Protecting Anonymity in Dynamic Peer-to-Peer Networks

Krishna P. N. Puttaswamy, Alessandra Sala, Christo Wilson, Ben Y. Zhao
Computer Science Department, University of California at Santa Barbara
{*krishnap, alessandra, bowlin, ravenben*}@*cs.ucsb.edu*

***Abstract*—Peer-to-peer anonymous networks offer the resources to support today's Internet applications. In today's dynamic networks, the key challenge to these systems arises from node dynamics and failures that disrupt anonymous routing paths, forcing them to be frequently rebuilt. Not only do these path rebuilds interrupt application sessions, but they also leak information to logging attacks such as the predecessor attack, leading to significant degradation of anonymity over long sessions. In this paper, we propose Bluemoon, a new anonymous protocol that provides strong resilience against the predecessor attack through the use of persistent anonymous links called *hooks*. When chained together, these links create robust anonymous paths that avoid path disruptions and rebuilds across node failures. Through detailed analysis, we show that relative to prior approaches, Bluemoon provides significantly stronger resistance against predecessor attacks. Finally, we implement and deploy a prototype on both local and Internet-scale network testbeds, and show that it provides high throughput even in high-load environments such as PlanetLab.**

I. Introduction

Today's Internet applications extend well beyond traditional email and web browsing to contemporary applications such as VoIP, streaming media and web services. However, protection of personal privacy in network applications has not kept pace. The most popular deployed anonymous network is Tor [10]. Infrastructure-based solutions such as Tor offer limited resources, thus supporting only TCP-based applications that can survive in a resource-constrained environment [16]. To support resource-intensive applications, one alternative is to use an open peer-to-peer anonymous infrastructure such as Tarzan [13], where each application endpoint acts as an anonymous relay and brings additional resources to the network.

However, the dynamic nature of peer-to-peer networks poses significant challenges to the reliability and security of anonymous protocols. Since nodes can enter or leave the network at any time, this "churn" breaks existing anonymous paths and forces them to be rebuilt by communication endpoints. Each path rebuild not only disrupts communication and consumes resources, but also exposes the flow to passive logging attacks such as the predecessor attack [30], [32]. The predecessor attack is a robust traffic analysis technique where colluding attackers such as Sybil identities [12] correlate knowledge of participants in a flow across its path rebuilds, allowing them to infer the identity of the source and receiver from their frequent appearances on multiple paths. Predecessor attacks are practical, and can be successfully launched by even low-resource nodes [1]. Furthermore, today's popular applications such as VoIP and web-services (*e.g.* GoogleDocs) require lengthy communication sessions. Long sessions in these applications increase the number of path rebuilds per session, further increasing their vulnerability to the predecessor attack.

Prior work in this area has proposed limited defenses against the predecessor attack. Wright et al. showed that using persistent nodes in anonymous routes can protect paths against the predecessor and other passive logging attacks [31]. This is similar to the use of trusted "guard nodes" deployed in the popular Tor network. Unfortunately, persistent nodes are difficult and costly to maintain, and are generally unavailable in today's dynamic networks. In addition, their persistence and scarcity makes them obvious targets for attacks, and their compromise would lead to significant loss of anonymity [1].

In this work, we propose Bluemoon, a novel anonymous communication protocol that provides strong resilience against predecessor attacks in dynamic network conditions. Our protocol defends against the predecessor attack using persistent anonymous links, referred to as *hooks* from here onwards, built from reliable peer-to-peer storage. These hooks include encryption keys necessary for relaying traffic to the next hop, and are stored reliably despite nodes entering and leaving the network. This reliable storage provides the abstraction of a persistent anonymous link (hook). Routing through these hooks provides robust anonymous paths that eliminate path rebuilds except in the case of massive correlated node failures. Not only does this provide reliable anonymous sessions, but it also means attackers have minimal information with which to compromise the identity of the communication endpoints.

This paper makes three key contributions. First, we propose the use of persistent anonymous links (or hooks) to build end-to-end anonymous paths. We describe our protocol called Bluemoon, and give details on an implementation of hooks that relies on reliable storage from Distributed Hash Tables (DHTs). Second, we perform detailed analysis to measure Bluemoon's resilience against the predecessor attack, and also quantify its overall anonymity using an entropy-based anonymity metric. Our analysis shows that compared to popular protocols such as Onion Routing, Bluemoon provides several orders of magnitude increase in resilience against the predecessor attack. Finally, we evaluate our Bluemoon protocol using both detailed simulations and measurements of a deployed prototype. Measurements from both a local cluster and the PlanetLab network testbed show that our prototype is efficient, supporting throughputs up to 100Mb/s on our cluster and 4Mb/s on bandwidth limited PlanetLab nodes.

II. Attack Model and Assumptions

We use the following terminology in this paper. A *node* or *peer* is a single instance of the Bluemoon router running on a single IP address and port. The *source* node sends messages to the *receiver* node. Continuous communication between a source and receiver is called a "session."

Goals. We designed our system with several goals in mind, listed here by priority:

1) *Anonymity against malicious peers:* provide source and receiver anonymity against colluding attackers, including those performing the predecessor attack [30], [32].
2) *Deterministic stability under network churn:* provide highly stable end-to-end paths in the presence of node failures and dynamics.
3) *Performance:* maximize routing performance to meet the throughput requirements of today's bandwidth-hungry applications.

Attack Model and Assumptions. We assume that an *attacker* is a participant in the network who passively monitors and logs all interactions with other participants. Attackers in the network will collude and share all logs with zero delay. A fraction of the network can be malicious at any given time and hence can monitor a fraction of the links in the network. However, attackers in our model do not perform active attacks on messages like tampering, dropping, or reverting encryption. Hence, we further assume that the attackers can perform timing analysis to identify messages that belong to the same session. This enables attackers to perform the predecessor attack.

Similar to prior work on passive logging attacks [30], [31], we do not assume a global adversary able to monitor and correlate *every packet on every link* in the network. Such an adversary equipped with timing analysis capability can track down the packets despite the presence of cover traffic [15].

Finally, our work focuses on protecting anonymity at the application level. While attackers can also target message routing at the overlay layer, a number of new protection mechanisms have been proposed to secure the overlay routing layer [14], [4], [17]. We believe these mechanisms can sufficiently protect against dropping and mis-routing messages. We also assume that every node joining the anonymous network is assigned a nodeID securely, either through a secure hash of a non-forgeable secret, or through assignment via a central CA [4].

III. Design and Implementation Details

Our primary goal is to build an anonymous routing protocol that provides anonymity levels similar to prior systems such as Onion Routing [10], and also provides strong resistance against passive logging attacks such as predecessor attacks in a peer-to-peer (P2P) environment.

Impact of Predecessor Attacks. One of the most powerful and fundamental attacks on anonymous systems is the *Predecessor Attack* [30], [32]. In this attack, colluding attackers use timing analysis to determine that they are on the same path, then work together to gain information about the identity of the source and receiver. As the path is repeatedly rebuilt, identities of the source and receiver can be inferred because they must be present on each of the anonymous paths. For traditional Chaum-mix systems, the path must be rebuilt after a single node failure on the anonymous path. If we call the time period between two path rebuilds a "round," it is shown that it only takes $(\frac{N}{c})^2 log(N)$ rounds to derive the source and the receiver of a path [30], [32] w.h.p., in a network of N nodes out of which c are colluding attackers.

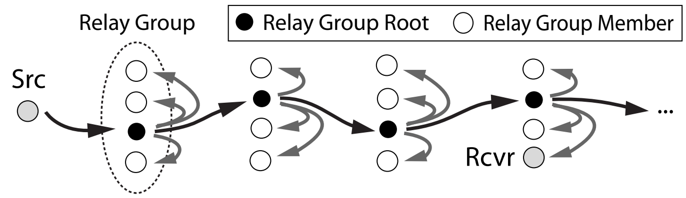

Fig. 1. A source forwards a message along an anonymous path consisting of a chain of relay groups. The root of each relay group broadcast the message to group members while forwarding it on to the next hop. The receiver is hidden inside one of the relay groups.

We achieve the goal of avoiding path rebuilds with one key insight: that using *persistent anonymous links* (or hooks) as building blocks in a Chaum-mix style multi-hop anonymous network leads to significant reduction in path failures. Each hook acts as a persistent link due to the robust DHT storage layer it is built on, and is persistent unless there is a massive failure in the DHT layer. Because our paths are constructed from these hooks, they are significantly robust to peer-to-peer node dynamics. Thus paths do not need to be rebuilt, eliminating information leakage that make attacks such as the predecessor attack possible.

In the rest of this section, we will describe our new anonymous routing protocol called Bluemoon built on hooks. We describe an implementation that relies on Distributed Hash Tables [7] for reliable storage, and then discuss the issue of path construction through hook insertion. Finally, we describe management and expiration of hooks, and how to construct anonymous reply paths in Bluemoon.

A. Realizing Hooks via Relay Groups

Our protocol forwards traffic across multiple relays, each identified by an application level ID. We introduce the notion of Hooks. Conceptually, a hook can be stored as a tuple ($<R_i, K_i, R_{i+1}>$) that includes the ID of the current relay (R_i), an encryption key used to decrypt traffic (K_i), and the ID of the next relay in the path (R_{i+1}).

Since single node relays can easily fail, we use groups of nodes to ensure reliability. These "relay groups" are identified by a single ID. All members of the group store the hook tuple, and are therefore capable of forwarding traffic on to the next relay group. We call the node that actually forwards each message to the next group the "relay group root." We will explain the implementation of relay groups and how roots are chosen in Section III-B. In addition to forwarding traffic to the next hop, each relay group root also broadcasts the message to all members of the relay group.

The receiver of an anonymous session is hidden in one of the relay groups on the path. It receives its messages when they are broadcasted by the local relay group root. The message source encrypts its messages with the appropriate symmetric keys such that, when they arrive at the receiver's relay group, the packets are encrypted only with a session key shared by the source and receiver. This symmetric key is chosen by the source and delivered to the receiver encrypted by its public RSA key (obtained from an offline CA). Message forwarding continues even past the receiver's group, and terminates when the end of the path is reached, indicated by a R_{i+1} whose value is null. We illustrate an anonymous path in Figure 1.

B. Building Hooks using Distributed Hash Tables (DHTs)

DHTs are scalable overlay networks that provide highly reliable storage based on a key-value hashtable interface [7], [27], [22]. All nodes in the overlay are assigned node IDs chosen uniformly at random from a large namespace, *e.g.* the 160-bit integer space. Given a key-value tuple, the DHT routes the tuple to a node whose own ID most closely matches the key. The DHT also replicates and maintains the tuple by storing it on a small group of nodes. For each key-value tuple associated with a key K, the DHT proactively replicates the tuple across the g nodes in the network closest to K in node ID. Increasing the replication factor g improves the availability of the stored data. Numerous studies have shown that DHTs provide highly reliable storage across node failures [6].

We assume that all nodes in our anonymous protocol are members of a DHT network. So we implement our relay groups by defining each relay group as the g nodes associated in the DHT with a key K, the relay group ID. Therefore, relay group IDs are chosen from the same namespace as all overlay nodes in the DHT. We use the reliable storage of DHTs to store and maintain the hook across the ongoing node dynamics in a P2P network. The DHT guarantees that incoming messages for a relay group will arrive at one of the nodes storing the hook tuple, and therefore is processed and forwarded on to the next hop.

The DHT greatly simplifies our management of relay groups and group roots. Members of the relay group are the g nodes actively maintaining a copy of the hook tuple. The relay group root is the node whose ID most closely matches the group ID, and can easily be identified by sending a DHT message to the group ID. As nodes come and go, the DHT ensures that the hook tuple is stored on the g nodes closest in ID to the relay group ID. Finally, the DHT allows us to create relay groups on the fly by storing a hook tuple using the desired relay ID.

Note that while our protocol leverages functionality of a DHT, it does not expose the native DHT interface to Bluemoon nodes. For example, attackers cannot perform *get* or *put* operations to retrieve or modify hook tuples. While attackers can repeatedly request node IDs to gain access to a given relay group, recent work has shown this attack to be extremely expensive in terms of node IDs required, and increases quickly in larger networks [19].

C. Anonymous Communication using Hooks

Recall that Bluemoon routes messages through a chain of hooks, each equivalent to an anonymous hop, as shown in Figure 1. During path construction, all relay groups IDs are selected uniformly at random, except the group containing the receiver. To include the receiver, the receiver's group must be chosen within a limited identifier space range away from the receiver. We present detailed proofs of how far the receiver group ID can be from the node ID of the receiver in Section IV.

Anonymous Hook Setup. To maintain anonymity during hook insertion, the source uses a probabilistic random walk similar to Crowds [21], where the message is forwarded through random nodes and has a constant probability p of being delivered to the destination R_i after each hop. While the path setup in Crowds is shown to be vulnerable to the predecessor attack [30], our hook setup process is less vulnerable for two reasons. First, the attack on Crowds [30] assumes that the source repeatedly talks to the same receiver, while our hook insertion is a one-time operation. Second, Crowds path setup includes the receiver's information in plain-text, where our hook setup messages include only the hook's key, and not the identifier of the real receiver. As a result, if a fraction f of the network is malicious, the attackers will see, with probability f, the source node send one message to relay group. But unlike Crowds, attackers have no information to correlate different hook setup messages with the same flow.

Forwarding Traffic. After the path is constructed, a source node A performs multiple layers of encryption on the message using the keys in the hooks in the right order and forwards the message along the chain of hooks, much like Onion Routing. When this data message wrapped in layers of encryption arrives at each hook, the DHT delivers this message along with hook identifier R_i to the relay group root. This root node then locates the hook tuple associated with identifier R_i, and uses the key K_i to decrypt one layer of the message M_i, obtaining the new message M_{i+1}, forwards the new message to the next hop, and finally broadcasts the new message to its broadcast group as illustrated in Figure 2.

To ensure the confidentiality of the message sent to the receiver B, all messages sent by A are encrypted with a symmetric session key. At path setup time, A generates the session key for the path, encrypts a magic number and the session key with the public key of B obtained by contacting an offline CA, and attaches the encrypted session key to the data transferred. Thus, the payload looks like:

$$P_i = <PubKey_B(MagicNumber, SessionKey), P_{i-1}>$$

All members of the relay group examine the message they receive and try to decrypt the encrypted session key in the payload using their private key. If successful (indicated by the presence of the magic number in the decrypted message), this member is the intended receiver and it decrypts the original message using the decrypted session key. Members cache

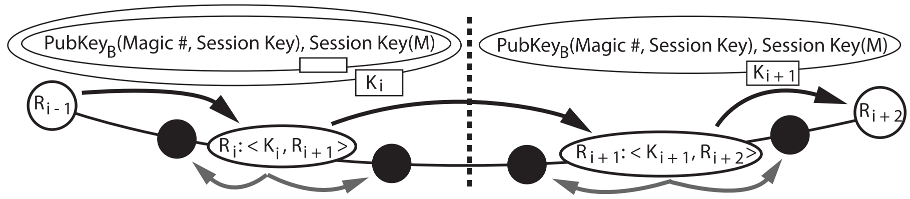

Fig. 2. Message decryption in Bluemoon. Each relay group root removes a layer of encryption using its symmetric key K_i, and forwards the message to the next relay group R_{i+1} and to the other members of its group.

the result of this decryption to avoid expensive asymmetric decryption of future payloads with the same session key.

Hook Management. Each hook has a pre-defined lifespan L^S, a system-wide parameter, after which it expires and is automatically removed. This "soft-state" approach allows implicit hook removal without divulging the source identity. Note that we do not perform explicit hook revocation, since those messages can be logged and used to compromise the source node's anonymity.

The use of hooks breaks the traditional point-to-point relay path abstraction. Since a hook can be inserted on any identifier, multiple forward hooks can latch on to the same identifier, essentially providing light-weight anonymous multicast. This can be used to repair or replace parts of an anonymous path without destroying the entire path. Take for example a sequence of two hooks that route traffic from identifier R_{i-1} to R_i, then from R_i to R_{i+1}. To reroute traffic around R_i, the source would insert a hook $<K_i, R_{i+1}>$ at identifier R_n, then a hook $<K_{i-1}, R_n>$ at identifier R_{i-1}. Then, the relay group root at R_{i-1} will multicast each message to R_i and R_n until the original hook expires, effectively modifying the path.

Receiving Anonymous Replies. The chaining of hooks can be easily extended to support anonymous replies from the receiver B to the source A. To receive replies, source A sets up a reverse chain of hooks going from B to A. Each of these hooks perform encryption on the data packets they receive instead of decryption as on the forward chain. A sends in its payload to B the location of the first hook C on the reverse path and the symmetric session key for the reverse path. All replies are forwarded to C and onwards back to A. routed via the reverse chain of hooks back to A.

IV. Analysis of Bluemoon's Anonymity

In this section, we perform detailed analysis to evaluate Bluemoon's resistance against the predecessor attack [30], [32], and analyze our source anonymity and unlinkability using the entropy metric [9], [24].

Our attack model is the same model used to analyze the impact of predecessor attacks [30], [32], thus allowing us to directly compare our analytical results with those of prior systems. Our analysis assumes nodes relay traffic, but do not perform mix operations such as padding, mixing or traffic shaping. We also assume Internet links introduce latency variance and prevent attackers from detecting their precise position on the routing path through perfect timing analysis.

A. Formalizing Our System

At a high level, our approach improves path robustness by breaking a traditional Chaum-mix path into multiple links, each represented by a single persistent anonymous link we call a *hook*. The source forwards its messages through a number (R) of hooks, each identified by a unique ID in the namespace. The nodeIDs are m-bit numbers assigned uniformly at random by a central offline CA.

Each hook forwards messages to a broadcast group of size g, where each member receives the message. We refer to the node whose nodeID best matches a hook's ID as the *hook root*. Each hook root node forwards messages to its $g-1$ broadcast neighborhood, which on average covers a range on the overlay network equal to $I = \frac{g2^m}{N}$ nodeIDs.

The destination or receiver can be hidden in the broadcast group of any hook in the path. Therefore, one of the hook IDs must be chosen such that the receiver's nodeID is close enough for it to be included in the broadcast group. This receiver ID is a random ID chosen from a restricted interval near the receiver. In particular, let x be the receiver's ID. Then the desired receiver ID is a random number in $I' = [x - \lfloor\frac{(g-1)2^m}{2N}\rfloor, x + \lceil\frac{(g-1)2^m}{2N}\rceil]$. We use this interval because the source does not know which live nodes are in that interval, but must make sure that the receiver will be part of the $g-1$ nodes that will receive its messages.

Anonymous Routing Overhead in Bluemoon. The overhead of routing a message in traditional anonymous routing systems like Onion Routing is equal to the length of the anonymous path. In Bluemoon, however, each message is broadcasted to g hook members. As a result, the overhead is g times more than Onion Routing's overhead for the same path length. However, this extra overhead provides significant benefits in terms of anonymity and reliability of the path. We quantify this in detail later in this Section, and experimentally validate in Section V.

B. Analysis of the Predecessor Attacks and Network Dynamics

In this section, we provide two analytical results. First, we determine the number of rounds required for attackers to perform a successful predecessor attack on Bluemoon. In this analysis, the worst case result for Bluemoon (and the best case for attackers) is when the receiver always stays within its broadcast group. We explore this assumption to obtain the worst case result for Bluemoon. Next, we seek to understand the scenario when a receiver is "pushed" out of its intended receiver broadcast group by nodes entering the network. This

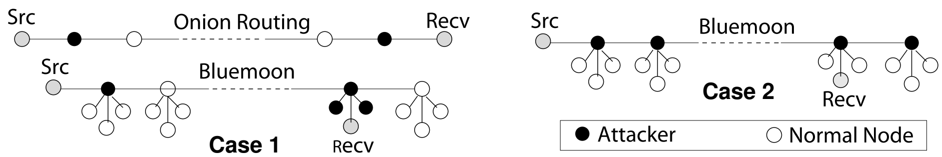

Fig. 3. Scenarios for successful predecessor attacks on Onion Routing and on Bluemoon (group size 4).

occurs when more than g nodes join during the "broadcast interval" defined as I. Using a stable network model, we bound the time the receiver spends outside of its broadcast group.

For our analysis of Bluemoon, we conservatively define a round. Recall that the time period between two path rebuilds in Chaum-mix systems is called a "round." But we define a round in Bluemoon as the time between failure or leave events of *any* two nodes in a relay group-path. Therefore, our analysis measures degradation in anonymity as a function of any changes to the groups on the end-to-end path, not just changes to roots of relays groups.

1) Resilience to Predecessor Attacks: To understand the highest probability of success of predecessor attack on Bluemoon, we must determine the optimal configurations that allow attackers to observe and link a source and the receiver. In systems like Crowds or Onion Routing, the attacker configurations are simple. In Crowds, for example, each node that forwards a particular message knows its receiver. Therefore attackers can easily identify each other on the same path and identify the same path across rounds. In Onion Routing, the receiver is always the last node on the path. Thus the successful attacker configuration is where two attackers reside on the path, one immediately after the source and one immediately before the receiver as shown in Figure 3. For Bluemoon, the receiver can reside in any of the hook groups along the path, and hence we need to customize the attacker positions for them to correlate paths across multiple rounds.

Let us consider the situation when there are just two attackers on the path, $k = 2$, (assume $R > 3$; otherwise, it degenerates to the case we analyze in Lemma 2) in two separate relay groups. Assume that they are the roots of two relay groups in the path (best for the attackers). Across rounds, the attackers can observe their predecessor group root, successor group root and the members of their groups. However, as the receiver can be in any of the groups in the path, the confidence with which the attackers can identify the receiver and correlate with the source is low indicating that we need more than two attackers.

Suppose there are more than two attackers, $k > 2$ (and $R > 3$). Attackers are served best in both these cases when one attacker is the root of the first relay group. Then, there are two ways to place the remaining attackers. First case is when they are all the group members of a single relay group, and the second case is when they are lined up as the root of relay groups on the path. In the first case, when $k < g-1$ (at least one group member is not an attacker), the attackers have to disambiguate between the other two group nodes one of which might be the potential receiver. This means the attack has the highest probability when $g-1$ members of a given relay group are attackers. In the second case, when $k < R-1$ (at least one relay root is not an attacker), there is at least one group for which the attackers cannot get any information to correlate. Since this group might potentially contain the receiver, the attack is most benefited when all relay group roots are attackers.

Therefore, there are two scenarios when Bluemoon is most vulnerable to the predecessor attack. First is when the attackers control all other nodes in the receiver's broadcast group, as shown in the "Case 1" of Figure 3, which we examine in Lemma 1. The second scenario is when attackers control the root nodes of all hook broadcast groups, as shown in the "Case 2" of Figure 3. They can then identify all nodes in all groups in a path and correlate them across rounds. We examine this scenario in Lemma 2. We examine these two scenarios separately in the two lemmas and combine our analysis to show that the predecessor attack succeeds in $O\left(min\left[\frac{(N-R+1)^R \log N}{c^R}, \frac{(N-g+1)^g \log N}{c^g}\right]\right)$ rounds in Theorem 1. Note that we obtain the worst case for Bluemoon by assuming that receivers cannot leave their groups.

Lemma 1. *The probability that the c colluding attackers control all other nodes in the receiver's broadcast group, and also the group root immediately following the source is $O(\frac{c^g}{(N-g+1)^g})$.*

Proof: We assume the probability to occupy a particular position in the namespace is uniformly distributed w.r.t. the total number of positions. The probability that given c nodes, $g-1$ of them are in consecutive positions within the receiver's group, and one node is the first group root after the source is: $\frac{\binom{N-g}{c-g}}{\binom{N}{c}} = \frac{(N-g)!(N-c)!c!}{N!(N-c)!(c-g)!} < \frac{c^g(N-g)!}{N!} < \frac{c^g}{(N-g+1)^g}$ ■

Lemma 2. *The probability that c colluding attackers control all relay roots is $O(\frac{c^R}{(N-R+1)^R})$.*

Proof: Assuming that the c colluding attackers can join the network with the same probability in each position, the probability that they control the R relay roots that the source has chosen for an anonymous path is: $\frac{\binom{N-R}{c-R}}{\binom{N}{c}} = \frac{(N-R)!(N-c)!c!}{N!(N-c)!(c-R)!} < \frac{c^R(N-R)!}{N!} < \frac{c^R}{(N-R+1)^R}$ ■

Theorem 1. *With high probability, the number of rounds that the c colluding attackers need to perform a successful predecessor attack is $T = O\left(min\left[\frac{(N-R+1)^R \log N}{c^R}, \frac{(N-g+1)^g \log N}{c^g}\right]\right)$.*

We do not include detailed proofs here due to space con-

	Crowds	Onion Routing	Bluemoon	Ideal Network
Rounds req. w.h.p.	$O(\frac{N}{c}\log N)$	$O((\frac{N}{c})^2 \log N)$	$O\left(min\left[\frac{(N-R+1)^R \log N}{c^R}, \frac{(N-g+1)^g \log N}{c^g}\right]\right)$	$O((\frac{N}{c})^l \log N)$

TABLE I

ASYMPTOTIC WORST CASE BOUNDS ON THE NUMBER OF ROUNDS REQUIRED TO COMPROMISE THE PROTOCOLS USING PREDECESSOR ATTACKS, WITH HIGH PROBABILITY. IDEAL NETWORK RESULTS ARE ACHIEVED WHEN TIMING ANALYSIS IS NOT POSSIBLE.

straints. Full proofs are available in our technical report [18].

2) Impact of Network Dynamics: We now consider the scenario where the receiver can be excluded from its broadcast group by the addition of new nodes. When this occurs, attackers cannot see who is the receiver even if they controlled all other nodes in the receiver group. We assume a stable network where the node join and leave rates are equal.

Our goal is to understand the probability that the receiver is kicked out of its group, and how long it has to wait before rejoining and resume receiving messages from the source. Under dynamic network churn more than g nodes could join the broadcast group interval of $\frac{g2^m}{N}$ IDs, pushing the receiver outside the group that receive the messages from the source.

To compare Bluemoon fairly against competing protocols, we assume that the total number of nodes involved in a path between the source and the receiver is a constant L. In Chaum-mix protocols these L nodes are organized as a chain. In Bluemoon, the L nodes are organized as a chain of R groups, each containing one root node and $g-1$ leaf nodes.

Theorem 2. *With high probability, the receiver can leave its broadcast group for up to $O(L)$ rounds.*

Please refer [18] for detailed proof.

We note that while receivers are outside their groups, attackers can observe no information about the receiver, thereby maximizing unlinkability. To maintain message delivery, however, the receiver's peers in the group must maintain a FIFO cache of messages from the last L rounds to be delivered to the receiver upon return. Alternatively, the source can construct a reverse path leading to itself using hooks, and then get ACKs from the receiver to ensure the receiver is in the path, and construct a new path when the receiver leaves its group. Path rebuilds, however, increases the information leaked and hence can be undesirable for applications. We leave it as future work to study how a receiver can leave its group to take advantage of this to improve its anonymity, and what is the best strategy to deliver packets when the receiver is outside the group.

C. Anonymity via Entropy Metric

We now evaluate Bluemoon's source anonymity and unlinkability using the entropy-based anonymity metric [9], [24]. Let the "anonymity set" be a set in which single elements are not identifiable. In a network of N nodes, the anonymity set Ω is a subset of $\{N\}$ whose members all have the same probability to be the source or receiver. Clearly, "ideal anonymity" is achieved when $|\Omega| = N$, meaning all nodes are not identifiable as the source or the receiver. The following two definitions from literature quantify the "Entropy of a system" and the "Anonymity of a system" respectively:

Definition 1. *Let Ω be a finite set of all nodes in the network. An attacker can compute $\forall$ node $v \in \Omega$ a probability p_v as being the source or receiver of a message, using information leaked from the system. Therefore the system entropy is defined as: $H(\Omega) = -\sum_{v\in\Omega} p_v \log p_v$*

From the above definition it is clear that the maximum entropy $H_{max}(\Omega)$ of a system is achievable when $\forall\ v \in \Omega,\ p_v = \frac{1}{|\Omega|}$ and so $H_{max}(\Omega) = \log |\Omega|$.

Definition 2. *The anonymity of a system can be measured as:* $\frac{H(\Omega)}{H_{max}(\Omega)} = \frac{-\sum_{v\in\Omega} p_v \log p_v}{\log |\Omega|}$

It follows from this definition that a system's anonymity is measured as a value $\in [0, 1]$ where 0 means no anonymity and 1 means the system is completely anonymous. It is now possible to measure "unlinkability" (i.e. the probability to bind source and receiver):

Definition 3. *$\forall$ pair of nodes $(u, v) \in \Omega$ let p_u be the probability that u is the source and p_v the probability that v is the receiver. Let $p_{u,v}$ be equal to $p_u p_v$, then the unlinkability of our system is:* $\frac{\sum_{(u,v)\in\Omega} p_{u,v}(\log p_{u,v})}{N^2(\frac{1}{N^2}\log\frac{1}{N^2})}$

1) Source Anonymity: In our analysis, c colluding attackers share data gathered from locally observed traffic and perform timing analysis. Our results show that source anonymity is strictly a function of the path length and the number of malicious nodes on the path.

Theorem 3. *The source anonymity proportionally increases with the rise of the number of relay groups.*

See our technical report [18] for the detailed proof.

2) Receiver Anonymity:

Theorem 4. *Receiver anonymity increases as a function of the size of the hook groups.*

Proof: We separate our analysis into two cases, when the receiver is in a group where the root is controlled by an attacker, and when the receiver's group root is not controlled by an attacker. Let A be the number of all nodes that the attackers can "see" on the path, where all nodes in a group are "seen" if and only if an attacker controls the group root. In case 1: the probability that attackers can assign to each node to be the receiver is: $\frac{1}{Rg-\frac{cA}{N}}$. Clearly, the larger the group, the stronger the receiver anonymity. In case 2: the attacker can infer that a node is the receiver with probability: $(\frac{1}{N-A})(\frac{Rg-A}{Rg-\frac{c}{N}A})$. ■

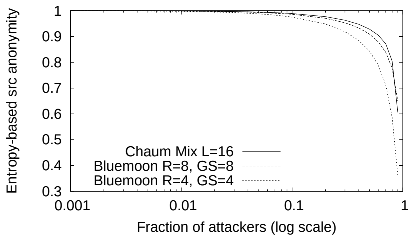

Fig. 4. Entropy-based source anonymity of Bluemoon (for relay group size 4 and 8) and Chaum Mix.

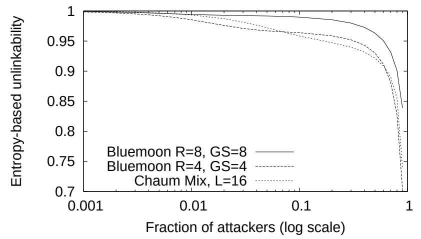

Fig. 5. Entropy-based unlinkability of Bluemoon (for relay group size 4 and 8) and Chaum Mix.

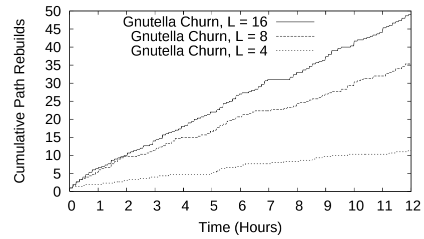

Fig. 6. Path rebuilds in Onion Routing under churn for different path lengths over a period of 12 hours.

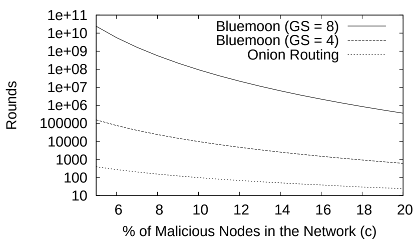

Fig. 7. Ideal number of rounds to break the unlinkability of Onion Routing and Bluemoon.

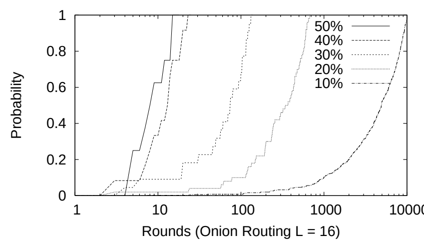

Fig. 8. Onion Routing's anonymity degradation under predecessor attack, for different percentage of attackers in the network.

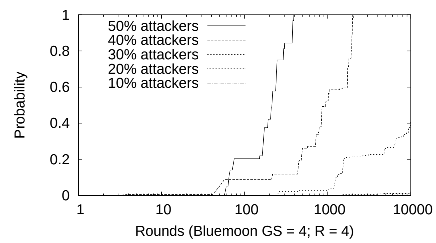

Fig. 9. Bluemoon's anonymity degradation under predecessor attack, for different % of attackers in the network, relay group size=4.

D. Simulation Results using Entropy Analysis

We simulate the entropy-based source anonymity in Figure 4 for Bluemoon and Chaum-mix under different number of relay nodes (R) and group sizes (g). Figure 4 shows that with different ratios of colluding attackers in the network, Bluemoon has source anonymity similar to those of Chaum-mix systems. We also see that we can improve source anonymity by increasing values of R and g. Figure 5 shows that for the same total number of nodes (L=16 or R=4, GS=4), Bluemoon has similar unlinkability to Chaum-mix. We also see that as expected, increasing the number of nodes on the path (by increasing R and g) significantly improves unlinkability.

V. Evaluation

In this section, we first simulate path robustness of Bluemoon and Onion Routing using configurations that incur equal bandwidth overhead. We then compare their predicted and measured resilience against predecessor attacks. Finally, we present throughput measurements of our prototype deployed on a LAN and on PlanetLab.

A. Simulation Results

We implemented Bluemoon and Onion Routing in OverSim [2], an event-driven overlay simulator. All simulations run a network of 1000 nodes, each assigned a random 160-bit nodeID. Unless specified otherwise, each node entering the network is randomly assigned a lifetime based on traces or an exponential distribution. Nodes leave once their lifetime expires, and are replaced by a new node entering the network with a new ID and lifetime. In addition to churn, our network also maintains a constant percentage of malicious nodes. We keep the sender and the receiver alive for the test duration.

All our simulations terminate after 10K rounds; we count the number of times the attackers were in the right positions to perform a successful predecessor attack over time (each occurrence is called a hit) and convert this hit count to the probability of success of the attackers based on the theoretical number of hits required by the attackers. Each point plotted in our graphs is an average of 20 simulation runs.

1) Bluemoon Path Robustness to Churn: We measure the number of path reformations under churn for both Onion Routing and Bluemoon, using node lifetimes extracted from a recent measurement trace of the Gnutella network [23].

Comparison under Equal Bandwidth Overhead. Like in analysis, the death of any relay on the path is considered a round for Onion Routing simulation, and a new path is built. For Bluemoon, the death of any path member (group root or member) signals a new round. For fair comparison, we ensure that both protocols use the same number of nodes in a path: for each Onion Routing path of L hops, we compare it to a Bluemoon path of L/g hooks, with g nodes in each hook's relay group. Thus, we fairly compare the robustness of these protocols under *exactly the same* amount of total bandwidth overhead.

Robustness Results. We plot the cumulative path rebuilds over time in Figure 6 for different path lengths. The number of rebuilds increases significantly for Onion Routing paths with the passage of time. On the same trace, Bluemoon did not require a single path rebuild, confirming the effectiveness of its stable links. Compared to the Cashmere [33] protocol that provides probabilistic path resilience, Bluemoon achieves the same level of stability deterministically, and therefore achieves equal or better path robustness for a given path.

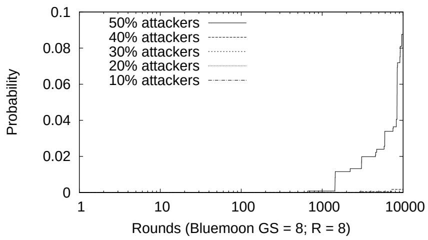

Fig. 10. Bluemoon's anonymity degradation under predecessor attack, for different % of attackers in the network, relay group size=8.

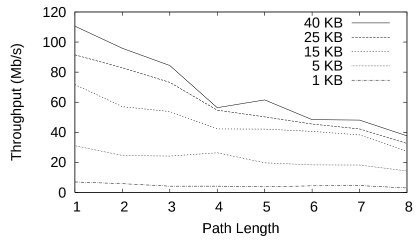

Fig. 11. Throughput measurement results from Bluemoon prototype deployment on the cluster.

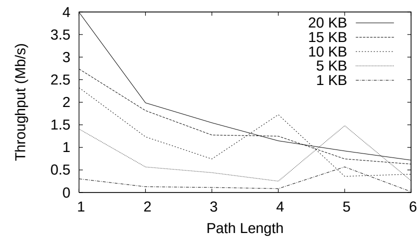

Fig. 12. Throughput measurement results from Bluemoon prototype deployment on the Planetlab.

2) Anonymity Degradation under the Predecessor Attack: Analytical Results Visualized. To better visualize the analytical results in Table I, we plot in Figure 7 the theoretical number of rounds required by attackers to break the unlinkability of Onion Routing and Bluemoon. With 5% of nodes attacking, Bluemoon with group size 8 is more than eight orders of magnitude better than Onion Routing. The gap slowly shrinks as the ratio of attackers increases. At 20%, Bluemoon is four orders of magnitude better. Attack resiliency shrinks with relay group size. At group size 4 the advantage of Bluemoon ranges from three orders of magnitude to two orders of magnitude at 20% attackers in the network. Our simulation results presented next confirm these analytical results.

Simulation Results. We verify our analytical results via simulation by comparing the unlinkability of Bluemoon and Onion Routing under a globally colluding predecessor attack. For this long running simulation we derive node lifetimes following an exponential distribution with average of 3000 secs. A simulation of 10K rounds corresponds to a real-world measurement of around 12 days. This extreme churn leads to around 300K total node leave events in a 1000 node network during the simulation. In each graph, we plot the number of rounds on the x-axis and the probability of attacker's success for the given number of rounds on the y-axis. We also look at resilience for different amount of attackers in the network. For a fair comparison, we set the Onion Routing path length to 16, and the Bluemoon path length to 4 with each group having 4 nodes (totally 16 nodes). Thus, paths in Bluemoon and Onion Routing incur exactly the same bandwidth overhead.

We plot the attack resilience for Onion Routing and Bluemoon in Figure 8 and Figure 9 respectively. Although both systems are more easily compromised with more attackers, Bluemoon's degradation is at least an order of magnitude slower than that of Onion Routing. Bluemoon's resilience is exceptional for low level of attackers in the network. Lines for 10% and 20% attackers are on the x-axis and hence invisible. We increase Bluemoon group size and path length to 8, and see from Figure 10 that it improves the attack resilience by more than two orders of magnitude compared to group size of 4, and confirms our analytical results. Note that we had to zoom in on the y-axis in Figure 10 to see any results. Except the 50% attacker graph, all others are on the x-axis and hence are invisible.

The step function-like increase in Bluemoon graphs occur because once the attackers get into the right positions their hits keep incrementing for every round until one of the attackers gets out of position. In the case of Onion Routing the increase in probability is more uniform as the entire path is reset in each round and the hits do not have this cumulative effect.

B. Prototype Deployment and Measurement Results

We implemented our prototype as a library on top of a C-based DHT. Including the DHT storage and replication layer, Bluemoon totals less than 1500 lines of C++ code. We used openSSL v0.9.7f and 128-bit keys (Blowfish) as our symmetric cipher. We also implemented on Bluemoon an anonymous email service which took less than 200 lines of C++.

Measurement Setup We deployed and measured our Bluemoon prototype on both cluster and wide-area networks. Our local experiments ran on a cluster of 30 Dell PowerEdge 1750 servers connected via Gigabit Ethernet LAN. Our wide-area experiments ran on 90 world-wide nodes on PlanetLab.

Once we established our anonymous overlay, we choose one machine as the data collector D, and insert a hook H into the network with D as the receiver. The rest of the machines sent bursts of 20 packets to H using paths varying in length from 1 to 8 hops. The collector would record the inter-arrival times of the 20 packets in each burst and use the total time coupled with the packet size to compute the throughput. Our graphs represent the average throughput obtained for each packet-size path-length combination over a large number of test runs.

Cluster Measurements. Figure 11 plots throughput obtained on the cluster experiment generated by measuring approximately 250,000 packet bursts. Each line represents throughput of a particular packet size as we vary the number of hooks. Throughput is high for low path lengths, and lowers gradually as each hop adds additional overhead for encryption and forwarding. Since the cryptographic operations in Bluemoon are bound by CPU speed rather than bandwidth, larger packets are able to reach higher throughputs. These results show that our prototype adds little overhead, providing throughput in the hundreds of megabits. Even for today's network applications, this should be sufficient to support multiple flows per node.

PlanetLab Measurements. Due to PlanetLab's popularity as a test environment, the large majority of its nodes are constantly under extremely heavy processing load. PlanetLab machines also experience significant bandwidth congestion and packet loss. Figure 12 shows the throughput results obtained from 13,000 packet bursts on our PlanetLab deployment. Average throughput on PlanetLab is significantly lower due to heavy CPU contention on each machine. Path lengths greater than 6 hops provided negligible throughput. The two outliers on the graph can be attributed to variance in machine load on PlanetLab. Data runs for these points were initially lost due to failed nodes, and were repeated on a separate day.

Our results demonstrate the robustness of the Bluemoon prototype under extreme conditions. We observed CPU loads on PlanetLab nodes ranging from 10.0 to 40.0 during the course of our experiments. Maintaining connectivity and garnering up to 4 megabits of bandwidth under these conditions attests to the feasibility of deploying Bluemoon applications on resource constrained end hosts.

VI. Advanced Attacks and Defenses

We proved earlier that Bluemoon guarantees stronger anonymity against the standard predecessor attack compared to other systems such as Crowds and Onion Routing. We now examine the effectiveness of other attacks, including a variant of the predecessor attack highly customized for Bluemoon.

A Stateful Predecessor Attack. We can modify the predecessor attack to focus on obtaining control of relay roots, nodes that actually perform the message forwarding (and local broadcast) at each relay. Each relay root knows the IDs of the previous and next relay roots on the path. Since our paths are stable, colluding attackers need to control each relay root on the path at least once and record the information they gained. If the length of the path R is known, then knowing information at all R relays reveals the identity of the source node. While this is possible, it is difficult to achieve in practice for large networks. Note that increasing the path length increases the cost of controlling all relay roots.

Even if the attack succeeds in this first step, obtaining the identity of the receiver is much harder. For R-hop paths, where keys are replicated to $g-1$ non-root nodes, the receiver can be any one of $R(g-1)$ nodes observed. Clearly, increasing the path length also increases the size of the effective anonymity set. More importantly, attackers can only identify the receiver if they control all other nodes in the receiver's group. Using the same analysis as the proof for Theorem 1, the attack can be successful in $O((\frac{N}{c})^{g-1}\log N)$ rounds. This is less than the rounds required for the basic predecessor attack. However, this attack on the receiver can be prevented. We note that multiple flows can use the same relay group with different keys. Thus when attackers try to control the receiver's group, they can only identify nodes in a flow by its key. If Bluemoon periodically sends out new keys (via path setup messages) to the receiver's relay group or a nearby group it belongs to, attackers cannot be sure that they are logging the members of the right flow and must restart the process.

Intersection Attack. A successful intersection attack requires full knowledge of all nodes online at different times during a session, using the intersection of these online sets to reduce the anonymity of the source and receiver nodes. Unlike prior systems [13], [21] where each node has access to global network membership, a node in Bluemoon knows only of its overlay neighbors, which make up a small fraction of the large peer-to-peer network. In addition, we can use mechanisms proposed by Salsa [17] to further limit the amount of network membership information known at each node.

Path Construction Attacks. Attackers can mis-route path setup messages to other attackers, thereby increasing the number of attackers on a given path. In addition, attackers can drop setup messages. We can defend against attacks at the overlay layer by leveraging recent work [17], [14] that provides bounds checking and redundancy mechanisms in the overlay. Attackers can also snoop on path setup messages to obtain keys. We can defend against this attack by encrypting each setup message S_m with a random symmetric key k', and delivering S_m and k' to the relay root using separate random walks, each starting at nodes far apart in the overlay.

The Sybil Attack. Attackers can enhance their influence in the network through the use of the Sybil attack [12]. We can mitigate its impact by leveraging previously proposed mechanisms such as limiting node identities through IP address mapping, per-node monetary or computational costs, or user CAPTCHAs [13], [17], [4]. Uniform random selection of relay or hook identifiers means that attackers' power will be limited to the ratio of the network it controls, *i.e.* (c/N) if controlling c nodes in a network of size N.

Denial of Service (DoS). Broadcast-based protocols [25], [26] are highly vulnerable to DoS attacks, where malicious nodes broadcast dummy messages in the group to disrupt legitimate communication. Since the impact is proportional to the group size, Bluemoon's small group sizes limits the severity of these attacks. Attackers can also perform DoS by becoming relay roots, dropping messages and forcing paths to be rebuilt. Recent work [3] suggests a fundamental tradeoff between reliability and anonymity, and shows that selective DoS can significantly reduce anonymity. Further work to defend against these attacks is ongoing, and details are beyond the scope of this paper.

VII. Related Work

Cashmere. Cashmere [33] improves the reliability of anonymous routes using flexible naming in structured peer-to-peer overlays to construct *probabilistic* forwarding groups. Cashmere requires a central CA to distribute large numbers of prefix keys. More importantly, Cashmere's improved resilience to node failures is non-deterministic, and in practice, will provide little or no protection for a significant portion of all sessions. Since nodeIDs in these overlays are randomly assigned, Cashmere group sizes can vary dramatically from empty or single node groups to very large groups.

Theorem 5. *The probability that a Cashmere prefix group has at most one node is* $\frac{g+1}{e^g}$.

Proof: Let N be the number of nodes in the overlay and nodeIDs are assigned to nodes uniformly at random. The size of relay groups defined by a k-digit groupID is Poisson distributed with parameter $g = \frac{N}{2^k}$. The expected size of the relay group is g. Using the Poisson distribution, the probability that at most one node is in a group G is: $P[|G| \leq 1] = \frac{e^{-g} g^0}{0!} + \frac{e^{-g} g^1}{1!} = \frac{g+1}{e^g}$. ■

This shows a significant probability that a given group is empty, resulting in a broken path, or has only one member, resulting in a brittle path similar to prior systems. For the recommended group size $g = 5$, the probability of a given group having zero or one node is 4%. In a 5-hop path, the probability that one or more groups along the path fails (zero nodes) or is unreliable (one node) is 18.6%. While increasing g reduces the probability of an empty group, doing so generates proportionally large groups that incur high overheads in intra-group communication. In contrast, the hook abstraction provides deterministic reliability guarantees that prevent independent node failures from breaking links and triggering end-to-end path rebuilds.

Other Anonymous Protocols. Chaum's seminal Mix-Net design [5] inspired many anonymous systems proposed to date. Many low-latency relay-based anonymous systems are designed for interactive communication [11], [8], including the Onion Routing system [20], [29]. Onion Routing relies on traffic redirection between a set of dedicated routers that maintain pair-wise symmetric keys. The source routes the messages through a selected set of currently active routers. Tor [10] is the second generation onion routing system, and is deployed globally on a network of trusted infrastructure servers. These systems experience scalability problems with increase in the number of users due to limited resources. P2P anonymous systems [13], [17] have been proposed to address the scalability problems of infrastructure-based systems.

Finally, hooks resemble the trigger abstraction in i3 [28]. While i3 is designed to ease the implementation of mobility and multicast, hooks are built to defend against predecessor attacks in anonymous systems.

VIII. Conclusion

The predecessor attack poses a real and significant risk to peer-to-peer anonymous networks. In this work, we describe a proposal that uses hooks to defend against traffic analysis attacks such as the predecessor attack. We show that by leveraging persistent storage on DHTs, we can make anonymous links stable and minimize the impact of network dynamics on anonymous communication paths. Our analysis and simulations show that compared to prior approaches, our approach provides several orders of magnitude improvement in resilience against predecessor attacks. Measurements of a deployed prototype shows that provide reasonable throughput even under contentious high load environments.

Acknowledgements

We thank Mudhakar Srivatsa and Lili Cao for providing thoughtful comments on the earlier drafts of this paper.

References

[1] Bauer, K., et al. Low-resource routing attacks against tor. In *Proc. of WPES* (Alexandria, VA, 2007).

[2] Baumgart, I., Heep, B., and Krause, S. Oversim: A flexible overlay network simulation framework. In *Proc. of GI* (2007).

[3] Borisov, N., et al. Denial of service or denial of security? how attacks on reliability can compromise anonymity. In *CCS* (2007).

[4] Castro, M., et al. Security for structured peer-to-peer overlay networks. In *Proc. of OSDI* (December 2002).

[5] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *CACM 24*, 2 (1981).

[6] Chun, B.-G., et al. Efficient replica maintenance for distributed storage systems. In *Proc. of NSDI* (May 2006).

[7] Dabek, F., et al. Towards a common API for structured P2P overlays. In *Proc. of IPTPS* (February 2003).

[8] Danezis, G. Mix-networks with restricted routes. In *Proc. of PET* (March 2003).

[9] Diaz, C., Seys, S., Claessens, J., and Preneel, B. Towards measuring anonymity. In *Proc. of PET* (April 2002).

[10] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *Proc. of the USENIX Security Symposium* (Aug 2004).

[11] Dingledine, R., and Syverson, P. Reliable MIX cascade networks through reputation. In *Proc. of FC* (March 2002).

[12] Douceur, J. R. The Sybil attack. In *Proc. of IPTPS* (Cambridge, MA, March 2002).

[13] Freedman, M. J., and Morris, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of CCS* (Nov. 2002).

[14] Kapadia, A., and Triandopoulos, N. Halo-high assurance locate for distributed hash tables. In *Proc of NDSS* (2008).

[15] Levine, B. N., Reiter, M. K., Wang, C., and Wright, M. K. Timing attacks in low-latency mix-based systems. In *FC* (Feb 2004).

[16] Murdoch, S. J. Economics of tor performance. Light Blue Touchpaper Blog, July 2007.

[17] Nambiar, A., and Wright, M. Salsa: A structured approach to large-scale anonymity. In *Proc. of CCS* (Nov 2006).

[18] Puttaswamy, K. P. N., et al. Protecting anonymity in dynamic peer-to-peer networks. Tech. Rep. 2008-12, UCSB, Sept. 2008.

[19] Puttaswamy, K. P. N., et al. Securing structured overlays against identity attacks. Tech. Rep. 2008-01, UCSB, Jan. 2008.

[20] Reed, M. G., Syverson, P. F., and Goldschlag, D. M. Anonymous connections and onion routing. *IEEE JSAC 16*, 4 (May 1998).

[21] Reiter, M., and Rubin, A. Crowds: Anonymity for web transactions. *ACM TISS 1*, 1 (June 1998).

[22] Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware* (November 2001).

[23] Saroiu, S., Gummadi, P. K., and Gribble, S. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (January 2002).

[24] Serjantov, A., and Danezis, G. Towards an information theoretic metric for anonymity. In *Proc. of PET* (April 2002).

[25] Sherwood, R., et al. P5: A protocol for scalable anonymous communication. In *IEEE Sec. and Priv.* (May 2002).

[26] Sirer, E. G., et al. Eluding carnivores: File sharing with strong anonymity. In *Proc. of ACM SIGOPS European Workshop* (Sept. 2004).

[27] Stoica, I., et al. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM* (2001).

[28] Stoica, I., et al. Internet indirection infrastructure. In *Proc. of SIGCOMM* (August 2002), ACM, pp. 73–86.

[29] Syverson, P. F., et al. Anonymous connections and onion routing. In *IEEE Sec. and Priv.* (May 1997).

[30] Wright, M., et al. An analysis of the degradation of anonymous protocols. In *Proc of NDSS* (Feb. 2002).

[31] Wright, M., et al. Defending anonymous communications against passive logging attacks. In *Proc. of IEEE Sec. and Priv.* (May 2003).

[32] Wright, M., et al. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM TISS 7*, 4 (2004).

[33] Zhuang, L., et al. Cashmere: Resilient anonymous routing. In *Proc. of NSDI* (Boston, MA, May 2005).