Hashing it out in public

Common failure modes of DHT-based anonymity schemes

Andrew Tran
Carnegie Mellon University
Pittsburgh, PA 15213
qtran@andrew.cmu.edu

Nicholas Hopper
University of Minnesota
Minneapolis, MN 55455
hopper@cs.umn.edu

Yongdae Kim
University of Minnesota
Minneapolis, MN 55455
kyd@cs.umn.edu

ABSTRACT

We examine peer-to-peer anonymous communication systems that use Distributed Hash Table algorithms for relay selection. We show that common design flaws in these schemes lead to highly effective attacks against the anonymity provided by the schemes. These attacks stem from attacks on DHT routing, and are not mitigated by the well-known DHT security mechanisms due to a fundamental mismatch between the security requirements of DHT routing's put-get functionality and anonymous routing's relay selection functionality. Our attacks essentially allow an adversary that controls only a small fraction of the relays to function as a global active adversary. We apply these attacks in more detail to two schemes: Salsa and Cashmere. In the case of Salsa, we show that an attacker that controls 10% of the relays in a network of size 10,000 can compromise more than 80% of all completed circuits; and in the case of Cashmere, we show that an attacker that controls 20% of the relays in a network of size 64000 can compromise 42% of the circuits.

Categories and Subject Descriptors

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

General Terms

Security

Keywords

Anonymity, Selective Denial of Service, Peer-to-Peer Networks

1. INTRODUCTION

Anonymous overlay networks allow arbitrary Internet hosts to communicate while attempting to conceal the correspondence between initiators and responders from the members of the overlay, that is, hiding who communicates with whom. These networks are an important tool for online privacy; censorship resistance, in the form of blocking circumvention; surveillance evasion; and safeguarding freedom of expression online. The most popular such network, Tor, has approximately 1,700 simultaneous servers and an estimated 100,000 simultaneous users.

One challenge in the design and deployment of anonymous overlay networks is the relationship between the level of security provided and the number of users: ceterus paribus, hiding among more users provides stronger anonymity. Thus features such as usability, latency, and scalability also have a direct effect on the level of security provided by a system. In fact, it has been observed that several anonymity schemes with differing capacities have stabilized with similar performance, perhaps because any further load results in unacceptable performance; thus scalability in particular is an important feature for anonymity.

Among the deployed anonymous overlays, the most scalable design for achieving anonymity seems to be Tor [15]. Tor achieves anonymity by having clients route traffic over virtual "circuits" that are established by randomly choosing a sequence of three relay servers from a global list. Tor clients establish a circuit by successively exchanging keys with the relays in the sequence, in a "telescoping" fashion so that only the first relay communicates directly with the client. Several other schemes, including Tarzan [17] and Crowds [32], have a similar flavor but vary in the length of the route, level of encryption, and distinction between clients and servers. All of these schemes, however, rely on a global list of all participants being circulated to all participants so that selecting a series of relays requires quadratic communication costs, limiting to some extent the scalability of these schemes.

One natural alternative to using a global list of relays is to organize routers into a distributed hash table (DHT). A DHT is a decentralized, distributed system that assigns a logical identifier to a set of N nodes and provides algorithms to efficiently locate the node responsible for an arbitrary identifier, typically given knowledge of $O(\log N)$ other nodes. Typically this "routing" layer is used to implement a hash table put/get interface allowing peers to store a value associated to a key at several "replica" nodes with identifiers closest to the key. In principle such schemes would present a scalable method to find relays for building tunnels; it is thus not surprising that many proposals for anonymity schemes based on DHT overlays appear in the literature [29, 30, 44, 24, 20, 22, 7, 23, 43, 2].

Unfortunately, schemes based on DHTs have proven hard to secure. Several previous works [25, 4] have suggested that existing redundancy mechanisms for securing DHT lookups seem to provide a trade-off between active and passive attacks. In this paper, we show that the problems with this approach are much more fundamental: the "security" goals of DHT lookup (contact at least one replica) and relay selection (select an unbiased node) are fundamentally mismatched, and as a result the schemes based on these mechanisms are dramatically more vulnerable than previously suspected. In particular, we describe new active attacks that are more

WPES'09, November 9, 2009, Chicago, Illinois, USA.

powerful than the previously-identified passive attacks, regardless of the redundancy level. Our attacks target two common "failure modes" of such schemes that allow an adversary who controls a fraction f of the nodes to compromise a fraction larger than f of the circuits (compared with fraction f^2 of circuits[1] in case every node knows all other nodes). The common theme between these failure modes is that a DHT lookup necessarily involves $O(\log n)$ nodes, so that with very high probability the results of any lookup can be influenced by some adversarial node. In essence, while previous attacks show how a particular routing mechanism can give a local attacker some of the capabilities of a global passive adversary, our attacks show that essentially every DHT-based anonymity scheme promotes a local attacker to a global *active* adversary.

Several schemes have additional mechanisms that are intended to deal with this phenomenon and the related problem of honest nodes leaving the network suddenly. We show how two such schemes, Salsa [29] and Cashmere [44], are vulnerable to these failure modes in spite of their additional complexity. In each case our attacks significantly improve on the best previous attacks in the literature. For example, Mittal and Borisov [25] show that an attacker that controls 20% of a 1024-node network can compromise 25% of the circuits in Salsa. Our attack compromises this fraction of the circuits when it controls only 5% of the nodes, and given control of 20% of the nodes, we compromise 99% of the circuits. For Cashmere, we show that with 64,000 peers an adversary that controls 20% of the nodes can completely compromise 42% of the circuits, whereas the analysis in [44] suggested that 90% of malicious nodes are required for effective traffic analysis.

Our attacks have significant implications for the construction of Peer-to-peer anonymity protocols: *using a "secure" or "robust" DHT routing layer is not sufficient for anonymity*. Indeed, the sense in which a lookup must be secure is quite different between the two, because in a robust DHT the content can be independently verified whereas when DHT routing is used for relay selection it is the destination that must be verified, and furthermore the source of a lookup itself must be confidential. The central question in designing DHT-based anonymous overlays then becomes whether DHTs can be adapted to support these requirements.

Outline

The remainder of this paper is organized as follows. In Section 2 we describe in more detail the key notions from DHTs and anonymous communication necessary for understanding our attacks. Section 3 describes our new attacks in terms of a "generic" model of a DHT-based anonymity scheme. We demonstrate two variations of this model that give rise to the different failure modes and compute the success of attacks exploiting these failure modes. In Sections 4 and 5 we present specific attacks based on DHT routing and denial of service on Salsa and Cashmere. Finally, section 6 discusses in more detail the relationship between our attack and other attacks proposed in the literature.

2. BACKGROUND

2.1 Relay-based anonymity

All of the anonymity schemes we consider in this paper are based on a "circuit" model inherited from Onion Routing and Tor [2] and are intended to provide "low-latency" connections for interactive applications such as web browsing and file sharing. In this model, the initiator of a session builds a "circuit" of nested encrypted channels over which the session will be relayed to the responder. Thus, the initiator initially selects a random relay R_1 and establishes a secure channel with R_1 to build the first stage of the circuit. At each subsequent stage the circuit is extended to another randomly selected relay; each stage of the circuit adds another (inner) layer of encryption to the messages sent by the initiator. These nested secure channels can be established noninteractively by repeated decryption and forwarding of a single message, or interactively, by passing messages along the already established portion of the circuit. By default, a Tor circuit has three relays; where the length of a circuit is relevant we analyze an arbitrary circuit length ℓ. In most of the schemes we analyze, the final node in the circuit can relay the session contents to an arbitrary Internet host, but several schemes allow communication only between members of the overlay network.

In schemes based on circuits, the anonymity comes primarily from the fact that the initiator of the circuit is known only to the first relay and the responder is known only to the final relay. It is assumed that a global adversary can use timing analysis to discover the correspondence between the initiator and the responder, so the typical goal is to resist attacks by a "local" adversary that controls only a fraction f of the nodes in the system (and is possibly a responder for one or more sessions) and does not see any other traffic.

In this case, it is known that timing analysis will allow an adversary who controls the first and last relay of a circuit to discover that two streams belong to the same circuit and link the initiator to the responder. We call this result - linking the initiator and responder of a circuit with high confidence - a circuit *compromise*. An adversary that controls f fraction of nodes can thus compromise f^2 fraction of circuits without any additional attacks. In some of the schemes we consider, an attacker may be able to significantly reduce the number of possible initiator-responder pairs for a circuit. If the adversary can produce a list of ω possible initiator-responder pairs with high confidence, we say that a circuit is ω-compromised. (Thus a complete compromise is a 1-compromise). We use the fraction κ of circuits ω-compromised for a given fraction of adversarial nodes as our metric of the effectiveness of an attack, with higher values of κ and lower values of ω indicating more effective attacks.

2.2 Distributed Hash Table Overview

A distributed hash table (DHT) is a distributed system that provides efficient lookup of key-value pairs. DHTs usually have several common properties that allow them to achieve this.

Each node in a DHT is assigned a unique identifier, called its *nodeID*, uniformly distributed in a large key space. Application specific data are also assigned identifiers, called *keys*, from the same ID space. Often, DHTs have a notion of distance between two nodes based on prefix matching, where the more bits of the prefix two nodes share the closer they are. The overlay designates ownership of a set of keys to a single unique live node. If a node owns a key, then the node is said to be the $root$ of that key.

To route queries efficiently, every node maintains a routing table consisting of multiple entries of the form nodeID, IP address, and port. In general, the entries in a node's routing table are chosen to efficiently route queries while maintaining limited network state. For example, in Chord [37], a node with nodeID K maintains a routing table with $O(\log n)$ distinct entries of the form $entry_i = (nodeID_i, IP_i, port_i)$, where $nodeID_i = root(nodeID+2^{i-1})$.

Queries can proceed either iteratively or recursively. In recursive routing the source will delegate control of its query to a node in its routing table closer to the target. The node, who receives a query,

[1]Under selective DoS [4] the fraction of compromised circuits is actually $\frac{f^2}{f^2+(1-f)^3} \approx f^2 + 4f^3$

[2]We note that several of the schemes substitute a "group" of nodes in place of a relay at some points in our description

passes control of the query to another node with a longer prefix match with the key. Nodes repeat this process until a node decides if it is the root of the key and passes the answer back (either by reversing the route from the source or by directly sending the answer to the source). For example, in Figure 1 (a), the source S wants to recursively find the root of the key K. First, S contacts A and control of the look up passes to A. Second, A contacts B and control of the look up passes to B. B contacts R. Then, R decides that it is the root of K and notifies B that it is the root of K. B tells A that R is root of K. Finally, A tells S that R is the root of K. In some cases, R sends the message directly to S.

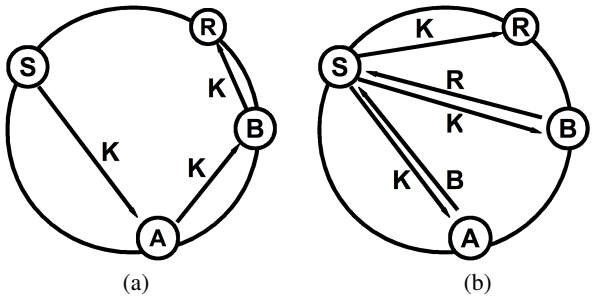

Figure 1: Recursive and Iterative Routing

Iterative routing starts with the source asking the node in its routing table with the greatest prefix match to the target for a node with an even longer prefix match. The source repeatedly queries the results for nodes with longer prefix matches until the source cannot find any node closer to the key. For example, in Figure 1 (b), the source S wants to iteratively find the root R. First, S contacts A and A responds with B. Second, S contacts B and B responds with R. Finally, S contacts R and decides R is the answer.

In both cases, at each step of the algorithm the distance to the target root is halved, and, therefore, the process takes $O(\log n)$ steps.

2.3 Attacks on Distributed Hash Tables

2.3.1 *Query Capture*

DHT routing has two main security problems: the first is that the chance of a query passing through a malicious node is very high, and the second is that it is very hard to determine whether the result of a query is correct.

Suppose a query uses on average p hops to traverse the DHT. If any of the p hops are controlled by the attacker, then the query is considered captured. Thus, the probability that a query is captured is $1-(1-f)^p$, where p is $O(\log n)$ in most DHTs and n is the size of the network. This is an overestimate since it assumes every query takes $O(\log n)$ hops every time. For example, Chord's routing uses the maximum number of hops on half of the queries, the maximum number of hops minus one on one fourth of the queries and so on. Adjusting for a Chord-like routing scheme, the probability that a query is captured is

$$1-\sum_{i=1}^{p}\frac{(1-f)^{p-i+1}}{2^i} = 1-\frac{(1-f)^{p+1}((2(1-f))^p-1)}{(2(1-f))^p(1-2f)}, \quad (1)$$

where p is the maximum number of hops and f is the fraction of malicious nodes.

Figure 2 shows the fractions of queries captured by at least one malicious node by plotting eq. (1) for varying values of f and p. Significantly more queries will be captured as f and p grow; in particular when $p=\Omega(\log n)$ then the probability of query capture is $1-O(n^{-f})$.

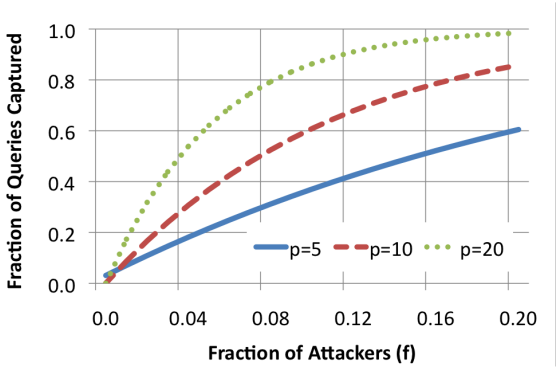

Figure 2: Fraction of queries captured for maximum path length $p \in \{5, 10, 20\}$.

The second attack involves a malicious node forwarding queries to other malicious nodes or even non-existent nodes. For instance, upon receiving a request for a key K the adversary simply returns the IP address of a malicious node. Unless the victim has detailed knowledge of the neighborhood of the destination ID space, it is difficult for the victim to verify whether or not the returned result is correct. A malicious node on receiving a query has three options: 1) to forward the query to a malicious node, 2) to drop the query and 3) to log the query. The probability of successfully routing between two honest nodes is only $O((1-f)^{\log n})$ approximately.

2.3.2 *Mitigating Attacks on Routing*

One way to prevent attackers from claiming an arbitrary portion of the ID space is to make nodeIDs verifiable. For example, a node's ID could be defined as the hash of its IP address. Hash of IP address provides a weak form of authentication between nodes because everyone can compare the hash of the IP address with a node claiming to be a given nodeID. Due to this property, attackers cannot attempt to place a node somewhere in the ID space without controlling an IP address that maps to the desired space. Using hash of IP address as nodeID has the drawback of being unable to cleanly support NAT boxes without making some sort of security trade off: for example, if nodeID is the hash of IP address and port, then an attacker can generate many uniformly distributed nodeIDs as before.

One way to reduce the chance of query failure is to send queries through multiple routes simultaneously. The chance of a query failing may be reduced from the probability that the attacker controls any of the nodes on a given path to probability that the attacker controls a node on every path. In order for redundant routing to be useful special precautions must be taken such that the redundant paths do not converge. Furthermore, redundant routing has the drawback of increasing bandwidth overhead. More importantly, in the context of anonymous networks redundant routing is "noisier" than single path routing: many more nodes are contacted by the initiator, potentially compromising the initiator's identity [25]. Furthermore, redundant routing can improve the odds of at least one query reaching the target, but may not resolve the issue interpreting conflicting results if different paths report different roots. This is not a trivial task because without detailed information about the target's neighborhood it is hard for the source to determine which results are the correct ones and furthermore, on average the majority of the paths will be adversarially controlled.

The "density check" method can be used to partially mitigate a node's lack of detailed knowledge about different regions of the ID space. The checks are made on the assumption that malicious nodes are less dense in the ID space versus honest nodes [34]. The density check tests whether the distance between a result node and

the key is consistent with the distribution of node IDs near the initiator; if the distance is greater than some multiple (e.g. $1.5\times$) of the average distance between nodeIDs (near the initiator) the result is rejected. A drawback of the density check is that it requires large n to be accurate. If n is too small, then the variation in density between honest nodes can be too high to make a useful check.

3. GENERIC FAILURE MODES OF DHT-BASED RELAY SELECTION

In this section, we discuss two related failure modes of DHT-based anonymous overlays: *insecure relay selection* and *overlay circuit extension*. We describe new attacks on generic schemes exhibiting these failure modes. Recall that in a generic relay-based anonymity scheme a circuit is established by iteratively extending the circuit to a series of ℓ randomly chosen relays; essentially the "algorithm" for such a scheme is:

1. Set $R_0 = I$ (the initiator), $i = 0$
2. While $i < \ell$:
   (a) Select relay R_{i+1}
   (b) Extend the circuit from R_i to R_{i+1}

The "relay selection problem" is to efficiently and privately choose an unbiased relay for the next hop of a circuit. As noted in the introduction this problem is easily solved if every node has a complete list of other nodes in the system, but this approach does not scale. Using a distributed algorithm to address efficiency concerns may introduce the possibility of failures in unbiasedness or privacy.

In particular, a "generic" DHT-based anonymity scheme implements step 2a by uniformly choosing a key k, using the DHT to look up the root of k, and selecting $R_{i+1} = root(k)$. In expectation, when all nodes follow the protocol, this results in an unbiased relay selection. However, the "randomness" (lack of bias) of the query depends critically on the *correctness* of the query result: if lookups are susceptible to attacks that return incorrect or nonexistent results, the relay selection may be biased towards adversarial nodes. Additionally, if a DHT queries can be linked to its initiator, then the privacy of the relay selection may be undermined, allowing an adversary who controls the final relay of the circuit to determine the initiator without controlling the initial relay.

In some of the schemes we consider, step 2b is also accomplished via the DHT: rather than contact R_{i+1} directly, R_i sends its communications over the DHT routing layer. This introduces additional opportunities for both adversarial observation and interference.

3.1 Insecure Relay Selection

Among anonymous overlays that use recursive routing for relay selection, and direct connections for circuit construction, a common failure mode is a lack of proper security measures applied to DHT lookups. While some schemes include traditional DHT security methods such as redundant routing and verifiable addresses, this is not in general sufficient to prevent a bias towards malicious nodes. This is due to a mismatch between the security goals of these measures and the security requirements for anonymity: Castro *et al.* [5] proposed these measures assuming the presence of "content authentication:" the ability to verify that a particular value stored for a key is "correct." Under this assumption, it is sufficient to guarantee that at least one query for a given key k reaches at least one neighbor of $root(k)$; if other queries are dropped or return incorrect results, content authentication can be used to recognize the correct response.

In contrast, in the relay selection setting, there is no clear method to verify whether a particular peer is the current root of a key. In the most simplistic setting, this means that a malicious node could simply always claim to be the correct result of a query, compromising a large majority of circuits. Even if measures are taken to make it difficult to claim proximity to arbitrary keys, most of the nodes in a peer-to-peer system are offline at any one time, so an attacker may effectively cause queries to be dropped by returning "close" nodes that are offline; when the initiator (or the current endpoint of a circuit) attempts to extend a circuit to such a node it will fail, requiring a new lookup or other fallback option (for example, in Tor, if a circuit extension fails, the circuit is torn down.) Using threshold or voting-type schemes to determine the best result may be unreliable as well, since asymptotically the probability of query capture approaches one, meaning that in expectation the majority of redundant routes will pass through a malicious node. This ability to effectively drop queries can significantly bias the selection of relays toward malicious nodes.

In particular, consider the following selective denial of service attack on a generic anonymous peer to peer system's circuit construction using recursive routing for random node selection. Source S wants to build a tunnel out of the roots of the keys K_1, K_2, and K_3. First, S does a lookup for K_1. With probability q the attacker can capture S's query. Suppose the attacker captures the query. If the attacker controls malicious node M_1 close enough to K_1, an event with probability at least f, the attacker returns M_1 and otherwise the attacker drops the request. Now S requests M_1 to lookup the root of K_2. M_1 returns this node, N, whether it is malicious or not. Finally, N queries the DHT for K_3; with probability q this query is captured as well, and if the attacker controls a nodes M_3 close enough to K_3 then M_3 is returned and otherwise the query is dropped. If N extends a circuit to M_3 the circuit is compromised; if it does not, then M_1 drops the circuit. Adversarial nodes always drop circuit requests that come from relays that are not part of circuits that begin at adversarial nodes.

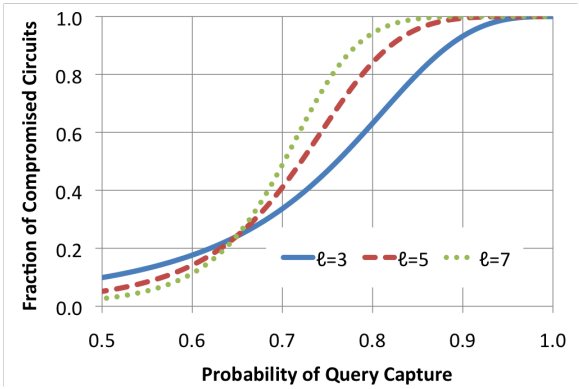

Figure 3: Fraction of circuits compromised as a function of probability of query capture, given $f = 0.1$.

Under this scenario, a circuit can only be built successfully in two cases. In the first case, none of the nodes $root(K_1)$, $root(K_2)$, $root(K_3)$ are adversarially controlled and none of the queries for these nodes is captured; the probability of this event is $(1-q)^3(1-f)^3$. Second, the adversary controls nodes sufficiently close to K_1 and K_3 to be accepted as the roots of these keys; in this case (whether or not the queries are captured) the circuit is compromised, and the event has probability at least f^2. Thus, conditioned on successfully building a circuit, the probability of compromise is $\frac{f^2}{f^2+(1-q)^3(1-f)^3}$. Generalizing to circuits of length ℓ gives a fraction $\frac{f^{\lfloor \ell/2 \rfloor+1}}{((1-q)(1-f))^\ell+f^{\lfloor \ell/2 \rfloor+1}}$ of compromised circuits. Figure 3 shows how the probability of compromise grows as a function of q when $f = 0.1$.

Remark. In recursive routing, it is difficult to attribute query failure to the responsible node, since the source can not see each step of the recursive query. If a node fails somewhere along the path from the source's perspective only the next node has failed. Suppose source S has a recursive query that uses nodes A, B, C. S can not distinguish between failures at A, B, and C, because from S's perspective it only appears that A has failed. As a consequence, malicious nodes can be selectively uncooperative without being blacklisted.

Iterative routing has the property that it is easier for the source to recover from node failure. The source can see all of the nodes in an iterative query because it talks to them directly. If a node fails, the source can just timeout on that node, and ask other node. Since the source talks to all of the nodes in the query directly the source can decide not to use poorly performing nodes. However, use of iterative routing leads to passive attacks on anonymity similar to those discussed by Mittal and Borisov [25]. Combined with selective DoS at the relay level these attacks (on iterative routing) are similar in effectiveness to the one described above.

3.2 Overlay Circuit Extension

Many DHT-based anonymous overlays make additional use of the DHT overlay: rather than simply looking up relays over the DHT and then connecting directly to the roots, the actual traffic carried by the overlay is delivered over the DHT by recursive routing. The most common motivation for this decision is to improve reliability and resistance to predecessor attacks [32, 42]: if nodes join and leave the network at a high rate as in other peer-to-peer systems, then circuits established over these nodes may be unreliable, so circuits are instead established over "logical" keys so that when one relay churns out its DHT neighbor can continue to receive and process messages.

Unfortunately this opens the door to potentially more powerful attacks, since adversarial nodes gain the ability to *conditionally* allow circuits to be constructed and then close these circuits (by dropping all messages sent over the DHT to a particular ID) when additional information becomes available. In particular, when considering the "insecure relay selection" attack from section 3.1 we observe that it is no longer necessary to control any of the "interior" relays on a circuit to compromise the circuit. If for every i, there is a malicious node on the DHT path between R_{i-1} and R_i, or R_i is malicious, the adversary can discover the entire relay path $R_1, \ldots, R_\ell$. In order to compromise the circuit, it is then only necessary to control R_ℓ and to identify the initiator.

Identifying the initiator is not as straightforward due to the use of recursive routing between the initiator and the first relay, but we note that in the worst case of Chord-style routing, $\frac{1}{2}$ of all routes take the maximum number of DHT hops and the first hop can identify the initiator with certainty. Many other DHTs use "wide" routing tables that have the effect of decreasing the maximum number of DHT hops but increasing the fraction of keys that require the maximum number. So assuming Chord-style routing and a probability of q of DHT route capture, a circuit is compromised with probability $(f^2/2)(q + f - qf)^{\ell-2}$. On the other hand, the malicious nodes can allow all circuit establishment messages to be delivered, and if the adversary does not observe a circuit with ℓ links such that the first DHT hop and final relay are malicious, the traffic along the circuit can be dropped. Thus in order to successfully route traffic over an uncompromised circuit, none of the relays can be malicious, and none of the routes between them can be captured, which happens with probability $((1-q)(1-f))^\ell$. Figure 4 shows how the fraction of compromised circuits (conditioned on successful completion) varies with f, N, and ℓ.

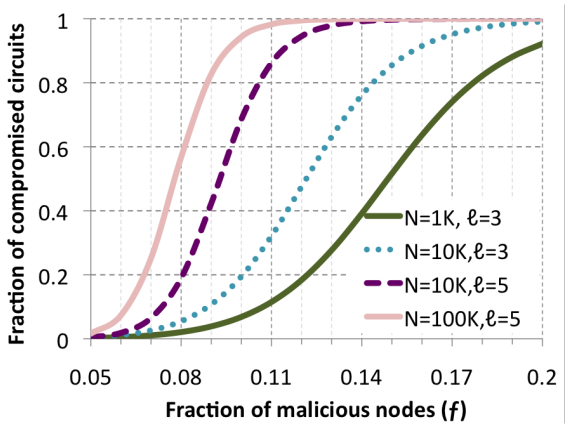

Figure 4: Overlay circuit extension: Fraction of circuits compromised as a function of fraction of compromised nodes, for varying network size (N) and circuit length (ℓ), assuming Chord-style routing.

3.3 Survey of vulnerable schemes

In a literature survey, we identified 10 DHT-based anonymous overlay networks. While two of these schemes specify mechanisms to prevent DHT lookup failures through redundancy [24, 29], the majority use overlay circuit extension with no provisions for redundant routing and provide some sort of robustness mechanism at a level above the overlay [30, 44, 43, 2, 23]. The remainder make no provisions for robustness [22, 7, 20], and are thus vulnerable to our generic attacks with no modification. Table 1 summarizes the schemes and which failure modes they exhibit; in the remainder of this paper we give examples showing how our generic attacks can be adapted to the specific robustness mechanisms of two representative schemes: Salsa [29] and Cashmere [44].

Insecure relay select	CORE [22], Salsa [29], AP3 [24]
Overlay circuits	Bluemoon [30], Information Slicing [20], Cashmere [44], TAP [43], GAP [2], WonGoo [23], NEBLO [7]

Table 1: Summary of failure modes in 10 DHT-based anonymity schemes

4. SALSA

4.1 Overview of Salsa

Salsa [29] is a DHT-based layered circuit anonymity scheme. Salsa relays are organized into a Chord-like recursive DHT for node selection, where each node's ID is determined by applying a cryptographic hash function to its IP address. Salsa is explicitly designed to resist attacks on its DHT functionality, assuming a fraction $f < 0.2$ of malicious nodes, through the aforementioned use of verifiable IDs, bounds checking and redundant lookup.

Salsa's use of bounds checking is straightforward: a node discards a lookup result if the result is greater than a threshold distance d (based on the average distance between IDs of the node's DHT neighbors) away from the target ID. When the bounds check fails the lookup is aborted and the node searches for a new key. The bounds check may generate false positives that cause Salsa to drop correct results, but the authors show that the bounds check has reasonable accuracy when the number of attackers is below 20%. In addition to bounds checking, Salsa uses a binary tree structure that ensures that nodes share very few global contacts on average. This

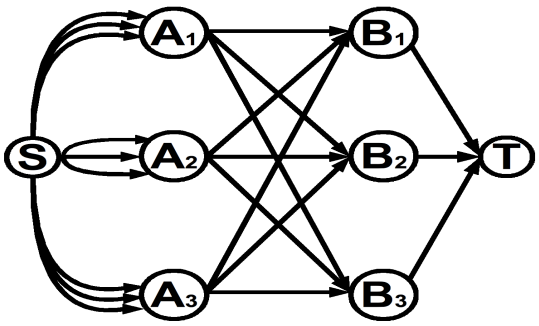

Figure 5: Salsa Circuit Construction

ensures that the paths taken by redundant lookups are likely to be disjoint so that path failures are independent.

Salsa's circuit construction process is shown in Figure 5. Circuits are built in ℓ stages, using redundancy parameter r. In the first stage, the initiator I asks r nodes (including itself) in its local DHT neighborhood to perform recursive lookups for r keys $k_1, \ldots, k_r$, and receives public keys and IP addresses for r relays $A_1, \ldots, A_r$. I establishes a single-hop circuit with each of these nodes, and the next stage begins. At each subsequent stage, I asks each of these r circuit endpoints to recursively lookup r additional keys $k'_1, \ldots, k'_r$, to learn the IP addresses and keys of r additional relays $B_1, \ldots, B_r$. I extends circuits from each A_i to each B_j, $1 \leq i, j \leq r$, and then randomly chooses one of the successfully extended circuits to each B_j, discarding the rest. This process continues until the final stage, in which each of the r circuit endpoints is asked to recursively lookup a single key k'', obtaining the IP address and public key of a single relay T; I extends each circuit from B_i to T and randomly picks one of the successfully extended circuits for its anonymous session.

The critical remaining issue is how the initiator resolves conflicting redundant lookup results. Recall that in the generic case of DHTs storing key/value pairs it is assumed that there exists some "content authentication" scheme that can identify the correct result of a lookup, which means it is sufficient to ensure that one redundant lookup is not captured. In the case of Salsa, there is no such scheme. Salsa nodes resolve conflicting lookup results as follows: the initiator uses the public cryptographic hash function to compute the ID for each IP address returned. The ID that is closest to the target key is selected, and the other results are discarded. Finally, this ID is subjected to the density test and if it passes the lookup succeeds.

4.2 Attack on Salsa

Our attack is a variant of the generic selective DoS attack of Section 3.1. It relies on three shortcomings of Salsa's lookup defense mechanisms. First, the bounds check is ineffective in case the adversary controls the closest live node to a key. Second, Salsa's "closest ID" conflict resolution means that an adversary can cause a redundant lookup to fail by being capturing *one* of the redundant queries and returning the IP address of the non-participating node with the closest hash to a given key (If fraction c of all IP addresses participate in Salsa, the probability that one of the non-participating IP addresses is closest to a given key is greater than $1 - e^{-1/c}$.) Finally, Salsa has no mechanism to verify the binding between a public key and IP address; if a first-stage node is malicious it can masquerade as any nonparticipating node simply by returning a public key of its choosing for that node.

Our attack uses a dictionary of IP address and hash pairs to defeat Salsa's ID is hash of IP check. The attacker generates the dictionary by hashing all of the IP addresses. The storage required for a dictionary of 2^{32} hashes is only 16 GB. (It is only necessary to remember which 4-byte IP address produces the closest hash to any 32-bit ID prefix)

The attack proceeds as follows. The adversary attacks all DHT lookup requests. Upon capturing a query for key k, malicious node M first determines whether the correct response to the query is another malicious node M' and if so, M immediately returns M' as the recursive lookup result. If not, M identifies the IP address N with the hash closest to k and returns N as the recursive lookup result. Since with overwhelming probability N is not a participant, the initiator will fail to extend a circuit to N and choose a different key instead. This heavily biases the choice of first-stage relays towards malicious nodes.

When malicious node M receives a circuit extension message it always responds, and then determines whether it is a first-stage relay or not (If r separate nodes extend circuits to M, it is not a first-stage relay.) If M is not a first-stage relay, it returns bogus results for all subsequent lookups, so that the circuit cannot be completed. If M is a first-stage relay, it uses the dictionary to return IP addresses for each of the keys $k'_1, \ldots, k'_r$, with public keys chosen so that M knows the corresponding private keys. Since I cannot contact any of the returned IP addresses directly, it cannot verify these public keys. Since the IP addresses $B_1, \ldots, B_r$ returned by M resolve to the closest IDs to $k'_1, \ldots, k'_r$ and are nonexistent, I will ask each of the first-stage nodes to extend circuits to $B_1, \ldots, B_r$ and only malicious nodes will succeed (by simply decrypting the key-exchange messages with no forwarding). In the following stage, all of the (fictional) relays are adversarially controlled and the dictionary attack (with ad hoc public keys) can be repeated, ensuring all of the next-stage relays are compromised. Thus the entire circuit will be compromised as a result of this behavior.

We note that Salsa's behavior in case of conflicting or failed circuit extensions is not specified; an alternative interpretation to our own is that any failure to extend a circuit from some A_i to some B_j causes the entire process to be aborted. In this case, however, it is trivial for a first-stage relay to abort the process unless at least one malicious relay is selected for the second stage, so that the "generic" attack analysis applies directly, replacing the probability of query capture by the probability that at least one out of r queries is captured and replacing the probability of relay compromise by the probability that at least one out of r relays in a single stage is compromised. This applies even if Salsa is augmented with a PKI binding public keys to IP addresses or IDs, since the majority of Salsa nodes at any time are likely to be offline and thus the adversary can cause queries to nonmalicious nodes to be dropped by returning the certificate of a closer, offline node.

4.3 Salsa Simulation

To demonstrate the effectiveness of our attack, we simulated Salsa networks of varying size using a 64-bit hash space, neighborhoods of size 128, fraction $f = 0.1$ of malicious nodes and circuits of length $\ell = 3$. For each network size, and redundancy level $r \in \{3, 5, 7\}$, we simulated the creation of 100000 circuits and recorded the fraction of completed circuits compromised. The results appear in Figure 6. As expected, the fraction of compromised circuits grows as the network size increases (since the expected number of DHT hops, and thus the number of opportunities to capture a DHT query, increases). Interestingly, higher levels of redundancy actually reduce the security of Salsa against our attack; this is because the attacker only needs to capture a single redundant query to drop it; more queries give the attacker additional opportunities. This remains true in the presence of a PKI or with the "abort on conflict" strategy discussed above.

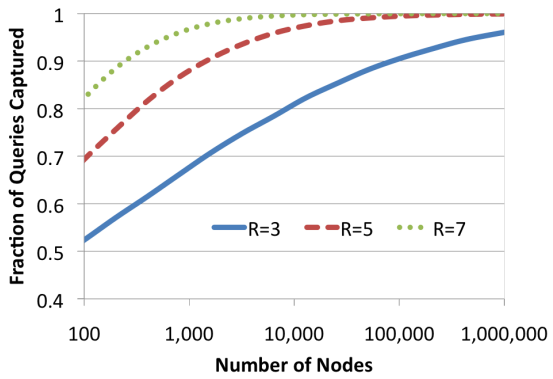

Figure 6: Results of simulated attack on Salsa with malicious fraction $f = 0.1$, where conflicting redundant lookups are resolved to the closest node.

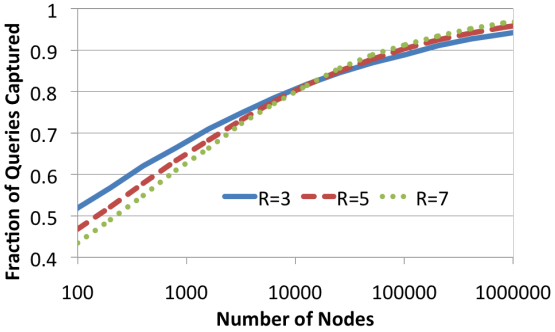

Figure 7: Results of simulated attack on Salsa with malicious fraction $f = 0.1$, where conflicting redundant lookups are resolved by a majority vote

An alternative to Salsa's "closest node" strategy for resolving conflicting redundant lookups is to take the relay (that passes the bounds check) returned by the majority of lookups. In this case, the best strategy for the adversary is to "stack the vote" towards a malicious node that can pass the bounds check for a key if one exists, and "drop" the query otherwise by returning an offline node that can pass the bounds check. Since asymptotically the majority of redundant lookups will be captured this increases the probability of having a malicious first-stage relay by a factor of the bounds check multiplier. Figure 7 shows the result of simulations using this conflict resolution method. We note that the modified attack is still quite successful, and that there is a range of network sizes for which increased redundancy slightly reduces the impact of the attack. This is due to the fact that for very small network sizes the probability of query capture is close to 50%, so the probability of capturing more than half of the r queries decreases as r increases. Once the probability of capture exceeds 50% by a significant margin, however, increasing r reduces the chance that a majority of lookups are not captured.

5. CASHMERE

5.1 Overview of Cashmere

Cashmere is a recursive DHT-based anonymous communications overlay built on top of the Pastry DHT [34]. Cashmere uses *virtual relays* composed of sets of nodes (or *relay groups*) for resilience. A Cashmere node with a k-bit nodeID has an m-bit groupID for every $1 \leq m \leq k$. A node belongs to relay group if the groupID is a prefix of the nodeID. Every relay group must have a public/private key pair, and every member of the relay group has the public/private key pair for that group. Cashmere assumes all of the keys are both generated and distributed by a trusted offline CA. When a user bootstraps into Cashmere, the user obtains a signed k-bit nodeID and the set of k keys associated with prefixes of that nodeID. The user must also obtain a public key for all other prefixes.

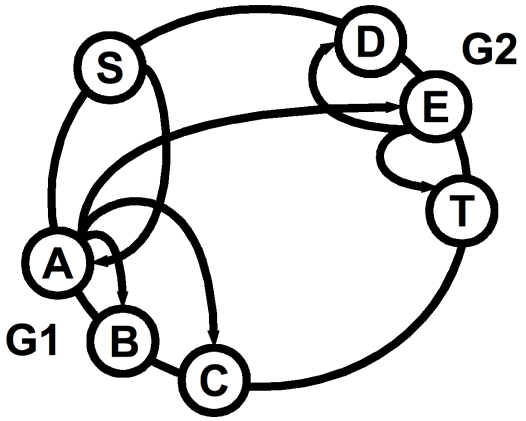

Figure 8: Cashmere Forwarding

The overlay passes messages through relay groups. The overlay uses the groupID as a key to route the message using prefix matching. The first node found with a prefix matching the groupID is called the relay group root. The relay group root is responsible for processing the message on behalf of the relay group; in particular the root broadcasts the message to the other members of the group and performs any forwarding operations required. Thus routing to a groupID functions similarly to any cast to the relay group members.

Cashmere's forwarding process is shown in Figure 8. Circuits consist of ℓ stages. Suppose peer S wants to forward a message to peer T. S chooses a path consisting of ℓ relay groups $G_1, \ldots, G_\ell$, such that T is in relay group G_d for a randomly chosen $1 \leq d \leq \ell$. S then encrypts the forwarding path in layers using each relay group's public key.

Cashmere decouples a message's payload from the forwarding path and encrypts each separately. This allows a source to reuse a forwarding path reducing the computational overhead of multiple encryption. By caching the forwarding path the source is able to send multiple messages to the destination at the cost of a single public key encryption per key.

The i^{th} relay group root receives the message $[Path_i, Payload_i]$ from the previous relay group. The i-th relay group root uses the group's secret key to decrypt the outer layer of $Path_i$, revealing $Path_{i+1}$, the identity of the next relay group G_{i+1}, and a symmetric key to decrypt $Payload_i$. The d_th relay group root may not be the destination because any member of the relay group can receive the message. Therefore, each relay group root must broadcast $Payload_i$ to all members of the relay group.

Cashmere uses end-to-end acknowledgments to detect failures and malicious nodes: if the source receives no acknowledgments, it can use timeouts to guide retransmission. Return messages are sent by including an encrypted return path, along with the symmetric keys necessary to encrypt the responder's payload, in the initiator's first payload.

5.2 Attack on Cashmere

The attack on Cashmere is similar to the generic attack on overlay circuit extension described in section 3.2 with some small modifications. As in that section, the adversary attempts to discover the entire path by either being on the forwarding path to a relay group

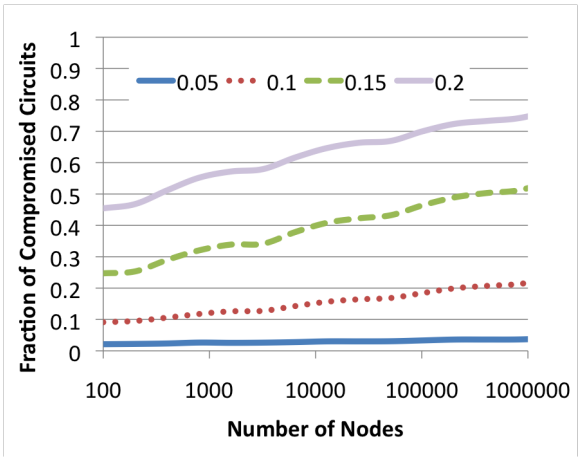

Figure 9: Results of simulated attack on Cashmere with malicious fraction $f = 0.05, 0.1, 0.15, 0.2$ with varying number of nodes, using path length $\ell = 3$ and group size $\rho = 5$.

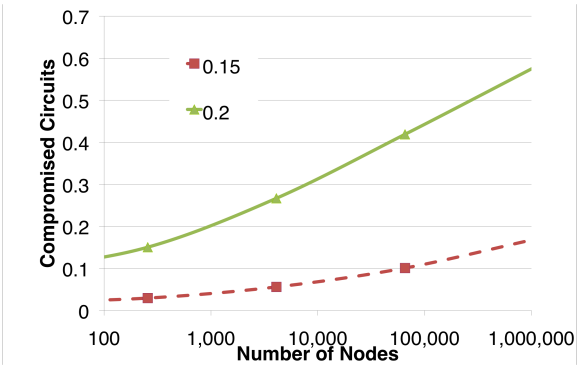

Figure 10: Estimated fraction of complete compromises for $\ell = 3$, $\rho = 5$ and $f \in \{0.15, 0.2\}$

or being a member of the previous group. Compared to the generic attack, the use of relay groups increases the probability of learning the next group by being a member of the previous group. If the adversary sees message m_1 delivered to group G_1 and then sees m_2 delivered to G_2, it can test whether G_1 and G_2 are neighbors on the same path by constructing its own path message including G_1 and G_2 and forwarding this message from the point it observed m_1. If it sees its message at the same point it observed m_2, the amount of path diversity in the Pastry overlay gives us high confidence that m_2 came from G_1.

If the adversary observes the construction of a chain of ℓ relay groups then malicious nodes are instructed to deliver subsequent messages using the same path only if the initiator can be determined. Due to Pastry's "wide" routing table, with probability $\frac{15}{16}$ a query's first hop is to a node with a single-digit prefix match. Thus when a malicious node receives a recursive query with a one-digit prefix match it knows that the forwarding node is the query initiator; if the target of the query is the first relay group in an observed chain of ℓ then the query initiator is also the circuit initiator. In all other cases (either the circuit initiator cannot be determined or some of the relay groups belonging to a circuit are not observed) subsequent messages using the same path are dropped.

Thus, a circuit can only successfully deliver subsequent (non-reply block) payloads if the adversary learns the initiator and all of the relay groups or if none of the DHT paths between the initiator and the recipient are captured. In the first case, the circuit can be considered to be $\ell\rho$-compromised, where ρ is the size of a relay group; the recommended parameters of $\ell = 4$ and $\rho = 5$ mean that when a circuit is compromised the initiator can be linked to a list of 20 possible recipients. If we let p_c be the probability of the first case and p_{nc} the probability of the latter, then this attack will $\ell\rho$-compromise fraction $\frac{p_c}{p_c+p_{nc}}$ of the circuits.

Figure 9 shows how the success rate of this attack varies with n and f, when $\ell = 3$ and $\rho = 5$. For each value of n and f we simulated the creation of 100,000 successful circuits and recorded the fraction of successful circuits compromised. As expected, larger fractions of malicious nodes and larger network sizes result in a higher fraction of compromised circuits. Note that for $f = 0.1$ the fraction of $\ell\rho$-compromised circuits exceeds f when the $N > 400$.

We note that it is possible to use application-level semantics to improve these parameters. In particular, the authors [44] state that application-level acknowledgments (sent using the encrypted reply path) are used to detect failures; if no acknowledgment is received, a new forwarding path must be established. Thus a circuit is only useful if both the forwarding and reply path can successfully transmit payloads. Suppose that the adversary observes a path from node S in group G' to relay groups $G_1, \ldots, G_\ell$. If no corresponding path is observed from a node $T \in \bigcup_i G_i$ to G', then the adversary concludes that either the forwarding or reply path has not been compromised and drops subsequent messages along the path, causing the entire path to be abandoned. On the other hand, if such a path is observed, then with high confidence T is the responder to S's connection. In this case, the attack completely compromises fraction $\frac{p_c^2}{p_c^2+p_{nc}^2}$ of the circuits. Figure 10 shows how the fraction of completely compromised circuits varies with f and N.

6. RELATED WORK

The most directly related work to our own is that of Mittal and Borisov [25]. They considered two DHT-based anonymity schemes that use redundant routing for "secure" lookup – AP3 [24] and Salsa [29] – and showed that the redundancy in routing introduces "information leaks" – extra opportunities to learn the initiator of a lookup – that significantly weaken the anonymity provided by each scheme. Furthermore, they show that for a network of size 1000, when the redundancy factor is small there is an increased risk of circuit compromise by active attacks that return adversarial nodes whenever they pass the bounds check. Mittal and Borisov conclude that increasing lookup redundancy offers a tradeoff between security against "passive" attacks based primarily on information leaks[3] and "active" attacks against the queries. In contrast, we show that because of the difference in security goals between key-value lookup and relay selection, *there is no tradeoff*: redundancy does not significantly improve the security of DHT-based relay selection against active (selective-DoS) attacks. As a result our attacks are significantly more effective.

The basic "building blocks" of our attack have appeared in the literature before. In the context of anonymity, Borisov *et al.* [4] were the first to point out the effectiveness of selective denial of service against circuit-based anonymity schemes, including an attack against Salsa that did not rely on the DHT communication layer. Basic attacks on DHT routing, including the possibility of query dropping and misrouting, appear in the seminal works on P2P security by Sit and Morris [36], Wallach [40], and Castro *et al.* [5]. "Route capture" attacks using public-key modification are mentioned in [4] but the idea of route capture seems to be a "folklore" idea known to the anonymity community since at least 2002 [33, 8]. Reiter and Rubin [32] and Wright *et al.* [42] describe

[3]Mittal and Borisov's attack against Salsa involves "key substitution" after the first layer, so it is not purely passive

the predecessor attack, in which an initiator that repeatedly builds circuits to the same responder can be identified.

Several recent works have explored the security importance of details of other system components or layers that are typically hidden by interface abstractions. Feamster and Dingledine [16] consider the effects of AS-level topology, and Murdoch and Zieliński [28], consider traffic analysis by internet exchanges. Other features that have been exploited include clock skew [26] and internet path properties such as latency [19] and bandwidth [6]. Mittal and Borisov show that the use of redundancy to secure DHT lookups is an important factor; we show that the security of the individual lookups is even more important.

Finally, it is well-known that global, passive adversaries can defeat most low-latency anonymity designs [1, 31, 38]. Even high-latency "mixing" schemes are vulnerable to global passive intersection attacks that infer general patterns of recipients [9, 31, 3, 21, 10, 13, 39] and difficult to secure against targeted global active attacks [18, 35, 11]. More recent works by Wright *et al.* [41], Murdoch and Danezis [27], and Danezis, Clayton and Syverson [12, 14] as well as the previously mentioned work of Mittal and Borisov [25] have explored ways in which architectural or implementation defects can allow a local adversary to "simulate" a global passive adversary. Our work can be seen as showing that using DHT routing to select relays can promote a local adversary to a global *active* adversary.

7. CONCLUSION

The anonymity literature, including all of the schemes investigated here, is replete with claims that a peer-to-peer architecture is necessary in order to construct a scheme that will work at Internet scale. Distributed Hash Tables offer a scalable architecture for organizing and finding peers, and thus appear to be an obvious choice of peer-to-peer architecture. However, as we have shown there is not a clear bijection between the security and robustness requirements of a DHT's put-get interface and an anonymity scheme's relay selection mechanism. This leads to severe vulnerabilities in the existing schemes based on DHTs, limiting the deployability of such schemes. The critical question for future work in this line of research is whether a "DHT-like" algorithm can be designed to meet the specific requirements – in terms of privacy, availability, and correctness – of an anonymity scheme.

Acknowledgments

The authors thank Nikita Borisov, Prateek Mittal, Mike Reiter, Gene Tsudik, Eugene Vasserman, and Matt Wright for helpful discussions and comments about this work. This work was partially supported by the NSF under grants CNS-0546162 and CNS-0716025.

8. REFERENCES

[1] BACK, A., MÖLLER, U., AND STIGLIC, A. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of Information Hiding Workshop (IH 2001)* (April 2001), I. S. Moskowitz, Ed., Springer-Verlag, LNCS 2137, pp. 245–257.

[2] BENNETT, K., AND GROTHOFF, C. GAP – practical anonymous networking. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)* (March 2003), R. Dingledine, Ed., Springer-Verlag, LNCS 2760, pp. 141–160.

[3] BERTHOLD, O., AND LANGOS, H. Dummy traffic against long term intersection attacks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)* (April 2002), R. Dingledine and P. Syverson, Eds., Springer-Verlag, LNCS 2482.

[4] BORISOV, N., DANEZIS, G., MITTAL, P., AND TABRIZ, P. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In *Proceedings of CCS 2007* (October 2007).

[5] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation* (New York, NY, USA, 2002), ACM, pp. 299–314.

[6] CHAKRAVARTY, S., STAVROU, A., AND KEROMYTIS, A. D. Identifying proxy nodes in a tor anonymization circuit. In *SITIS '08: Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 633–639.

[7] CIACCIO, G. Improving sender anonymity in a structured overlay with imprecise routing. In *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)* (Cambridge, UK, June 2006), G. Danezis and P. Golle, Eds., Springer, pp. 190–207.

[8] CLAYTON, R. A Few Design Critiques. at *3rd Workshop on Privacy Enhancing Technologies*, 2003.

[9] COTTRELL, L. Mixmaster and remailer attacks. online essay: `http://web.archive.org/web/*/http://obscura.com/~loki/remailer-essay.html`, 1994.

[10] DANEZIS, G. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)* (Athens, May 2003), Gritzalis, Vimercati, Samarati, and Katsikas, Eds., IFIP TC11, Kluwer, pp. 421–426.

[11] DANEZIS, G. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)* (May 2004), vol. 3424 of *LNCS*, pp. 35–50.

[12] DANEZIS, G., AND CLAYTON, R. Route fingerprinting in anonymous communications. In *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 69–72.

[13] DANEZIS, G., AND SERJANTOV, A. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH 2004)* (Toronto, May 2004), LNCS.

[14] DANEZIS, G., AND SYVERSON, P. Bridging and fingerprinting: Epistemic attacks on route selection. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)* (Leuven, Belgium, July 2008), N. Borisov and I. Goldberg, Eds., Springer, pp. 133–150.

[15] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004).

[16] FEAMSTER, N., AND DINGLEDINE, R. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)* (Washington, DC, USA, October 2004).

[17] FREEDMAN, M. J., AND MORRIS, R. Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications*

security (New York, NY, USA, 2002), ACM Press, pp. 193–206.

[18] GÜLCÜ, C., AND TSUDIK, G. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96* (February 1996), IEEE, pp. 2–16.

[19] HOPPER, N., VASSERMAN, E. Y., AND CHAN-TIN, E. How much anonymity does network latency leak? In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security* (New York, NY, USA, 2007), ACM, pp. 82–91.

[20] KATTI, S., COHEN, J., AND KATABI, D. Information slicing: Anonymity using unreliable overlays. In *Proceedings of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI)* (April 2007).

[21] KESDOGAN, D., AGRAWAL, D., AND PENZ, S. Limits of anonymity in open environments. In *Proceedings of Information Hiding Workshop (IH 2002)* (October 2002), F. Petitcolas, Ed., Springer-Verlag, LNCS 2578.

[22] LANDSIEDEL, O., PIMENIDIS, L., WEHRLE, K., NIEDERMAYER, H., AND CARLE, G. Core: A Peer-To-Peer Based Connectionless Onion Router. In *Proceedings of IEEE GLOBECOM, Globalcommunications Conference* (Washington DC, USA, November 2007).

[23] LU, T., FANG, B., SUN, Y., AND CHENG, X. Wongoo: A peer-to-peer protocol for anonymous communication. In *PDPTA* (2004), H. R. Arabnia, Ed., CSREA Press, pp. 1102–1106.

[24] MISLOVE, A., OBEROI, G., POST, A., REIS, C., DRUSCHEL, P., AND WALLACH, D. S. Ap3: cooperative, decentralized anonymous communication. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop* (New York, NY, USA, 2004), ACM, p. 30.

[25] MITTAL, P., AND BORISOV, N. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)* (Alexandria, Virginia, USA, October 2008), P. Syverson, S. Jha, and X. Zhang, Eds., ACM Press, pp. 267–278.

[26] MURDOCH, S. J. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of CCS 2006* (October 2006).

[27] MURDOCH, S. J., AND DANEZIS, G. Low-cost traffic analysis of tor. *IEEE SP 00* (2005), 183–195.

[28] MURDOCH, S. J., AND ZIELIŃSKI, P. Sampled traffic analysis by internet-exchange-level adversaries. In *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)* (Ottawa, Canada, June 2007), N. Borosov and P. Golle, Eds., Springer.

[29] NAMBIAR, A., AND WRIGHT, M. Salsa: A structured approach to large-scale anonymity. In *Proceedings of CCS 2006* (October 2006).

[30] PUTTASWAMY, K., SALA, A., WILSON, C., AND ZHAO, B. Protecting anonymity in dynamic peer-to-peer networks. In *IEEE International Conference on Network Protocols (ICNP)* (Oct. 2008), pp. 104–113.

[31] RAYMOND, J.-F. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (July 2000), H. Federrath, Ed., Springer-Verlag, LNCS 2009, pp. 10–29.

[32] REITER, M., AND RUBIN, A. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security 1*, 1 (June 1998).

[33] RENNHARD, M., AND PLATTNER, B. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society* (New York, NY, USA, 2002), ACM Press, pp. 91–102.

[34] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware* (2001).

[35] SERJANTOV, A., DINGLEDINE, R., AND SYVERSON, P. From a trickle to a flood: Active attacks on several mix types. In *Proceedings of Information Hiding Workshop (IH 2002)* (October 2002), F. Petitcolas, Ed., Springer-Verlag, LNCS 2578.

[36] SIT, E., AND MORRIS, R. Security considerations for peer-to-peer distributed hash tables. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, 2002), Springer-Verlag, pp. 261–269.

[37] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., AND BALAKRISHNAN, H. Chord: A peer-to-peer lookup service for internet applications. In *SIGCOMM* (2001).

[38] SYVERSON, P. F., GOLDSCHLAG, D. M., AND REED, M. G. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy* (Oakland, California, 4–7 1997), pp. 44–54.

[39] TRONCOSO, C., GIERLICHS, B., PRENEEL, B., AND VERBAUWHEDE, I. Perfect matching statistical disclosure attacks. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)* (Leuven, Belgium, July 2008), N. Borisov and I. Goldberg, Eds., Springer, pp. 2–23.

[40] WALLACH, D. S. A survey of peer-to-peer security issues. In *ISSS* (2002), M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, Eds., vol. 2609 of *Lecture Notes in Computer Science*, Springer, pp. 42–57.

[41] WRIGHT, M., ADLER, M., LEVINE, B. N., AND SHIELDS, C. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (May 2003), pp. 28–43.

[42] WRIGHT, M., ADLER, M., LEVINE, B. N., AND SHIELDS, C. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC) 4*, 7 (November 2004), 489–522.

[43] ZHU, Y., AND HU, Y. Tap: A novel tunneling approach for anonymity in structured p2p systems. In *ICPP* (2004), IEEE Computer Society, pp. 21–28.

[44] ZHUANG, L., ZHOU, F., ZHAO, B. Y., AND ROWSTRON, A. Cashmere: resilient anonymous routing. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation* (Berkeley, CA, USA, 2005), USENIX Association, pp. 301–314.