Salsa: A Structured Approach to Large-Scale Anonymity

Arjun Nambiar
arjun.nambiar@uta.edu

Matthew Wright
mwright@cse.uta.edu

Dept. of Computer Science and Engineering
University of Texas at Arlington
Box 19015
Arlington, TX 76019-0015

Highly distributed anonymous communications systems have the promise of better distribution of trust and improved scalability over more centralized approaches. Existing distributed approaches, however, face security and scalability issues. Requiring nodes to have full knowledge of the other nodes in the system, as in Tor and Tarzan, limits scalability and leads to intersection attacks in peer-to-peer configurations. MorphMix avoids giving nodes complete system knowledge, but new research shows that a collaborating fraction of the peers can control the paths of many users.

To overcome these problems, we propose Salsa, a structured approach to organizing highly distributed anonymous communications systems for scalability and security. Salsa is designed to select nodes to be used in anonymous circuits randomly from the full set of nodes, even though each node has knowledge of only a small subset of the network. It uses a distributed hash table based on hashes of the nodes' IP addresses to organize the nodes into groups. With a virtual tree structure, limited knowledge of other nodes is enough to route node lookups throughout the system. We use redundancy and bounds checking when performing lookups to prevent malicious nodes from returning false information without detection. We show that our scheme prevents attackers from biasing path selection, while incurring moderate overheads, as long as the fraction of malicious nodes is less than 20%. Additionally, the system prevents attackers from obtaining a snapshot of the entire system until the number of attackers grows too large (e.g. 15% of 10000 peers, given 256 groups). The number of groups can be used as a tunable parameter in the system, depending on the number of peers, that can be used to balance performance and security.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

CCS'06 October 30–November 3, 2006, Alexandria, Virginia, USA.

General Terms

Security

Keywords

Anonymous Communications, Peer-to-Peer Networks, Privacy

1. INTRODUCTION

Anonymous communications systems on the Internet provide protection against eavesdroppers and others that seek to link users with their communications. These systems have many important applications in areas such as law enforcement, intelligence gathering, business privacy, anonymous publishing, and personal privacy. Current systems, such as Tor [11], rely on a relatively small set of advertised servers to forward messages for the user. These systems can suffer from scalability problems, with potentially large bandwidth and system overhead costs, and the servers themselves can be targets of direct attacks.

Peer-to-peer anonymous communications systems, such as Tarzan [13] and MorphMix [23], have been proposed as a way to alleviate these problems with a large and dynamic set of peers acting as servers. This makes direct attacks less effective and increases scalability. Tarzan, however, requires that each peer know the identity of all other peers, which makes it highly vulnerable to intersection attacks [28] and does not scale beyond 10,000 nodes [13]. MorphMix does not have this requirement, but it requires that users allow untrusted peers to choose the proxies that will forward the user's messages; attacker-controlled peers will always select other colluding peers to be on the path. Although the authors of MorphMix propose a collusion detection scheme, recent work has shown that this scheme can be fooled while attackers continue to control many paths in the system [26]. The fundamental problem facing these systems is one of selecting peers independently at random from the set of peers to ensure unbiased path selection, while not requiring full knowledge of the set of available peers.

To solve this problem, we propose a new peer-to-peer anonymous communications system using distributed hash tables (DHTs). Similar to peer-to-peer file-sharing systems that use DHTs, like the Chord system [24], our system maps each IP address to a point on the ID space using consistent hashing. We further divide the ID space into groups, conceptually organized as a binary tree for purposes of node

lookup. Each node has knowledge of all the nodes in its own group, as well as knowing a limited number of nodes in other groups. This knowledge is enough to effectively route lookups throughout the system. Nodes use redundancy and probabilistic checking when performing lookups to prevent malicious nodes from returning false information without detection.

We show that our scheme prevents attackers from biasing path selection, while incurring moderate overheads, as long as the fraction of malicious nodes is less than 20%. Additionally, the system prevents attackers from obtaining a snapshot of the entire system until the number of attackers grows too large (e.g. 15% for 10000 peers and 256 groups). The number of groups can be used as a tunable parameter in the system, depending on the number of peers, that can be used to balance performance and security.

In Section 2, we describe existing work in peer-to-peer anonymous communcations systems, relevant attacks on these systems, and related work in structured peer-to-peer systems. Section 3 describes the design of Salsa, including our novel network architecture. We give an analysis of Salsa's security properties in Section 4. We describe the simulation methodology and our results in Section 5 and conclude with future directions in Section 6.

2. BACKGROUND

In this section, we cover three areas that are critical to our work. First, we make a case for highly distributed anonymous communications in light of a broad overview of work in anonymity. Second, we describe relevant attacks and challenges facing current proposed systems for highly distributed anonymous communications. Third, we look at other structured peer-to-peer overlays and security considerations that have been studied to date.

Before this, we give some concepts that we use throughout the paper. As we are considering systems in which a user's node may also be a proxy, we define the *initiator* as the node of the user who initiates an anonymous connection. We call the node that the initiator contacts, such as a Web site, the *responder*. Most systems are based on Chaum's idea of mixes [6], in which there is a path of proxies, or *circuit*, between the initiator and the responder that adds indirection to help hide their connection. A technique called *layered encryption* limits an attacker's ability to track packets passing through the circuit. However, most of these systems are vulnerable to attacks in which the first and last proxies on a circuit, or a well-placed eavesdropper, can collaborate and use the timings of packets to correlate the initiator with the responder [9, 17, 29].

2.1 Highly Distributed Anonymity has Strong Potential

Experts in anonymous communcations do not have consensus on the issue of what types of mix-based systems are most secure. Some argue for *mix cascades*, such as Web Mixes [3], in which all messages pass through the same set of proxy servers [4]. Others argue for *mix networks*, such as Tor [11], in which users choose proxy servers randomly for each position on the path. More recently, starting with Crowds [22], the idea of peer-to-peer systems for anonymous communications has been developed. Rather than attempt to present complete arguments for these approaches, we describe here only the benefits of both the peer-to-peer approach and versions of mix networks with many servers. Our goal is only to demonstrate that this is a promising approach worth the present investigation – we expect that debate over which approach is best will remain open for some time.

Peer-to-peer anonymity systems provide a nice security property: the first proxy in the path does not know whether or not it follows the initiator or another peer serving as a proxy. This is the main argument for security against malicious peers in Crowds [22]. Although Crowds does not protect against what we believe to be reasonable attacker models, such as a substantial subset of the peers [27], this property also applies to systems that use layered encryption. A significant security property of both highly distributed mix networks and peer-to-peer networks with many servers comes from the large number of proxies. When there are fewer proxies, the relatively small number of them present a viable set of targets for direct and active attacks. Such attacks include eavesdropping, node corruption, node takeover, and legal subpeonas.

A particularly devastating attack for smaller, less distributed, systems was presented by Murdoch and Danezis in 2005 [20]. In this attack, which was demonstrated on the Tor network with 35 operating onion routers, a single corrupt client fills the available bandwidth of a server and watches for drops in the connection's bandwidth. If those drops in bandwidth correlate to the times when packets were received by the responder, the server was involved in forwarding the initiator's traffic. The attack succeeds by testing many servers and tracing back the initiator's path. In highly distributed systems, this and other direct attacks are significantly harder due to the sheer number of tests that must be conducted. Similarly, eavesdropping may require a truly global adversary with substantial capability to extract relevant data from vast amounts of traffic – placing eavesdropping equipment directly on the networks of even a fraction of the nodes is not practical for many attackers.

Another benefit of highly distributed systems is in the distribution of costs and the possibilities for beneficial incentive structures. Since some users are providing services as a cost of being in the system, they do not need to pay for service. This is important, because paying for anonymity can be challenging to do in a fair and anonymous way and may require special forms of digital cash [21]. Further, users who are particularly concerned about privacy have an incentive to provide service, which somewhat reduces the *freeriding* problem [14]. In particular, freeriders have weaker anonymity as they do not forward packets for other nodes, so all packets must be initiated by the freerider himself [1]. Other incentives issues may exist for peer-to-peer systems, but the current situation is promising.

2.2 Attacks on Highly Distributed Anonymous Communications

A significant issue with peer-to-peer systems, and any system that does not strongly verify the identity of the participants, is the Sybil attack [12]. This attack is essentially a recognition that an attacker can, at relatively low cost, construct many online identities and use them to control or attack the system.

The Sybil attack can be partially mitigated. For example, we can force the attacker to own many IP addresses by ensuring that each identity maps to a unique address; this has been used in the Tarzan system [13]. Another technique

pioneered by Tarzan and adopted by MorphMix [23] is the use of *hierarchical address selection*. In this scheme, a user who wants to select a node at random first chooses a subnet at random, based on subnets represented in the peer's IP addresses, and then select a node from the chosen subnet at random. This ensures that an attacker who controls one subnet cannot flood the system with nodes from that subnet to gain control in the system. An attacker must then control nodes in a large number of subnets to succeed.

In today's Internet, the threat of *botnets*, in which the attacker controls a large population of corrupted nodes widely distributed in the network, continues to make the Sybil attack a dangerous threat. Botnets of 100,000 nodes have been reported, but botnets of 20,000 nodes or less are more common, partly because of the smaller chance of being caught [15]. Against a widely-distributed botnet, the only known defense is to have a large number of semi-honest nodes that are not collaborating with the attacker. Thus, scalability is critical for open anonymous communications systems to succeed.

One of the most challenging attacks to defend against in anonymous communications is the intersection attack. In this attack, a passive observer takes logs of the set of users participating in the system at regular intervals. By comparing these logs with logs of the responders contacted via the system at the same times, the attacker can profile different users and undermine the users' anonymity [8, 18, 28]. In most systems, this attack is difficult to conduct, as it requires complete knowledge of all the users in the system. An attacker controlling a large subset of the proxies could not be sure that they have the complete list. Note that an incomplete list means that the attacker would be profiling users based on unseen users' activity. The attackers do not, however, need to witness all outgoing activity. If the attacker is only interested in a single responder (e.g., the owner of the responder wants to know who is contacting it anonymously), then it need only get information on when the responder is contacted via the system.

In the Crowds and Tarzan systems this attack is particularly dangerous. When joining the system, the user obtains a list of all peers, which is used to select proxies. This is considered necessary for the security of the system, as a partial list might be biased with a large fraction of attackers [13]. However, the list of peers provides a list of all users. Thus, an attacker need only join a single peer to the system to obtain a great deal of the information needed to perform the intersection attack. This attack has been shown to be very efficient in narrowing the possible initiators to a small set (e.g., five or ten peers) [28].

Using complete lists of all peers is also a scalability issue. Tarzan's gossiping protocol for propagating peer information throughout the network only scales well to about 10,000 nodes [13]. The operators of Tor have discussed, for scalability reasons, reconfiguring the system so that each proxy only knows and connects to a subset of other proxies (R. Dingledine, personal communication).

The MorphMix peer-to-peer system similarly has each peer maintain connection information about a subset of the other peers [23]. While this helps to avoid the intersection attack and improves scalability, choosing a path securely becomes challenging. For example, MorphMix initiators choose the first node on the path from their list of known peers. For the second node, however, the first node makes the choice from among its list of known peers. If the first node is an attacker, it can select an attacker for the second node, and this can continue throughout the path. In this way, whenever an attacker is selected for the first node, the entire path can be easily compromised. MorphMix employs a *witness* process to detect when this kind of collusion is used, but recent work by Tabriz and Borisov has shown that the witnesses can be deceived by intelligent selection of nodes [26]. This allows the attackers to control many paths in a system indefinitely and without detection.

In general, limited knowledge of the network can be a significant issue for secure paths, as the initiator may be deceived. This is a key motivation of our work.

2.3 Structured P2P Systems and Anonymity

A number of works have considered security and privacy issues for structured peer-to-peer systems. Danezis *et al.* propose an alternative routing strategy, called *zig-zag routing* for DHT-based systems that helps defend against Sybil attacks [10]. Zig-zag routing aims to avoid Sybil groups by ensuring diversity while getting close to the target. Both Borisov and Ciaccio have proposed adding anonymity to structured peer-to-peer systems [5, 7]. Borisov proposes the use of random walks on de Bruijn networks to help provide anonymity with reasonably short paths. Ciaccio proposes the use of *imprecise routing* in which the construction of neighbor tables is done from a range rather than with a precise value (as in Chord). This makes it difficult for an attacker to work backwards to determine the source of a request.

These systems all have in common the idea of adding randomness or diversity to the structured overlay, while keeping the general approach of rapid reduction in the distance to the target. Salsa also shares this feature, but it uses a unique structure that is designed to satisfy the different system requirements of providing a structured overlay for large-scale anonymity.

3. DESIGN OF SALSA

In this section, we describe the Salsa system design. The purpose of Salsa is to organize a large anonymous communications network to enable random and unbiased node selection for users. The basic design combines a structured overlay with redundant lookups to ensure that randomly selected nodes are not biased towards selection of attackers. With such a system, we argue, the initiator can select the nodes in her circuit at random without significant bias and without knowledge of the entire set of proxies.

The overall anonymous communications system may either be a peer-to-peer anonymous communications system or a server-based system similar to Tor. We assume that the system creates a circuit of proxies for each user by selecting a set of nodes from the available peers or servers. This is the basic approach used in most anonymous communications systems, including Tor (and Onion Routing), MorphMix, Tarzan, and Freedom [11, 25, 23, 13, 2]. We have intended Salsa to focus on node organization and selection, independent of building and maintaining anonymous communication circuits, forwarding traffic, and returning replies.

The fundamental challenge that motivates the Salsa design is the selection of nodes at random from the set of available peers with only limited knowledge of the nodes. As explained in Section 2.2, allowing every node to have only limited knowledge of the system helps protect the system

from the intersection attack (in peer-to-peer systems) and enhances the scalability of the system. It is also important to select nodes at random from the entire set of nodes. Selecting from subsets of the network would allow an attacker to focus an attack on any given user. However, selecting a node at random despite limited knowledge is not trivial.

3.1 A Naive Approach

We now describe a simple but naive approach to address the challenge of random node selection, based on the Chord system. Similar to Chord, we can give each node an identifier (ID) based on a cryptographic hash of the node's IP address [1]. The IDs are placed in sorted order around a circular *ID-space*, and each node N_i is said to *own* the fraction of the ID-space between itself and it's preceeding node N_{i-1}. A user makes a request for a specific ID, called the *target ID*, and a request for a target ID between N_{i-1} and N_i will be routed to N_i, as it owns that ID-space. In this case, we say that N_i is the *target owner*.

Also as with Chord, each node keeps a *finger table*, which stores routing information. The finger table includes the addresses of a series of nodes, called *fingers*, that are at varying distances away in the ID-space. Specifically, we can define the minimal distance between two adjacent nodes as one unit, and the fingers can then be said to at one unit, two units, four units, and further powers of 2 units away from the node, up to approximately one-half of the total ID-space (a more precise description may be found in the Chord paper [24]). This allows for fast routing of requests in the system through recursive propogation. If a node knows the address of the target owner, it will forward the request directly to that node. Otherwise it will forward the request to the node in its finger table that most closely preceeds the target ID. This scheme ensures that the number of nodes that need to be contacted is bounded by $O(\log n)$ [24]. Intuitively, this is because the distance is cut in half at each step until the owner is only one unit of ID-space away.

We now have what we need to select nodes at random with only limited knowledge of the system. First pick a random ID from the ID-space. Then send a request for the owner of that ID. This is not entirely the same as uniform random selection from the set of nodes, as each node may own different amounts of the ID-space, but the amounts are probabilistically bounded by $O((\log n)/n)$ and cannot be controlled by the owners [24]. In particular, an attacker who controls c nodes in the system will own, on expectation, c/n of the ID-space. For large values of c and n, the actual value will be close to expectation.

Due to the use of consistent hashing, the system has several other important features. Consistent hashing provides a weak form of authentication: every other proxy can quickly compare the hash of the IP address of the node it connects to with the node's claimed ID. As described in Tarzan, a simple two-way handshake is enough to ensure that the node controls the IP address (at least with respect to the connecter) [13]. Additionally, the attacker cannot attempt to place a node in a certain part of the address space without controlling an IP address that would map to the desired ID range. If the node density is high enough, this becomes difficult for most attackers. Another benefit of consistent hashing is that when a node joins the system, the ID-space can be determined from its successor node. The successor node cannot lie to the joining node to gain more space for itself – the space between them is defined by the IP addresses that they use to communicate. When a node leaves the system, it can inform its successor that the address space now belongs to the successsor. With high probability, no node owns more than $O((\log n)/n)$ fraction of ID-space.

However, this approach has an obvious security problem: since the user must rely on other nodes to forward the requests, any node along the path of a request can modify the results. In particular, attackers may stop propogating the request and return the identity of another attacker. To alleviate this problem, we propose the use of simultaneous redundant requests. Rather than relying on a single request, the requesting node asks each of its fingers to make the request on it's behalf. Although redundancy increases overhead in the system, the lookups can be done in parallel to keep the delay bounded, and we demonstrate in Section 5 that, with a different network structure, the amount of redundancy is also bounded at what we believe to be a reasonable level.

A particularly nice property of consistent hashing in this regard is that the target owner will be closer than any other node. This means that only one of the redundant requests needs to return the correct result for the requesting node get the IP address of the true target owner – if multiple results are returned, the closest node can be selected. Here, the attacker also must take some risk of being discovered in modifying lookup results unless it can be sure that it has modified all of the redundant lookups. It may be possible to use incorrect lookup results as part of a reputation system, but we do not study this possibility here.

The issue with using this Chord-like structure for redundant requests is that many requests may go through a few of the same nodes, who are then in a position to modify more than one request if they are malicious. To see this, we first note that all requests flow in a single direction around the circular ID-space. Second, as requests get closer to the target, the distance between hops decreases. This ensures a greater density of the requests close to the target. If some of the nodes close to the target are malicious, the attacker is much more likely to be able to modify all the returned results. See Section 5 to see how using Crowds falls short.

3.2 The Salsa Network Architecture

To improve the value of redundant requests, we propose the Salsa network architecture, a novel structured overlay designed to aid the random selection of nodes for anonymous communications. The Salsa architecture improves upon the naive approach by ensuring that redundant requests proceed on random pathways and do not converge on a few nodes. We show in Section 5 that when the fraction of attacker nodes in the system is less than 20%, between four to six redundant lookups is sufficient to find the correct node a very high percentage of the time.

There are some key similarities between the Salsa architecture and the Chord-based approach. Salsa is based on a DHT and has an ID-space to which we map nodes by taking a hash of their IP-address. Each node has a list of a subset of the nodes in the system, its *contacts* (akin to fingers in Chord), that is used to route requests throughout the ID-space. Also, each lookup is resolved by contacting

[1]Recent advances in the birthday attack on cryptographic hashes such as MD5 do not affect our system, as the IPv4 address space is too small to yield many collisions.

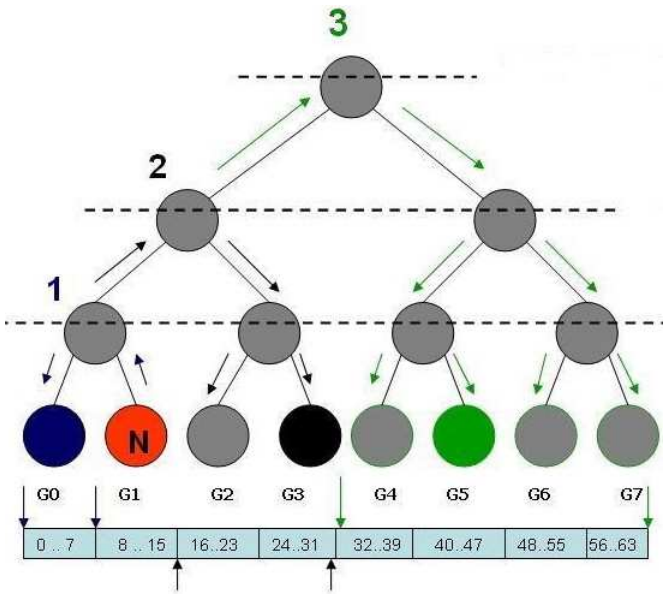

Figure 1: The binary tree structure of Salsa.

other nodes in a recursive manner until the target owner is reached. Each of the nodes contacted during a lookup is called a *lookup hop*.

However, rather than having a single circular ID-space, we divide the entire ID space into equal-sized groups that are defined by a contiguous portion of the ID-space. Each group's portion of the ID-space is cyclic, i.e., the ends of the ID-space within a group wrap around. Let us say that a node belongs to a group if its ID is in the group's ID-space; each node belongs to exactly one group. Within the group, each node *owns* the ID-space between it and the node preceeding it; the ID-space segment of the first node in the group loops around to the last node. A node knows the connection information and ID-space segment for all nodes in its own group, and these are the node's *local contacts*.

Groups are conceptually organized in the form of a binary tree, as illustrated in Fig 3.2. To facilitate lookups beyond a node's group, each node has a set of *global contacts* in different groups, according to the tree structure. In particular, a node has one global contact for each level of the tree. At each level, the global contact is selected at random from the subtree corresponding to the other child of the node's parent that level. For example, in Figure 3.2, a node in group 4 (black), will have one contact in group 3 (dark grey), one in either group 1 or group 2 (group 1 is chosen and shaded in grey), and one in any of groups 5 through 8 (group 8 is chosen and shaded light grey). To ensure that nodes cannot select arbitrary global contacts, the contact for a given segment of the address space is chosen by hashing the concatenation of the selecting node's IP address and the height of the tree. The selected node can easily verify that it should be a global contact for the requesting node.

With both global and local contacts, a lookup for any ID in the ID-space is possible. Lookups proceed recursively. The requesting node asks a global contact in the same subtree as the target owner to continue the lookup and return the results. The contacted node becomes the requesting node and repeats the request to a contact in a smaller subtree that includes the target owner. This continues until the target owner is in the requesting node's group, in which case the requesting node has the contact information and returns it along the path of the request.

Since each node has multiple local contacts, we utilize these for redundancy. The initiator asks a subset of the local contacts, chosen at random, to each find the owner of a given target ID. The result that is closest to the target ID is assumed to be the target owner and is selected. A key property of the Salsa architecture is that any two nodes, even from the same group, will share very few global contacts on average. Thus, redundant requests are likely to proceed along different paths on the way to the target owner's group. This property helps to reduce the possibility of a single node being on multiple paths, thereby reducing the chance that the attacker can modify all the results returned to the requesting node.

An additional security measure is a simple bounds check. Since the target owner should be close to the target ID, we can reject results that are too far from the target ID. This may lead to rejections of legitimate nodes, so we need to consider false positives as well as the false negatives in setting our boundary. We have found that when the number of attackers is less than 20%, it is possible to set a boundary with low error rates. Further, we can modify the boundary for different systems to balance the delay in finding an acceptable node with the security of finding only correct target owners. We show the effectiveness of this in Section 5.

3.3 Building a Circuit

An initiator must select a small set of nodes, e.g. three for Tor, to use as proxies in the circuit. Normally, the initiator would select these nodes privately. In Salsa, however, the node lookup can be linked to the initiator if any of the local contacts that is used for redundant look up is controlled by the attacker. If this happens, and the last node is an attacker, then it can link the initiator to her messages.

First, we calculate the chance of this attack succeeding and then provide a solution to decrease this chance. If there are n nodes in the system and c of them are controlled by the attacker, then any given node will be an attacker with probability c/n. Thus, the chance that at least one of k chosen nodes will be an attacker is $1-(1-c/n)^k$. For $k=3$, and $c=0.1*n$, i.e. 10% attackers, this is approximately 27%. For the same $k=3$, but $c=0.2*n$, i.e. 20% attackers, this is approximately 49%. The chance that the last node is an attacker is slightly more than the fraction of attackers in the system, as we show in Section 5. Let us assume that it is 11% and 22% for 10% and 20% attackers in the system, respectively. For $k=3$ and 10% attackers, there is a 3.0% chance of attacker success; for 20% attackers, it is 11%.

A more secure approach is to incorporate circuit-building into the redundant lookups. First, r nodes are looked up in the normal way. Keys are estabilished with each of these nodes. Second, each of the first set of nodes does a lookup for r additional nodes. A Tor circuit is built from the initiator, through one of the first nodes, to one of the second nodes. The second set of nodes does a redundant lookup for a final node. One of the paths created between the first and second sets of nodes is selected, and the final final node is added to the end of the circuit. In this approach, the first node does not learn the identity of the final node and vice versa.

The attack against this last approach would require one attacker in the first set and one attacker in the second set as well as the final node. For $r = 3$ and 10% attackers, this means an approximately 0.8% chance of success. For $r = 3$ and 20% attackers, this means a 5.2% chance of success. Note that the chance of getting the first and last nodes as attackers on the circuit is 1.0% and 4.0% for 10% and 20% attackers in the system respectively. Since this scenario is considered sufficient to break the user's anonymity with a timing analysis attack in low-latency anonymity systems [11, 17, 9], further defenses are not likely to be worth the cost.

When $r = 3$ and a circuit length of three proxies, the added cost of this technique, over using redundant lookups with $k = 3$, is four additional key establishments and six additional lookups. The key establishments can be done in parallel with existing steps and likewise with the additional lookup, so there is relatively little additional delay.

3.4 Initialization and Network Dynamics

We now consider how a Salsa system could be built up securely and handle nodes entering and exiting the system. One issue for any peer-to-peer system is that an attacker who adds a large botnet's worth of nodes (e.g., 20,000 nodes) could dominate the system when it is relatively small and control most of the system's functions. We see little hope in stopping this through the design of the system – Captcha-inspired Turing tests designed to ensure that a human is using each connection might be tried, but it is beyond the scope of this work [16]. For Tor-like server-based systems, with presumably fewer nodes, we note that a higher barrier to entry is required to ensure that most nodes are at least associated with a unique owner. Again, the best method of doing this is beyond the scope of this work.

We propose that nodes should, if possible, join through trusted friends that already have nodes in the network. This is also the best way of building a small network into a larger one. The new node N can have the friend perform lookups to identify a subset of the nodes in N's group, as determined via a hash of N's IP address. N can then contact these nodes with a *join* message, to which the nodes respond with a full list of the group members. Again we face a performance and security tradeoff between the number of lookups performed by the trusted friend and the correct identification of the full group. However, mismatched lists would alert users to a possible attacker presence in the group, so the attacker must be confident that N only contacts attackers and not other nodes in the group. Otherwise, the attacker nodes must also send the full list.

In the event that new node N has no friends using the system, we propose that a subset of nodes could be advertised on, e.g., a website or a bulletin board. For example, a node may post its own IP and those of its global contacts. Since the global contacts are random and verifiable by hashing the IP with the level of the tree, N has some assurance that they are not an attacker's hand-picked selection. For $g = 256$ groups, $n = 10000$ nodes, and $c = 1000$ attacker nodes, the chance that a specific set of global contacts is all corrupt is approximately $(c/n)^{\log_2 g} = 0.1^8$. The chance that any of the 1000 attackers has a full set of attacker global contacts is:

$$1 - \left(1 - (c/n)^{\log_2 g}\right)^c = 0.00001$$

The user must be sure that nearly all of the contacts connect correctly, lest an attacker node have multiple attacker contacts and provide fake addresses for the rest. Once N connects to the local contacts, she can redundantly request multiple addresses within her new group. The security of redundant requests applies as with normal lookups, and the joining proceedure follows as in the trusted friend case.

Finally we consider what happens when a node leaves the system. The node should, ideally, notify the other group members and nodes that have recently connected to it as a global contact. This should be sufficent to create a smooth transition, as nodes can update their global contacts prior to the next round of requests. The disconnect message must include a brief handshake to prevent spoofed *leave messages* from becoming a denial of service attack or a way to redirect traffic to corrupt nodes. If a node abruptly disconnects, the nodes that had used it as a global contact will find out in the next round of requests and must issue a set of requests to determine the identity of the new global contact.

4. SECURITY ANALYSIS

We now discuss how well Salsa protects against a variety of attacks, including standard attacks against anonymity systems and attacks specific to the Salsa system. One type of attacker that it is critical to defend against is one with many compromised hosts to put into the system. Although this is a major issue for peer-to-peer systems, it can also be an issue in highly-distributed server-based systems in which keeping track of the identity of server operators may not be feasible. With compromised nodes in the system, possible attacks include intersection attacks, control of the initiator's circuit, end-to-end timing analysis, predecessor attacks, and denial of service attacks. We study each in turn.

Intersection Attacks. Intersection attacks are particularly dangerous when the attacker can learn the membership of the set of all users currently active in the system [28]. In Salsa, obtaining this information is significantly harder than in Tarzan or Crowds, but may be somewhat easier than in MorphMix. In particular, the attacker can own one node in each group, which ensures that he knows the membership of each group and therefore the entire system.

We model this attack as a balls-and-bins probability problem, as the placement of an attacker node into a group is random, based on a hash of the IP address [19]. Each bin is a group, and each ball is an attacker node. If there are g groups, the expected number of attacker nodes required to get one in each group is $g \cdot H_g$, where H_x is the x-th harmonic number (roughly $log(x)$). To ensure that there is an attacker in each group with high probability $1 - 1/g$, $O(2g \ln g)$ attacker nodes are required. When $g = 256$, the expected number of attackers needed is approximately 1570, and 2800 attackers will ensure that the attacker belongs to each group with high probability 0.996. When $g = 4096$, the expected number of attackers needed is approximately 36,400, with 68,100 attacker nodes needed to ensure attacker success with probability 0.9998. Clearly, the number of groups is critical to the security of the system. Consequently, the number of honest nodes is critical, as tens of thousands of nodes would be needed to support 4096 groups.

Unfortunately, the attacker can also learn about other nodes by using the lookup process. We note that the attacker must establish keys with each contact except for the final hop, increasing the cost of the attack. To slow the

connection process, each lookup response could introduce a small delay before responding. This delay can keep the attacker from discovering every node quickly enough to be confident that older results still hold; new nodes may have joined in the meantime. A small delay can protect against high-volume attacks, while not greatly inconveniencing users. Secondly, the attacker can never be sure that he has discovered every node in the network without testing every point in the ID-space. Sometimes two nodes will be very close, meaning that the first node owns a very small segment of ID-space and is difficult to detect via random search. Critically, intersection attacks require a complete or nearly complete view of all users. Otherwise, the attacker could fail to find the initiator; even if the initator is observed later in the course of the attack, she will have been intersected out of the possible initiator set.

Control of the Initiator's Circuit. If the attacker can control all of the nodes on the initiator's circuit, he will be able to easily link the initiator with her communications. MorphMix is particularly vulnerable to this attack, as the initator cannot select nodes from the entire system. We claim that Salsa limits the initator's exposure to this attack.

In Salsa, the attacker may attempt to succeed by biasing the random selection of the proxies to get the initiator to select attacker-controlled nodes. Salsa mitigates this threat through redundancy and bounds checking. Despite these mechanisms, the attacker can still bias the selection; we show the extent of this in Section 5. If this bias is limited, the attack becomes similar to random selection. However, given that attackers selected early in the circuit have a greater chance to bias later results, even a moderate bias can lead to substantial increases in the attacker's success rate. We also study this effect in Section 5.

End-to-end timing analysis. The attacker can observe the timings of packets that enter and exit the system and find correlations between the two ends of the same stream [9, 29, 17]. If the attacker controls a subset of nodes, this can occur when the first and last nodes are controlled by the attacker. When each node is selected at random, with replacement, the chance of the attacker being in place is $(c/n)^2$; for a node to attacker ratio of 10.0, this is 0.01. Again, bias in the node selection can increase this chance, and it is critical to minimize this bias. When the attackers are in place, it is not guaranteed that timing analysis will work, but error rates for correlating streams without mixing or cover traffic are low [17]. Thus, we use this as a baseline for other attacks; if the attacker's chance of success in another attack is less than $(c/n)^2$ (and attacks can't be combined), the initiator's vulnerability has not been significantly increased.

Predecessor attacks. Crowds [22] introduced the idea of the *predecessor attack*, which was later extended in [27]. In this attack, the attacker takes observations of path order over time to link the initiator and the responder. The predecessor attack against mix-based systems relies on end-to-end timing or complete circuit control attacks. Thus, defenses against these attacks also inhibit the predecessor attack. In peer-to-peer mix-based systems, the predecessor attack requires $(1/S)\log n$ connections by the initiator to work with high probability, where S is the chance of attacker success in the placing nodes for the end-to-end timing attack [27, 17]. This further demonstrates the need to limit bias in the selection of nodes, as the length of the attack depends inversely on the attacker success rate.

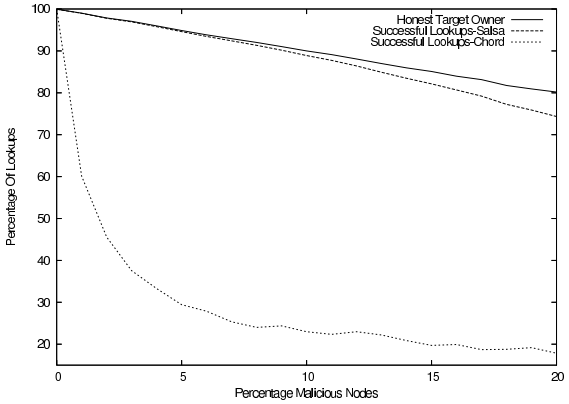

Figure 2: Lookup success: 1,000 nodes, 128 groups, Redundancy=5

Denial of Service Attacks. The last type of attack that we consider is a denial of service (DoS) attack. We do not intend to protect fully against such an attack, as determined attackers with enough resources can always flood the proxies. This is a strength of highly distributed anonymity systems, as there are many proxies to attack. However, we must be careful that our system design does not allow for attackers to easily take down the system without using substantial resources. One possibility for such an attack is to make excessive lookup requests in the network. Each request generates a series of up to $2\log G + 2$ messages, which is good leverage for a DoS attack. Intermediate nodes can introduce additional delay for each request before forwarding it. This provides dual benefits of slowing any DoS attempts and keeping an attacker from quickly learning about the entire network.

Another possibility for attack is to create many computationally expensive key establishment requests. This can be mitigated by starting with a simple handshake process. Each requester initiates key establishment by first sending send a challenge, e.g. a nonce, that the proxy should return along with a second nonce. The second nonce should be returned in the key establishment request, providing weak authentication that the requesting node can receive at its advertised IP address. This is a simple extension of the handshake protocol from Tarzan [13]. Here it provides some assurance that both the requester and the proxy own their respective addresses, and it can be a vehicle for exchanging public keys. The proxy will only begin expensive key establishment with nodes that it has done this handshake with and can limit each requester to one request per fixed time unit. Thus, an attacker has more work to do per request, and has limited requests per machine per time unit, making the DoS attack more difficult.

5. SIMULATION RESULTS

We performed a number of experiments aimed at testing the security properties of the Salsa system, as described in Section 3. Specifically, we demonstrate the effectiveness of redundant node selection and bounds checking in limiting the attacker's ability to bias node selection. Additionally, we show the extent to which the levels of attacker bias in the system lead to compromise of the initiator's path.

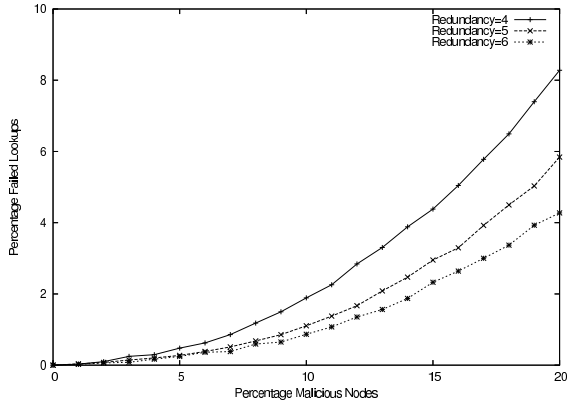

Figure 3: Percentage of Failed Lookups vs Redundancy: 1,000 nodes, 128 groups, 20% malicious nodes

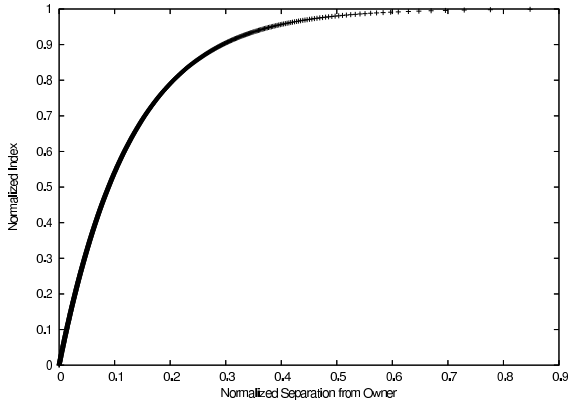

Figure 4: Probabilistic Checking: 1,000 nodes, 128 groups

For our tests, we used a 30-bit hash space and considered systems of 1,000 and 10,000 nodes. We used $G = 128$ and $G = 256$ groups for 1,000 nodes and $G = 256$, $G = 512$, and $G = 1024$ for 10,000 nodes. For each test, we simulated 1,000 separate systems and made 1,000 lookups per system. The system was simulated in Java; we do not simulate in detail the underlying network, encryption, or other system details. Rather, the experiments focus on the security of the system under varying degrees of attack. We tested with the percentage of malicious nodes ranging from 0% to 20%. Beyond 20% malicious nodes, greater than 4% of all paths are compromised due to end-to-end timing analysis attacks.

We claimed earlier that the id-space is organized such that it provides resilience to attacker-induced bias. Here we assume that once a node chooses a malicious node as a proxy on the circuit, that route fails. There are two ways to select a malicious node:

- When a node chooses an ID at random for one of the proxies on the circuit, that ID could be owned by a malicious node.
- While performing a lookup for an ID, the results of the lookup are modified by attackers to be a malicious node.

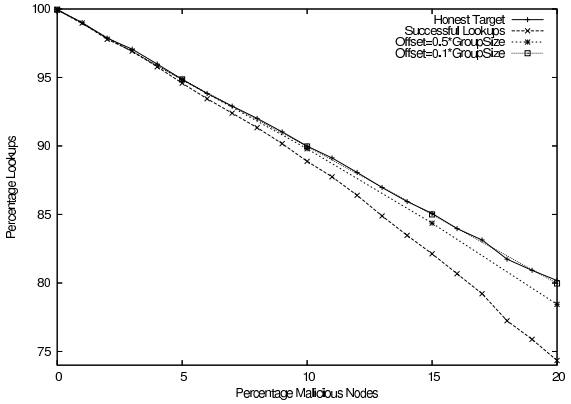

Figure 5: Lookup Success With Probabilistic Checking: 1,000 nodes, 128 groups

To perform a lookup we choose one node at random from the set of all nodes (assuming that it is honest for simplicity). Let us call this the *source node*. We then choose an ID at random from the entire ID-space, and we call this the *target ID*. The source node then performs a lookup for the target ID. The chance that the target ID is owned by a malicious node is approximately given by the fraction of malicious nodes in the system. As nothing can be done about the selection of these malicious nodes, we call these *abandoned lookups*. We aim to test the case where the target ID is owned by an honest node, as this is the chance for the attacker to bias the node selection to an attacker-controlled node. If we choose an honest target ID and the lookup result has been biased in this way, we say call it a *failed lookup*. Correspondingly, we define *successful lookups* as those in which the target ID was owned by an honest node and that node was returned. We checked the percentage of the total lookups that were successful, and compared the results with the number of lookups not abandoned. The difference– the percentage of failed lookups–shows the resiliency of the system to attackers randomly distributed over the ID-space.

We use redundancy to minimize the effect of running into malicious node while lookup up an ID. The level of redundancy, defined by the number of redundant lookup requests per lookup, is a tunable parameter to balance security with performance. The more redundant lookup requests the initiator uses, the greater the chance of getting a successful lookup. However, more lookups mean greater overhead in terms of numbers of messages. We show results for systems using between three and six redundant lookups.

To study the use of bounds checking, we evaluate the system using different bounds, with results for both false positives and false negatives. When a lookup result falls outside the bound, the initator must perform a lookup for a new ID. This increases the overhead of the system, but provides greater security, and we study this tradeoff as well.

5.1 Results

We now present the results of our simulations.

For the first set of experiments we use a redundancy of five, i.e., for groups of greater than five nodes, five of the local contacts were asked to do the lookup. As shown in Figure 2, about 20% of the target IDs were owned by ma-

R	Avg. min. group size	Avg max. group size	Avg. messages sent	Avg. max pathlength
4	1.68	15.85	39.63	8.92
5	1.74	15.94	48.53	8.94
6	1.68	15.98	56.42	8.94

Table 1: Group, Path, Message Statistics: System Details: 1000 nodes, 128 groups, 10% malicious nodes.

Max. offset	False Positives	False Negatives			
		5	10	15	20
0.5	2.0%	9.2%	17.0%	24.2%	29.9%
0.1	45.8%	0.95%	2.0%	2.9%	3.8%

Table 2: Probabilistic checking

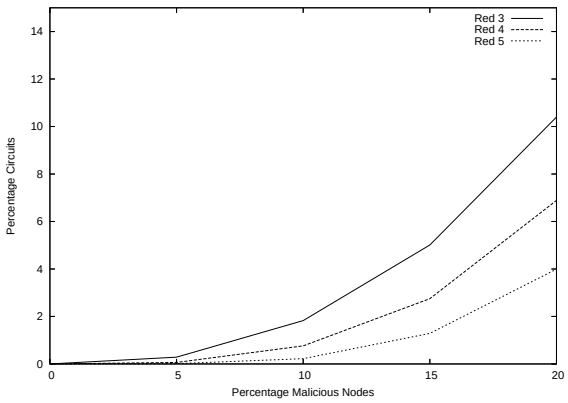

Figure 6: Lookup Success With Probabilistic Checking: 1,000 nodes, 128 groups

licious nodes when there were 20% malicious nodes in the system. The value is consistently close to the percentage of malicious nodes in the system, as expected. We see that even when there are 20% malicious nodes in the system, the number of failed lookups amounts to less than 6%. The naive system, using Crowds, fails badly. Figure 3 shows the results for varying levels of redundancy. As we increased the redundancy from four to five the percentage of failed lookups decreases by 2.44% and a further 1.57% as we increased the redundancy from five to six.

Next we did experiments to check general statistics of the lookups and systems. Table 1 shows information like the average minimum and maximum group population, the number of messages exchanged to perform lookups as well as the average maximum pathlength. The number of messages indicates the overhead the redundancy places on the system. The average maximum pathlength indicates the delay induced in the system. Since the lookups are similar to a binary search on the groups, we expect the number of lookup hops to be about $log_2(G)$. The results show that to be true. Here each of the group members asked to do a redundant lookup is considered the first lookup hop.

As is expected the number of messages exchanged during lookups increases as the redundancy is increased. Of course, the other values remain the same since the redundancy doesn't affect any of them.

We also show how well bounds checking works. Figure 4 shows how IDs and their corresponding owners are distributed. The values are normalized by the group sizes. Table 2 shows the results numerically. False positives mean that the honest pick was out-of-bounds. False negatives mean that the attacker pick was in-bounds. The false positive rate is high, which results in the requesting node needing to check multiple random IDs until she gets a good one. If the false positive rate is given by fp, the user would need to test $1/fp$ IDs on average and will succeed with probablity $1 - fp^k$ after testing k IDs. For example, with a 50% false positive rate, the user must test two IDs on average and no more than ten with 99.9% probability. As shown in Figure 2, at 20% malicious nodes in the system, only about 6% lookups turn false, so to pick the highest value of false negatives in the table i.e., 29.9% means 29.9% of those 6% or 1.8% of the total lookups returned malicious nodes and went undetected. This can be seen in Figure 5. With the offset set at $0.5groupSize$ there is a non-negligible false negative rate, but we still get a substantial decrease in the attacker's ability to bias the lookup results. If we set the offset more strictly, e.g. to $0.1groupSize$, we greatly reduce the false negative rate at the cost of more lookups due to more false positives.

Finally, we demonstrate that redundant checking continues to operate over the length of the path to prevent bias in node selection. In Figure 6, we show the chance that both the first and last nodes are compromised. As shown, the attacker does not gain much over systems with full random selection, such as Tarzan. For example, with 20% attackers, Tarzan would have 4% compromised paths, while for Salsa with redundancy of four, this chance is 6.9%. With redundancy of five, the chance is the same 4%.

6. CONCLUSIONS

In this paper we proposed a structured overlay architecture, based on a DHT, for securely and scalably organizing and selecting nodes in a highly distributed anonymous communications system. Nodes have limited knowledge of the system, while still being able to choose proxies at random from the entire system. This helps protect it from the intersection attack and enhances scalability. Due to the use of consistent hashing, attackers cannot influence where the nodes fall on the ID space. We have shown that the system organization helps its resilience to attack. The system uses redundant lookups to mitigate the risk of finding attackers on the path. This redundancy is a tunable parameter to balance performance and security. The system also uses distance checking to find malicious nodes returned by lookups. We show that we can reduce the false negatives to 3.8% in a system containing 20% malicious nodes. We also show that for an attacker to try to perform the intersection attack, he'd have to control a large fraction of the nodes in the system.

7. ACKNOWLEDGMENTS

We would like to thank the NSF for support of this project with grant CNS-549998. Additionally, thanks to Dawn Song for initial discussions about this project and Roger Dingledine for his input about Tor.

8. REFERENCES

[1] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In *Proc. Financial Cryptography*, Jan 2003.

[2] A. Back, I. Goldberg, and A. Shostack. Freedom 2.0 security issues and analysis. Zero-Knowledge Systems, Inc. white paper, Nov 2000.

[3] O. Berthold, H. Federrath, and M. Kohntopp. Project anonymity and unobservability in the Internet. In *Proc. Computers Freedom and Privacy*, April 2000.

[4] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix-routes and how to overcome them. In *Proc. Intl. Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[5] N. Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays*. University of California, Berkeley, CA, 2005. Ph.D Thesis.

[6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, Feb 1981.

[7] G. Ciaccio. Improving sender anonymity in a structured overlay with imprecise routing. In *Proc. Privacy Enhancing Technologies Workshop (PET)*, June 2006.

[8] G. Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proc. Security and Privacy in the Age of Uncertainty (SEC 2003)*, pages 421–426, May 2003.

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

[10] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant DHT routing. In *Proc. ESORICS*, Sep. 2005.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The next-generation Onion Router. In *Proc. 13th USENIX Security Symposium*, August 2004.

[12] J. Douceur. The Sybil attack. In *Proc. IPTPS*, Mar. 2002.

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

[14] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. 12th International World Wide Web Conference*, 2003.

[15] J. Kirk. Botnets shrinking in size, harder to trace. `http://tinyurl.com/nfxgk`, Jan. 2006.

[16] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proc. Eurocrypt*, 2003.

[17] B. N. Levine, M. Reiter, C. Wang, and M. Wright. Timing analysis in low-latency mix systems. In *Proc. Financial Cryptography*, February 2004.

[18] N. Mathewson and R. Dingledine. Practical traffic analysis: extending and resisting statistical disclosure. In *Proc. Privacy Enhancing Technologies workshop (PET 2004)*, May 2004.

[19] R. Motawani and P. Raghavan. *Randomized Algorithms*, chapter 3. Cambridge University Press, 1995.

[20] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.

[21] M. Reiter, X. Weng, and M. Wright. Building reliable mix networks with fair exchange. In *Proc. 3rd Applied Cryptography and Network Security Conference (ACNS)*, June 2005.

[22] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM TISSEC*, 1(1):66–92, Nov 1998.

[23] M. Rennhard and B. Plattner. Practical anonymity for the masses with MorphMix. In *Proc. Financial Cryptography (FC '04)*, February 2004.

[24] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for Internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[25] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of Onion Routing security. In *Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[26] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In *Proc. Privacy Enhancing Technologies Workshop (PET)*, June 2006.

[27] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proc. ISOC Sym. on Network and Distributed System Security*, Feb 2002.

[28] M. Wright, M. Adler, B. Levine, and C. Shields. Defending anonymous communications against passive logging attacks. In *Proc. IEEE Sym. on Security and Privacy*, May 2003.

[29] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proc. Privacy Enhancing Technologies (PET)*, May 2004.