In Search of an Anonymous and Secure Lookup

Attacks on Structured Peer-to-Peer Anonymous Communication Systems

Qiyan Wang
UIUC
qwang26@illinois.edu

Prateek Mittal
UIUC
mittal2@illinois.edu

Nikita Borisov
UIUC
nikita@illinois.edu

ABSTRACT

The ability to locate random relays is a key challenge for peer-to-peer (P2P) anonymous communication systems. Earlier attempts like Salsa and AP3 used distributed hash table lookups to locate relays, but the lack of anonymity in their lookup mechanisms enables an adversary to infer the path structure and compromise user anonymity. NISAN and Torsk are state-of-the-art systems for P2P anonymous communication. Their designs include mechanisms that are specifically tailored to mitigate information leak attacks. NISAN proposes to add anonymity into the lookup mechanism itself, while Torsk proposes the use of secret buddy nodes to anonymize the lookup initiator.

In this paper, we attack the key mechanisms that hide the relationship between a lookup initiator and its selected relays in NISAN and Torsk. We present passive attacks on the NISAN lookup and show that it is not as anonymous as previously thought. We analyze three circuit construction mechanisms for anonymous communication using the NISAN lookup, and show that the information leaks in the NISAN lookup lead to a significant reduction in user anonymity. We also propose active attacks on Torsk that defeat its secret buddy mechanism and consequently compromise user anonymity. Our results are backed up by probabilistic modeling and extensive simulations. Our study motivates the search for a DHT lookup mechanism that is both secure and anonymous.

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, attacks, information leaks, peer-to-peer

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.

1. INTRODUCTION

Anonymous communication hides the identities of communicating partners from third parties, or the user identity from a remote party. As a key privacy-enhancing technology, anonymous communication is gaining widespread popularity in an era of pervasive surveillance. The Tor network [8] is a deployed anonymous communication system that currently serves hundreds of thousands of users a day [13]. Tor is widely used to preserve privacy for journalists, whistleblowers, law enforcement, and even embassies [10].

A key problem in Tor's architecture is that it requires all users to maintain a global view of the system, which could be costly as the size of the Tor network increases. McLachan et al. [16] show that in the near future, the Tor network could be spending an order of magnitude more bandwidth in maintaining a global view of the system, than for relaying anonymous traffic. In order to address this problem, a peer-to-peer (P2P) architecture will likely be necessary. Indeed, several proposals for P2P anonymous communication have been put forward [17, 24, 9, 29, 16, 25, 19].

P2P networks present new challenges to anonymity, including the ability to locate random relays for anonymous communication. Earlier attempts at P2P anonymous communication like Salsa [24] and AP3 [17] used distributed hash table (DHT) lookups to locate random relays. Mittal and Borisov [18] showed that the lookup mechanisms of these systems leak information about the relationship between a lookup initiator and the communication destination, and thus are not anonymous. Moreover, such information leaks in the lookup could then be used to attack the circuit construction mechanisms and compromise user anonymity.

Recently, NISAN [25] and Torsk [16] P2P systems have been proposed to alleviate the scalability problems in the Tor network. Both NISAN and Torsk also use DHT lookups to locate relays for anonymous communication, but in contrast to Salsa and AP3, they include mechanisms that are specifically designed to mitigate information leak attacks [18]. In this paper, we show that these mechanisms are vulnerable to several passive and active attacks.

NISAN proposes the use of a custom secure and anonymous lookup mechanism to locate relays. The NISAN lookup relies on redundancy and bound checking [25] to defend against active attacks, while trying to keep the lookup destination secret from attackers. In this paper, we present passive attacks on the NISAN lookup and show that it is not as anonymous as previously thought. We analyze three circuit construction mechanisms for anonymous communication using the NISAN lookup, and show that information

leaks in the NISAN lookup lead to a significant reduction in user anonymity for all the three circuit construction strategies. For a network size of 10 000 nodes, with 20% compromised nodes, our analysis and simulation results show that our passive attacks can reduce entropy of the circuit initiator by 2.2 bits, which is close to the ideal passive attacks whose entropy reduction is 2.6 bits.

Unlike NISAN, Torsk [16] does not attempt to add anonymity into the lookup itself. Instead, nodes in Torsk perform random walks to obtain secret *buddy* nodes before the lookups are performed. The lookup querier can then use the secret buddy node as a proxy to perform the lookup on its behalf, thus hiding the relationship between itself and the lookup target. However, we shall show that the information leaks in the lookup itself allow an attacker to launch active attacks to defeat the secret buddy mechanism, and consequently compromise user anonymity. Our analysis and simulation results show that with 20% malicious nodes, the attacker can compromise over 80% of constructed circuits. We also study potential improvements to Torsk, but find that they do not fully defend against our attacks.

Our analysis of NISAN and Torsk shows that their key mechanisms to anonymously look up a node are vulnerable to either passive attacks or active attacks, motivating the search for a secure and anonymous DHT lookup. We note that the vulnerabilities in the current P2P anonymous communication designs are a result of some properties in their lookup mechanisms not being fully analyzed in the security evaluation of the system, leading us to conclude that system designers should explicitly model the mechanism for anonymous lookup and analyze its security.

The remainder of the paper is organized as follows. We describe background material including the threat model in Section 2. We show that both NISAN and Torsk lookups are not anonymous in Section 3. In Section 4, we propose circuit construction strategies using the NISAN lookup and analyze their anonymity. Section 5 describes our attacks on Torsk and considers potential improvements. Finally, we discuss related work in Section 6 and conclude in Section 7.

2. BACKGROUND

In this section, we present a brief overview of anonymous communication. We discuss why P2P systems have strong potential for anonymous communication, and describe the state-of-the-art systems that are based on DHT lookups. We also describe our adversarial threat model.

2.1 Low-Latency Anonymous Communication Systems

Anonymous systems are typically divided into high-latency and low-latency systems. High-latency anonymous systems, such as Mixminion [6] and Mixmaster [20], are designed to resist a powerful global attacker, but the communication latency for such systems could be up to several hours, which make them unsuitable for interactive communications, such as web browsing and instant messaging. The focus of this paper is on low-latency anonymous communication systems.

Tor [8] is a popular low-latency anonymous communication system, which serves hundreds of thousands of users every day [13]. Each Tor client obtains a list of servers from a central directory authority, and selects random relays from the list to construct a circuit for onion routing [31]. Tor requires each client to maintain a global view of all the servers. However, as the number of servers increases, maintaining a global view of the system become costly, since churn will cause frequent updates and a large bandwidth overhead. In fact, McLachan et al. [16] show that in the near future, the Tor network could be spending an order of magnitude more bandwidth in maintaining a global view of the system, than for relaying anonymous traffic. A P2P architecture will likely be necessary to address this problem.

Several designs for P2P anonymous communication have been proposed [9, 29, 17, 24, 19, 16, 25]. We can broadly classify these designs based on their mechanisms to locate relays. Designs like Tarzan [9], Morphmix [29] and ShadowWalker [19] perform random walks on restricted topologies to find relays, while Salsa [24], AP3 [17], NISAN [25] and Torsk [16] use DHT lookups to select random relays. In this work, we focus on DHT-lookup-based P2P anonymous communication systems.

Distributed hash tables, also known as structured peer-to-peer topologies, provide an attractive foundation for P2P anonymous communication. Structured topologies assign neighbor relationships using a pseudorandom but deterministic mathematical formula based on the IP addresses or public keys of nodes. This allows the relationships to be verified externally, presenting fewer opportunities for attacks. We now briefly describe the P2P anonymous communication systems that use DHT lookups.

The design of Salsa [24] is similar to Tor, in that a circuit is built by selecting three random nodes (or relays) in the network and constructing a circuit through them. Salsa uses a specifically designed secure lookup over a custom DHT to locate relays. The secure lookup uses redundant checks to mitigate potential attacks. These checks are able to limit the bias an adversary can introduce in the lookup, but make Salsa susceptible to information leak attacks: attackers can detect a large fraction of lookups and thus infer the path structure [18]. Salsa is also vulnerable to selective denial-of-service (DoS) attack, where nodes try to deny service for circuits that they cannot compromise [3].

AP3 [17] also relies on secure lookups to locate relays, but the design of AP3 is more similar to Crowds [28] than to Tor, with paths being formed by performing a stochastic expected-length random walk. The stochastic nature of AP3 makes it difficult for a rogue node to decide whether its preceding hop is the initiator or simply a relay on the path. However, for low-latency communication, timing attacks may make this decision simpler. Similar to Salsa, the secure lookup used in AP3 reveals a lot of information about the lookup initiator, and makes it vulnerable to passive information leak attacks [18].

NISAN and Torsk are the state-of-the-art P2P anonymous communication systems. Their designs include mechanisms that are specifically tailored to mitigate information leak attacks. NISAN proposes to incorporate anonymity into the lookup itself, and Torsk uses secret buddy nodes. However, we shall show that these mechanisms are vulnerable to either passive attacks or active attacks, resulting in a significant reduction in user anonymity.

2.2 Threat Model

Low-latency anonymous communication systems are not designed to resist a global passive adversary. We consider a partial adversary who controls a fraction f of all the nodes in the network. This set of malicious nodes collude and can

launch both passive and active attacks. We consider the set of colluding nodes is static and the adversary cannot compromise nodes at will.

Even in networks with a large number of nodes, f can be a significant fraction of the network size. Powerful adversaries, such as governments or large organizations, can potentially deploy enough nodes to gain a significant fraction of the network. Similarly, botnets, whose average size has grown in excess of 20 000 nodes [26], present a real threat to anonymous systems. In our analysis, we will consider the maximum value of f to be 0.2, since P2P systems are not designed to be secure at higher values of f.

3. INFORMATION LEAKS IN DHT LOOKUPS

In this section, we study two recently proposed DHT lookup mechanisms, Torsk [16] and NISAN [25], and analyze information leaks in both lookups. Since the Torsk lookup is "louder" and simpler to analyze, we start with discussing the Torsk lookup.

3.1 Torsk Lookup

The authors of Torsk [16] proposed a custom secure iterative lookup scheme (we refer to as the Torsk lookup) to construct Torsk. The design of the Torsk lookup is based on Kademlia DHT [15] and Myrmic [33]. Its main goal is to resist active attacks, rather than preserving information leaks.

In the Torsk lookup, a querier Q who wants to look up a target x first selects t fingers from its finger table (FT) that are closest to x (typically, $t = 3$), and uses them as starting points for t independent lookup branches. Q maintains a best list of closest fingers to x for each lookup branch. In each iteration, t fingers that have not been contacted are selected from each best list and are queried with x in parallel. Any requested node returns k fingers closest to x. The wide parallel lookup process terminates when any best list is unchanged at the end of one iteration. To resist active attacks, each node U keeps a certificate ($nCert_U$) issued by a trusted central authority. $nCert_U$ includes all fingers of U and its expiration time etc. Therefore, Q can verify whether the finally found node V is responsible for x by verifying $nCert_V$ and querying V's neighbors about the freshness of $nCert_V$. V is accepted by Q only when V passes all these checkings.

Since x is revealed to each queried node, an attacker can observe the lookup and associate V with Q, as long as one malicious node is queried in the lookup. We simulate the Torsk lookup using a simulator written in C++ with about 1000 lines. We measure the probability of associating the target with the querier in different sized networks ($n = 1000$, 2000, 5000). We choose typical values for parameters used in the simulation according to the Torsk paper [16] (ID space $= 2^{20}$, #buckets $= 16$, bucket size $= 20$, and $nList$ size $= 6$). Each data point is averaged over 100 random topologies with 10 000 independent runs. We can see from Figure 1 that when $f = 0.2$, the attacker can nearly observe all lookups and associate all the targets with the queriers. Also, as the size of the network grows, the attacker has a better chance to observe the lookup with the same fraction of compromised nodes. This is because as the length of the lookup path increases, a malicious node would more likely be involved in the lookup.

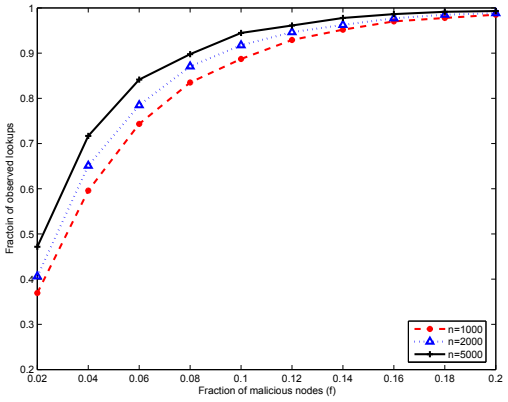

Figure 1: Simulation results: probability that the attacker can associate the querier with the target in the Torsk lookup.

3.2 NISAN Lookup

The NISAN lookup [25] is based on the Chord DHT [30], and is specifically designed to preserve anonymity. In particular, the querier Q maintains a top list (we refer to as *TopList*) of m best fingers that are closest to the target ID x during the lookup (typically m is set as the size of the FT). At each iteration, Q asks the nodes in the TopList that have not been queried for their whole FTs, rather than revealing the target x, in hopes of hiding x from passive attackers. The TopList is updated with the returned FTs if any node closer to x is discovered. The lookup process stops when the TopList is unchanged at the end of one iteration.[1]

To limit malicious nodes from replying with manipulated FTs, NISAN applies bound checking on each obtained FT: Q calculates the mean of the distance between the actual fingers in the FT and the optimal fingers (i.e., $ID_{owner}+2^i$). FTs with mean distance larger than a threshold is discarded by Q. To make it even harder for the attacker to learn the ultimately selected node, the authors suggest considering the whole TopList and picking a uniformly random finger out of the TopList as the final result, so that even if the attacker gains some knowledge about x, there is still some uncertainty about the finally picked node.

3.2.1 Passive Attacks on the NISAN Lookup

We show that the NISAN lookup is not as "anonymous" as expected: the attacker can learn both a lower bound and an upper bound of x. Since the ultimately selected node (denoted by T) is within $m-1$ hops preceding x, the attacker can estimate the range of T based on the knowledge of x.

Range estimation. The attacker's strategy is based on the fact that Q will not query a node succeeding x (except in the first iteration) since the Chord ring is directed. Initially, the lower bound and the upper bound of x are set as the direct successor and direct predecessor of Q, respectively, and the

[1]We note that the NISAN lookup is different from the Chord lookup. The Chord lookup finds the successor of the target ID. Whereas, in the NISAN lookup, all the fingers in the final TopList (including the finally chosen node) precede the target ID. This has been confirmed by the NISAN authors in a private communication.

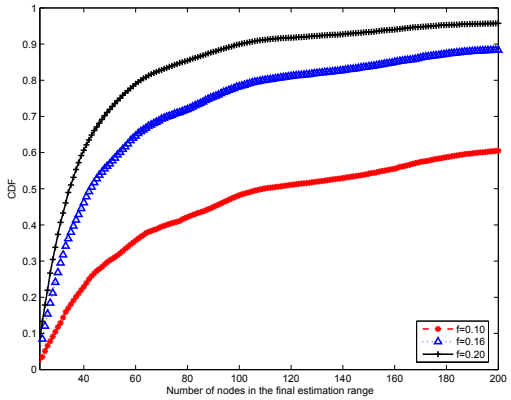

Figure 2: Simulation results: range estimation for the finally selected node in the NISAN lookup.

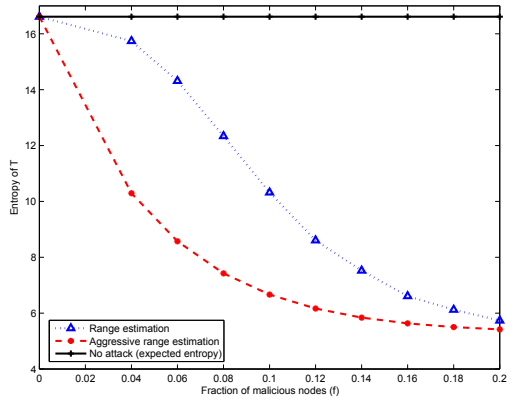

Figure 3: Passive attacks on the NISAN lookup: entropy of the finally selected node.

initial estimation range of T covers all nodes (except Q) on the ring. As the iterative lookup proceeds, whenever a malicious node W is queried by Q, the attacker can learn that W must precede x, and hence she can update the lower bound to be W. In addition, since queried malicious nodes will return their FTs to Q, the attacker can know that the malicious nodes in the returned FTs will be known to Q. Therefore, if W is contacted and some other malicious node Z known to Q is not queried, the attacker can be sure that Z must succeed x; among all the malicious nodes known to Q but not queried, the attacker chooses the closest one that succeeds the lower bound W as the new upper bound. The final estimated range of T includes all nodes between the upper and lower bounds as well as the nodes that are no more than $m-1$ hops preceding the lower bound.

We simulate the above estimation process using C++ with about 500 lines of code. We use a typical network size with #nodes = 10 000 nodes and ID space = 2^{20}, and choose $m = 20$. We use 100 random Chord rings. For each topology, we run 100 independent lookups. The simulation results (Figure 2) show that this passive attack can narrow T to a small number of possible nodes with high probability, compared with the large size of the network. We can calculate the average entropy of T as follows:

$$H(T) = \sum_{i=1}^{n} Pr(RangeSize = i) \cdot \log_2(i)$$

Aggressive range estimation. The range estimation strategy allows the attacker to reliably compute the range of T, which means that the probability that T belongs to the estimation range is 1. If small false positive rate is allowed, the attacker may further narrow the estimation range. Since all fingers in the final TopList are queried, the lower bound will belong to the TopList as long as there is a malicious node in the TopList. This happens with probability $1-(1-f)^m$ (98.85% with $f = 0.2$, $m = 20$). In this case, T is at most m-hops from the lower bound. If the attacker would like to take small risk of false positives (less than 1.2% in the above case), guessing that the lower bound is in the TopList, she can bound the size of the estimation range to be $2m-1$. We call this range estimation strategy as *aggressive range estimation*. Similarly, $H(T)$ is calculated as:

$$\begin{aligned} H(T) &= Pr(T \in TopList) \cdot \log_2(2m-1) \\ &\quad + Pr(T \notin TopList) \cdot \log_2(n) \end{aligned}$$

We can see from Figure 3 that when $f = 0.2$, the entropy of T is only 5.5 (while the expected entropy is over 16), showing that significant information is leaked to a passive attacker. [2]

4. NISAN

The authors of NISAN do not describe how to use the NISAN lookup to construct circuits. To concretely evaluate the effectiveness of the NISAN lookup in building anonymous communication systems, we consider three typical circuit constructions with the NISAN lookup, as shown in Figure 4. For each construction, we describe corresponding attacks to compromise anonymity by using information leaks in the NISAN lookup.

Since our goal is to compromise the whole circuit, the exit relay of the circuit (or tunnel) must be compromised; otherwise, the destination would be unknown to the attacker. Our following analysis is based on the pre-condition that the exit relay of the circuit is malicious. This happens with probability f (assuming the circuit construction is secure).

4.1 Construction I

We start with analyzing a simple approach to constructing circuits using the NISAN lookup. As we know, in Tor the initiator I picks three random relays A, B, and C from the router list provided by a central authority to build the tunnel. However, due to potential high costs of maintaining a global view of the system, such a centralized approach cannot scale to a large number of users. To address this problem, an alternative way could be to let I perform NISAN lookups to locate relays.

However, this construction allows the attacker to link the exit node C to I, since C is contacted by I directly. If C is compromised, then the attacker can learn both the destination and I, and thus break the tunnel. In order to formally calculate the attacker's success probability, we need to consider concurrent lookups when C is being looked up, in that if C is the finally picked node in multiple concurrent lookups,

[2]We note that the attacker could apply the passive attack together with other active attacks to further increase the entropy reduction. One example is to let malicious nodes return FTs that are crafted to have as many bad fingers as possible without tripping the mean distance threshold. For simplicity, in this paper we only focus on the most effective attack – the passive attack.

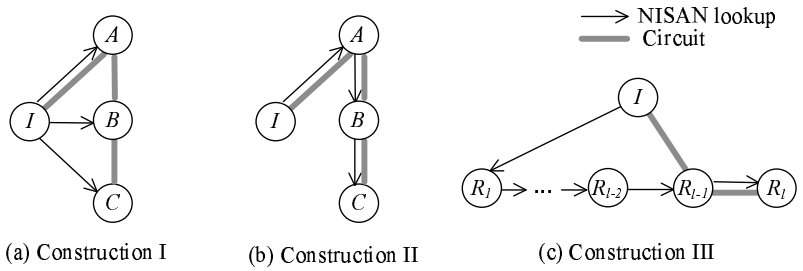

Figure 4: Circuit constructions using the NISAN lookup.

the attacker may not be able to correctly associate C with I. We let BR denote the event that the attacker can successfully break the circuit, FP denote the false-positive case in which C is not correctly linked to I, and M_C represent that C is malicious. Then, we have:

$$\begin{aligned} BR &\equiv M_C \wedge \neg FP \\ Pr(BR) &= f\,(1 - Pr(FP)) \end{aligned}$$

We define α as the number of nodes in the network that are performing lookups at the same time. A reasonable number for α could be $\alpha = n/100$, which means that during this lookup, 1% of all nodes are also performing lookups. A number much larger than this (e.g. $n/10$) would mean that nodes are spending a significant fraction of their time (10%) performing lookups, rather than using them for anonymous communication. Among the α concurrent lookups, $\frac{\alpha}{3}$ of them are searching for an exit node, and $\frac{f\alpha}{3}$ of them end up with a malicious exit node. Therefore,

$$Pr(FP) = 1 - \left(1 - \frac{1}{fn}\right)^{\frac{f\alpha}{3}}$$

Figure 5 shows $Pr(BR)$ as a function of f. We note that the attacker can rely on other observations to further reduce the false positive rate. When C is compromised, the ID of B is known to the attacker. If the attacker can link B to I with fairly high probability, she can significantly decrease $Pr(FP)$, since FP is true only when there is another initiator querying both B and C at the same time. In fact, we shall show in the next subsection that due to information leaks in the NISAN lookup, the chance of successfully linking B to I is very high even if B is honest. In this case, the false positive rate can be reduced to be nearly 0.

4.2 Construction II

The main reason for the weakness of Construction I is that the attacker is able to link the exit node C to the initiator I due to I contacting C directly in the NISAN lookup. One way to avoid this is to let I use the already found relay as proxy to extend the circuit. In particular, I first performs a NISAN lookup to find the first relay A and then establishes a partial circuit with A; next, I uses A as a proxy to look up the second relay B and extends the circuit to B; finally, I requests B to perform a lookup for C. Intuitively, the attacker cannot directly link C to I, since C is contacted by B rather than I.

Unfortunately, this construction alone is vulnerable to two attacks: *public-key modification attack* [3] and *route capture attack*. In public key modification attack, a malicious proxy gives I a manipulated public key of the next relay, and thus

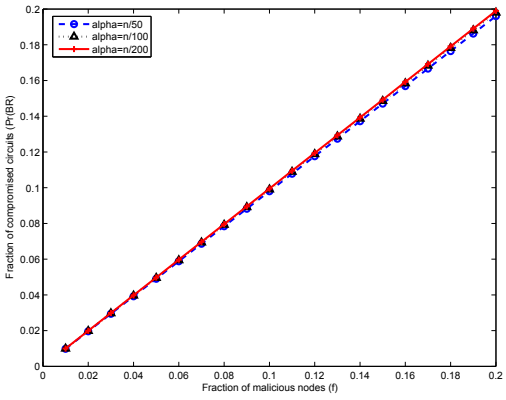

Figure 5: $Pr(BR)$ – Fraction of compromised circuits in Construction I with $n = 1\,000\,000$.

can intercept all messages routing through the tunnel. In route capture attack, the first malicious relay simulates all the remaining lookups, and hence becomes the end relay of the circuit. In order to show that Construction II is inherently vulnerable to passive information leak attack, we consider the worst case for the attacker: we assume there exists a trusted PKI available to the system (although the centralized PKI may reduce scalability of P2P systems and create a single point of failure) that enables I to check the results returned by proxy nodes, so that both the public-key modification attack and the route capture attack can be avoided.

4.2.1 Hop-By-Hop Tracing Attack

We present a passive attack to trace the circuit in a hop-by-hop manner from the exit relay C back to I, so that the attacker can learn significant knowledge of I. The idea is to make use of information leaks in the NISAN lookup to link successive relays on the path.

Suppose X and Y are two (honest) successive relays on the path, the ID of Y is known to the attacker, and the attacker attempts to trace back from Y to X. Recall that we have presented a passive attack on the NISAN lookup in Section 3.2 that allows the attacker to learn lower and upper bounds of the target ID. Let L denote the lower bound related to the lookup performed by X for Y. Since L is contacted by X in the lookup, the attacker could infer X as long as she can associate L with Y. Assuming there are α concurrent lookups being performed in the network, then there could be up to α concurrent lower bounds. The attacker's task is to guess the correct lower bound L associated with Y among all the concurrent lower bounds. More precisely, the attacker will assign probability to all the concurrent lower bounds:

the lower bounds that are more likely to be L will receive high probability, while the lower bounds that are less likely to be L will receive small probability.

Since L is the contacted malicious node laying closest to the lookup target and the Chord ring is directed, intuitively L would be very likely to lay close to Y. As we analyzed in Section 3.2, L would belong to the final TopList as long as there is a malicious node in the TopList. This happens with very high probability (98.85% when $f = 0.2$, $m = 20$). In this case, both L and Y are in the TopList, and hence there are at most $m-1$ nodes between L and Y on the ring. Note that, there could be very few lower bound nodes that are within m hops from Y. This is because to be a lower bound, a node must be malicious, queried in a concurrent lookup, and closest to the lookup target. Therefore, the attacker can make a good guess on L by assigning high probability on the lower bound nodes that are within m hops from Y, and consequently the entropy of X could be very low. The attacker can apply the hop-by-hop tracing attack multiple times until reaching the initiator I.

4.2.2 Analysis

We formally analyze the hop-by-hop tracing attack, and use the entropy of I as the metric to evaluate the anonymity. We show that with the hop-by-hop tracing attack, the entropy of I is much lower than the expected value.

The average entropy of I is calculated as follows (note that the exit relay C must be compromised):

$$\begin{aligned} H(I) &= Pr(M_C \wedge M_B \wedge M_A) \cdot H(I|M_C \wedge M_B \wedge M_A) \\ &+ Pr(M_C \wedge M_B \wedge \neg M_A) \cdot H(I|M_C \wedge M_B \wedge \neg M_A) \\ &+ Pr(M_C \wedge \neg M_B \wedge M_A) \cdot H(I|M_C \wedge \neg M_B \wedge M_A) \\ &+ Pr(M_C \wedge \neg M_B \wedge \neg M_A) \cdot H(I|M_C \wedge \neg M_B \wedge \neg M_A) \\ &+ Pr(\neg M_C) \cdot \log_2(1-f) \cdot n \end{aligned}$$

Since the attacker can break the tunnel when both A and C are compromised, the entropy of I in this case is 0. Hence, $H(I|M_C \wedge M_B \wedge M_A) = 0$, and $H(I|M_C \wedge \neg M_B \wedge M_A) = 0$. Now we calculate $H(I|M_C \wedge M_B \wedge \neg M_A)$ and $H(I|M_C \wedge \neg M_B \wedge \neg M_A)$, respectively.

(1) Calculation of $H(I|M_C \wedge M_B \wedge \neg M_A)$. In this scenario, the ID of A is known to the attacker, and she needs to link A to I. To compute this entropy, we consider two cases: either L belongs to the TopList, or L is outside the TopList. We let H_1 and H_2 denote the entropies for the two cases, respectively. Then, $H(I|M_C \wedge M_B \wedge \neg M_A)$ can be written as:

$$Pr(L \in TopList) \cdot H_1 + Pr(L \notin TopList) \cdot H_2$$

As we mentioned before, $Pr(L \in TopList) \gg Pr(L \notin TopList)$. Hence, for the case $L \notin TopList$, we use an upper bound $\log_2(1-f)n$ to compute H_2 for simplicity.

As for the case $L \in TopList$, we use the following notations when computing H_1. We let $L_{p(1)}, \cdots, L_{p(u)}$ denote the sequence of concurrent lower bounds that precede A in the TopList (if any), and let $L_{s(1)}, \cdots, L_{s(v)}$ denote the list of concurrent lower bounds that succeed A in the TopList (if any). It is possible that a lower bound node could be the lower bound in multiple concurrent lookups. Hence, we let $\mathcal{I}_{p(t)}$, $1 \le t \le u$ (or $\mathcal{I}_{s(r)}$, $1 \le r \le v$) denote the set of concurrent lookup queriers related with $L_{p(t)}$ (or $L_{s(r)}$).

We let $\theta(g)$ denote the probability that L is a lower bound in g concurrent lookups (excluding the one performed by I for A), $g = 0, 1, \cdots$. Since among the α concurrent lookups, $(1-f)^2\alpha$ of them are performed by a honest node and finish with a honest relay, we have $\theta(g) = bino\left(g, (1-f)^2\alpha, \frac{1}{fn}\right)$, where $bino(x, y, z) \equiv z^x(1-z)^{y-x}$. Then, we have:

$$\begin{aligned} H_1 &= -\sum_t \sum_g \sum_{I_{p(t),g} \in \mathcal{I}_{p(t)}} \theta(g) \cdot Pr(I = I_{p(t),g}) \\ &\quad \cdot \log_2\left(\theta(g) \cdot Pr(I = I_{p(t),g})\right) \\ &\quad -\sum_r \sum_h \sum_{I_{s(r),h} \in \mathcal{I}_{s(r)}} \theta(h) \cdot Pr(I = I_{s(r),h}) \\ &\quad \cdot \log_2\left(\theta(h) \cdot Pr(I = I_{s(r),h})\right) \end{aligned}$$

, where $I_{p(t),g}$ denotes an element in $\mathcal{I}_{p(t)}$ when $\mathcal{I}_{p(t)}$ contains $g+1$ elements (including the correct initiator I). Similar definition applies to $I_{s(r),h}$.

Based on the attacker's observation, she cannot distinguish the initiators related with a same lower bound, and thus she will make an even guess on these initiator candidates. Therefore,

$$\begin{aligned} Pr(I = I_{p(t),g}) &= \frac{1}{g+1} \cdot Pr(L = L_{p(t)}) \\ Pr(I = I_{s(r),h}) &= \frac{1}{h+1} \cdot Pr(L = L_{s(r)}) \end{aligned}$$

Now we do some preparation calculations before computing $Pr(L = L_{p(t)})$ and $Pr(L = L_{s(r)})$. We define S_L and S_A as the positions of L and A in the TopList (clockwise), respectively. Since L is the queried malicious node closest to the target, all fingers succeeding L in the TopList must be honest. Therefore, we have:

$$Pr(S_L = i) = f(1-f)^{m-i} \quad 1 \le i \le m$$

Since A is selected randomly from the TopList and can not be L (since L is malicious and A is honest), we have:

$$Pr(S_A = j) = \frac{1}{m-1} \quad 1 \le j \le m$$

We define β as the probability that a node is a lower bound. Then β is calculated as:

$$\beta = f\left(1 - \left(1 - \frac{1}{fn}\right)^{(1-f)\alpha}\right)$$

There are $r-1$ lower bounds laying between $L_{s(r)}$ and A, and thus $Pr(L = L_{s(r)})$ is:

$$\sum_{j=1}^{m-r} Pr(S_A = j) \cdot \sum_{i=j+r}^{m} Pr(S_L = i) \cdot bino(r-1, i-j-1, \beta)$$

Since all fingers succeeding L in the TopList are honest and are impossible to be lower bounds, only $L_{p(u)}$ (the closest lower bound that precedes B) is likely to be L. So,

$$\begin{aligned} Pr(L = L_{p(t)}) &= 0, \quad 1 \le t \le u-1 \\ Pr(L = L_{p(u)}) &= \sum_{j=2}^{m} Pr(S_A = j) \cdot \sum_{i=1}^{j-1} Pr(S_L = i) \end{aligned}$$

(2) Calculation of $H(I|M_C \wedge \neg M_B \wedge \neg M_A)$. In this scenario, the ID of B is learned by the attacker, and she needs to trace back two hops: B to A, and A to I. Using the same strategy, the attacker first finds a number of candidates for A and assigns them with probabilities. The calculation of

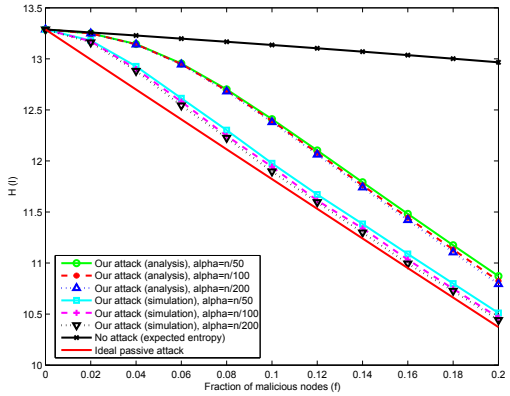

Figure 6: Entropy of I to the attacker. $n = 10\,000$.

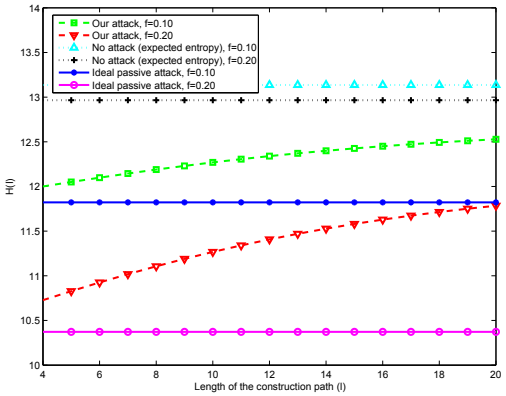

Figure 7: $H(I)$ with a long construction path. $n = 10\,000$, $m = 20$, **and** $\alpha = n/100$.

these probabilities for A is the same as the above scenario. Then for each candidate of A, the attacker further finds a number of candidates for I. The entropy is averaged over all candidates of I. Since the calculation is similar in spirit with scenario (1), we omit the details for simplicity.

4.2.3 Results

We calculate $H(I)$ using the above analysis, and the results are given in Figure 6. We also simulate the hop-by-hop tracing attack using the simulator we developed for the NISAN lookup. We adopt the same configuration as the simulation of the NISAN lookup (Section 3.2). We use $n = 10\,000$ nodes, and 100 random Chord rings. The simulation results are averaged over 10 000 independent runs. Figure 6 shows that our analysis results are upper bounds of the actual values. This is because we use an upper bound value for H_2 when computing $H(I|M_C \wedge M_B \wedge \neg M_A)$.

We compare our attack against the ideal passive attack, in which the attacker can obtain full knowledge of I as long as the exit relay is compromised[3], i.e., $H(I) = (1-f)\log_2(1-f)n$. We can see that the entropy of I achieved by our attack is very close to the optimal value of the ideal attack, showing that the hop-by-hop tracing attack is a very powerful attack.

4.3 Construction III: Further Enhancements to Construction II

One way to mitigate the attacker's knowledge of I is to use a longer path. To reduce the expense of relaying traffic over a long path and mitigate selective DoS attack, we adopt the idea of ShadowWalker proposed by Mittal and Borisov [19] to construct the circuit: I first follows Construction II to find a sequence of l relays, and then use the last two relays to build the circuit.

However, this approach can only mitigate attackers' knowledge of I to some extent, since timing analysis (on lookup traffic) allows the attacker to associate the first malicious relay and the last non-exit malicious relay on the path, and thus skip linking the relays in the middle. Figure 7 shows that the attacker can still reduce the entropy of I by 1.2 bits when $l = 20$ and $f = 0.2$. Furthermore, this construction substantially increases bootstrapping latency and renders the system vulnerable to DoS attack (i.e., preventing clients from conducting anonymous communication by dropping lookup traffic).

[3]Note that when the exit relay is honest, it is impossible for the attacker to learn both I and the destination, so the entropy in this case is the maximum value $\log_2(1-f)n$.

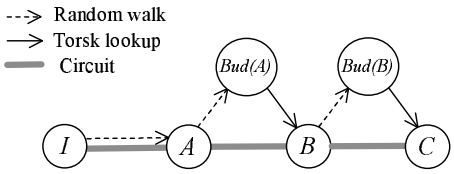

Figure 8: Torsk circuit construction.

5. TORSK

From the discussion of the NISAN circuit constructions, we can see that letting the initiator I or the end relay of a partial circuit to look up the next hop allows an attacker to launch hop-by-hop tracing attack to associate the destination with I. The authors of Torsk [16] proposed an alternative design for circuit construction by using secret buddy nodes. We show that although Torsk is immune to passive hop-by-hop tracing attack, introducing secret buddies renders the system seriously vulnerable to active attacks.

5.1 Secret Buddy

One important notion in Torsk is *secret buddy*. Torsk requires each node to privately select a number of random buddies from the global pool of nodes in the network; a lookup querier will request one of its buddies to perform the lookup on its behalf, so that an attacker cannot associate the lookup target with the querier. Then, hop-by-hop tracing attack cannot be applied, as long as the relationship between nodes and their buddies is kept secret from the attacker and the buddies are not compromised.

Torsk proposes four mechanisms to ensure secrecy of relationship between nodes and their buddies as well as the security of the buddy selection process. First, Torsk proposes to use anonymous random walks to select buddies, leaking little information about the relationship between the random walk initiator and the finally picked node. Second, the buddy selection process is performed off-line (before a lookup is performed), so that timing analysis attack cannot be applied. Third, Torsk requires that buddies must be one-time use. Otherwise, an attacker can associate a node U with its buddy, by requesting U to perform a lookup for x^* (for circuit extension), and the node that looks up x^* subse-

quently is learned as U's buddy node. Finally, Torsk applies certificate verification at each hop of the random walk, preventing attackers from biasing the random walk to increase the chance of malicious nodes being selected as buddies.

5.2 Buddy Exhaustion Attack on Torsk

Although using secret buddies to extend circuits breaks off the "clue" of tracing the path backwards, the one-time use of buddies makes the system vulnerable to *buddy exhaustion attack*: an attacker can prevent the circuits having honest entry nodes from being extended, by exhausting buddies of the end relays of these partial circuits. Consequently, a large fraction of constructed circuits would have malicious entry nodes. Among these circuits, the attacker can further apply selective DoS attack [3] by letting the malicious entry nodes tear down the circuits that have honest exit relays. As a result, the majority of built circuits would have both compromised entry and exit nodes, and hence the attacker can launch timing analysis attack to break the tunnel.

Now we describe buddy exhaustion attack in details. First, we let all malicious nodes choose colluding nodes as their buddies. Hence, when the attacker observes a lookup performed by a honest buddy, she is sure that this lookup is requested by a honest node. In particular, we consider the case when the entry relay A is honest (if A is malicious, then selective DoS attack can be applied directly). As shown in Section 3.1, due to information leaks, the attacker can observe nearly all lookups (over 99.5% when $f = 0.2$). Hence, when $Bud(A)$ looks up B, the attacker can observe the lookup and infer that A is honest[4]. Then, the attacker floods B with sufficiently many lookup requests, so that all cached buddies of B would be exhausted. Consequently, B will have no buddy to perform any lookup for A to extend the circuit.

The buddy exhaustion attack can successfully defeat the second mechanism of Torsk (off-line buddy selection), since the victim node B under the buddy exhaustion attack has to find new buddies during the circuit construction.

Now we show that the attacker can further prevent B from recovering from the buddy exhaustion attack, by blocking B from finding new buddies. We first briefly review the random walk process for buddy selection. First, the querier randomly picks a finger F_1 out of its FT as the first hop of the random walk, and then asks F_1 for its FT, from which the second hop F_2 is selected at random. This process is iteratively performed for l hops, and is followed by a geometrically distributed "tail" with expected length l. To prevent the attacker from biasing the random walk, the querier requests and verifies the certificates related to the incoming and outgoing links of F_i. If an invalid certificate is found, the random walk is restarted. Therefore, to prevent the querier from finding new buddies, the attacker can let malicious nodes involved in a random walk return invalid certificates to force the random walk to restart. Since the random walks are typically long (with $2l$ hops, $l = 14$ [5]), the attacker could have a very good chance to interrupt the buddy selection process.

[4]Note that even if $Bud(A)$ is not honest, the attacker can still learn that A is honest. The attacker can obtain the ID of A through $Bud(A)$, since A requests $Bud(A)$ to perform the lookup.

[5]Values of l are selected according to the Torsk paper [16]: $l = \lceil \frac{2\log_2 \frac{n\kappa}{\epsilon}}{3} \rceil$, where we use typical values $\kappa = 3$, $\epsilon = 0.01$.

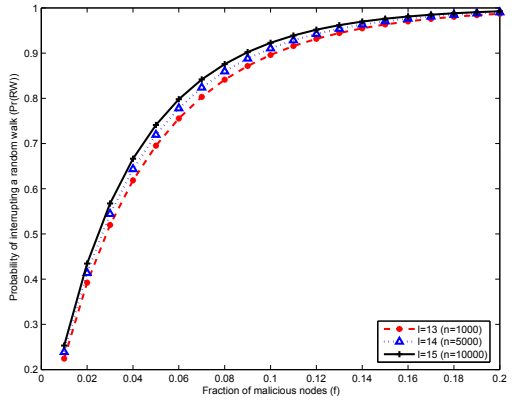

Figure 9: $Pr(RW)$ – Probability that the attacker can block a honest node from performing a random walk to select a buddy.

5.3 Analysis

Now we analyze the anonymity of Torsk under buddy exhaustion attack. We let RW denote the event that the attacker can interrupt a random walk. Then, $Pr(RW)$ equals the chance that there is at least one malicious node picked in the random walk. Therefore,

$$Pr(RW) = (1-f)^l \left(1 - \sum_{i=1}^{\infty}(1-p)^{i-1}p(1-f)^i\right) + \left(1-(1-f)^l\right) = 1 - \frac{p(1-f)^{l+1}}{1-(1-f)(1-p)}$$

where $p = \frac{1}{l}$, representing the probability that the random walk stops at a particular hop of the tail. Figure 9 shows that when $f = 0.2$, the attacker can successfully interrupt about 99% random walks.

A node under buddy exhaustion attack could try to repeatedly perform random walks in parallel, in hopes of finding new buddies before the client gives up the circuit construction. However, the number π of concurrent random walks a querier can perform is limited by its computational capacity. During the random walk, the querier needs to perform $2d$ certificate-verifying operations at each hop (the Torsk authors [16] suggest d is 8 or 16). We let φ denote the average latency between two hops in a random walk (according to the experimental results in [16], $\varphi \approx 0.2sec$). Suppose that the time needed to verify a public-key signature is τ (e.g., $\tau = 0.5ms$). Then, π is bounded by $\frac{\varphi}{2d\tau}$, which is equal to 25 with $d = 8$, $\varphi = 0.2sec$, and $\tau = 0.5ms$.

Apart from π, the maximum number of random walks the querier can perform in parallel before the client's timeout is also determined by the time σ needed to perform an interrupted random walk. σ is determined by the number of hops that a random walk travels before meeting the first malicious node. Hence, the expected value of σ, $E(\sigma)$, is calculated as:

$$\varphi\left(\sum_{j=1}^{\infty}(j+l)(1-p)^{j-1}p(1-f)^{l+j-1}f + \sum_{i=1}^{l} i(1-f)^{i-1}f\right)$$

We assume the attacker floods B with $\nu + \mu$ lookup requests, where μ is the number of buddies maintained by each node. We let EX denote the event that the attacker can successfully prevent B from extending the circuit, and let Φ

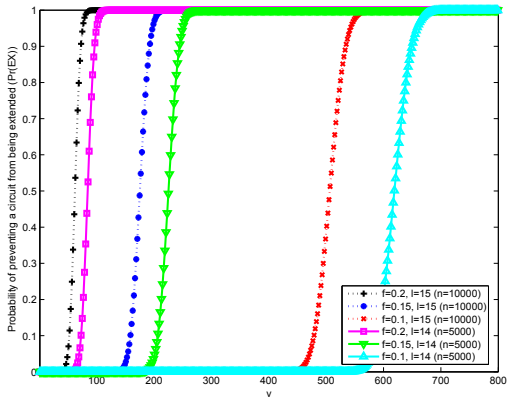

Figure 10: $Pr(EX)$ – Probability that the attacker can prevent a honest node from extending a circuit. $\Phi = 5min$, and $\pi = 25$.

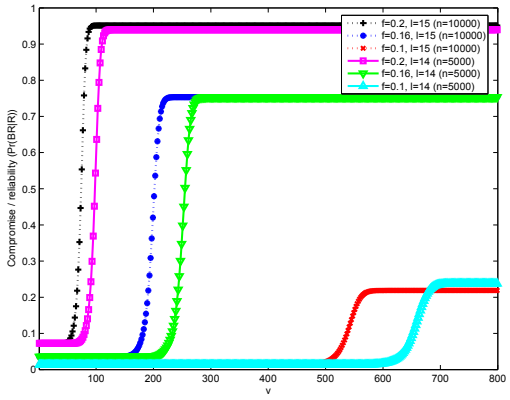

Figure 11: Effect on varying the number of flooding lookup requests. $\Phi = 5min$, and $\pi = 25$.

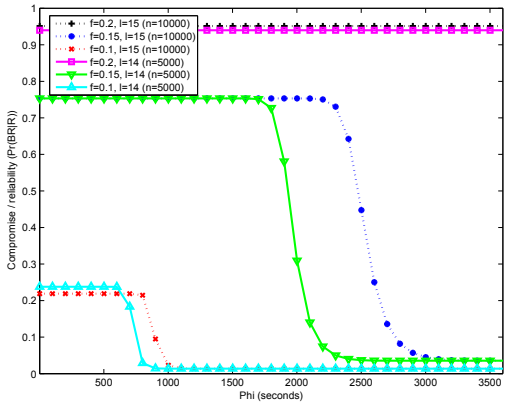

Figure 12: Effect on varying the client's maximum waiting time Φ. $\nu = 500$, and $\pi = 25$.

denote the client's maximum waiting time before giving up the circuit construction (e.g., $\Phi = 5min$). Then,

$$Pr(EX) = \sum_{i=0}^{\nu} bino\left(i, \frac{\pi\Phi}{E(\sigma)}, 1 - Pr(RW)\right)$$

Figure 10 shows that when ν is larger than a threshold value (e.g., 200 when $f = 0.15$, $n = 10\,000$), the attacker can prevent a particular relay from extending the circuit with success probability nearly 1. We note that it is reasonable to consider fairly large ν, in that the attacker may ask her controlled malicious nodes to flood a particular node in collaboration. For instance, in a network with $n = 10\,000$ nodes and 20% malicious nodes, there would be 2000 flooding requests if each malicious node contributes one.

Now we analyze the anonymity of Torsk under buddy exhaustion attack. We let R denote the event that a circuit is reliable (either when both the entry and exit nodes are compromised, or when the attacker fails to launch buddy exhaustion attack). R is equivalent to:

$$(M_A \wedge M_C) \vee \neg M_A \wedge \neg M_B \wedge \neg M_C \wedge \neg(LK \wedge EX)$$

We let LK denote the event that a lookup is observed by the attacker, and let BR represent that a circuit is compromised. Then, the fraction of compromised circuits out of reliable circuits, $Pr(BR|R)$, is calculated as:

$$\frac{Pr(BR \wedge R)}{Pr(R)} = \frac{f^2}{f^2 + (1-f)^3(1 - Pr(LK)Pr(EX))}$$

Figures 11, and 12 present the analysis results for $Pr(BR|R)$ with different impact factors. We simulate the buddy exhaustion attack on Torsk. We generate 20 random topologies, and for each topology we perform 1000 circuit constructions. Figure 13 shows the simulation results, with $\nu = 500$, $\Phi = 5min$, and $\pi = 25$. We can see that with buddy exhaustion attack, the attacker can break over 80% of all constructed circuits.

5.4 Improvements to Torsk

The authors [16] suggested that the random walk process for buddy selection needs to start over whenever a queried malicious node returns an invalid certificate. This allows the attacker to block honest nodes from finding new buddies. Nevertheless, we think restarting the random walk upon seeing an invalid certificate is not a necessity. Instead, the random walk can step back one hop and choose another random finger as the next hop. If all fingers of a particular hop are malicious, the random walk can step back one more hop and try other branches. By doing so, it would be infeasible for the attacker to interrupt a random walk, and thus honest nodes can find buddies as expected.

However, it is important to note that the querier under buddy exhaustion attack may still need substantial time to find enough buddies to get through the flooding lookup requests, even though random walks are not interrupted. For example, when $\nu = 1200$, $l = 15$ and $\pi = 25$, it will take the victim node $\frac{2l\varphi\nu}{\pi} \approx 5min$ to find enough buddies to perform a lookup. It is very likely that the client will not wait $5min$ before starting communication.

We can speed up the process of buddy selection via some cryptographic mechanisms. For example, if the certificates associated with each hop are formed into Merkle trees, the computational latency at each hop could be reduced[6]. However, the effectiveness of this strategy still depends on the attacker's capability of sending flood requests. Finally, we note that each bogus lookup request may generate consid-

[6]Forming certificates into Merkle trees increases maintenance costs, since whenever a node joins/churns or a signature expires, all related Merkle trees have to be reconstructed.

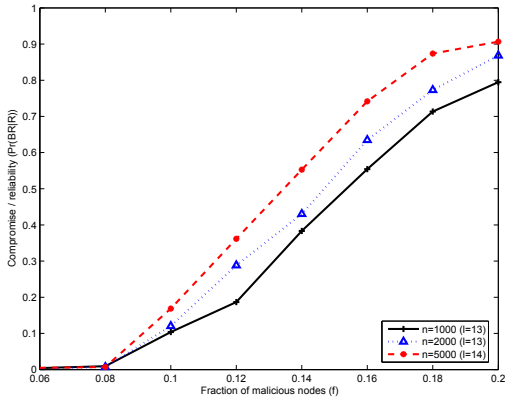

Figure 13: Simulation results: fraction of compromised circuits in the reliable circuits. $\nu = 500$, $\Phi = 5min$, **and** $\pi = 25$.

erable traffic due to the buddy selection process, and therefore buddy exhaustion attack could substantially degrade network performance and may lead to system overload.

6. RELATED WORK

P2P anonymous communication systems have been the subject of a lot of research [17, 24, 9, 29, 19, 25, 16]. AP3 and Salsa are among the first attempts at using DHT lookups for anonymous communication. Their secure lookup protocols rely heavily on redundancy and are shown to be vulnerable to information leak attacks [18]. NISAN and Torsk are also based on DHT lookups, but they are specifically designed to mitigate information leak attacks. We have proposed a variety of passive and active attacks on NISAN and Torsk that significantly reduce user anonymity.

An alternate approach to building circuits for anonymous communication is to connect relays into a restricted topology and construct circuits along paths in this topology [9, 29, 19]. For example, in Tarzan [9], each node has a small set of *mimics*, and all circuits must be created on links between mimics. To verify that paths are constructed correctly, nodes in Tarzan maintain a global view of the system. This limits Tarzan to networks of about 10 000 or fewer nodes. MorphMix [29] is designed to eliminate such scalability constraints by creating a randomized, unstructured overlay between relays, with circuits built on paths along the overlay. Instead of maintaining a global view, MorphMix uses a collusion detection mechanism involving witness nodes to verify neighbor information. However, the collusion detection mechanism can be circumvented by a set of colluding adversaries who model the internal state of each node, thus violating anonymity guarantees [32]. In [25], the authors of NISAN also considered a random walk construction to complement their DHT-lookup-based design, which is found to be vulnerable to active attacks. ShadowWalker [19] is a recent design that proposes the use of redundant structured topologies to enable verification of neighbor information while mitigating information leak attacks.

Danezis and Clayton [5] studied attacks on peer discovery and route setup in anonymous P2P networks. They showed that if the attacker learns the subset of nodes known to the initiator (by observing lookups, for example), its routes can be fingerprinted unless the initiator knows about the vast majority of the network. Danezis and Syverson [7] extended this work to observe that an attacker, who learns that certain nodes are *unknown* to the initiator, can carry out attacks as well and separate traffic going through a relay node.

Reiter and Rubin [28] proposed the predecessor attack, which was later extended by Wright et al. [34, 35, 36]. In this attack, an attacker tracks an identifiable stream of communication over multiple communication rounds and logs the preceding node on the path. To identify the initiator, the attacker uses the observation that an initiator is more likely to be the predecessor than any other node in the network. Similar to predecessor attacks, there is a thread of research that deals with degradation of anonymity over a period of time. Berthold et al. [2] and Raymond [27] proposed intersection attacks that aim to compromise sender anonymity by intersecting sets of users that were active at the time the intercepted message was sent, over multiple communication rounds. Similarly, Kesdogan et al. [12] used intersection to find recipients of a given user's message. A statistical version of this attack was proposed by Danezis [4] and later extended by Mathewson and Dingledine [14].

An important point of our paper is that, when building anonymous systems, it is important not to abstract away the properties of the system that can affect anonymity. Similar in spirit to ours, a lot of recent research has focused on details abstracted away by conventional analysis models to break the anonymity of the system. Such details include congestion and interference [22, 1], clock skew [21], heterogeneous path latency [11, 1], the ability to monitor Internet exchanges [23], and reliability [3]. For example, Murdoch and Zieliński [23] showed that Internet exchange-level adversaries are capable of observing a vast majority of user traffic and could degrade user anonymity by performing end-to-end timing analysis. Borisov et al. [3] proposed selective-DoS attack and showed that attackers could selectively affect the reliability of the system to degrade user anonymity.

7. CONCLUSION

In this paper, we have analyzed mechanisms that hide the relationship between a user and its selected relays in two state-of-the-art anonymous communication systems NISAN and Torsk. We presented passive attacks on the NISAN lookup and show that it is not as anonymous as previously thought. Information learned from the NISAN lookup can be used degrade anonymity of constructed circuits.We have also shown that the information leaks in the Torsk lookup allow an attacker to launch active attacks to defeat its secret buddy mechanism, and substantially compromise user anonymity. Our analysis of NISAN and Torsk shows that their key mechanisms to anonymously look up a node are vulnerable to either passive attacks or active attacks, motivating the search for a secure and anonymous DHT lookup.

8. ACKNOWLEDGMENTS

We would like to thank Andriy Panchenko and Nicholas Hopper for helpful discussions to let us better understand the NISAN and Torsk schemes. We are very grateful to Andriy Panchenko for sharing their code for the NISAN simulation, and George Danezis for motivating us to work on this problem. We also thank the anonymous reviewers for their invaluable feedback on the draft manuscript. This research was supported in part by NSF grants: CNS-0627671 and CNS-0524695.

9. REFERENCES

[1] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *IH*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.

[2] O. Berthold, H. Federrath, and M. Köhntopp. Project anonymity and unobservability in the Internet. In *CFP*, pages 57–65, New York, NY, USA, 2000. ACM.

[3] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? *ACM CCS*, 2007.

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

[5] G. Danezis and R. Clayton. Route Fingerprinting in Anonymous Communications. *IEEE Int. Conf. on Peer-to-Peer Computing*, pages 69–72, 2006.

[6] G. Danezis, R. Dingledine, and N. Mathewson. Maxminion: Design of a type iii anonymous remailer protocol. *in IEEE Symposium on Security and Privacy*, May 2003.

[7] G. Danezis and P. Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In *PETS*, pages 151–166, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, August 2004.

[9] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. *ACM CCS*, 2002.

[10] D. Goodin. Tor at heart of embassy passwords leak. *The Register*, September 10 2007.

[11] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? In *ACM CCS*, October 2007.

[12] D. Kesdogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In F. Petitcolas, editor, *IH*. Springer-Verlag, LNCS 2578, October 2002.

[13] K. Loesing. Measuring the tor network: Evaluation of client requests to the directories. *Tech. Report*, 2009. `https://git.torproject.org/checkout/metrics/master/report/dirreq/directory-requests-2009-06-26.pdf`.

[14] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *PET*, volume 3424 of *LNCS*, pages 17–34, 2004.

[15] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *IPTPS*, 2001.

[16] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. *ACM CCS*, November 2009.

[17] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. Ap3: Cooperative, decentrialized anonymous communication. *ACM SIGOPS European Workshop*, 2004.

[18] P. Mittal and N. Borisov. Infomation leaks in structured peer-to-peer anonymous communication systems. *ACM CCS*, 2008.

[19] P. Mittal and N. Borisov. Shadowwalker: Peer-to-peer anonymous communication using redundant structured topologies. *ACM CCS*, November 2009.

[20] U. Moller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol – version 2. *IETF Internet Draft*, July 2003.

[21] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *ACM CCS*, October 2006.

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

[23] S. J. Murdoch and P. Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In N. Borisov and P. Golle, editors, *PETS*, Ottawa, Canada, June 2007. Springer.

[24] A. Nambiar and M. Wright. Salsa: A structured approach to large-scale anonymity. *ACM CCS*, 2006.

[25] A. Panchenko, S. Richter, and A. Rache. Nisan: Network information service for anonymization networks. *ACM CCS*, November 2009.

[26] L. D. Paulson. News briefs. *IEEE Computer*, 39(4):17-19, April 2006.

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

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

[29] M. Rennhard and B. Plattner. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. *WPES*, 2002.

[30] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[31] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. *International Workshop on Design Issues in Anonymity and Unobservaility, vol. 2009, LNCS, Springer*, July 2000.

[32] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In G. Danezis and P. Golle, editors, *PET*, pages 368–384, Cambridge, UK, June 2006. Springer.

[33] P. Wang, I. Osipkov, N. Hopper, and Y. Kim. Myrmic: Secure and robust dht routing. *Tech. Rep. Univerity of Minnesota DTC Research Report*, 2006.

[34] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *NDSS*. IEEE, February 2002.

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

[36] M. Wright, M. Adler, B. N. Levine, and C. Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM TISSEC*, 4(7):489–522, November 2004.