Preventing Active Timing Attacks in Low-Latency Anonymous Communication
[Extended Abstract]

Joan Feigenbaum[1⋆], Aaron Johnson[2⋆⋆], and Paul Syverson[3⋆⋆⋆]

[1] Yale University Joan.Feigenbaum@yale.edu
[2] The University of Texas at Austin ajohnson@cs.utexas.edu
[3] Naval Research Laboratory syverson@itd.nrl.navy.mil

Abstract. Low-latency anonymous communication protocols in general, and the popular onion-routing protocol in particular, are broken against simple timing attacks. While there have been few proposed solutions to this problem when the adversary is active, several padding schemes have been proposed to defend against a passive adversary that just observes timing patterns. Unfortunately active adversaries can break padding schemes by inserting delays and dropping messages.
We present a protocol that provides anonymity against an active adversary by using a black-box padding scheme that is effective against a passive adversary. Our protocol reduces, in some sense, providing anonymous communication against active attacks to providing a padding scheme against passive attacks.
Our analytical results show that anonymity can be made arbitrarily good at the cost of some added latency and required bandwidth. We also perform measurements on the Tor network to estimate the real-world performance of our protocol, showing that the added delay is not excessive.

1 Introduction

Anonymous communication protocols are designed primarily to allow *users* to communicate with *destinations* anonymously. They face, however, the challenge of optimizing over several competing criteria: anonymity, latency, and bandwidth. High latency and limited bandwidth are unacceptable for many popular Internet applications, and onion routing [12], despite its vulnerability to correlation attacks, has become a successful protocol for anonymous communication on the Internet. To design a useful protocol, we focus on providing better anonymity than onion routing while maintaining acceptable latency and bandwidth.

Low-latency protocols in general have been vulnerable to several attacks based on the timing of events in the system. Typically, the user in these protocols chooses a set of *routers* to mediate between the user and the destination,

⋆ Supported in part by NSF grants 0331548 and 0716223 and IARPA grant FA8750-07-0031.
⋆⋆ Supported by NSF grant 0716223.
⋆⋆⋆ Supported by ONR.

forwarding data between the two and obscuring their relationship. The essential problem is that timing patterns in these data are conserved between the source and destination. Therefore an adversary only needs to observe the incoming *stream* of data (the consecutive messages exchanged during a communication session), from the user and the outgoing stream of data to the destination to use patterns to link the two. In a *passive* timing attack, an adversary relies on timing patterns that are generated by the user. Because the user creates these patterns, he can prevent this attack by adding dummy packets and delays into the stream to make his traffic look similar to the traffic of other users [27]. However, the adversary can defeat this by performing an *active* attack, in which he inserts timing patterns into the traffic as it passes through routers under his control.

As a result of this sort of active attack, existing low-latency anonymity protocols do not provide anonymity when the adversary controls the routers that the user communicates with directly and the routers that the destination communicates with directly. Suppose the adversary controls a fraction b of the network. In onion routing, users select routers uniformly at random, and the adversary compromises anonymity with probability b^2.

This probability is fixed and cannot be improved by trading off performance elsewhere, and it can be quite insufficient. Consider Tor [7], the popular implementation of onion-routing and the associated volunteer network. In Tor, a user sends a message over a sequence of routers he sets up in advance called a *circuit*. Suppose the adversary runs just two routers. If we take into account the way Tor chooses circuits, the size of the network [28], and the number of users observed on Tor in one day [17], we expect the adversary to compromise 15 users at least once in that day. If the adversary provides the top two routers by bandwidth, the expected number of compromised users increases to 9464.[1] Thus, the system provides poor anonymity against a wide variety of realistic opponents, such as governments, ISPs, and criminals willing to purchase the use of botnets.

We consider the very weak anonymity provided by low-latency protocols against an active adversary to be a fundamental and critical challenge in anonymous communication. In this paper, we present a low-latency protocol that pro-

[1] Roughly, circuits are selected in Tor as follows: the first hop is chosen from a set of *guard* routers, the second hop is chosen from the entire network, and the third and final hop is chosen from a set of *exit* routers. As of April 2010, the Tor network consists of around 1500 routers, of which around 250 are guard routers and around 500 are exit routers. Suppose that the adversary runs one guard router and one exit router. McCoy et al. observed 7571 unique clients while running a guard router for one day. The expected number of these that would lose anonymity is $7571/500 = 15.142$. Moreover, Tor weights by bandwidth, and so suppose that the adversarial routers are the top two by bandwidth. McCoy et al. observed that the top 2% of routers transported about 50% of the traffic. Then, very roughly, the probability of choosing the adversary in a guard set would increase from 1/250 to .5/(250*.02), and so the expected number of users observed would be 25*7571=189275. By similar rough approximation, for every circuit, the adversary's exit router would be selected with probability .5/(500*.02)=.05, and so the expected number of deanonymized users would be 189275*.05=9463.75.

vides arbitrarily good anonymity against an adversary that can observe and create timing patterns. The protocol makes black-box use of a padding scheme to prevent passive timing attacks. Several padding schemes that defeat passive timing attacks have been proposed [27, 25, 30], and furthermore we believe that there is still potential for substantial improvement. The protocol provides two-way stream communication.

A two-way protocol requires different defenses, depending on the direction of communication, because of an asymmetry in the communication between the user and destination. The user can talk directly to many routers and will add padding correctly. We will require that the destination communicate with just one router, and that router can't be trusted to pad the stream correctly. As a result, our protocol uses a somewhat different scheme for traffic on the way *from* the user as on the way *to* the user. The essential features of our solution are

1. Packets have timestamps with their intended send time.
2. Packets from the user to the destination are sent in several copies over a *layered mesh* topology . This balances limiting view of the stream to a small number of routers while providing redundancy against malicious delays
3. Packets from the destination to the user are sent over a path that performs in-stream padding.

For simplicity, we describe forming the layered mesh as a *cascade*: a fixed arrangement of routers that all users use to send data. The biggest drawback to using cascades is that the resource constraints of the cascade routers obviously limit the number of feasible users and therefore limit anonymity. Another drawback is that cascades make long-term intersection attacks easier because only two known endpoints need to be watched. Giving users freedom to choose the meshes, analogous to *free routes*, is an important future extension to our scheme.

We evaluate the anonymity provided by our protocol in a network model that incorporates timing and an active adversary. The theoretical results suggest that the approach has good asymptotic efficiency and that a promising next step is to optimize within the framework of the scheme we describe. Moreover, because we are concerned with eventual practicality, we do measure a component of the system over which we have no control and which could have made our protocol unusable, delay variations in the host network. Specifically, we measured the likely latency costs of running our protocol on the existing Tor [7] network. This provides a strenuous, real-world scenario to evaluate our protocol's performance.

Our results are therefore a mix of the theoretical and experimental:

1. We show that the user can be made anonymous with arbitrarily high probability as long as b is less than $1/2$. The user is anonymous within the set of users with identical traffic patterns as produced by the input padding scheme.
2. We prove that our mesh topology in the forward direction is optimal for anonymity in a limit sense.
3. The latency of our protocol is proportional to the length of the mesh and return path. We show that the probability of compromise decreases to zero

polynomially in that length. This compares well with onion routing, which adds latency proportional to its path length.
4. The bandwidth used is $2w + (l-1)w^2 + 1$, where l is the mesh/path length, and $w = \Theta(\log l)$. This compares well asymptotically to the $l+2$ copies that are sent in an onion-routing path of the same total length.
5. For most of our measurements, we observe that added packet delay would need to be less than a factor of two to achieve satisfactory reliability.

The results suggest that our approach indeed has the potential to mitigate active timing attacks. Our results are presented here without proofs and detailed measurement procedures because of space limitations. Please see our Technical Report [9] for these details.

2 Related work

Timing attacks are a major challenge in low-latency anonymous communication [16]. They have been observed in some of the earliest low-latency systems [1], including initial versions of onion routing [12]. These attacks are also closely related to traffic analysis in mix networks [24].

In a passive timing attack, the adversary observes timing patterns in a network flow, and then correlates them with patterns in other traffic that it observes. If the adversary is able to observe both the user and the destination, he can thereby link the two. The ability of the adversary to perform this correlation has been experimentally demonstrated several times [32, 16, 23, 21, 2].

A solution to passive timing attacks is to get rid of identifying patterns in the traffic by padding and delaying it. The drawbacks to such an approach are added latency and bandwidth overhead. Our protocol relies on the existence of some acceptable and effective padding scheme. Constant-rate padding, in which traffic is sent at a constant rate by filling in the gaps with dummy packets, is probably the most obvious such scheme. It has appeared multiple times in the literature [27, 11, 16]. Levine et al. [16] propose a "defensive dropping" mechanism which adds dummy packets at the start of the circuit and drops them at various routers before the end. This reduces the correlation between any patterns in the incoming streams and patterns in the outgoing streams. Shmatikov and Wang [25] propose a variable-rate padding scheme. In their scheme, packets from the user are forwarded with little delay, and dummy packets are added by the intermediate routers according to a probability distribution on the packet inter-arrival times. Wang et al. [30] describe a link-padding scheme for low-latency systems, but their system is designed for a situation in which the adversary is not active and the destination participates in the protocol. This situation does not reflect the kind of Internet communications that have proven useful and that we target.

All of these schemes are vulnerable to an adversary that actively delays packets from the user. Yu et al. [31] show that this can be done in a way that makes it difficult for the user or the system to detect that the attack is occurring. One approach to this problem is to change the timing patterns within the network

by adding, dropping, or delaying packets ([25, 16]). However, dropping or delaying packets can't hide very long delays without adding unacceptable latency or bandwidth overhead. Adding dummy packets can, in our case, be detected by the final router, and therefore do not help. A final router can detect them because, in our case, the destination does not participate in the protocol; there must be one last router in the system that provides the point of contact to the destination. Moreover, Wang et al. [29] experimentally show that many such schemes for internal traffic shaping are still vulnerable to an active timing attack.

Simply delaying packets that pass directly through adversarial routers isn't the only active timing attack that has been demonstrated. Murdoch and Danezis [20] show that in onion routing the adversary can actively add delay patterns to the data by sending bursts of traffic through a router. This can be used to determine the routers on a given circuit. Fu et al. [10] describe how the presence of a flow on a router can also be determined by ping times to the router. Borisov et al. [4] look at the case that the adversary doesn't just delay, but drops packets in a denial-of-service (DoS) attack aimed at forcing users to move to circuits that the adversary can deanonymize. Such an attack was also discussed by Dingledine et al. [8]. We do not address such attacks in this paper, and they are outside of our model.

A related timing attack by Hopper et al. [13] uses congestion to exploit different network latencies between hosts. They show that the latencies from multiple hosts to a user can be very identifying. The user in our protocol communicates with several routers as a first hop, and in light of this attack we took care not to allow the adversary to infer these latencies.

One key feature of our protocol is the use of a layered mesh to provide redundancy. The use of redundancy for several purposes has been also explored in previous protocols. Syverson [26] suggests using router "twins," pairs of routers that share the same key, to provide a backup in case a router fails. Two redundancy schemes to manage failures, K-Onion and Hydra-Onion, are proposed by Iwanik et al. [14]. Redundancy to support failure is not our goal, and such schemes are in some ways complementary to our own. However, the redundancy in our protocol does protect against honest node failures as well as malicious ones. Nambiar and Wright [22] use redundancy in the host lookup of Salsa to protect against route capture. Interestingly, an analysis by Mittal and Borisov [19] of this technique uncovers the tradeoff between preventing active and passive attacks that we face as well.

Another critical feature of our protocol is the use of explicit timing to coordinate the users and routers. This is similar to the timing instructions of Stop-and-Go mixes [15]. Such mixes are given a time window within which the packets must arrive, and they delay forwarding by an exponentially-distributed amount of time. Although the techniques are similar, this scheme is designed for mix networks and not stream communication, and this scheme does give the adversary some slack time within which a timing signature could possibly be placed. Moreover, the lack of any redundancy means that any slowdowns within the network, even of benign origin, can quite easily kill a connection.

The timing approach we take is also similar to the use of synchronicity in mix networks by Dingledine et al. [8]. They describe synchronous batching on free routes and show that it typically provides better anonymity than cascades. Our scheme can be viewed as low-latency synchronous batching.

3 Model

We will express and analyze our anonymity protocol in a model of network and adversary behavior. A particular advantage of this approach is the ability to make convincing guarantees of security when we cannot predict the tactics that an adversary will use.

3.1 Network

Let the network consist of a set of onion routers R, a user population U, and a set of destinations D. The network is completely connected, in that every host can send a message directly to every other host. Each event in the network occurs at some global time. We assume that each user and router has a local clock that accurately measures time, and that these clocks have run some sort of synchronization protocol [18]. Let δ_{sync} be the largest difference between two synchronized clocks in the network.

There is some probabilistic network delay $d_{net}(r, s)$ between every pair (r, s) of routers. This models the unpredictable delay between routers due to factors such as route congestion and route instability. There is also some probabilistic processing delay $d_{proc}(r)$ at every router r. This reflects changes in delay at a router due to local resource contention among anonymous messages and among multiple processes. The delay of a message is the sum of delays drawn independently from these two distributions. We also let the delay of one message be independent of the delays of the other messages. We assume the distributions of both sources of delay is known to the system. In practice, this may be achievable via network measurements.

We assume that all hosts (in particular, all destinations) respond to a simple connection protocol. One host h_1 begins the connection by sending to another host h_2 the pair $< n, M >$, where $n \in N^+$ is a number and M is a message. Any responses M' from h_2 are sent back as $< n, M' >$. Once established, connections in the anonymity protocol cannot be closed by anyone to prevent distinctions of one from another by open and close times. All connections thus stay open for a fixed amount of time and then close automatically. Application connections running over these can of course be closed by the ultimate source and destination; although this will not close the anonymity circuit, and padding messages will continue to be sent until connection timeout.

3.2 Users

User communication drives the operation of the anonymity network. We view the communication of a user as a sequence of connections. Each connection is

to one destination, and it includes messages to and from the destination. 'User' refers to both human users and software running on their behalf.

3.3 Adversary

The adversary controls some subset $A \subseteq R$ of the routers, where $b = |A|/|R|$. It seems plausible that an adversary can run routers that are at least as fast as the other routers on the network, and that it may dedicate them to running the protocol. Therefore, in contrast to non-adversarial routers, we pessimistically assume that the adversary has complete control over the processing delay $d_{proc}(a)$ of routers $a \in A$. If needed, we reflect compromise of links between routers by the compromise of an adjacent router.

3.4 Padding scheme

The padding scheme $\mathcal{P}$ is a black box that initially takes as input a connection start time and outputs the timing of the return traffic. Then every time step it takes the presence of data from the user and returns whether or not a packet should be sent. Note that this requires the scheme to determine in advance the length of the connection. Let S_u be the set of users that start connections at the same time as user u and have the same traffic pattern. Our proposed protocol relies on the effectiveness of the padding scheme. At best, it makes u indistinguishable within the set S_u supplied by $\mathcal{P}$.

Some of the padding schemes previously proposed in the literature can provide the black box $\mathcal{P}$. For example, to use basic constant-rate padding, in which packets get sent at constant rate in both directions, we simply need to choose a fixed length for the connection when it starts. This approach typically causes high added latency and/or message overhead, however. As another example, the padding scheme of Shmatikov and Wang [25] could be used by fixing the connection length and return scheme. In this padding scheme, inter-packet delays are sampled from a distribution, which is adjusted if a packet arrives early. Dummy packets are sent after the sampled delay, and real packets from the user are sent immediately. In the direction from the user, this scheme could be used directly. In the return direction, we could just skip shifting the distribution for early packets. Then we would send each return router the same sequence of random bits to use in sampling the distribution. Alternatively, we could relax our requirements for the return scheme and allow it to be updated periodically. The user could then use the forward mesh to update the distribution of packet arrival times.

It is not hard to conceive of novel padding schemes that might satisfy these requirements, although getting a good mix of anonymity and performance certainly does not seem easy.

4 Problem

The problem in this model is to design an anonymity protocol that supports the low-latency, two-way, stream communication that has made Tor [7] popular.

In order to allow communication with hosts that are ignorant of the protocol, we require that only one host communicates with the final destination, and that the communication is only the original messages generated by the user. We evaluate our protocol by three criteria: anonymity, latency, and the amount of data transferred. We evaluate the anonymity in our protocol according to its *relationship anonymity*, that is, the extent to which it prevents an adversary from determining which user-destination pairs are communicating. For latency, we consider the amount of time it takes for a message to reach the destination from a user. For the amount of data transferred, we consider the total amount of data that to be transferred during a single user connection.

5 A Time-stamping Solution

The padding scheme gives us sets of users that have traffic streams with identical timing patterns. However, the model we have described gives the adversary the ability to modify these patterns as the traffic travels through its routers towards the destination. To prevent this, we try to enforce the desired timing pattern on packets sent by including the times that the routers should forward them. Any honest node that receives the packet will obey the instructions, removing any delays inserted by the adversary. For traffic sent from the user to the destination we can trust the user to correctly encode the padding-scheme times. Traffic sent from the destination to the user must, by our requirements, pass initially through a single router. Because it may be compromised, we cannot trust it to use a proper padding scheme. However, this traffic is destined for an anonymity-protocol participant, the user; therefore, unlike traffic from the user, we can destroy inserted timing patterns by re-padding it all the way to the user. Observe that re-padding does not work for traffic from the user, because the final router sees which packets are real and which are padding.

5.1 From the user

First, consider what we could do if propagation and processing delays were deterministic. The user could send through a path in the network a layered data structure called an *onion* which, for each packet, includes in the ith layer the time that the onion should arrive at the ith router. Then each router on the path could unwrap the onion to make sure that the initial timing sequence was being preserved and, if so, forward the onion.

Unfortunately, in real networks, delays are somewhat unpredictable. For example, an onion might be delayed by congestion in the underlying network. However, if the distribution of delays is known, we know how long we need to wait at a router for onions to arrive with any fixed probability. We will set that probability to balance decreasing added latency with decreasing the chance of a successful timing attack. Then we add in this buffer time to the send time.

Another problem is that the adversary could drop onions entirely in a pattern the propagates down the path. Our approach to this problem is to send multiple

copies of an onion down redundant, intersecting paths. A router on a path needs only one copy of the onion to arrive in time from any incoming path in order to forward it by its send time.

This approach has limits, because each redundant router adds another chance for the adversary to observe an entire path from source to destination. For example, suppose that we simply send onions over k paths of length l that intersect at a final router, where every router is chosen uniformly at random. Let b be the fraction of routers that are controlled by the adversary. The probability that at least one path is entirely composed of compromised routers is $b(1-(1-b^{l-1})^k)$. This quickly goes to b as k increases. We use a layered-mesh topology to balance the ability of the adversary to passively observe a path with his ability to actively perform a timing attack.

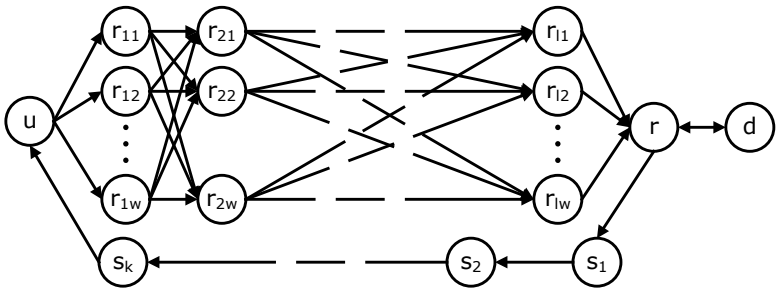


Fig. 1. Layered-mesh topology

Topology The layered-mesh topology we propose is pictured in Figure 1. For some length l and width w, user u sends a copy of each onion to the w members r_{1j} of the first layer. Then, in layer i, each router r_{ij} sends one copy of every onion it receives to each router $r_{(i+1)k}$ of the next layer. Finally, the routers r_{lj} in the last layer send a copy of each onion received to a single router r, which finishes decrypting them and sends the data on to the destination d. We call this structure the *layered mesh*.

Timestamps As described, the user sets timestamps to instruct routers to maintain a specific timing pattern. A user may have different latencies to the different routers in the first layer. If the time that one of these routers is instructed to forward the packet only depended on the network and processing delays of that router, the first-layer routers could send copies of the same packet at different times. This would provide information to the next layer about the identity of the user. Similarly, the adversary could use different times between layers to link other layers in the mesh. Therefore, we set the send time for each layer to be the send time of the previous layer plus the longest delay at our chosen reliability level p. We must also add some extra delay to allow for clock skews.

Let $d^*(r, s)$ be the amount of delay we need to ensure that a packet from r is processed at s with success probability p:

$$p = Pr[d_{net}(r, s) + d_{proc}(s) \leq d^*(r, s)].$$

At time t, let user u be instructed by $\mathcal{P}$ to send a message. The user chooses the same send time for all routers in the same layer. The send time for routers r_{1j} in the first layer is

$$t_1 = t + \max_j d^*(u, r_{1j}) + \delta_{sync}.$$

The send time for routers r_{ij} in the ith layer is

$$t_i = t_{i-1} + \max_{j,k} d^*(r_{(i-1)j}, r_{i,k}) + \delta_{sync}.$$

If a router receives its first copy of an onion after the send time has passed, it immediately forwards the onion to the routers in the next layer. At worst, the delay is the result of attempts by the adversary to delay certain packets. Sending the packet later or not at all in that case would only make it easier for the adversary to observe its lateness later in the mesh. Forwarding it immediately might even allow the onion to eventually get back on schedule. At best, the delay is just a result of network delays and forwarding has no effect on anonymity.

Onions Let M be the message to be sent. We will encrypt the message with a public key shared by all members of a layer. Given that the layers are set up in advance and known to all, such a key can be generated by a trusted third party or by electing a leader to do it. Let $\{M\}_{r_i}$ denote the encryption of M with the public key of layer i. Let $n_{r_i}, n_{s_1} \in \mathbb{N}$ be random numbers and k_r be a private key. Then the onion that u sends to the routers in layer 1 is

$$\{n_{r_1}, t_1, \{n_{r_2}, t_2, \cdots \{n_r, d, n_{s_1}, k_r, M\}_r \cdots\}_{r_2}\}_{r_1}$$

For each layer i, a user generates the random number $n_{r_i} \in \mathbb{N}$ as an onion identifier. The routers keep track of the onion identifiers they have seen. When they receive an onion, they decrypt it and examine the identifier. Routers only forward an onion if its identifier n_i has not been seen before. n_{s_1} is the identifier that r should use with s_1 when sending back any reply, and k_r is a private key that will let r encrypt the return message for u.

The onion encoding and forwarding scheme should hide routing information, prevent forgery, prohibit replay attacks, and hide message content. For clarity of presentation, we have described a simple scheme that achieves this. We observe, however, that several improvements to the protocol could be made. For example, the protocol could do a more explicit stream open and close to reduce the lists of identifiers that routers have to maintain. Also, symmetric keys could be exchanged to speed up onion processing. Another improvement that we could incorporate is forward secrecy. Numerous cryptographic details must be carefully set out (*e.g.* as in [5]) for our protocol to have a cryptographically secure and efficient implementation. These are not the focus of this paper.

5.2 To the user

Traffic returning to the user from the destination must first pass through the one router that is selected to communicate directly with the destination. This router may be compromised, and it may try to insert timing patterns in the return traffic. We manage this by giving the intermediate routers the pattern of the return traffic. They enforce it by fitting the return onions into the pattern, adding dummy packets when necessary. We note again that this doesn't work for the traffic from the user because any added delays translate into delays in the underlying data, and this can be viewed by the final router. We choose a simple path of length k for the return traffic (Fig. 1), because there is no anonymity advantage to adding redundancy here. We call this structure the *return path*.

To communicate the desired traffic pattern to the return path, we take advantage of the one-way communication protocol already developed. The user takes the return traffic pattern that is given by the padding scheme $\mathcal{P}$ and sends it via the layered mesh to every router in the return path. At the same time, the user generates the random numbers $n_{s_i} \in \mathbb{N}, 1 \leq i < k$, and sends two random numbers $n_{s_i}, n_{s_{i+1}}$ and a key k_{s_i} to each router s_i. The numbers will be the incoming and outgoing identifiers for the onion. The user also sends n_k and u to s_k. Let M be the message to be sent back from d to u. The return onion sent to s_i is

$$O_i =< n_{s_i}, \{\cdots\{\{M\}_{k_r}\}_{k_{s_1}}\cdots\}_{k_{s_{i-1}}} > .$$

After r receives M from d, it will take n_{s_1} and k_r out of the inbound onion from the user and send O_1 to s_1. When s_i receives O_i it looks for a matching n_{s_i} in the pairs of number it has received and then forms O_{i+1} to send when the padding scheme instructs it to.

5.3 Choosing the routes

A simple method to select the layered mesh and return path is for it to be chosen by the network and shared among all users. This is analogous to cascades in mix networks [3]. A disadvantage of cascades is that number of users that can be confused with one another, *i.e.*, the size of the anonymity set, is at most the number of users that can simultaneously be handled by a router [8]. Allowing users to choose the cascades with some freedom should allow anonymity sets to grow with the size of the network, similar to free routes in onion routing, but we leave this to future work.

6 Analysis

The purpose of our protocol is to provide better anonymity than onion routing at reasonable cost in latency and bandwidth. A major drawback to onion routing is that the probability of compromise is b^2 and cannot be improved, *e.g.* by choosing a longer path. We will show how in fact our scheme can provide arbitrarily good probability by increasing the length and width of the layered mesh.

First, the design of the protocol onions essentially limits the adversary to traffic analysis. For traffic from the user, the use of encryption and unique identifiers forces onions to be passed through the required layers and limits them to being forwarded by an honest router at most once. It also hides the source and destination from all but the first and last routers, and it makes messages leaving one layer unlinkable to messages leaving another layer. For traffic to the user, the source and destination are not included in the packets, and encryption prevents messages leaving one router from being linked with messages leaving another router.

For the adversary to break a user's anonymity, then, he will need to either observe traffic on an entire path between source to destination or link traffic at different steps on that path. The latter depends on his ability to introduce delays in the packet stream. To evaluate this possibility, we will make the simplifying assumption that the network and processing delay between two routers r, s never falls above the time allowed for it $d^*(r, s)$. In our model, such failures can happen with probability $1 - p$, where p can be set arbitrarily close to 1. Certainly, if the adversary can deanonymize a user even under this assumption, he can do so with possible link failures. When such failures do occur, they open up the chance for an adversary to successfully delay packets and insert a timing pattern. However, late packets are typically irrelevant because the same packet will have been forwarded by another router in the previous layer. Also, late packets that are the first of their type to arrive are immediately forwarded, thus benign delays are somewhat self-correcting, and we estimate that they do not open up a large opportunity for an active timing attack. However, if we wish to make a conservative estimation, we can expect that for any given packet a fraction $1 - p$ of the packet copies will fail to arrive in time. We can estimate the combined probability of malicious or benign delay or dropping of packets by $(1 - p) + b$.

Assuming no link failures, then, the anonymity of a user only depends on which routers the adversary controls. Because all users on the same cascade use the same routers, the adversary can either deanonymize all users in the anonymity set S_u, or he can not deanonymize any of them. Because the routers in the cascade are selected randomly, there is some probability that the adversary can deanonymize the users. Let $\mathcal{C}$ be the event that the adversary can compromise anonymity. We can quantify the probability of $\mathcal{C}$.

Theorem 1

$$Pr[\mathcal{C}] = b^{k+1} + b(1 - b^k)\left[b^w \frac{1 - (1 - (1 - b)^w - b^w)^l}{b^w + (1 - b)^w} + (1 - (1 - b)^w - b^w)^l\right]$$

Theorem 1 shows that, when less than half of the routers are compromised, we can make the probability that a user is anonymous arbitrarily high by setting w, l, and k large enough. (Recall that all proofs are in our technical report [9].)

Corollary 2

$$\lim_{w,l,k\to\infty} Pr[\mathcal{C}] = \begin{cases} 0 & b < 1/2 \\ 1/4 & b = 1/2 \\ b & b > 1/2 \end{cases}$$

Corollary 2 shows that anonymity can be made arbitrarily good when $b < 1/2$, but that it is worse than onion routing when $b \geq 1/2$. Therefore assume from now on that $b < 1/2$. We would like to determine how big the layered mesh and return path grow as we increase our desired level of anonymity. First, we will consider how wide the mesh must be for a given depth to achieve optimal anonymity. This affects the number of messages that need to be sent. Also, it will allow us to evaluate anonymity as a function of the lengths k and l, quantities which we would like to keep small to provide good latency. Luckily, it turns out that the optimal width w^* grows slowly as a function of l.

Theorem 3 $w^* = O(\log(l))$

Thm. 3 shows that the total number of messages sent for every message from the user is $O(2\log(l) + (l-1)\log(l)^2 + 1)$. This compares well asymptotically to the $l + 2$ copies that are sent in an onion-routing path of the same total length.

Network users are particularly sensitive to latency, and each additional step in the layered mesh and return path represents a potentially global hop on our network. To keep the lengths l and k of the mesh and return path small, then, we would like for $Pr[\mathcal{C}]$ to converge quickly to its limit. As suggested by Thm. 3, let $w = \log l$, and consider the convergence of $Pr[\mathcal{C}]$. It is clear that the first term shrinks exponentially with k. The convergence time of the second term is polynomial, although it does improve as b gets smaller.

Theorem 4 *Let* $c_1 = \log b$ *and* $c_2 = \log(1-b)$. *Then*

$$Pr[\mathcal{C}] = \Theta(l^{c_1 - c_2}).$$

Table 1 compares the performance of our mesh topology to that of onion routing using some reasonable parameter values. In it, we let both the mesh length and the onion-routing-path length be l, we let the length of the return path from the mesh equal the mesh length (*i.e.* $k = l$), and the width w of the mesh is set to optimize anonymity. We use small values of l to make the number of hops close to the three hops that have proven to be usable in the current Tor system. The numbers show clear decreases in the probability of compromise when using the mesh, especially with larger values of l. We can see that larger compromised fractions b will require somewhat longer paths for significantly improved anonymity. The total number of messages sent in each scheme for every message-response pair between the user and destination is also given.

			Mesh		**Onion Routing**	
b	l	w	$Pr[\mathcal{C}]$	Msgs.	$Pr[\mathcal{C}]$	Msgs.
.05	3	3	.0002	29	.0025	8
.05	4	3	.00003	39	.0025	10
.1	4	3	.0007	39	.01	10
.25	4	2	.0303	22	.0625	10

Table 1. Mesh routing vs. Onion routing

Our analysis shows how our scheme can provide arbitrarily good probability for $b < 1/2$. Is it possible to improve this to include values of b greater than

1/2? First, we observe that some other plausible topologies do not perform as well. For example, suppose the user sends onions to k routers, each of which forwards it to the same next router, and then from then on there is a path to the destination. The probability that anonymity is compromised in this situation is $b^2(1-(1-b)^k + b^{k-1}(1-b))$. As k grows, the anonymity goes to b^2, the probability that the second-layer router and final router are both compromised. As another example, consider using a binary tree, where the user sends to the leaves of the tree, and each node forwards to its parent at most one copy of the onions it receives. It can be shown that as the depth of the tree increases the probability that anonymity is compromised goes to zero when $b \leq 1/4$, $b(4b-1)$ when $1/4 \leq b \leq 1/2$, and b when $b \geq 1/2$.

The following theorem shows that the layered mesh is optimal in the limit among all topologies.

Theorem 5 *Let $c(b)$ be the probability of anonymity compromise in some forwarding topology when the fraction of adversarial routers is b. Then, if $b < 1/2$, $c(b) < \epsilon$ implies that $c(1-b) > 1 - b - \frac{1-b}{b}\epsilon$.*

Theorem 5 implies that if the probability of compromise for a topology goes to zero for $b \in [0, \beta]$, then it must go to b for $b \in [1-\beta, 1]$. This is achieved by the layered-mesh topology for the largest range of b possible, $[0, 1/2)$.

7 Latency Measurements

Our protocol requires routers to hold messages until a specified send time. Latency between routers varies, and therefore this send time must be set high enough to guarantee that with sufficiently high probability that it occurs after the packet actually arrives. The amount of time that packets spend being delayed before the send time depends on the variability of latency. Because network performance is critical to realizing effective anonymous communication [6], we wish to evaluate the latency in our protocol.

In order to do so, we have measured latency in the Tor [7] network. Performance data in the Tor network gives us reasonable estimates for the performance of our protocol because the essential router operation in both protocols is decrypting and forwarding packets and the network is globally distributed and therefore includes a wide variety of network and host conditions.

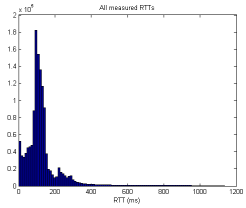
7.1 Methodology

During each experiment, we made three measurements on all Tor routers from our test host. First, we measured *round-trip time* (RTT) by opening TCP connections to the hosts. Second, we measured the *connection delay*, that is, the time between sending the stream-open request to the router and receiving the TCP connection from the router, by creating a circuit from our test host to the router and opening a TCP stream over that circuit from the router back to the

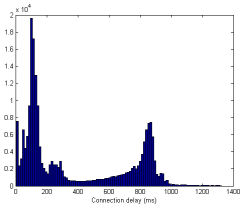
test host. Third, we measured *packet delay* by sending five 400-byte segments of data up the established TCP stream.

We took measurements hourly in the Tor network from February 22, 2009, to March 21, 2009.

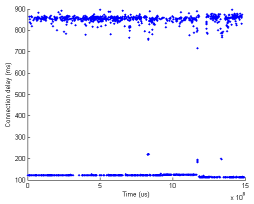
7.2 Results



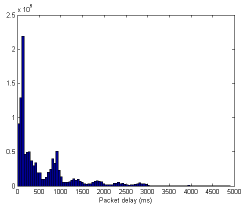
(a) Round-trip times (w/o top 1%)



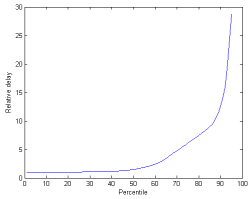
(b) Connection delays (w/o top 5%)



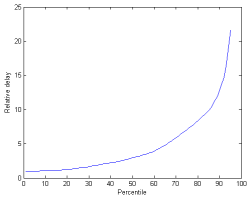
(c) Connection delays - *b0b3r*



(d) Packet delays (w/o top 1%)



(e) Relative connection delays, p=0.95



(f) Relative packet delays, p=0.95

Fig. 2. Measurement results

Round-trip times for the experiments are shown in Figure 2(a), with the top 1% removed to show the rest in greater detail. The mean of these times is 161ms and the median is 110ms. We see a peak bin centered at 100ms. A histogram of all the connection delays measured is shown in Figure 2(b), with the top 5% of delays removed. It shows that nearly all delays are less than 1s. Also, we can see that the distribution is bimodal, with peaks at about 120ms and 860ms. The connection delays over time for a one such router - *b0b3r* (193.221.122.229) - is shown in Figure 2(c). The clear and uniform timing stratification suggests a cause other than varying Internet-route congestion or host-resource contention. We believe that this is due to read/write rate-limiting that Tor servers manage by periodically filling global token buckets. We can use the RTT and connection delay measurements, assuming they are independent, to estimate the distribution of host processing delays. Considering the distribution over all routers, there is almost a 40% probability of having a processing delay of nearly zero. Thus processing delays are due to limited resources at the routers and not to inherent costs of the protocol.

The delays of the 400-byte packets that were successfully forwarded is shown in Figure 2(d). We again see times that cluster around certain levels. Here, there are six obvious levels. If we examine the delay time series of individual routers we see that again the different levels are interleaved. Thus, this phenomenon is probably due to the same mechanism underlying the similar pattern in connection delays.

From the delay measurements we can estimate the added delay that would have resulted from our protocol. The protocol chooses a time t that must elapse from the time the packet is originally sent before the packet is forwarded. It is set such that with some success probability p the packet arrives at the forwarding router in less than t time. In our data, we examine the tradeoff that varying t sets up between the fraction p of packets that arrive in time and the forwarding delay that gets added to them.

We divide the delay measurements by router and into 6 hour periods. Within each period, to achieve success probability p we set the send time t to be the smallest value that is larger than at least a fraction p of delays. We look at the relative increase in delay, *i.e.*, the total new delay divided by the original delay.

The distribution over all successfully-opened streams of relative connection delays to achieve a success probability of 0.95 is shown in Figure 2(e). At the 50th percentile, the relative connection delay is less than 1.48. Also, at the 50th percentile, we observe a failure rate of less than 0.005. The data for relative packet delays appears in Figure 2(f). At the 50th percentile, the relative packet delay is just over 2.95. The failure rate stays below 0.005 until the 99th percentile. The reason for this is that packet sends are not even attempted when the connection fails.

8 Future Work

There are several developments that fit within our approach and have to potential to make it truly useful and effective. Foremost among these is to design and evaluate a usable padding scheme, with large anonymity sets and low overhead. It should also allow a predetermined, or perhaps only periodically updated, return padding scheme. Also, we have avoided for now optimizing the efficiency of processing at the routers. Onion routing, in particular, has developed good techniques to make this aspect of the protocol fast. For example, we could use persistant circuits to speed up detecting duplicate packets, or we could switch to private keys. Our analysis could be improved in some areas as well. First, we could consider the positive effect of forwarding late packets immediately. Understanding this process better could improve the expected anonymity of the protocol. Also, Tor is not optimized for latency, and therefore understanding its resource congestion issues would help us better determine the added latencies of our protocol.

References

1. Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding, 4th International Workshop (IH 2001)*, pages 245–257, 2001.
2. Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES 2007)*, pages 11–20, 2007.
3. Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45, 2000.
4. Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007)*, pages 92–102, 2007.
5. Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Advances in Cryptology Conference – CRYPTO 2005*, pages 169–187, 2005.
6. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *5th Workshop on the Economics of Information Security (WEIS 2006)*, 2006.
7. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.
8. Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Privacy Enhancing Technologies, 4th International Workshop (PET 2004)*, pages 186–206, 2004.
9. Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Preventing active timing attacks in low-latency anonymous communication. Technical Report TR-10-15, The University of Texas at Austin, 2010. ftp://ftp.cs.utexas.edu/pub/techreports/TR-1965.pdf.
10. Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Active traffic analysis attacks and countermeasures. In *2003 International Conference on Computer Networks and Mobile Computing (ICCNMC'03)*, pages 31–39, 2003.
11. Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.
12. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Information Hiding, First International Workshop (IH 1996)*, pages 137–150, 1996.
13. Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, 13(2):1–28, 2010.
14. Jan Iwanik, Marek Klonowski, and Miroslaw Kutylowski. DUO–onions and hydra–onions – failure and adversary resistant onion protocols. In *Communications and Multimedia Security: 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, pages 1–15, 2004.
15. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go-MIXes providing probabilistic anonymity in an open system. In *Information Hiding, Second International Workshop (IH 1998)*, pages 83–98, 1998.

16. Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems (extended abstract). In *Financial Cryptography, 8th International Conference (FC '04)*, pages 251–265, 2004.
17. Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the Tor network. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 63–76, 2008.
18. David Mills. Network time protocol (version 3) specification, implementation. RFC 1305, Internet Engineering Task Force, March 1992.
19. Prateek Mittal and Nikita Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 267–278, 2008.
20. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (SP 2005)*, pages 183–195, 2005.
21. Steven J. Murdoch and Piotr Zieliński. Sampled traffic analysis by internet-exchange-level adversaries. In *Privacy Enhancing Technologies, 7th International Symposium (PET 2007)*, pages 167–183, 2007.
22. Arjun Nambiar and Matthew Wright. Salsa: a structured approach to large-scale anonymity. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006)*, pages 17–26, 2006.
23. Lasse Øverlier and Paul Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (SP 2006)*, pages 100–114, 2006.
24. Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2000.
25. Vitaly Shmatikov and Ming-Hsui Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security ESORICS 2006, 11th European Symposium on Research in Computer Security*, pages 18–33, 2006.
26. Paul Syverson. Onion routing for resistance to traffic analysis. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III)*, volume 2, pages 108–110, 2003.
27. Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114, 2000.
28. TorStatus - Tor network status. http://torstatus.kgprog.com/, April 2010.
29. Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Network flow watermarking attack on low-latency anonymous communication systems. In *2007 IEEE Symposium on Security and Privacy (SP 2007)*, pages 116–130, 2007.
30. Mehul Motani Wei Wang and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 323–332, 2008.
31. Wei Yu, Xinwen Fu, Steve Graham, Dong Xuan, and Wei Zhao. DSSS-based flow marking technique for invisible traceback. In *2007 IEEE Symposium on Security and Privacy (SP 2007)*, pages 18–32, Washington, DC, USA, 2007.
32. Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Privacy Enhancing Technologies, 4th International Workshop (PET 2004)*, pages 207–225, 2004.