a set of options T (defined below), a header H_1 containing the next-hop IP address and port, and a recursively defined body B_1 encrypted using the next hop's session key. The message may contain an optional second header and body, H_2 and B_2, in the case of a fan-out message. Finally, as each layer of the message is removed, padding P is appended so the total message length remains constant.

We define the following options T on a message:

set-rate A *set-rate* option has a field that indicates the rate at which the mix should forward every subsequent message.

delay A *delay* option instructs a mix to wait for some time period d before processing the rest of the message, including any other options that may be set.

fan-out A message with the *fan-out* option set contains in its body two next-hop messages, H_1, B_1 and H_2, B_2. Typically, one of these next-hop messages is standard cover traffic which is forwarded to the next hop in the ring; the other is routed to the responder.

Each option may be set in any of the encrypted layers in a message, and a single layer may have multiple options. Note that *set-rate* applies to all future messages received by the mix, whereas *delay* is specific to the message layer containing that option.

3.5 Constructing the ring

An initiator μ_0 uses a trusted, well-known, directory server (similar to the Tor[10] directory server) to arbitrarily select a set of mixes μ_1, μ_2, ..., μ_n that will comprise the ring. The initiator then selects a second set of mixes ν_1, ν_2, ..., ν_n which form a tunnel through the mixes to μ_i, an arbitary mix in μ. Through this tunnel, the initiator negotiates a session key with each mix in the ring be used for encrypting the layers of each message (see Figure 2).

Note that when a mix is sent a message with no option set (as are the mixes in the tunnel) it behaves as a standard continuous-time mix. The purpose of the tunnel at startup is to protect against a local adversary (an adversary

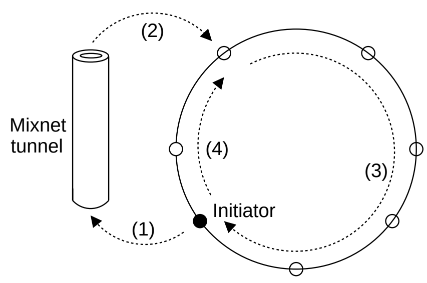

Fig. 2. Injecting cover traffic into the ring. The initiator tunnels all cover traffic (1) to an arbitrarily chosen mix on the ring (2), from which the traffic is routed around the ring (3). When the traffic returns to the initiator, it is replaced with cover traffic routed directly through the the ring (4), rather than through the tunnel.

situated on the initiator's local network) who can otherwise observe the initiator constructing the ring. See Section 4.1 for further discussion.

Now with a session key for each mix in the ring, the initiator begins transmitting cover traffic at some rate r, routed through the tunnel and around the ring, destined for the initiator itself. The first of these messages has the *set-rate* option defined on each of its layers, instructing each mix on the default rate r to transmit each subsequent message. As each cover message returns to the initiator, it is seamlessly replaced with a new cover message routed through μ_1 rather than the tunnel. When the desired number of cover messages has entered the ring, the initiator closes the tunnel. The desired number of cover messages is a factor of the bandwidth requirements of the initiatior and the bandwidth limitations of the links on the ring itself. It can be as small as one, or as large as the network infrastructure of the ring can handle.

Reiter *et al.* showed that all anonymity systems must reset periodically to allow new nodes to join[18, 21]. We use the standard network time protocol (NTP) to coordinate periodic resets across the mix ring system for this purpose. We assume that, after each reset, every participating mix builds a new ring. This protects from an adversary observing the initiator's local network who might otherwise be able to identify the initiator by observing which mix is the first to send traffic (since this would have to be the initiator). With the above method, that same adversary can only determine that the initiator is a participant in one or more rings, but not that he or she is a given ring's creator.

At this point, the construction of the ring is complete. The cover traffic is flowing around the ring, each message advancing at a rate of r. This is the default behavior of the ring when no message is being sent.

3.6 Default behavior

The default behavior occurs when no message is being sent. At some rate r the initiator sends cover messages through the ring. As each cover message traverses the ring and returns to the initiator, it is seamlessly replaced with a new one. If a cover message fails to arrive within some window of its appointed time, or if the messages arrive out of order, the initiator considers such messages lost and does *not* transmit a new cover message; instead it uses the method described in Section 3.7 to refill these gaps in the cover traffic.

If the initiator filled each gap as it appeared, an adversary could drop one or more cover messages and then observe at what point the messages were replaced; whichever mix replaced the messages would be the initiator. Thus, we use another method, described below.

3.7 Filling gaps in the cover traffic

When the initiator discovers a gap in the cover traffic, it cannot simply insert a new cover message because it would be clear to an adversary that the mix which filled the gap was the initiator. Instead, the initiator chooses a random mix μ_j on the ring and instructs *that* mix to fill the gap. This instruction is performed using tools already described: on the gap's second trip around the ring, the initiator replaces the message immediately preceding it with a message constructed in the following manner. In the message layer corresponding with μ_j, the initiator sets the *fan-out* flag. The message headers H_1 and H_2 have the same next-hop destination (the next mix in the ring) but each is set with a different *delay* value, one $d = r$ and one $d = 2r$, such that the gap is filled when the second message is relayed. For larger gaps, this procedure is repeated as necessary.

3.8 Varying the traffic rate

Under normal circumstances, the traffic rate in the ring is set to a low volume to avoid consuming unnecessary bandwidth. Only when the initiator wishes to send traffic is the rate increased. When the initiator wishes to increase or decrease the traffic rate, it constructs a message to each mix in the ring with carefully calculated *delay* and *set-rate* values such that all mixes in the ring execute the *set-rate* at approximately the same time (this instruction actually takes the form of a single message, with each layer in the message instructing that particular hop on its rate and delay values). Perfect coordination is not required so long as the initiator's rate change is not statistically unique with respect to the rate changes of the other mixes.

Over time, due either to properties of the network or actions of an adversary, the cover traffic may become unevenly distributed around the ring. That is, the traffic may become grouped together in "clumps." We do not consider it further here, but in future versions of the system, the initiator may counter this action through judicious use of the *delay* option.

3.9 Sending a message

To send a message, the initiator chooses a random member of the ring μ_e as the exit point for that message. It then constructs a message which is identical to the other cover messages except that the layer of the message destined for μ_e has the *fan-out* option set. One branch of the fan out, the data, is destined for the responder; the other, cover traffic, is routed to the next hop on the ring. The initiator can vary the ratio of data to cover messages to increase or decrease the traffic bandwidth.

3.10 Receiving a message

The current mix rings implementation does not include a mechanism for a return channel. However, we include in this section a draft for this mechanism which will be built in future work. Briefly, the initiator includes in the message a reply block which the responder may use to send messages back to the initiator. The reply block is essentially an empty onion, designed to travel along the return route of the ring. The responder fills in the body of the message and sends it to the first hop of the return path (*i.e.*, the exit point on the ring). When the initiator receives the message, the return path is complete, but the initiator must generate a new cover message that continues along the ring in place of the return message. Otherwise an adversary could simply observe the path of return message to identify the initiator.

4 Analysis

In this section, we analyze the mix ring architecture by comparing it with onion routing and mixnet architectures. We show performance advantages of nearly 40% over batching mixnets and demonstrate a stronger form of anonymity than strict onion routing.

Table 1. Comparison of initiator exposure in onion routing, mix rings, and mixnets. The columns represent, respectively, a malicious first-hop router, a malicious intermediate mix or set of mixes, and a malicious responder.

	Local eavesdropper	Bad mixes	End host
Onion routing	Exposed	No	No
Mix rings	Possibly	No	No
Mixnets	Possibly	No	No

Table 1 gives a comparison of the three architectures with respect to various adversaries, each of which will be analyzed further in the remainder of this section. We examine attack vectors by breaking them into three classes: adversaries at or near the initiator (a local eavesdropper), adversaries in the middle of the network (compromised mixes), and adversaries at or near the responder. We then consider some specific attacks against the cover traffic in the ring, and attacks by a global adversary.

4.1 Attacks from a local eavesdropper

In onion routing, an initiator is immediately exposed to an adversary of this type, since the adversary can observe the construction of the onion route. In mixnets, the adversary can detect initiator activity, but generally cannot distinguish whether the initiator is truly an initiator or simply an intermediate node on a mix route. Initiator activity in mix rings is identical to that of mixnets – the initiator is simply building up mix routes, and thus the initiator is no more or less exposed than an initiator in a mixnet.

4.2 Attacks from compromised mixes

In both onion routing and mixnets, a malicious intermediate mix can identify its predecessor and successor on a given route. Also, if the responder is a node outside the network and the compromised mix is the last hop on the route before the responder, the adversary is trivially able identify the responder.

Similarly, a malicious mix in a ring can identify its predecessor and successor. It has knowlege of the traffic rate in the ring, and whether or not it is an exit point. If the mix is an exit point, it can identity the responder.

When a compromised mix knows it is on a route of interest, it can record its predecessor on that route. Since the initiator has to be on the route between itself and the responder after every reset, while all other mixes will only be on that route with some probability less than one, over time the initiator will be the predecessor of the compromised mix more often than any other mix. This attack is called the predecessor attack and was first proposed by Reiter and Rubin[18].

Of course, the compromised mix must identify when it is in a route of interest; this is only achieved when multiple compromised mixes are on the same route. In onion routing, if a compromised mix is the exit point for a route, it uses timing analysis to identify whether any of its collaborators are also on that route. If there is one, then the collaborator records its predecessor. In this case, Wright, *et al.*, showed that after $O(n^2 \ln n)$ resets, the adversary can with high probability identify the initiator[20].

The mixnet attack is similar, but requires all mixes in a route of length l be compromised before recording a predecessor data point. Wright showed that mixnets survive for $O(n^l \ln n)$ resets before the adversary can identify the initiator with high probability .

Under the predecessor attack, mix rings behave as mixnets. The portion of the ring past the fan-out is irrelevant since the attack takes place over the portion of the ring between the initiator and the exit point. This is a segment of the ring and a segment of the ring is also a mix route. If the distance from initiator to exit point is i, then the initiator in a mix ring is compromised after $O(n^i \ln n)$ resets.

4.3 Attacks from the responder

Consider a ring of circumference $2n$ and an onion or mix route of length n. (These are comparable since, on average, assuming randomly chosen exit points, a message will traverse n mixes before it reaches the exit point and leaves the ring.) A malicious responder in a standard onion or mix route must unroll the route, compromising each of the n mixes in turn, to identify the initiator. Similarly, a malicious responder in a mix ring must, on average, compromise n mixes from the exit point back to the initiator to make positive indentification. Mix rings, mixnets and onion routing are equally strong against attacks of this form from the responder.

4.4 Attacking the cover traffic

Cover traffic is a key difference between mix rings and mixnets. The cover traffic in a mix ring is cryptographically transformed in the same fashion as the data-carrying traffic. An adversary is therefore unlikely to be able to distinguish cover traffic from data traffic without reversing the crytographic transforms. The adversary can, however, attempt to inject new cover traffic, or drop, delay, modify or replay.

New cover traffic introduced by the adversary will be detected and dropped at its first hop since we assume only the initiator possesses the session key required to construct a valid message. However, if the adversary cuts an arbitrary link for a short time but does not drain the cover traffic completely, then the mechanism described in Section 3.7 is sufficient to fill any gaps. Meanwhile, the remaining cover traffic is still useful for carrying data. If the adversary drains the cover traffic completely, the ring must be reconstructed as in Section 3.5. This is effectively a denial of service.

If, instead, the adversary attempts to delay the traffic (this may also occur due to properties of the underlying network) the cover traffic may become unevenly distributed around the ring. This is handled as described in Section 3.8.

4.5 Attacks from a global adversary

Similar to mix nets and onion routing, a global adversary is able to observe the construction of the ring, and this compromises the initiator from the start.

4.6 Further analysis

An important consideration in mix rings is that there must be members of the community willing to operate nodes in a mix ring in which they, themselves, are not the initiator. Given that there are members of the community who have shown willingness to operate Tor and Mixminion nodes, for example, we believe it a reasonable assumption that like-minded individuals would also be willing to operate nodes in a mix ring.

5 Performance

To compare the performance of onion routing, mix rings, and mixnets, we designed and deployed a simulator on PlanetLab, a worldwide experimental overlay network. The simulator is designed to emulate onion routing, mix rings, and

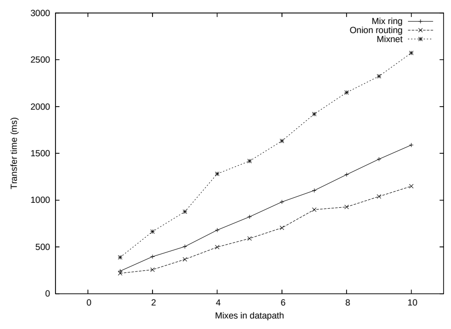

Fig. 3. Transfer time for a 50kb file over varying route lengths.

mixnets. It is important to note that we only consider peer-to-peer anonymity systems; we do not consider mix cascades. The simulator is a daemon that runs on each node. At startup, the daemon is configured to operate as a node in an onion routing system, a mix ring system, or a mixnet system. The difference between these modes of operation consist primarily of batching for mixnets and handling the message options for mix rings. More specifically, the daemon behaves as follows.

– When configured as an onion-router, the daemon listens for messages on a pre-defined port. On receipt of a message m, the daemon decrypts the message using its private key to obtain next-hop information. The message is then immediately forwarded to the next hop.
– When configured as a router in a mix net, the daemon collects messages arriving on a pre-defined port until the specified number of messages has arrived (the batch size). Then each message is decrypted to obtain next-hop information and forwarded onwawrd.
– Finally, when configured as a node in a mix ring, the daemon behaves as an onion router, but obeys any options that may be set in the message (see Section 3.4).

We deployed the simulator on approximately 30 PlanetLab nodes. For testing mixnets and onion routing, we chose six nodes at random to behave as onion routers or mixes, respectively. In the case of mix rings, we chose twelve of the nodes at random, since in a mix ring a one-way message only traverses half the ring, on average. We configured the mixnet with a batch size of 10 messages; we configured the mix ring with a default cover traffic rate of 10ms between messages. Each message is 8192 bytes.

In each of the following tests, we chose one node as the initiator and a node outside the route or ring as the responder. The measurements do not take into account interactions with other users, nor do they compare startup times; these will be the subject of future analysis.

Figure 3 compares the transfer time of a 50kb file under each of the three schemes as the length of the route increases. In the case of the mix ring, the exit point on the mix ring was pegged at the $\frac{n}{2}$ node. It is clear that, while the mix ring does not outperform onion routing, it has considerably lower latency than the mixnet, on the order of 40% improvement for a given path length.

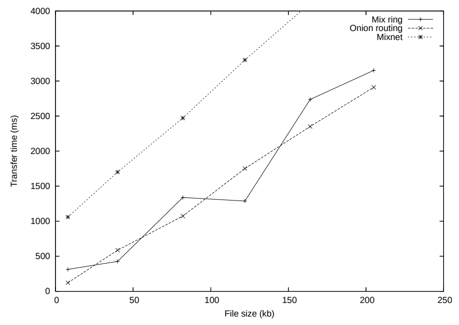

Fig. 4. Transfer time as a function of file size in a 6-node mix route, 6-node onion route, and 12-node ring.

Figure 4 compares the three schemes as the file size increases. In this case, the mix ring exit point is chosen randomly, so the variance of the mix ring plot has increased due to the randomly chosen exit points. However, the trend is still clear: mix ring performance is considerably better than mixnets and approaches that of onion routing.

6 Conclusion

This paper presents mix rings, an anonymity system stronger than onion routing, with better performance than mixnets. Mix rings are a compromise between the weaker anonymity/low latency of onion routing and the strong anonymity/high latency of mixnets. Mix rings use cover traffic to protect against timing analysis and provide a mechanism to vary the cover traffic rate to prevent bandwidth overuse. We show that mix rings provide anonymity that is stronger than onion rings, and comparable to mixnets. For file transfers, we show that mix rings exhibit nearly 40% performance improvements over mixnets. This work makes strong anonymity a possibility for low-latency applications that may not have previously had the option available.

In future work, we will consider bidirectional traffic, perhaps using reply blocks. We will also give futher study to the bandwidth interactions between multiple rings.

7 Acknowledgments

This research was supported by NSF grant NSF CCF-05-41093.

References

[1] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1):25–39, 2003.

[2] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[3] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[4] W. Dai. Pipenet 1.1. Usenet post, August 1996.

[5] G. Danezis. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2004)*, LNCS, May 2004.

[6] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[7] G. Danezis and B. Laurie. Minx: A simple and efficient anonymous packet format. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.

[8] G. Danezis and L. 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.

[9] C. Díaz, G. Danezis, C. Grothoff, A. Pfitzmann, and P. F. Syverson. Panel discussion - mix cascades versus peer-to-peer: Is one concept superior? In *Privacy Enhancing Technologies*, pages 242–242, 2004.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[11] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[12] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[13] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[14] The NymIP Effort. http://nymip.velvet.com.

[15] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.

[16] A. Pfitzmann and M. Waidner. Networks without user observability – design options. In *Proceedings of EUROCRYPT 1985*. Springer-Verlag, LNCS 219, 1985.

[17] J.-F. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[18] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[19] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003*, October 2003.

[20] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium (NDSS '02)*. IEEE, February 2002.

[21] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.