TRUSTWORTHY AND SCALABLE ANONYMOUS COMMUNICATION

BY

PRATEEK MITTAL

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Assistant Professor Nikita Borisov, Chair
Assistant Professor Matthew Caesar
Dr. George Danezis, Microsoft Research
Adjunct Professor P. R. Kumar
Professor Nitin Vaidya

ABSTRACT

The architectures of deployed anonymity systems such as that of the Tor network suffer from the problems of (a) limited scalability, (b) reliance on a few central points of trust, and (c) trust issues due to Sybil attack. In this thesis, we investigate the design of novel approaches to anonymous communication that are scalable, decentralized, and Sybil-resilient.

First, we begin by investigating security vulnerabilites in existing P2P anonymity systems, and find fundamental limitations in their designs. Second, we propose novel protocols for P2P anonymous communication that can successfully overcome these limitations. Third, we describe a protocol for detecting malicious Sybil identities using information about social network trust relationships. Fourth, we present protection mechanisms for DHTs that also leverage social network trust relationships to defend against the Sybil attack while preserving the privacy of social contacts and providing a basis for pseudonymous communication. Finally, we describe a protocol for trustworthy and scalable anonymous communication that can directly leverage users' trusted social contacts. We evaluate the effectiveness of our protocols using theoretical analysis, simulations, implementations and a Facebook application.

To my family

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of my advisor Nikita Borisov. I would like to thank Nikita for his mentorship and support throughout my doctoral studies. Nikita has been a role model for my research career, and his approach to problem solving and mentoring students has been a great learning experience. I would also like to thank Matthew Caesar, who has been like a second advisor to me. I learned a lot about systems research by working with Matt, and have benefited tremendously from our conversations over these years.

I am grateful to my doctoral committee members George Danezis, P. R. Kumar and Nitin Vaidya for their feedback. My internship with George at Microsoft Research introduced me to the wonderful world of social networks, which has helped shape my research.

I would like to thank my fellow students and faculty at Illinois for all their help. In particular, I am grateful to my colleagues in the Hatswitch research group - Amir Houmansadr, Robin Snader, David Albrecht, Nabil Schear, Qiyan Wang, Joshua Juen, Sonia Jahid, Giang Nyugen, Xun Gong, and Anupam Das. Thanks for many insightful discussions, and making grad school a memorable experience.

I would like to thank all of my collaborators, both at Illinois and outside. My internships at ICSI and Philips Research broadened my research interests. I am grateful to my mentors, Vern Paxson, Robin Sommer, Vinay Varadan, Nevenka Dimitrova, Angel Janevski, Sirtharthan Kamalakaran, and Nilanjana Banerjee. I am also grateful to Matthew Wright, who is my collaborator on the Pisces protocol in this dissertation, and whose paper on Salsa inspired my doctoral research.

I would like to thank the Tor project and the PETS community for their feedback, in particular, Roger Dingledine, Nick Mathewson, Paul Syverson, Apu Kapadia, Ian Goldberg, Steven Murdoch, Nick Hopper, Carmela Tron-

coso, Femi Olumofin, Eugene Vasserman, and Mike Perry.

My work was supported in part by NSF CNS 06-27671, 08-31488, 09-53655, and an HP Labs IRP grant. I would also like to thank the ECE department at Illinois for its support in the form the RAMBUS award and M. E. VanValkenburg award.

Parts of this dissertation have been published in ACM CCS 2007, ACM CCS 2008, ACM CCS 2009, NDSS 2009, ACM CCS 2010, USENIX HotSec 2010, USENIX Security 2011, NDSS 2012 and ACM TISSEC 2012. The Pisces protocol is currently under submission at ACM CCS 2012.

Finally, I would like to thank my family and friends. I am especially grateful to Amelia, for her love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Anonymous communication is a key privacy enhancing technology, and is gaining widespread popularity in an era of pervasive surveillance [2]. Anonymous communication hides the identity of communication partners from third parties, or hides user identity from the remote party. The Tor network [3], deployed in 2003, now serves hundreds of thousands of users [4] and carries terabytes of traffic a day [5]. Originally an experimental network used by privacy enthusiasts, it is now entering mainstream use; a recent attack showed a number of foreign consulates were using Tor to avoid surveillance by their host countries [6].

The Tor network comprises approximately 2 000 relays as of April 2011 [7]. Tor clients first download a complete list of relays from *directory servers*. The relay information is signed by trusted *directory authorities* to prevent directory servers from manipulating its contents. Clients select three random relays to build *circuits* for anonymous communication. We note that there are several problems with Tor's architecture. First, Tor requires all users to maintain a global view of all the relays. As the number of relays increases, maintaining a global view of the system becomes costly, since churn will cause frequent updates and large bandwidth overhead. Second, the Tor protocol relies on a few centralized directory authorities, which makes them an attractive target for the attackers. Finally, Tor is also vulnerable to Sybil [8] attacks, in which a single attacker can insert a large number of Tor relays and break the anonymity guarantees provided by the network.

In order to address the first problem, a peer-to-peer architecture will likely be necessary. However, peer-to-peer networks present new challenges to anonymity, one of which is the ability to securely locate relays for anonymous traffic. Additional challenges include defending against the Sybil attack to prevent a single entity from compromising the security guarantees of the system, as well as eliminating central points of trust from the design. *In this*

thesis, we investigate the design of a trustworthy (Sybil-resilient) peer-to-peer (P2P) anonymous communication system.

1.1 Contributions

1.1.1 Information leaks in existing mechanisms

As a first step towards our goal, we perform a security evaluation of the state-of-art P2P anonymous communication system called Salsa [9]. Salsa [9] is built on top of a distributed hash table (DHT), and uses a specially designed secure lookup operation to select random relays in the network. The secure lookups use redundant checks to mitigate attacks that try to bias the result of the lookup. We show that Salsa is vulnerable to information leak attacks: as the attackers can observe a large fraction of the lookups in the system, a node's selection of relays is no longer anonymous and this observation can be used to compromise user anonymity [10, 11]. Salsa was designed to tolerate up to 20% of compromised nodes; however, our analysis shows that in this case, over one quarter of all paths will be compromised by using information leaks. We show that Salsa is also vulnerable to a selective denial-of-service attack, where nodes break circuits that they cannot compromise. Selective DoS attack is devastating for user anonymity in Salsa; at 20% compromised nodes, the probability of path compromise is 0.7, thus rendering the system insecure for most proposed uses.

1.1.2 P2P anonymity using structured topologies

Next, we propose ShadowWalker, a P2P anonymous communication system that satisfies all of our design goals (secure against a Byzantine adversary, decentralized design) assuming an external Sybil defense mechanism. ShadowWalker is based on a random walk over structured topologies. The key challenge in this system is to prevent attackers from biasing the results of the random walk process (*route capture* attack). Our main idea is the creation of *shadow nodes*, which redundantly verify the correctness of a given nodes' routing table and certify it as correct (to prevent attackers from biasing the results). Such certificates can then be used to check the steps of a random

walk; by using certificates rather than online checks, we can avoid information leak attacks [12]. We show that our design is effectively able to prevent route capture attacks by employing a small number of shadows per node. In particular, the anonymity levels achieved by our system are much higher than those of Salsa [9] when 20% of all nodes are compromised.

1.1.3 Sybil detection using social networks

Next, we propose SybilInfer, a protocol to detect Sybil identities using social network trust relationships. Our main contribution is to propose a formal model for detecting Sybil nodes in a social network, that makes use of all information available to the defenders. Our model casts the problem of detecting Sybil nodes in the context of Bayesian inference: given a set of stated relationships between nodes, the task is to label nodes as honest or dishonest. Based on some simple and generic assumptions, like the fact that social networks are fast mixing [13], we sample cuts in the social graph according to the probability they divide it into honest and dishonest regions. These samples not only allow us to label nodes as honest or Sybil attackers, but also to associate with each label a measure of uncertainty. We demonstrate the practical efficacy of our approach using both synthetic scale-free topologies as well as real world livejournal data. We show an order of magnitude security improvements over state-of-art systems like SybilGuard and SybilLimit.

1.1.4 Sybil-resilient distributed hash table using social networks

SybilInfer is a generic tool for Sybil defense that does not provide support for P2P routing. Moreover, SybilInfer requires knowledge of the full social network topology, presenting challenges pertaining to scalability as well as privacy of social contacts. Next, we propose X-Vine, a protection mechanism for distributed hash tables that does not require knowledge of the full social graph. X-Vine is resilient to denial of service via Sybil attacks, and in fact is the first Sybil defense that requires only a logarithmic amount of state per node, making it suitable for large-scale and dynamic settings. X-Vine also helps protect the privacy of users social network contacts and keeps their IP

addresses hidden from those outside of their social circle, providing a basis for pseudonymous communication. X-Vine operates entirely by communicating over social network links, and leverages ideas from network layer distributed hash tables such as virtual ring routing [14]. We evaluate X-Vine using theoretical analysis, simulations, PlanetLab implementation and a Facebook plugin. We find that X-Vine allows Sybil-resilient communications with low stretch and communication overhead.

1.1.5 P2P anonymity using unstructured social network topologies

We note that a combination of ShadowWalker with SybilInfer yields a Sybil-resilient protocol for P2P anonymous communication, and satisfies the objectives of this thesis. However, such a design is not optimal. Malicious nodes accepted by social network based Sybil defenses such as SybilInfer are localized with respect to honest nodes. ShadowWalker arranges all accepted nodes into a structured topology, resulting in malicious nodes being uniformly distributed in the topology, and losing information about the localized nature of the adversary. We propose PISCES, a protocol for P2P anonymous communication that directly performs secure random walks on unstructured social network trust topologies. Pisces brings together the benefits of trust relations, in the form of a social network, with the advantages of secure random walks in a P2P environment. The main technique that we leverage for our random walks is the *reciprocal neighborhood policy (RNP)*. The RNP states that all links in the graph must be bi-directional; for social networks, this means that social relationships must be mutual (friends, not followers). By providing techniques that enforce this policy (leveraging the X-Vine protocol), we ensure that a random walk on the topology is truly random. Using a real world social network topology and a reasonable set of trust assumptions, we find that Pisces significantly outperforms ShadowWalker, and provides up to 6 bits higher entropy in a single communication round.

1.1.6 Organization

We first discuss the state of art in P2P anonymity, as well as mechanisms leveraging social network trust relationships in Chapter 2. We present information leak attacks in existing P2P anonymity designs in Chapter 3. We propose the ShadowWalker protocol that resists information leak attacks in Chapter 4. In Chapter 5 and Chapter 6, we present the SybilInfer and X-Vine protocols for Sybil defense using social networks. We present the Pisces protocol for anonymous communication using social networks in Chapter 7, and conclude in Chapter 8.

CHAPTER 2

LITERATURE REVIEW

In this section, we review related work on anonymous communication, peer-to-peer anonymous communication, as well as Sybil defense mechanisms. We also describe our threat model.

2.1 Low-Latency Anonymous Communication

Anonymous communication systems can be classified into low-latency and high-latency systems. High-latency anonymous communication systems like Mixminion [15] and Mixmaster [16] are designed to be secure even against a powerful global passive adversary; however, the message transmission times for such systems are typically on the order of several hours. This makes them unsuitable for use in applications involving interactive traffic like web browsing and instant messaging. The focus of this dissertation is on low-latency anonymous communication systems, such as Tor [3], Anonymizer.com [17], AN.ON, I2P [18], Freedom [19] and Freenet [20].

Most of these networks rely on *onion routing* [21–23] for anonymous communication. Onion routing enables anonymous communication by using a sequence of relays as intermediate nodes to forward traffic. Such a sequence of relays is referred to as an onion routing *circuit*. A key property of onion routing is that each relay on the circuit only sees the identity of the previous hop and the next hop, but no single relay can link both the initiator and the destination of the communication.

Anonymizer.com [17] is effectively a centralized proxy server with a single point of control. If the proxy server becomes compromised or is subject to subpoena, the privacy provided by the system would be lost. AN.ON [24] distributes the trust among three independently-operated servers; again, the compromise of just a few nodes suffices to undermine the entire system. Both

Anonymizer.com and AN.ON are prone to simple, flooding-based denial of service attacks.

Tor is the most popular anonymous communication system in use today. Tor serves hundreds of thousands of users [4], and carries terabytes of traffic every day [5]. The Tor network is substantially more distributed than either Anonymizer.com or AN.ON, with over 2 400 onion routers as of November 2011 [7]. This helps to protect against direct attacks and eavesdropping on the entire system.

Tor relies on trusted entities called *directory authorities* to maintain up-to-date information about all relays that are online in the form of a network consensus database. Other low-latency anonymity systems such as I2P [18] and Freedom [19] also use centralized directory servers. Users download the full database, and then locally select *random* relays for anonymous communication using onion routing. This database is periodically downloaded every three hours to handle relay churn.

There are several problems with Tor's architecture. First, the requirement for all users to maintain global information about all online relays becomes a scalability bottleneck. As the number of servers increases, maintaining a global view of the system becomes costly, since churn will cause frequent updates and a large bandwidth overhead. In fact, McLachlan et al. [25] showed that under reasonable growth projections, the Tor network could be spending more bandwidth to maintain this global system view than for the core task of relaying anonymous communications. Second, the trusted directory authorities are attractive targets for attack; in fact, some directory authorities were recently found to have been compromised [26].

We note that our recent proposal for PIR-Tor [27] might address the networking scalability issues, but it still does not mitigate the basic trust and denial of service issues in a centralized approach. In order to address these problems, we argue that a peer-to-peer architecture will be necessary.

2.1.1 Threat model

We consider a partial adversary who controls a fraction f of all the nodes in the network. This set of malicious nodes colludes and can launch both passive and active attacks. In terms of the standard terminology introduced

by Raymond [28], our adversary is internal, active, and static. We note that the adversary may also launch Sybil attacks [8] (inserting multiple identities in the system) to attain an f arbitrarily close to 1, and thus any secure solution would need to mitigate the Sybil attack.

2.1.2 Attacks

Low-latency anonymous communication systems are not designed to to be secure against a global passive adversary. In particular, an adversary who can observe the entry and exit of a circuit can use end-to-end timing analysis [29–32] to break user anonymity. Thus. traditional security analyses of Tor [3] assume that a user who controls (or observes) a fraction f of the network can compromise the anonymity of f^2 of all circuits by end-to-end timing analysis (by observing the entry and exit point of a stream). Note that due to bandwidth-weighted relay selection, f is best thought of as the fraction of Tor bandwidth controlled or observed by an adversary.

There is a thread of research that deals with degradation of anonymity over a period of time. Reiter and Rubin [33] proposed the predecessor attack, which was later extended by Wright et al. [34–37]. 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 the initiator is more likely to be the predecessor than any other node in the network.

In similar spirit to the predecessor attack, Berthold et al. [38] and Raymond [28] propose 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. [39] use intersection to find recipients of a given users message. A statistical version of this attack was proposed by Danezis [40] and later extended by Mathewson and Dingledine [41]. These attacks typically require an adversary to observe a significant fraction of the network.

Conventional anonymity analysis often abstracts away many important properties of anonymity systems. Recent research has shown that, when these properties are properly considered, the potential for anonymity compromise is significantly greater than predicted by conventional models. Ex-

amples of such properties include congestion and interference [42–46], clock skew [47, 48], heterogeneous path latency [43, 49, 50], the topology of the underlying Internet paths used to forward traffic between relays [51–53], and the reliability of relays [54]. Borisov et al. [54] proposed a selective-DoS attack on anonymous communication and showed that attackers could selectively affect the reliability of the system in states that are hardest to compromise. The selective-DoS attack affects peer-to-peer anonymous communication the most, because of the added complexity of knowing only a subset of the nodes in the network.

2.2 Peer-to-Peer Anonymous Communication

Several designs for decentralized peer-to-peer anonymous communication have been proposed. The key challenge in these systems is to securely locate random relays. Based on the mechanism used to locate random relays, we can broadly classify these systems into two categories.

2.2.1 Random paths in unstructured topologies

The first approach to scale anonymous communication is to connect relays into a restricted (non-clique) topology and construct circuits along paths in this topology. For example, in Tarzan [55], each node has a small set of *mimics*, and all circuits must be created on links between mimics. The use of a restricted topology has the advantage that the local view at each hop is sufficient to extend the circuit.[1]

Though communication in Tarzan is carried out over links between mimics, to be able to verify that paths are constructed correctly (to prevent route capture attacks), each node needs to maintain a global view of the system, updated using a gossip protocol. This limits Tarzan to networks of about 10 000 or fewer nodes. MorphMix [56] was designed to eliminate such scaling constraints by creating a randomized, unstructured overlay between relays, with circuits built on paths along the overlay. MorphMix faced a similar

[1] They also provide an opportunity for cover traffic to be sent along all the links in the restricted topology, something that would be infeasible for the full clique topology even of the current size of Tor, let alone much larger P2P networks of the future.

challenge in needing to trust a node to correctly specify its neighbors when extending a circuit. Instead of maintaining a global view, MorphMix designed a mechanism involving witness nodes and a collusion detection mechanism 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 [57].

Nagaraja [58] and the recent Drac [59] system describe a compelling vision for P2P anonymity by leveraging such random walks over unstructured social network topologies. However, neither of these designs provide any mechanisms to verify the authenticity of neighbor information returned by intermediate nodes in the system, and are thus vulnerable to malicious insider attacks. Securely leveraging random walks for anonymous communication is an open question, which we solve in this thesis in Chapter 4 and Chapter 7.

2.2.2 Random lookups in structured topologies

Structured peer-to-peer topologies, such as Chord [60] or Pastry [61] (also known as distributed hash tables, or DHTs), have been used as a foundation for peer-to-peer anonymous communication. Each node in a structured peer-to-peer topology is assigned a collection of neighbors, also known as fingers. Finger relationships are assigned using a mathematical formula based on node identifiers. A node maintains a routing table, which consists of the IP addresses and the public keys of its fingers. DHTs provide a lookup operation which takes an identifier as an input, and returns the node closest to the input identifier that is online. The main idea behind using DHTs is that lookup for a random identifier approximately corresponds to a random node in the system.

However, by default, DHTs are extremely vulnerable to active attacks on the lookup mechanism [62, 63]. Attackers can intercept lookup requests and return incorrect results by listing a colluding malicious node as the closest node to a key. For this reason, P2P anonymity systems built of top of DHTs must include mechanisms for secure routing (lookups). We now discuss the state of art in P2P anonymous communication.

The design of Salsa [9] is similar to Tor, in that a circuit is built by selecting three random nodes in the network and constructing a circuit through them.

For scalability reasons, Salsa does not maintain a global view; instead, it uses a specially designed secure lookup operation over a specially designed distributed hash table (DHT) to locate forwarder nodes. The secure lookups use redundant checks to mitigate potential attacks; these checks are able to limit the bias an adversary can introduce in the lookup.

AP3 [64] has a similar structure where paths are built by selecting random relays using a secure lookup mechanism [65]. The design of AP3 is more similar to Crowds [33] 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 in the path; however, for low-latency communication, timing attacks may make this decision simpler.

In Chapter 3, we present information-leak attacks that break the security guarantees of Salsa and AP3. Our key idea is that DHT lookups are not anonymous, and information revealed from DHT lookups can be used to break user anonymity. Danezis and Clayton [10] studied attacks on peer discovery and route setup in anonymous peer-to-peer networks. They show 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 [11] extend 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. These attacks are similar in spirit to the ones we propose, but rather than absolute knowledge of the initiator's routing state, we use probabilistic inferences based on observed lookups. Upon publication of our information-leak attacks, several new designs were proposed, that were aimed at specifically mitigating our attacks.

Panchenko et al. proposed NISAN [66] in which information-leak attacks are mitigated by a secure iterative lookup operation with built-in anonymity. The secure lookup operation uses redundancy to mitigate active attacks, but hides the identity of the lookup destination from the intermediate nodes by downloading the entire routing table of the intermediate nodes and processing the lookup operation locally. However, in followup work to our information leak attacks [67], we were able to drastically reduce the lookup anonymity by taking into account the structure of the topology and the deterministic nature of the paths traversed by the lookup mechanism.

Torsk, introduced by McLachlan et al. [25], uses secret *buddy nodes* to mitigate information leak attacks. Instead of performing a lookup operation themselves, nodes can instruct their secret buddy nodes to perform the lookup on their behalf. Thus, even if the lookup process is not anonymous, the adversary will not be able to link the node with the lookup destination (since the relationship between a node and its buddy is a secret). However, our aforementioned work [67] also showed some vulnerabilities in the mechanism for obtaining secret buddy nodes.

There have been some attempts to add anonymity to a lookup. Borisov [68] proposed an anonymous DHT based on Koorde [69], which performs a randomized routing phase before an actual lookup. Ciaccio [70] proposed the use of imprecise routing in DHTs to improve sender anonymity. These lookups were designed to be anonymous, but not secure: an active adversary could easily subvert the path of the lookup. As such, neither lookup mechanism can be used to build anonymous circuits.

Thus we can see that all current approaches to P2P anonymous communication that leverage DHTs are vulnerable to attacks, motivating the need for new approaches to P2P anonymous communication. Our ShadowWalker protocol proposed in Chapter 4 solves this open problem; ShadowWalker can be used to perform a randomized routing phase before an actual lookup (as proposed by Borisov [68]), to yield a lookup mechanism that is both secure and anonymous.

2.3 Sybil Defenses and Social Networks

In a Sybil attack [8], a single malicious entity emulates the behavior of multiple identities to compromise system security. Any attempt to build fault tolerance in the system is then doomed, since Sybil identities can obtain an arbitrary fraction of all identities in the system.

Sybil defenses must fundamentally impose a cost on participation in the network [8]. One approach, advocated by Castro et al. [65], requires users to provide identity credentials and/or payment to a centralized authority, who then issues certificates allowing users to participate. This authority, of course, becomes a central point of trust. Decentralized approaches instead allow nodes to directly verify some resource expenditure by other nodes,

such as CPU computation, or the possession of a unique IP address [71, 72]. All these solutions face a tradeoff between creating too high a barrier for participation by honest users and making it too easy for malicious users to create Sybil identities. More recent work has recognized that it is expensive for a malicious adversary to establish trust relationships with honest users and thus social network topologies can be used to detect and mitigate social Sybil attacks. The focus of this dissertation is on Sybil defense mechanisms based on social network trust relationships.

2.3.1 Generic Sybil defense

SybilGuard [13] and SybilLimit [73] are decentralized systems for Sybil defense. The main insight in these systems is the use of special random walks called *random routes* for Sybil defense. For example in SybilLimit, the authors show that as long as the number of attack edges is less than a threshold ($g = o\left(\frac{n}{\log n}\right)$, where n is the number of honest nodes), then with high probability, a short random walk of $O(\log n)$ steps is likely to stay within the set of honest nodes. Nodes in SybilLimit perform $\sqrt{e}$ short random walks (where e is the number of edges amongst the honest nodes) and keep track of their last edges (tails). By the birthday paradox, two honest nodes will share a common tail with high probability. Each node allows only a certain number of random routes to traverse it, thereby limiting the number of Sybil identities that are validated by the honest nodes. We note that SybilGuard suffers from high false negatives, while the guarantees of SybilLimit are also not optimal. In Chapter 5 we propose the SybilInfer protocol that reduces the number of accepted Sybil identities by an order of magnitude. Finally, we note that all of these systems are stand-alone Sybil defenses and do not provide support for secure peer-to-peer routing.

2.3.2 Sybil-resilient routing

Sybil-resistant DHT routing [74] uses the concept of a *bootstrap graph* (also called introduction graph) to defend against Sybil attacks. A bootstrap graph describes which node introduced which other nodes to the network. The protocol is based on the insight that the adversary will be connected to

the graph at a limited number of points which can be turned into trust bottlenecks. The lookup protocol is based on iterative routing and attempts to ensure that a diverse set of nodes are used for the lookup, which minimize the reliance on a localized set of malicious nodes.

Whanau [1] is the state-of-art Sybil-resilient DHT routing system that leverages the short random walk primitive from SybilLimit to architect a secure DHT where nodes can communicate with only one intermediate hop. Each node performs $\sqrt{e}$ random walks to sample nodes for construction of their routing tables; the Sybil resistant property of short random walks ensures that a high fraction of the sampled nodes are honest. By querying routing table entries, nodes can construct their successor lists. While Whanau does provide a secure routing primitive, it does so at the cost of maintaining $\sqrt{e} \log n$ state at each node. The large state requirement means that the system has difficulty maintaining accurate state in face of social network churn and node churn. The proposed solution outlined by the authors envisions that nodes perform random walks once per day—which would mean that on an average, new nodes would face about a twelve hour delay before being validated by the system, presenting a significant usability problem. Whanau also requires the social graph to be public, presenting significant privacy concerns. In contrast, our proposal X-Vine (described in Chapter 6) builds upon *network-layer* DHTs, embedding the DHT directly into the social network fabric. This enables X-Vine to provide good security while achieving improved scalability and privacy of social relationships.

2.3.3 Other systems

The concept of a bottleneck cut between a fast-mixing social network and Sybil nodes has been used in a number of other systems, such as SumUp [75], a protocol for online content rating that is resilient to Sybil attacks; Ostra [76], a system to prevent unwanted communication from nodes; and Kaleidoscope [77], a system for censorship resistance.

There is a class of systems that augments traditional peer-to-peer networks with social connections. Sprout [78] proposed augmenting the finger tables in traditional DHTs, such as Chord, with social network links. The authors showed that the added connections could improve the security of the routing

mechanism. However, Sprout does not defend against Sybil attacks, and is not concerned with user privacy. OneSwarm [79] is a deployed peer-to-peer communication system for improving user privacy where routing is performed by combining trusted and untrusted peer relationships. Tribler [80] increases download speed in BitTorrent by discovering and downloading file chunks stored at peers. Similarly, Maze [81] leverages a social network to discover peers and cooperatively download files. These three systems leverage flooding to provide any-to-any reachability, and thus cannot scale to large networks. These hybrid systems are not resilient to Sybil attacks.

Finally, we discuss a class of systems where all communication is over social links. This enables participants in the network to be hidden from each other, providing a high degree of privacy. Such a network is commonly known as a *darknet*. WASTE [82] is a deployed decentralized chat, instant messaging, and file sharing protocol, and is widely considered to be the first darknet. WASTE does not attempt to scale beyond small networks, and its suggested size is limited to 50 users. Turtle [83] is a deployed decentralized anonymous peer-to-peer communication protocol. Nodes in Turtle do not maintain any state information other than their trusted friend links and use controlled flooding to search for data items. Flooding methods create significant overhead as network size increases. Freenet [20] is a deployed decentralized censorship-resistant distributed storage system. Version 0.7 of Freenet nodes can be configured to run in darknet or opennet mode; the latter allows connections from untrusted nodes, and is expected to be used by less privacy-sensitive users. Freenet's routing algorithm is heuristic and does not guarantee that data will be found at all; it has also been shown to be extremely vulnerable even against a few malicious nodes [84]. Membership concealing overlay networks (MCONs) (formalized by Vasserman et al. [85]), hide the real-world identities of the participants through the use of overlay and DHT-based routing. However, their design makes use of a trusted centralized server and also requires flooding when a new user joins the network. In addition to these limitations, none of the above systems are resilient to Sybil attacks.

CHAPTER 3

INFORMATION LEAKS IN STRUCTURED PEER-TO-PEER ANONYMOUS COMMUNICATION SYSTEMS

The state of art in anonymous communication systems leverage DHT lookups to find random relays for anonymous communication. Such a lookup, however, can be subject to attack: malicious nodes can misdirect it to find relays that are colluding and violate the anonymity of the entire system. All of the P2P anonymous communication designs therefore incorporate some defense against such attacks; e.g. AP3 [64] uses secure routing techniques developed by Castro et al. [65], and Salsa uses redundant routing with bounds checks [9].

These defenses, however, come at a cost. They operate by performing extra checks to detect incorrect results returned by malicious nodes. These checks cause many messages to be exchanged between nodes in the network, some of which might be observed by attackers. As a result, a relatively small fraction of attackers can make observations about a large fraction of lookups that occur in the P2P network, acting as a near-global passive adversary. As most modern anonymity systems assume that a global passive adversary is too costly, they are not designed to resist such attacks. Therefore, this small fraction of attackers can successfully attack anonymity of the system.

We examine this problem through a case study of the Salsa anonymous communication system. Defenses against active attacks create new opportunities for passive attacks. Salsa makes heavy use of redundancy to address active attacks, rendering it vulnerable to passive information leak attacks. Further, increasing the levels of redundancy will improve passive attack performance, and often make the system weaker overall. We find that even in the best case, Salsa is much less secure than previously considered. Salsa was designed to tolerate up to 20% of compromised nodes; however, our analysis shows that in this case, over one quarter of all circuits will be compromised by using information leaks.

We studied potential improvements to Salsa that can be achieved by in-

creasing the path length or introducing a public key infrastructure (PKI). We found that these tools offer only a limited defense against our attacks, and the system is still not secure for practical purposes. Our results demonstrate that information leaks are an important part of anonymity analysis of a system and that new advances in the state of the art of P2P anonymous communication are needed.

The rest of the chapter is organized as follows. We discuss information leaks from lookups in Section 3.1 and show the tradeoff between security and anonymity. In Section 3.2 we present attacks based on information leaks from lookups on Salsa. We summarize in Section 3.3.

3.1 Information Leaks via Secure Lookups

It has been recognized that unprotected DHTs are extremely vulnerable to attacks on the lookup mechanism. First of all, malicious nodes can perform a Sybil attack [8] and join the network many times, increasing the fraction f. Second, they can intercept lookup requests and return incorrect results by listing a colluding malicious node as the closest node to a key, increasing the fraction of lookups that return malicious nodes. Finally, they can interfere with the routing table maintenance and cause the routing tables of honest nodes to contain a larger fraction of malicious nodes; this will increase the chance that a lookup can be intercepted and the result can be subverted.

3.1.1 Castro et al.'s secure lookup

Castro et al. [65] designed a suite of mechanisms to counter these attacks. We discuss their mechanisms in context of Pastry [61], a structured peer-to-peer overlay network, though they are applicable to other DHTs. They proposed:

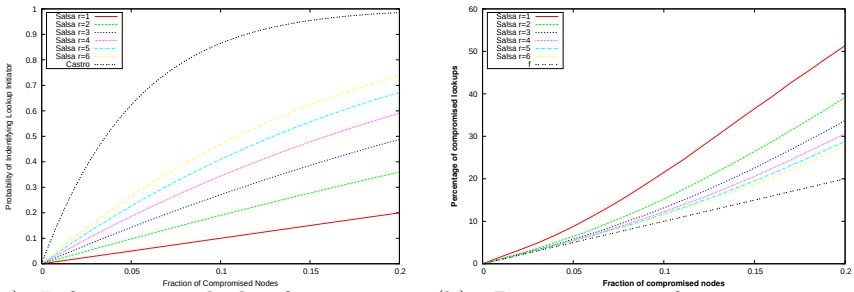
- *Secure node identifier assignment:* Each node is issued a certificate by a trusted authority, which binds the node identifier with a public key. The authority limits the number of certificates and prevents Sybil attacks.

- *Secure routing table maintenance:* Even with secure node ID assignment, attackers can maliciously influence routing table construction.

The Pastry routing algorithms allow flexibility in selecting a neighbor for each slot, which is used for optimizing latency or other metrics. Attackers can exploit this flexibility by suggesting malicious choices for these slots. Secure routing table maintenance eliminates this flexibility by creating a parallel, constrained routing table where each slot can have only a single possible node, as verified by secure lookup. This solution ensures that, on average, only a fraction f of a node's neighbors will be malicious.

- *Secure lookups (secure message forwarding):* For secure lookups, a two-phase approach is employed. The message is routed via the normal routing table (optimized for latency) and a routing failure test is applied. If the test detects a failure, redundant routing is used and all messages are forwarded according to the constrained routing table. The failure test makes use of the observation that the density of honest nodes is greater than the density of malicious nodes. The idea behind redundant routing is to ensure that multiple copies of messages are sent to the key root via diverse routes. Note that Castro et al. consider the problem of securely routing to the entire replica set, for which a neighbor anycast mechanism is also used. We refer the reader to [65] for a detailed explanation of the techniques.

Used together, these techniques are quite effective at ensuring that a lookup returns the actual closest node to the randomly chosen identifier, which in turn suggests that it is malicious with probability f. However, the secure lookup mechanism generates many extra messages: the routing failure tests involves contacting the entire root set of a node (L immediate neighbors in the node ID space), and redundant routing sends a request across several paths. These messages let attackers detect when a lookup has been performed between two honest nodes with high probability. The probability of detecting the lookup initiator can be approximated as $1-(1-f)^{L+\log_{2^b} N}$, which is quite high for the typical values of $L = 16$ and $b = 4$. In Figure 3.1(a), we plot the probability of detection of the lookup initiator as a function of the fraction of compromised nodes f. We can see that a small fraction of 5% compromised nodes can detect the lookup initiator more than 60% of the time. Moreover, when the fraction of compromised nodes is about 10%, the lookup initiator is revealed 90% of the time.



(a) Information leak from secure lookups

(b) Percentage of compromised lookups

Figure 3.1: Salsa lookup mechanism.

This shows the fundamental tension that is encountered by a DHT lookup. The default Pastry mechanisms provide little defense against active adversaries who try to disrupt the lookup process, dramatically increasing the probability that a lookup returns a compromised node. Castro et al.'s mechanisms solve this problem, but introduce another, as the lookup is no longer anonymous and can be observed by malicious nodes. A relatively small fraction of malicious nodes can, therefore, act as a near-global passive adversary and compromise the security of anonymous communication systems. The secure lookup exposes nodes to increased surveillance; we note that this may have consequences for protocols other than anonymous communication that are built on top of secure lookup.

3.1.2 Salsa secure lookup

Salsa [9] is based on a custom-built DHT that maps nodes to a point in an ID space corresponding to the hash of their IP address. The ID space in Salsa is divided into groups, organized into a binary tree structure. Each node knows all the nodes in its group (local contacts), and a small number of nodes nodes in other groups (global contacts).

Similar to Pastry, nodes must rely on other nodes to perform a recursive lookup. A malicious node who intercepts the request could return the identity of a collaborating attacker node. Salsa makes use of redundant routing and bounds checks to reduce the lookup bias. The Salsa architecture is designed to ensure that redundant paths have very few common nodes between them (unlike Pastry or Chord [60]). This reduces the likelihood that a few nodes

will be able to modify the results for all the redundant requests. A lookup initiator asks r local contacts (chosen at random) to perform a lookup for a random key. The returned value that is closest to the key is selected and a bounds check is performed. If the distance between the prospective owner and the key is greater than a threshold distance b, it is rejected, reasoning once again that malicious nodes are less dense than honest ones and thus will fail the bounds check much more frequently. If the bounds check test fails, the result of the lookup is discarded and another lookup for a new random key is performed. Redundant routing and the bounds check work together: an attacker would need to both intercept all of the redundant lookups and have a malicious node that is close enough to avoid the bounds check.

Salsa is resistant to conventional attacks that target the lookup mechanism as long as the fraction of malicious nodes in the system is less that 20%. Since Salsa does not provide adequate security for higher values of f, we shall limit our analysis to low values.

In Figure 3.1(b), we study the effect of varying redundancy on the lookup bias. To compute our results, we used a simulator developed by the authors of Salsa [86].[1] The simulator was configured to simulate 1000 topologies, and in each topology, results were averaged over 1000 random lookups. The lookup bias is sensitive to the average lookup path length, which in turn is about $\log_2 |G|$, where $|G|$ is the number of groups. This is because longer path lengths give attackers more opportunities to intercept the lookup and subvert the result. We therefore used 128 groups, which would be a typical number in a large network, and 1000 nodes in our simulation. We can see that increasing r clearly reduces the fraction of compromised lookups, thus increasing security. For $f = 0.2$, the fraction of compromised lookups drops from 39% to 27% when r is increased from 2 to 6.

The initiator of a lookup can be identified by the attackers if any of the local contacts used for redundant lookups are compromised. The probability of detecting the lookup initiator is $1 - (1 - f)^r$, as depicted in Figure 3.1(a). Clearly, increasing r increases the chance that a lookup initiator is detected. This illustrates the tradeoff between security and anonymity of a lookup.

In this section, we observed that secure lookups leak information about the lookup initiator. Furthermore, we observed a tradeoff between the security

[1] Our results differ slightly from those shown in [9] because of a bug in the simulator. We have communicated the bug to the authors and it has been accepted.

and anonymity of a lookup. A relatively small fraction of compromised nodes are able to observe a large fraction of lookups. Next, we shall use this to break the anonymity of AP3 and Salsa.

3.2 Salsa

We shall now analyze Salsa's path building mechanism. For anonymous communication, a path is built between the initiator and the recipient via proxy routers (nodes). Layered encryption ensures that each node knows only its previous and next hop in the path. The nodes used for the paths are randomly selected from the global pool of nodes, even though each node has only local knowledge of a small subset of the network.

3.2.1 Salsa path building

To build a circuit, the initiator chooses r random IDs ([9] sets $r = 3$) and redundantly looks up the corresponding nodes (called the first set/stage of nodes). Keys are established with each of these nodes. Each of the first set of nodes does a single lookup for r additional nodes (second set of nodes). A circuit is built to each of the nodes in the second group, relayed through one of the nodes in the first group. Again, the initiator instructs the second set of nodes (via the circuits) to do a lookup for a final node. One of the paths created between the first and the second set of nodes is selected and the final node is added to the circuit. We use the parameter l to refer to the number of stages in the circuit ([9] sets $l = 3$). Figure 3.3(a) depicts the Salsa path building mechanism for $r = 3$ and $l = 3$. Note that redundant lookups are used only to look up the nodes in the first stage; later lookups rely on the redundancy in the path building mechanism itself.

3.2.2 Active path compromise attacks on Salsa

Active attacks on the lookup mechanism can bias the probability that nodes involved in Salsa's path building mechanism are compromised. Borisov et al. [54] noted that Salsa path building is also subject to a public key mod-

ification attack.[2] If all the nodes in a particular stage are compromised, they can modify the public keys of the next set of nodes being looked up. This attack defeats Salsa's bounds check algorithm that ensures the IP address is within the right range, since it cannot detect an incorrect public key. Also, since the traffic toward the node whose public key has been modified is forwarded via corrupt nodes, the attackers are guaranteed to intercept the messages. They can then complete the path building process by emulating all remaining stages (and hence, the last node). The public key modification attack and attacks on Salsa lookup mechanism are active attacks. Now, by end-to-end timing analysis, the path will be compromised if the first and last nodes in the circuit are compromised. Conventional analysis of anonymous communication typically focuses on minimizing the chance of path compromise attacks. By increasing the redundancy in the path building mechanism, this chance can be minimized. This is because increasing r decreases the chance of both active attacks on lookups as well as public key modification attacks.

We now describe three types of passive information leak attacks on Salsa. We shall also show that increasing redundancy increases the effectiveness of the information leak attacks, resulting in a tradeoff between robustness against active attacks and passive information leak attacks.

3.2.3 Conventional continuous stage attack

A path in Salsa can be compromised if there is at least one attacker node in every stage of the path. Suppose that there are attacker nodes A_1, A_2, A_3 in the three stages respectively. In the path building mechanism, a node performs a lookup for all r nodes in the following stage implying that A_1 would have looked up A_2 and A_2 would have looked up A_3. Hence the attacker can easily (passively) bridge the first and last stages, thereby compromising the anonymity of the system. This attack was mentioned in [9]. Note that if we increase redundancy as per conventional analysis, the effectiveness of the continuous stage attack also increases. This is because increasing redundancy increases the chance that attackers are present in each stage (which is $1 - (1 - f)^r$), giving them more opportunities to launch this attack. Next, we

[2]Their analysis did not take into account the lookup bias.

shall describe two new bridging attacks also based on information leaks from lookups.

3.2.4 Bridging an honest first stage

This attack is based on the observation that the initiator performs redundant lookups for the nodes in the first stage. If the adversary can deduce the identities of the nodes in the first stage (they need not be compromised), and detect any of initiator's redundant lookups for nodes in the first stage, the anonymity of the system is compromised. Consider the Figure 3.3(a); malicious nodes are depicted in black. The first stage (A_1, B_1, C_1) is comprised solely of honest nodes, the second stage (A_2, B_2, C_2) has all malicious nodes and the third stage node A_3 is also compromised. The attackers know the identities of A_1, B_1, C_1 because of key establishment with them. Now if they detect a node performing a lookup for either A_1, B_1, or C_1, they can identify that node as the initiator. Since the initiator performs 9 lookups for the first stage nodes, the probability of detecting this initiator is $1 - (1 - f)^9$, which translates into a probability of 0.87 for $f = 0.2$. A similar attack strategy is applicable when only 2 or even one node in the second stage is compromised. In the latter scenario, the second stage knows the identity of only a single node in the first stage, and if the initiator is detected looking up that node, then the path is compromised. This occurs with probability $1 - (1 - f)^3$, which is 0.49 for $f = 0.2$. Similar to the continuous stage attack, notice that an increase in r increases the probability that attackers can detect a lookup by the initiator for the first node.

It is important to note that there are some false positives in the attack. The false positives occur when a node (say A_1) in the first stage is involved in building more than one path. In such a scenario, more than one node will look up A_1, and the attackers may detect a lookup for A_1 not done by the actual initiator. Using the variable x to model the amount of lookup traffic by other nodes, we can compute the false positives as:

$$1 - \left(\frac{N-1}{N}\right)^{x(1-(1-f)^r)}.$$

Figure 3.2 depicts the false positives for varying r using $f = 0.2, N = 1000$.

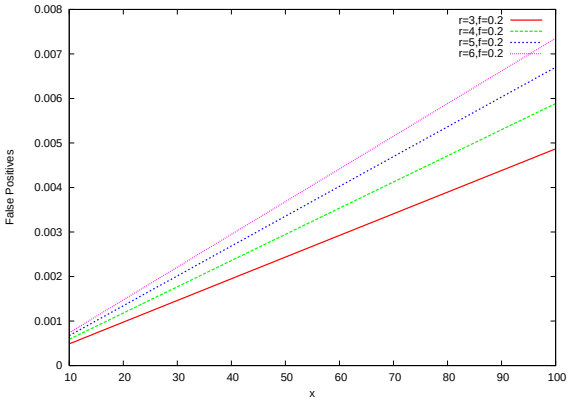
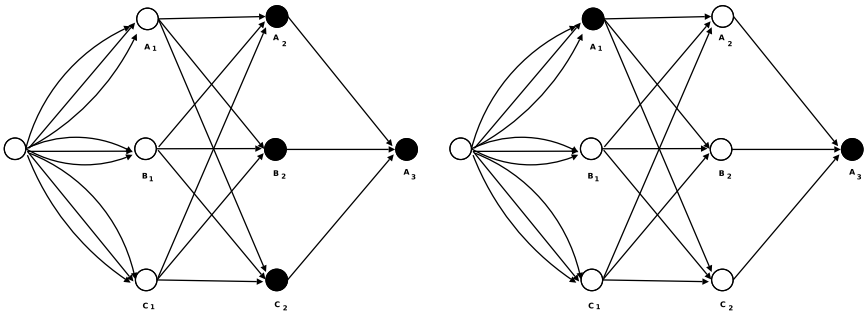


Figure 3.2: False positives in bridging an honest first stage.

Note that for $x < \frac{N}{100}$, the false positives are less than 0.1%.

3.2.5 Bridging an honest stage

Salsa is also vulnerable to a bridging attack where attacker nodes separated by a stage with all honest nodes are able to deduce that they are on the same path. Consider the arrangement of nodes depicted in Figure 3.3(b). The first stage has one malicious node A_1, the second stage consists solely of honest nodes, and the last node A_3 is compromised. A_1 knows the identities of all three nodes in the second stage, as it has performed a lookup for them. Also, as part of the path building mechanism, one of the nodes in the second stage will establish a key with the compromised third stage node A_3. In such a scenario, A_1 and A_3 can deduce that they are part of the same path as they both observe a common honest node. Similarly, if any of the nodes in the first stage are compromised and the last node is compromised, the path is compromised. In such an attack the compromised nodes in the first stage need not be selected as relays. Again, recall that increasing r increases the chance of an attacker being present in a stage, resulting in a higher probability of bridging an honest stage. The probability of false positives in this scenario can be analyzed as $1 - (\frac{N-1}{N})^x$, which for $x = N/100$ and $N = 1000$ is less than 1%.



(a) Bridging an honest first stage (b) Bridging an honest stage

Figure 3.3: Information leak attacks on Salsa.

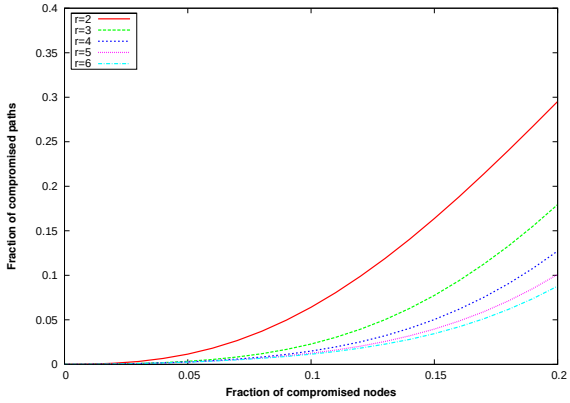


Figure 3.4: Conventional path compromise attacks: Increasing redundancy counters active attacks.

3.2.6 Results

We now present experimental results for active path compromise attacks and information leak attacks on Salsa. Our results have been computed by modeling the Salsa path building mechanism as a stochastic activity network in the Möbius framework [87]. For a fixed f and r, the input to the model is the lookup bias, which was computed using the Salsa simulator [86], with simulation parameters $N = 1000, |G| = 128$.

Figure 3.4 shows the chance of active path compromise attacks on Salsa for varying levels of redundancy. It is easy to see that increasing r reduces the fraction of compromised paths. For instance, at $f = 0.2$, 17% paths are compromised using $r = 3$. The corresponding value for $r = 6$ is approximately 8%. This is not surprising, as increasing r reduces the chance of both active attacks on lookups and attacks involving public key modification.

The continuous stage attack and both our bridging attacks are examples of

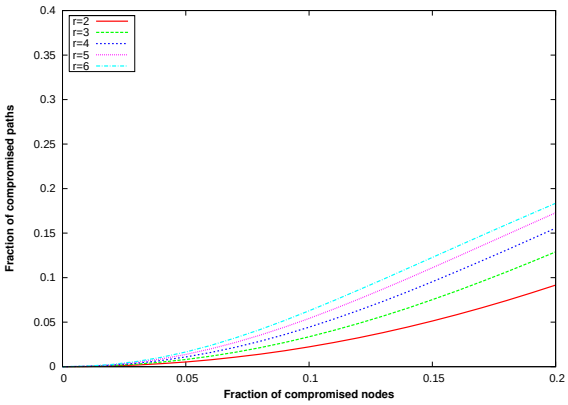


Figure 3.5: Information leak attacks: Increasing redundancy makes the passive adversary stronger.

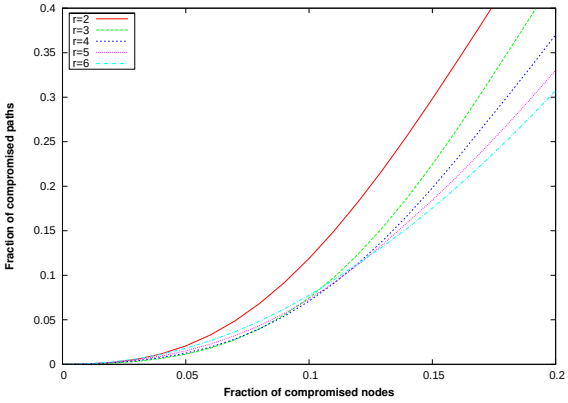


Figure 3.6: All conventional and information leak attacks: For maximal anonymity, $r = 3$ is optimal for small f. Note that there is a crossover point at $f = 0.1$ when $r = 6$ becomes optimal.

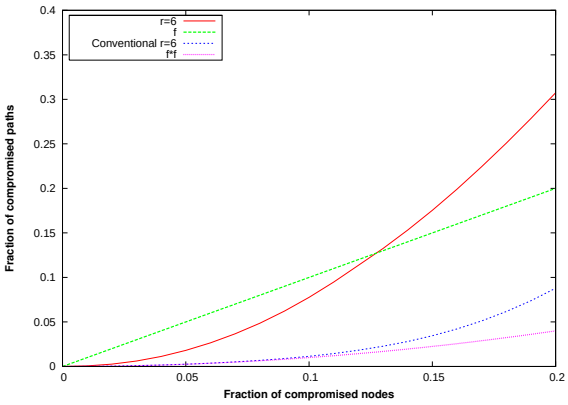


Figure 3.7: Comparison of all attacks with conventional active attacks: Note that for $f > 0.12$, fraction of compromised paths is greater than f.

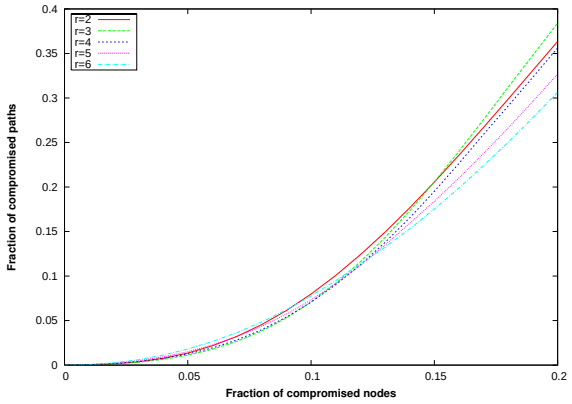


Figure 3.8: Salsa with a PKI—All conventional and information leak attacks. Even with a PKI, the security of Salsa is much worse as compared to conventional analysis.

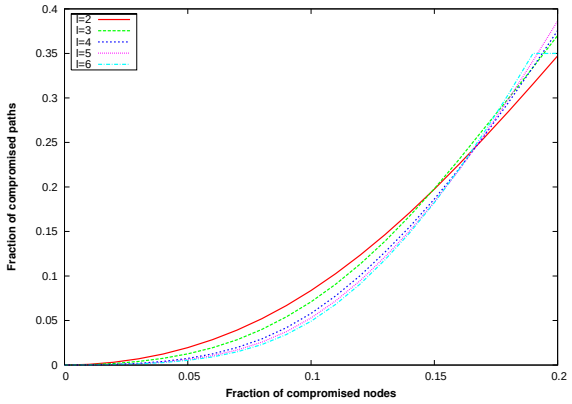


Figure 3.9: Effect of varying the path length: Note that there is only limited benefit of increasing path length.

passive attacks. Figure 3.5 shows the fraction of compromised paths under the passive attacks. We can see that an increase in r increases the effectiveness of the passive attacks, and is detrimental to anonymity. For 20% attackers, even for a small value of $r = 3$, the initiator can be identified with probability 0.125. Higher values of r can increase the probability of identifying the initiator to over 0.15. Note also that the bridging attack significantly improves upon the previous attacks on Salsa: using only the continuous stage attack, for $r = 3, f = 0.2$, anonymity is broken with a probability of only 0.048, less than half of what is possible with bridging.

The active path compromise attacks can be combined with passive information leak attacks. Figure 3.6 shows the fraction of compromised paths for all passive and active attacks. An interesting trend is observed where increasing redundancy (beyond $r = 2$) is detrimental to security for small values of f. This is in sharp contrast to conventional analysis; the inclusion of information leak attacks have made the effect of passive attacks more dominant over the effect of active attacks. There is a crossover point at about 10% malicious nodes, after which increasing r reduces to probability of path compromise. This is because active attacks are dominant for higher values of f. Note that $r = 2$ results in significantly worse security because of poor resilience to both lookup attacks and public key modification attacks.

This shows the tension between the passive and active attacks. There is an inherent redundancy in Salsa path building mechanism to counter active attacks. However, the redundancy makes the passive adversary stronger and provides more opportunities for attack. From Figure 3.7 we can see that by conventional analysis, security provided by Salsa is close to that of Tor (f^2). With our information leak attacks taken into account, for $f > 0.12$, the security provided by Salsa is even worse than f.

3.2.7 Improvements to Salsa

We next consider whether simple changes to Salsa's mechanisms would provide a defense against our attacks. First, we consider Salsa using a PKI, as in AP3. The public key modification attack would no longer work; however, other active attacks on the lookup mechanism and our passive information leak attacks would still apply. Figure 3.8 depicts the probability of identi-

fying the initiator under all active and passive attacks in Salsa with PKI. Again, we can see the tension between active and passive attacks. Increasing redundancy (beyond $r = 2$) is detrimental to security for small values of f, because of the dominance of our information leak attacks. There is a crossover point, after which active attacks become dominant, and increasing r increases security. With the public key modification attack gone, $r = 2$ becomes a more reasonable parameter, but even with a PKI, the fraction of compromised paths increases from 8% under conventional active attacks to more than 30% with our information leak attacks taken into account.

Finally, we explore the effect of increasing the path length (l) on the anonymity of Salsa. Figure 3.9 depicts the probability of identifying the initiator for varying values of l. There is an interesting tradeoff in increasing the path length. On one hand, increasing l reduces the chance of information leak attacks, because the attacker needs to bridge all stages. On the other hand, increasing l gives attackers more opportunities to launch active attacks, thereby increasing the probability that last node is compromised, which in turn gives attackers more observation points. This is basically a cascading effect: the presence of a malicious node in each stage increases the probability of presence of malicious nodes in the next stage. For small values of f, passive attacks are stronger, therefore increasing l increases security, but for higher f, the active attacks and the cascading are dominant, therefore increasing l decreases security.

We have proposed passive bridging attacks on Salsa that are based on information leaks from lookups, and can be launched by a partial adversary. Moreover, we have shown a tradeoff between defenses against active and passive attacks. Even at the optimal point in the tradeoff, the anonymity provided by the system in significantly worse than what was previously thought. This tradeoff is present even in Salsa with a PKI. Moreover, increasing path length in Salsa has only a limited benefit on the user anonymity.

3.3 Summary

We showed that lookup mechanisms in DHTs are not anonymous, and reveal information about the initiator of the lookup message. We showed that such information leaks can be used to break the security guarantees of DHT

lookups based peer-to-peer anonymous communication systems like Salsa.

CHAPTER 4

SHADOWWALKER: PEER-TO-PEER ANONYMOUS COMMUNICATION USING REDUNDANT STRUCTURED TOPOLOGIES

We propose a low-latency peer-to-peer anonymous communication system that is based on a random walk over redundant structured topologies. Our main idea is the creation of *shadow nodes*, which redundantly verify the correctness of a given node's routing table and certify it as correct. Such certificates can then be used to check the steps of a random walk; by using certificates rather than online checks, we can avoid information leak attacks. We show that our design is effectively able to prevent route capture attacks by employing a small number of shadows per node. We also analytically model the effects of a restricted topology on the anonymity of the system and show that, with an appropriate choice of an underlying topology, we can mitigate this effect and achieve strong anonymity. In particular, the anonymity levels achieved by our system are much higher than those of Salsa [9] when 20% of all nodes are compromised.

We present an extension to our system that improves anonymous communication performance at the cost of slightly weakening the anonymity protection. This extension should result in latency and bandwidth constraints similar to those achieved by Tor [3]. It also provides an effective defense against the selective denial-of-service attack on anonymous systems [54]. We also verified our analytic model with the help of simulations. We show that our system has manageable communication and computation overheads, and is able to handle a moderate amount of churn in the network.

The chapter is organized as follows. We propose the ShadowWalker scheme based on a redundant structured topology in Section 4.1 and analytically evaluate the anonymity provided by our scheme in Section 4.2. We describe our experimental results in Section 4.3, and summarize in Section 4.4.

4.1 ShadowWalker

To motivate our design, we first briefly describe a simple random walk–based anonymity protocol and discuss the attacks on it. In a random walk–based protocol, an initiator first sets up a circuit with a random finger A. To further extend the circuit, initiator sends A a random index i, and A returns the public key of the finger B corresponding to the index i (i'th entry in the routing table). The initiator can then extend the circuit via A to B. By iterating these steps, a circuit of arbitrary length can be established. The above protocol is susceptible to the following attacks:

Route Capture: An intermediate node A in a circuit may lie when asked about its finger B and return an incorrect public key. Since traffic for B will be forwarded through A, A can give its own public key and then pretend to be B. Further, it can perform the same attack in the subsequent steps, emulating the rest of the hops.

Restricted Topology: The terminus of the random walk in restricted topologies reveals some information about the initiator of the random walk [68,88]. This is because only a subset of the nodes in the network can reach the terminus in a given number of hops. For instance, suppose that the first hop in a two-hop random walk is not compromised, but the second hop is compromised. In this scenario, although the initiator cannot be directly identified, the attacker can be certain that the initiator lies in the set of nodes which have the first hop as fingers. Because of route capture attacks, the random walk can be thought to terminate after encountering the first malicious node (say A). If the walk has traversed i hops so far, then the initiator of the random walk must be within the set of nodes that can reach the previous hop of node A in $i-1$ hops. For fixed-length random walks, the number i can be determined by emulating the rest of the random walk; for randomized-length walks, timing analysis would need to be used to guess i.

4.1.1 Overview

We now describe our ShadowWalker protocol for peer-to-peer anonymous communication. Our main idea is the creation of *shadow nodes* that redundantly verify the correctness of a given node's neighbor table and certify it as correct. Such certificates can then be used to check the steps of a

random walk; by using certificates rather than online checks, we can avoid information leak attacks [12]. We first describe the concept of introducing redundancy into the topology itself, which lies at the heart of our solution. Next, we describe two circuit construction protocols for anonymous communication that perform random walks on redundant structure topologies in a secure manner. Finally we present a secure lookup protocol for routing table maintenance and algorithms to handle node churn.

4.1.2 Redundant structured topology

We first define the concept of a *shadow*. Each node A has several shadows, and each shadow is required to independently maintain the neighbor information for A. The shadows will provide this information as a way to verify that A is not attempting to perform a route capture attack. For a redundancy parameter r, the shadow nodes of A are denoted as $A_1, \ldots, A_r$. The shadow relationship is a deterministic, verifiable relationship that is calculated by applying a mathematical formula to the node identifier. As an example, for $r = 2$, the shadows for a node A can be considered to be its successor and its predecessor. For a generic r, the shadows for a node A can be considered to be its $\lfloor \frac{r}{2} \rfloor$ predecessors and $\lceil \frac{r}{2} \rceil$ successors in the DHT.

Using the shadow relationship, we can define a transformation to make any P2P topology into a redundant one:
Property 1: In addition to fingers, a node A maintains secure information about the shadow nodes of the fingers. This means that if $A \rightarrow B$ is an edge in the structured topology, $A \rightarrow B_j$ is also an edge in the redundant structured topology, for $j = 1, \ldots, r$ (r shadows of B).
Property 2: If a node A_j is the shadow of node A, it maintains a copy of the fingers (as well as the shadows of the fingers) of A. In other words, if $A \rightarrow B$ is an edge in the structured topology, then $A_j \rightarrow B$ and $A_j \rightarrow B_k$ are also edges in the redundant structured topology, for $j = 1, \ldots, r$ and $k = 1, \ldots, r$.

Figure 4.1 depicts the transformation of an edge $A \rightarrow B$ into a redundant structured topology with redundancy parameter $r = 2$. Danezis [88] analyzed the use of random paths along a restricted topology for mix networks and proposed the use of topologies with high expansion so that the route length necessary to provide maximal anonymity grows only logarithmically in the

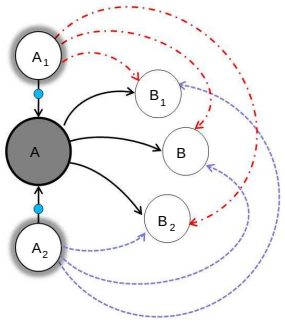


Figure 4.1: Redundant structured topology.

number of nodes in the network. Borisov [68] analyzed random walks on structured P2P topologies and proposed the use of the de Bruijn [89] topology to provide anonymity with small path lengths. We use the de Bruijn topology in our design. Note that nodes must be able to maintain the links in a redundant structured topology securely, as described later in Section 4.1.6.

4.1.3 Circuit construction

We use the shadows of a node A to verify the information reported by A during circuit construction. Note that an initiator I cannot contact the shadows directly, since the shadows would learn that it was building a circuit through A. I could use the circuit it has established with A to communicate with A_j, similar to how MorphMix contacts its witness nodes. But this still lets the node A_j know that a circuit is being built through node A.

We can completely avoid this information leak by having each shadow A_j digitally sign its view of the routing table of A and transmit the signature to A. Since the initiator knows the public key of all the shadows (by Property 1), it can verify the signatures without having to contact the shadows at all. Thus, we are able to redundantly check the information provided by A without contacting any other node. We now describe our secure random walk protocol based on redundant structured topologies. Figure 4.2 shows the pseudocode for our protocol. The initiator I first establishes a circuit to a random finger A. Next, it queries node A for a finger B with random index i (i'th entry in the routing table). A returns the following information to I:

1. IP address and public key of B, and B_k for $k = 1..r$

2. Signatures about the above information from A_j, $j = 1..r$

The initiator I then verifies that signatures of all A_j are correct. Note that since A is a finger of I, A_j are also maintained by I (Property 1). Thus I knows about the public keys of all A_j and can verify the signatures. If the signatures are correct, I can extend the circuit to node B. Now, I can query B for finger C with a random index i', verify it using signatures from B_k and repeat the process. The above example is illustrated in Figure 4.3. If the signatures do not match, the circuit construction is aborted.

$I.circuit\_setup(l)$

Let A be a random finger of I
Let A_j be the shadows of A, $\forall j = 1..r$
Let $Pub(A_j)$ be the public key of A_j, $\forall j = 1..r$
Create circuit between I and A
for $count = 1$ to $l - 1$ **do**
  Let B be a random finger of A with index i
  Let $Pub(B)$ be the public key of B
  /* The random finger is chosen by I*/
  Let B_k be the shadows of B, $\forall k = 1..r$
  Let $Pub(B_k)$ be the public key of B_k, $\forall k = 1..r$
  Let $Signature_j$ be the signature given by A_j for A's routing state.
  I obtains $B, Pub(B),$ all $B_k, Pub(B_k),$ and
  all $Signature_j$ from A via the established circuit.
  if $B, Pub(B)$, and all $B_k, Pub(B_k)$ are verified by all $Signature_j$ **then**
    extend circuit to B
    $A = B$
    $A_j = B_j$, $\forall j = 1..r$
    $Pub(A_j) = Pub(B_j)$, $\forall j = 1..r$
  else
    abort
  end if
end for

Figure 4.2: The pseudocode for circuit establishment of length l.

4.1.4 Using shorter circuits

Relaying an interactive stream over 5 or 6 nodes may be expensive; we propose a modification to our protocol where the initiator uses only the last two hops in the random walk to relay traffic. In essence, we use the random walk

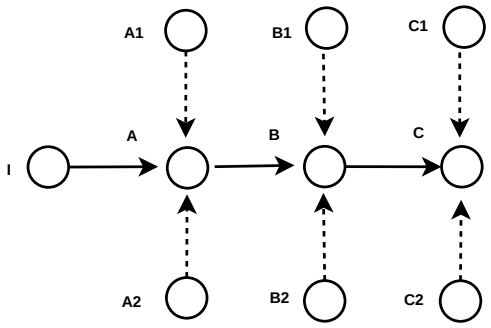


Figure 4.3: Circuit construction.

as an anonymous peer discovery protocol.

Let us consider our modification to the protocol: a node performs a secure l-hop random walk, and then uses the last two hops for anonymous communication, by building a circuit directly to the second to last hop and then extending it to the last hop.[1] Using only the last two hops will improve the system performance as compared to using all l hops for anonymous communication, at the cost of a slight loss of anonymity. Viewed from another perspective, our extension improves anonymity as compared to an 2-hop random walk. In general, if the initiator is interested in building a circuit of length k, it can increase anonymity by performing a l-hop random walk for $l > k$, and then use only the last k hops for anonymous communication. (As long as $l < \log_d N$, since beyond that point, longer random walks provide a limited improvement of anonymity [68]. Here d denotes the average node degree in the topology.)

4.1.5 Using Merkle hash trees

Our circuit construction protocol requires that a node obtains signatures for its routing state from its shadow nodes. We can do this efficiently by creating a Merkle hash tree [90] over the set of fingers and have A_j sign the root of the tree. Then when queried about a finger B, A can send the signature on the root along with $\log_2 d$ hashes to I, proving that B was part of the Merkle hash tree signed by A_j.

[1] If the initiator is unable to connect to the second to last hop because of non-transitive connectivity, the circuit construction is aborted.

4.1.6 Secure lookup

In Chapter 3, we described techniques for secure lookups like Halo [91] and Castro et al. [65], which are effective at ensuring that a lookup returns the actual closest node to a chosen identifier. However, in the context of redundant structured topologies, these mechanisms are not very efficient. For instance, in a redundant structured topology, a node needs to maintain shadows of its fingers. To achieve this, the above lookup protocols need to be invoked multiple times for each shadow node, the overhead for which is significant. We propose a secure lookup protocol that is specifically tailored for redundant structured topologies.

Say a node I wishes to securely lookup an identifier ID. Let A be the closest preceding node for ID in the finger table of I. Following the iterative routing strategy, I will query A for its finger, B, which is the closest preceding node for ID. Since I also knows all of the shadows of A, I can verify this information with them. In this way, I can learn the correct identity of B, as well all of its shadows. It can now proceed iteratively, asking B and its shadows for the closest preceding finger for ID. Note that as long as one node among A and its shadows is honest, I will learn the true identity of B; in case of conflicting answers, I should pick the closest one to ID.[2] Thus, a lookup is successful if there is at least a single honest node in each step of the lookup.

An important consequence of our secure lookup protocol is that along with the node corresponding to the chosen ID, its shadows are returned as well! This significantly reduces the communication overhead of our protocol because it obviates the need for performing multiple secure lookups for the shadows of fingers.

4.1.7 Handling churn

Handling node churn is a major issue in peer-to-peer systems. Existing DHT designs like Chord have developed algorithms that provide robustness guarantees in presence of churn. A *stabilization protocol* is used periodically to ensure that the information about new nodes is propagated to the other

[2]Note that for an anonymous lookup, all nodes must agree for the lookup to proceed. In the non-anonymous case, however, I can verify the existence of B directly, preventing attackers from responding with fake nodes.

nodes in its neighborhood. Periodically, nodes perform a lookup for chosen identifiers to keep their finger tables up to date. A successor list is also maintained to handle the case of node failures. We refer the reader to [60] for a detailed description of how Chord handles churn.

Now, to accommodate a redundant structured topology, the following changes are to be made:

1. A node periodically performs secure lookups to determine the identity of nodes (say the set S) for which it is the shadow.

2. A node periodically performs secure lookups for the fingers of the nodes in the set S.

This above steps suffice to maintain a redundant structured topology because secure lookups return the shadows of the fingers as well. Moreover, for the purpose of anonymous communication, a node also periodically sends signatures to nodes in the set S over their respective routing states.

4.2 Anonymity Evaluation

4.2.1 Anonymity metric

Low-latency anonymity systems are often studied from the point of view of *path compromise* attacks by counting the fraction of compromised circuits. This metric shows whether attackers are able to identify the initiator of a circuit or not. However, in P2P systems, there may be observations that reveal some information about the initiator even when complete identification is impossible. Therefore, rather than using the binary concept of path compromise, we use the entropy-based anonymity metric [92, 93]. This metric considers the *distribution* of potential initiators of a circuit, as computed by attackers, and computes its entropy:

$$H(I) = -\sum_i p_i \log_2 p_i, \tag{4.1}$$

where p_i is the probability that the node i was the initiator of the circuit. Note that a colluding set of attackers can launch a variety of attacks in order to infer the initiator of the circuit. Under some observation o, we can compute the probability distribution given o and compute the corresponding

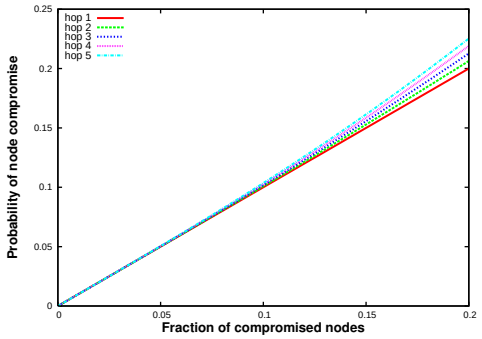


Figure 4.4: $P(k$'th hop is compromised).

entropy $H(I|o)$. To model the entropy of the system as a whole, we compute a weighted average of the entropy for each observation (including the null observation):

$$H(I|O) = \sum_{o} P(o)H(I|o), \tag{4.2}$$

where $P(o)$ is the probability of the observation o occurring, and O is the set of all observations. This is also known as the conditional entropy of I based on observing O.

4.2.2 Circuit construction

Our protocol is subject to the following attacks:

Route Capture Attacks: A single malicious intermediate node cannot launch route capture attacks, because its information is verified by its shadows. However, if an intermediate node and all of its shadows are compromised, they can launch a route capture attack by returning colluding malicious nodes as next hops, or by modifying the public keys of the remaining hops to emulate them. This means that if an intermediate node in the circuit and all of its shadows are are malicious, then the remaining nodes in the circuit are also malicious. Thus the initiator anonymity is compromised if the first node in the circuit and all its shadows are malicious.

End-to-End Timing Analysis: Like other low-latency schemes, ShadowWalker is also vulnerable to end-to-end timing analysis, where malicious nodes on both ends of the circuit can use timing correlations of the packets to infer that they are on the same circuit and compromise the initiator anonymity. If

the first and the last nodes are compromised, the circuit anonymity is broken.
Restricted Topology Attack: In a simple random walk design, the first malicious node is also the terminus of a random walk, due to the route capture attack. However, in our protocol, the random walk may continue past the first malicious node in case one of its shadows is honest. In particular, if the last node in the circuit is honest, the malicious nodes will not learn the destination of the circuit, and as such will gain nothing by learning (or guessing at) the identity of the initiator. However, if the last node is compromised, then the first malicious node in the circuit can perform timing analysis to establish that the two nodes are on the same circuit. It can then assign probabilities to the initiator as before, by considering all nodes that can reach its previous hop within $i-1$ hops. Thus if the last hop is compromised, and the first malicious node is at the i'th position, then it can infer that the initiator lies in the set of nodes who can reach its previous hop in $i-1$ hops. (i will have to be found out by timing analysis between the first malicious node and the last.)

We first study the effect of route capture attacks by modeling the *sampling bias*. We can think of an k-hop random walk as sampling a node that is k hops away from the initiator. If the walk proceeded undisturbed, then the probability that this sampled node would be malicious would be f. However, the route capture attack introduces a bias into this sampling, such that the longer the random walk, the larger the possibility of the route being captured and thus the last node being compromised. We now compute the bias we can expect when sampling nodes using a k-hop random walk. The k'th hop will be definitely malicious if any of the first $k-1$ stages are able to launch a route capture attack. The probability of launching route capture is given by $1-(1-f^{r+1})^{k-1}$. If the attacker is unable to launch the route capture attack in the first $k-1$ hops, then the k'th hop is malicious with probability f. We can now compute the probability that the k'th hop is compromised as follows:

$$\begin{aligned} P(k\text{'th hop is compromised}) &= \left(1-\left(1-f^{r+1}\right)^{k-1}\right)\cdot 1 \\ &\quad + \left(1-f^{r+1}\right)^{k-1}\cdot f. \end{aligned} \tag{4.3}$$

Figure 4.4 shows the probability that the k'th hop is compromised, for

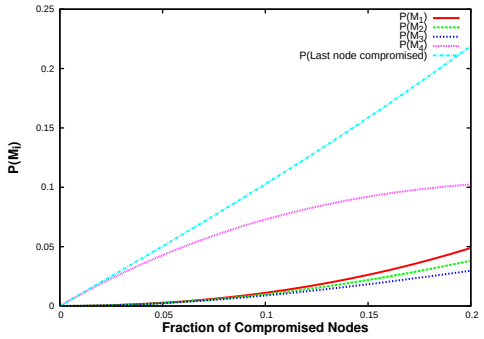


Figure 4.5: $P(M_i)$: Note that the probability of end-to-end timing analysis $P(M_1)$ is less than 5% for $f = 0.2$.

$r = 2$. We can see the cascading effect due to route capture attacks: as the random walk length is extended, the probability that the next hop is compromised becomes higher. Note that there is hardly any sampling bias for $f < 0.1$, and even when $f = 0.2$, the sampling bias is less than 3% for 5 hops. Thus, even at a small redundancy level, our protocol mitigates the route capture attack.

We will now quantify the anonymity that our design provides. Let M_i be the event that the first malicious node on the circuit (say A) is at the i'th position, and the last node is also compromised. Under the event M_i, let the entropy in the choice of the initiator be $H(I|M_i)$. Then, the conditional entropy for the simple random walk protocol can be computed as:

$$H = \sum_{i=1}^{l} P(M_i) H(I|M_i) + \left(1 - \sum_{i=1}^{l} P(M_i)\right) \log_2 N. \tag{4.4}$$

Let us compute $P(M_i)$. The first malicious node is at the i'th position with probability $(1-f)^{i-1}f$. Given this, we now need to compute the probability of the last node being malicious. The last node will be malicious with probability 1 if the attackers are able to launch the circuit capture attack between stages i to $l-1$ (capture). Otherwise (no capture), the last node is malicious with probability with f. $P(\text{no capture})$ is given by $(1-f^r)\cdot(1-f^{r+1})^{l-(i+1)}$. Also, $P(\text{capture}) = 1 - P(\text{no capture})$. Thus, we can express $P(M_i)$ as:

$$
\begin{aligned}
P(M_i) &= f(1-f)^{i-1}\left(P(\text{capture}) + P(\text{no capture})f\right) \\
&= f(1-f)^{i-1}\left(\left(1-(1-f^r)\left(1-f^{r+1}\right)^{l-(i+1)}\right)\right. \\
&\quad \left. + (1-f^r)\left(1-f^{r+1}\right)^{l-(i+1)} f\right). \qquad (4.5)
\end{aligned}
$$

Figure 4.5 shows the values of $P(M_i)$ as a function of f for $l = 4, r = 2$. $P(M_1)$ is the probability of end-to-end timing analysis, and is about 5% for $f = 0.2$. This is close to the current state in Tor, where the probability of end-to-end timing analysis is 4%.[3] Also note that M_l is the dominating event, because unlike other events, it only requires a single node (last node) to be compromised.

We now need to compute $H(I|M_i)$. Note that $H(I|M_i)$ depends on the particular network topology. Though any topology can be used for the random walk, we have considered the de Bruijn [89] topology in this paper because it has optimal mixing properties. In this topology, the expected number of nodes who can reach a particular node in i hops is given by d^i, where d is the average node degree in the topology.[4] We compute $H(I|M_i)$ as follows.

$$
H(I|M_i) = \min(\log_2 d^{i-1}, \log_2 N). \qquad (4.6)
$$

We can now compute the conditional entropy using Equation (4.4). Figure 4.6 shows the plot of entropy with varying circuit length for $r = 2$, $N = 1\,000\,000$ and $d = 20$. We can see that increasing circuit length results in a significant increase in entropy. In our secure random walk design, the sampling bias due to route capture is small, and the restricted topology attack dominates. Increasing circuit length mitigates the restricted topology attack and thus increases anonymity. (Note that increasing the circuit length past $l = 6$ will offer no benefit, unless $\log_d N > 6$.[5]) Finally, we study the effect of increasing redundancy. Figure 4.7 shows the plot of entropy with varying redundancy for $l = 3$. We can see that increasing redundancy beyond

[3]This is a slight simplification, as the exact fraction of compromised tunnels will depend on the share of bandwidth and the guard/exit status of compromised nodes.

[4]Fingers of fingers do not overlap in a regular de Bruijn topology.

[5]In real networks, the lack of perfect load-balancing will result in somewhat worse mixing, and thus values of $l > \log_d N$ may still make sense.

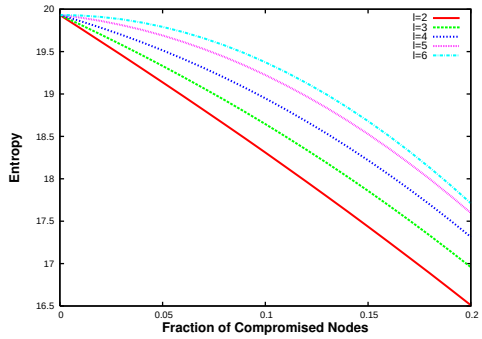


Figure 4.6: Effect of varying circuit length: Increasing circuit length increases entropy.

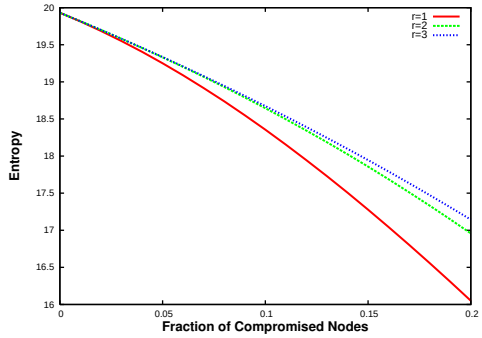


Figure 4.7: Effect of varying redundancy: There is little advantage in increasing redundancy beyond $r = 2$.

2 does not have any significant benefit. We use $r = 2$ in the remainder of our analysis.

4.2.3 Using shorter circuits

First, consider a two-hop random walk. Let us denote the first hop as A and the second hop as B. If both A, B are malicious, then the initiator anonymity is compromised. When only B is compromised, the initiator can be narrowed to the set of nodes that have A as their fingers. The expected size of this set is quite small (d), resulting in poor anonymity. Also note that the latter event happens frequently, with probability about f, where as both A and B are malicious with probability about f^2.

Now, let us consider our modification to the protocol: a node performs a secure three-hop random walk (A, B, C), and then uses the last two hops (B, C) for anonymous communication, by building a circuit directly to B

and then extending it to C. Again, the dominant event is when only C is compromised. Under this event, the attacker can narrow the choice of the initiator to the set of nodes who have B within two hops. The expected size of this set is now d^2. Thus our modification results in an increase in anonymity, while keeping the circuit length constant.

Note that in the anonymity analysis of the modified two hop random walk protocol, the entropy is 0 when the last two nodes are compromised. Thus let us redefine M_i for $(i \leq l-2)$ to be the event such that the first malicious node on the circuit is at the $i'th$ position, the last node is also compromised, but the second last node is honest. We define M_{l-1} as the event that the last two nodes are compromised, regardless of whether any previous nodes were compromised as well. $P(M_{l-1}) = f^2$, and $H(I|M_{l-1}) = 0$, since the initiator contacts the second last node directly. We keep the definition of M_l the same as before; i.e., only the last hop is compromised. For $i \leq l-2$, $P(M_i)$ can be expressed as:

$$P(M_i) = f(1-f)^{i-1}(1-f^r)\left(1-f^{r+1}\right)^{l-2-i}(1-f)f. \tag{4.7}$$

Figure 4.8 shows the plot of entropy for our modified protocol, computed as:

$$\begin{aligned} H &= \sum_{i=1}^{l-2} P(M_i)H(I|M_i) + P(M_l)H(I|M_l) \\ &\quad + \left(1 - \sum_{i=1}^{l} P(M_i)\right)\log_2 N. \end{aligned} \tag{4.8}$$

Here, l = 2–6 refers to our modified protocol where a node performs a 6 hop random walk and then uses only the last two hops for anonymous communication. We can see that that our modification allows a user to derive higher anonymity using longer random walks, but keeping the circuit length constant. Viewed from another perspective, this extension creates a trade-off between anonymity and performance. Using all l hops for anonymous communication is more secure, but introduces higher latency on the communication and uses more system resources. Using only the last two hops will improve the system performance, at the cost of revealing the identity of the initiator to the second-to-last hop. As can be see in Figure 4.8, the loss of

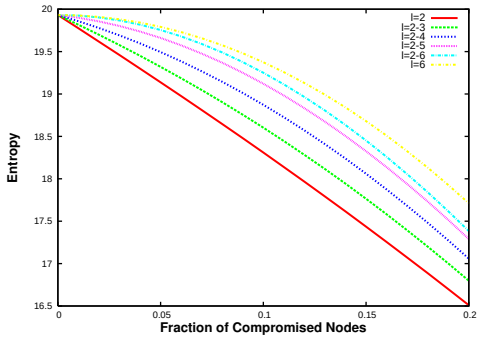


Figure 4.8: Using last two hops for anonymous communication: Mitigating restricted topology attacks while keeping circuit length constant.

anonymity is slight: using $l = 2$–6 results in anonymity that is only slightly lower than $l = 6$.

4.2.4 Comparison with Salsa

We will now compare our ShadowWalker protocol with Salsa [9]. Salsa uses secure lookup as a primitive to build a circuit for anonymous communication, which makes Salsa susceptible to information leak attacks [12]. To compute the effect of active attacks on lookups, we used a simulator developed by the authors of Salsa [86]. The simulator was configured to simulate 1000 topologies, and in each topology, results were averaged over 1000 random lookups. The Salsa architecture divides the identifier space into groups, where the number of groups is denoted by $|G|$. We used the parameters $N = 10,000$ and $|G| = 128$ for the simulation (it is difficult to scale the simulations beyond $10,000$ nodes). Next, we modeled the Salsa path building process as a stochastic activity network in the Möbius framework [87]. Figure 4.9 compares the anonymity provided by ShadowWalker and Salsa. In our system, we use the degree $d = 13$ and $r = 2$. In the next section, we will see that this translates into an effective degree of 39 ($(r + 1) \cdot d$). This is comparable to the effective degree of Salsa in this configuration, which is 85 ($10000/128 + \log_2 128$). We can see that for $f = 0.2$, our protocol using $l = 5$ has an entropy of 12, while Salsa only has an entropy of 7.5. Even our modified protocol which uses only two hops for anonymous communication, gives much better anonymity than Salsa.

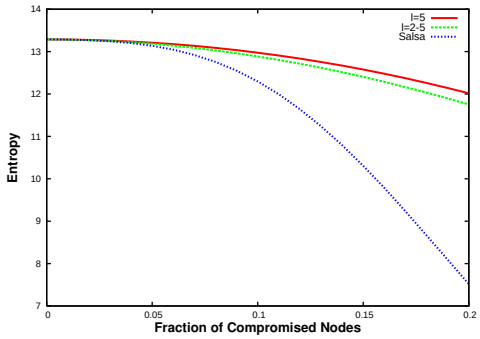


Figure 4.9: Comparison with Salsa: For $f = 0.2$, our protocol has 4.5 bits more entropy than Salsa.

4.2.5 Selective DoS attack

Recently, Borisov et al. [54] proposed a selective denial-of-service attack on anonymous communication. In this attack, malicious nodes can selectively drop packets in order to shut down any circuits that they are a part of, but which they cannot compromise. Borisov et al. found that selective DoS attack is most effective against peer-to-peer anonymous communication systems, because the circuit construction in P2P systems is complex and may provide many nodes with the opportunity to selectively deny service. Our design is vulnerable to the selective DoS attack in two ways:

Selective DoS by shadows: As a shadow node, a malicious node M may refuse to give signatures to honest nodes, or may give incorrect signatures to honest nodes. This attack will ensure that the honest nodes who have a malicious node as a shadow will never get selected in the random walk as an intermediate node, since the initiator will not be able to verify the neighbor relationships.

Selective DoS during circuit construction: Malicious nodes can also selectively break any circuits that they cannot compromise. Whenever malicious nodes find that they are part of a circuit in which they are unable to infer any information about the initiator, they stop forwarding packets on the circuit, causing a new circuit to be constructed. This attack is similar to the selective-DoS attack on Tor described by Borisov et al.

We can mitigate the first attack by using a symmetric shadow relationship. This means that if node A is a shadow of node B, then node B is a shadow of node A. If a node stops receiving signatures from its shadow, it can reciprocate by no longer certifying the shadow's routing information. As a result,

malicious nodes that do not follow the protocol and refuse to provide signatures will themselves be excluded from the circuit construction process. An adversary may decide to sacrifice its nodes, and in this process DoS (atmost) r honest nodes. However, since small redundancy levels of $r = 2, 3$ suffice for the security of our protocol, this strategy does not benefit the adversary.

For the second attack, the best strategy for malicious nodes is to shut down any circuit in which the last node is honest, since there is no hope of compromising it. Thus the only circuits that will be built are those where the last node is compromised, or where all the nodes are honest. The following equation quantifies the effect of the selective DoS attack on our protocol.

$$\begin{aligned} H &= \sum_{i=1}^{l} \frac{P(M_i)}{\sum_{j=1}^{l} P(M_j) + (1-f)^l} H(I|M_i) \\ &\quad + \frac{(1-f)^l}{\sum_{i=1}^{l} P(M_i) + (1-f)^l} \log_2 N \end{aligned} \tag{4.9}$$

Figure 4.10 plots the entropy for our protocol under the selective DoS attack. There is an interesting tradeoff here. On one hand, increasing circuit length mitigates the restricted topology attack and increases anonymity. On the other hand, increasing circuit length gives more opportunities to the attackers to launch a selective DoS attack. We can see that for small values of f, the former effect dominates, and increasing circuit length increases anonymity. There is a crossover point at about 18% of compromised nodes, when increasing circuit length beyond $l = 4$ becomes counterproductive, because of the selective-DoS attack. We note that our modified protocol, in which the initiator only chooses the last two hops for anonymous communication, provides a good defense against the selective-DoS attack. This is because the intermediate hops do not decide to abort until the circuit construction has reached the last hop. However, at that point, only the second-to-last hop can perform denial-of-service on the circuit. We can see from the figure, that $l = 2$–5 is most resilient to selective-DoS attack. Also note that selective-DoS presents a significant problem for Salsa. Salsa is able to provide only 4 bits of entropy at $f = 0.2$, as compared to about 11.5 bits of entropy for $l =$2–5.

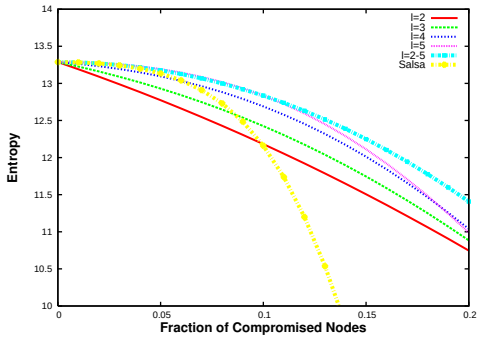


Figure 4.10: Selective-DoS attack: Using l =2–6 resists selective DoS attack.

4.3 Experimental Results

We implemented our protocol using an event-based simulator in C++ with 1.2KLOC. We consider a WAN setting, where latencies between each pair of nodes are estimated using the King data set [94]. This data set contains measured latencies between Internet domain name servers (DNS) and is highly heterogeneous. The average round trip time (RTT) in the data set is around 182ms and the maximum RTT is around 800ms. To handle churn, we run the stabilization protocol every second. The time period for refreshing fingers and signing certificates is also set to 1 second. We simulate our protocol for $N = 1000$ nodes with a redundancy parameter $r = 2$ and $d = 10$.

Studies have shown that in many popular peer-to-peer networks, the mean value of node uptime is about 60 minutes [95, 96]. We considered two widely used synthetic models for node uptime 1) PDF $f(x) = \lambda e^{-\lambda x}$. We set $\lambda = 1/60$. This is an exponential distribution with mean 60 minutes. 2) PDF $f(x) = \frac{ab^a}{(x+b)^{a+1}}$. We set $a = 1.5, 2, 3$ and b fixed so that the distribution had mean 60 minutes. This is a standard Pareto distribution, shifted b units (without the shift, a node would be guaranteed to be up for at least b minutes).

4.3.1 Communication overhead

Topology maintenance: As compared to a structured network, the overhead for topology maintenance in our protocol is higher due to the inherent redundancy in topology. The transformation from a structured topology to a redundant structured topology increases the effective node degree from d to

$(r+1)^2 d$. (Each finger has r shadows, and each node is a shadow for (around) $r+1$ nodes.) An important consequence of our secure lookup protocol is that along with the node corresponding to the chosen ID, its shadows are returned as well. This significantly reduces the communication overhead of our protocol because it obviates the need for performing multiple secure lookups for the shadows of fingers. The use of our secure lookup protocol reduces the effective node degree to $(r+1) \cdot d$. In the previous section, we had seen that our system provides better anonymity than Salsa with similar effective node degree. For $N = 1000$ nodes and $r = 2$, the mean communication overhead per node was measured to be 5980 bytes/sec.

Circuit Setup: To establish a circuit of length l, the initiator performs l key establishments and rl signature verifications. The corresponding figure for Salsa is $r(l-1)+1$ key establishments and $r^2(l-1)+r$ lookups. The table below shows the mean circuit setup latency. We can see that even for $l = 6$, the circuit setup time is less than 4 seconds. Since we avoid the use of lookups, the circuit setup latency for our protocol is smaller than Salsa.

Mean Circuit Setup Latency (ms)				
l=2	l=3	l=4	l=5	l=6
546	1092	1820	2730	3822

4.3.2 Reliability of circuit construction under churn

Due to churn, the routing states at different nodes may be inconsistent at times, resulting in different views of the network. This will mean that corresponding signatures by shadow nodes for the routing state of a node A may not be consistent, and our circuit construction protocol may fail.

Figure 4.11 shows the effect of churn on the reliability of our circuit construction protocol. Let us first consider the exponential distribution for node uptimes. We can see that increasing path lengths increases the probability of failure. This is because there is an higher chance of a node and its shadows having an inconsistent view of the network. For a path length $l = 6$, the probability of failure is about 0.05. Next, observe that the probability of failure increases if we model node churn as a Pareto distribution. Moreover, smaller values of the exponent a lead to higher probabilities of failure. To get some intuition for this, observe that Pareto distributions with smaller

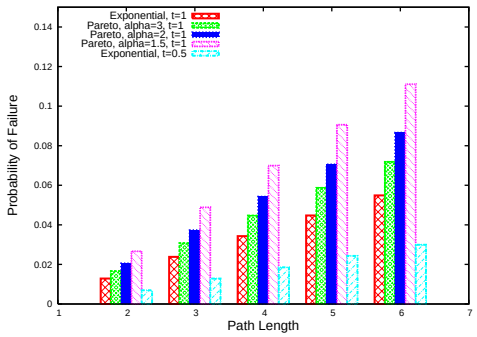


Figure 4.11: Impact of churn on reliability.

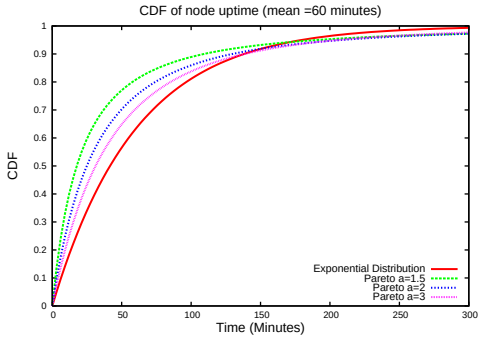


Figure 4.12: Churn distributions.

exponents a have a longer 'tail' in the CDF, as depicted by Figure 4.12. This results in a larger number of node arrivals and node departures (even though the mean node uptimes are the same), leading to an decrease in reliability of circuit construction.

We note that reliability of circuit construction can be increased by being more aggressive in topology maintenance (i.e., reducing the time period t for refreshing fingers and signing state). Figure 4.11 depicts this tradeoff between bandwidth use and reliability of circuit construction. We can see that for exponential distribution of node uptimes, by reducing the time period from $t = 1$ seconds to $t = 0.5$ seconds, the probability of failure has been approximately halved.

4.3.3 Secure lookup

A lookup is successful if there is at least a single honest node in each step of the lookup. For a lookup of path length l, the probability that a lookup succeeds can be modeled as:

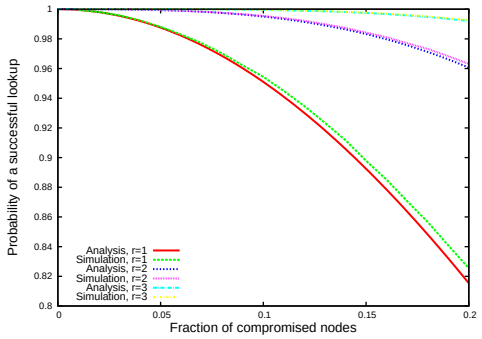


Figure 4.13: Lookup security.

$$P(\text{secure lookup}) = \left(1 - f^{r+1}\right)^{l}.$$

Figure 4.13 plots of probability of a successful lookup for varying values of r. For $f = 0.1, r = 2$, the probability of a successful lookup is 0.99. Even when we increase the value of f to $f = 0.2$, the lookup is still successful with probability 0.95. The lookup security improves exponentially with increasing r, because the chance that a node and all its shadows are malicious falls exponentially in r. Thus for $f = 0.2$ and $r = 3$, the lookup succeeds with probability 0.99. Note that for small values of r, the lookup security can also be improved by performing redundant versions of the above lookup.

4.3.4 Anonymity

Finally, we present simulation results for the anonymity provided by ShadowWalker. Using simulations, we have performed a whole system evaluation of ShadowWalker to check for any hidden correlations not captured by our analytic model. Our simulator also captures real world behavior like the effect of irregular topologies, which is not considered in our model. Figure 4.14 depicts the anonymity provided by ShadowWalker for $l = 4$ and $l = 2 - 4$. We can see that our simulation and analytic results closely match.

4.4 Summary

We proposed ShadowWalker: a new design for low-latency P2P anonymous communication. ShadowWalker is able to effectively defend against common

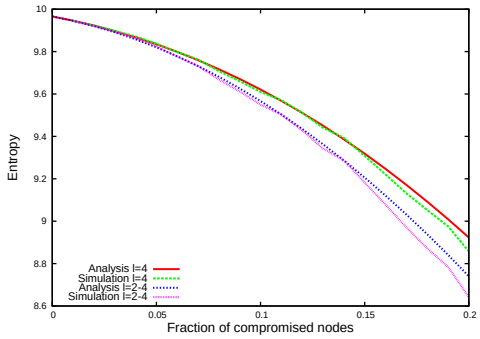


Figure 4.14: Anonymity.

attacks on peer-to-peer systems and achieve levels of anonymity superior to the state of the art in P2P anonymous communication.

CHAPTER 5

SYBILINFER: DETECTING SYBIL NODES USING SOCIAL NETWORKS

The peer-to-peer paradigm allows cooperating users to enjoy a service with little or no need for any centralised infrastructure. While communication [60] and storage [97] systems employing this design philosophy have been proposed, the lack of any centralised control over identities opens such systems to Sybil attacks [8]: a few malicious nodes can simulate the presence of a very large number of nodes, to take over or disrupt key functions of the distributed or peer-to-peer system. Any attempt to build fault-tolerance mechanisms is doomed since adversaries can control arbitrary fractions of the system nodes. This Sybil attack is further made practical through the use of the existing large number of compromised networked machines (often called *zombies*) being part of bot-nets.

Similar problems plague Web 2.0 applications that rely on collaborative tagging, filtering and editing, like Wikipedia [98], or `del.icio.us` [99]. A single user can register under different pseudonyms, and bypass any elections of velocity check mechanism that attempts to guarantee the quality of the data through the plurality of contributors. On-line forums, starting with USENET [100], to contemporary blogs or virtual worlds like Second Life [101], always have to deal with the issue of disruption in the discussion threads, with persistent abusers coming back under different names. All these are forms of Sybil attacks at the high-level application layers.

There are two schools of Sybil defence mechanisms, the *centralised* and *decentralised* ones. Centralised defences assume the existence of an authority that is capable of doing admission control for the network [65]. Its role is to rate limit the introduction of 'fake' identities, to ensure that the fraction of corrupt nodes remains under a certain threshold. The practicalities of running such an authority are very system-specific and in general it would have to act as a public key certification authority as well as a guardian of the moral standing of the nodes introduced – a very difficult problem in practice.

Such centralised solutions are also at odds with the decentralisation guiding principle of peer-to-peer systems.

Decentralised approaches recognise the difficulty in having a single authority vouching for nodes, and distribute this task across all nodes of the system. The first such proposal is Advogato [102], which aimed to reduce abuse in on-line services, followed by a proposal to use introduction graphs of DHTs [74] to limit the potential for routing disruption in those systems. The state of the art SybilGuard [13] and SybiLimit [73] propose the use of social networks to mitigate Sybil attacks. As we will see, SybilGuard suffers from high false negatives, while SybilLimit makes unrealistic assumptions about the knowledge of number of honest nodes in the network. In both cases the systems Sybil detection strategies are based on heuristics that are not optimal.

Our key contribution is to propose SybilInfer, a method for detecting Sybil nodes in a social network, that makes use of all information available to the defenders. The formal model underlying our approach casts the problem of detecting Sybil nodes in the context of Bayesian Inference: given a set of stated relationships between nodes, the task is to label nodes as honest or dishonest. Based on some simple and generic assumptions, like the fact that social networks are fast mixing [58], we sample cuts in the social graph according to the probability they divide it into honest and dishonest regions. These samples not only allow us to label nodes as honest or Sybil attackers, but also to associate with each label output by our algorithm a degree of certainty.

The proposed techniques can be applied in a wide variety of settings where high reliability peer-to-peer systems, or Sybil-resistant collaborative mechanisms, are required even under attack:

- Secure routing in DHTs motivated early research into this field, and our proposal can be used instead of a centralised authority to limit the fraction of dishonest nodes, that could disrupt routing [65].

- In public anonymous communication networks, such as Tor [3], our techniques can be used to eliminate the potential for a single entity introducing a large number of nodes, and de-anonymize users' circuits. This was so far a key open problem for securely scaling such systems.

- Leader election [103] and Byzantine agreement [104] mechanisms that were rendered useless by the Sybil attack can again be of use, after Sybil nodes have been detected and eliminated from the social graph.

- Finally, detecting Sybil accounts is a key step in preventing false email accounts used for spam, or preventing trolling and abuse of on-line communities and web-forums. Our techniques can be applied in all those settings, to fight spam and abuse [102].

SybilInfer applies to settings where a peer-to-peer or distributed system is somehow based on or aware of social connections between users. Properties of natural social graphs are used to classify nodes are honest or Sybils. While this approach might not be applicable to very traditional peer-to-peer systems [60], it is more and more common for designers to make distributed systems aware of the social environment of their users. Third party social network services [105, 106], can also be used to extract social information to protect systems against sybil attacks using SybilInfer. Section 5.4 details deployment strategies for SybilInfer and how it is applicable to current systems.

We show analytically that SybilInfer is, from a theoretical perspective, very powerful: under ideal circumstances an adversary gains no advantage by introducing into the social network any additional Sybil nodes that are not 'naturally' connected to the rest of the social structure. Even linking all dishonest nodes with each other (without adding any Sybils) changes the characteristics of their social sub-graph, and can under some circumstances be detected. We demonstrate the practical efficacy of our approach using both synthetic scale-free topologies as well as real-world LiveJournal data. We show very significant security improvements over both SybilGuard and SybilLimit, the current state of the art Sybil defence mechanisms. We also propose extensions that enable our solution to be implemented in decentralised settings.

This chapter is organised as follows: in Section 5.1 we present an overview of our approach that can be used as a road-map to the technical sections. In Section 5.2 we present our security assumptions, threat model, the probabilistic model and sampler underpinning SybilInfer; a security evaluation follows in Section 5.3, providing analytical as well as experimental arguments supporting the security of the method proposed along with a comparison

with SybilInfer. Section 5.4 discusses the settings in which SybilInfer can be fruitfully used, followed by a summary in Section 5.5.

5.1 Overview

The SybilInfer algorithm takes as an input a social graph G and a single known good node that is part of this graph. The following conceptual steps are then applied to return the probability each node is honest or controlled by a Sybil attacker:

- A set of traces T are generated and stored by performing special random walks over the social graph G. These are the only information retained about the graph for the rest of the SybilInfer algorithm, and their generation is detailed in Section 5.2.1.

- A probabilistic model is then defined that describes the likelihood a trace T was generated by a specific honest set of nodes within G, called X. This model is based on our assumptions that social networks are fast mixing, while the transitions to dishonest regions are slow. Given the probabilistic model, the traces T and the set of honest nodes we are able to calculate $\Pr[T|X \text{ is honest}]$. The calculation of this quantity is the subject of Section 5.2.1 and Section 5.2.2.

- Once the probabilistic model is defined, we use Bayes' theorem to calculate for any set of nodes X and the generated trace T, the probability that X consists of honest nodes. Mathematically this quality is defined as $\Pr[X \text{ is honest}|T]$. The use of Bayes' theorem is described in Section 5.2.1.

- Since it is not possible to simply enumerate all subsets of nodes X of the graph G, we instead sample from the distribution of honest node sets X, to only get a few $X_0, \ldots, X_N \sim \Pr[X \text{ is honest}|T]$. Using those representative sample sets of honest nodes, we can calculate the probability any node in the system is honest or dishonest. Sampling and the approximation of the sought marginal probabilities are the subject of Section 5.2.3.

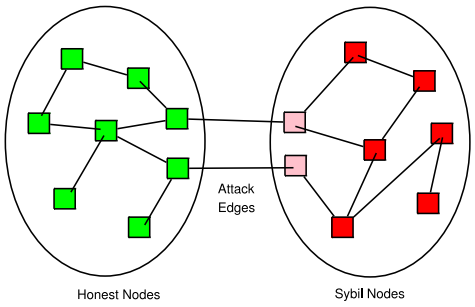


Figure 5.1: Illustration of honest nodes, Sybil nodes and attack edges between them.

The key conceptual difficulty of our approach is the definition of the probabilistic model over the traces T, and its inversion using Bayes' theorem to define a probability distribution over all possible honest sets of nodes X. This distribution describes the likelihood that a specific set of nodes is honest. The key technical challenge is making use of this distribution to extract the sought probability each node is honest or dishonest, that we achieve via sampling. Section 5.2 describes in some detail how these issues are tackled by SybilInfer.

5.2 Model and Algorithm

Let us denote the social network topology as a graph G comprising vertices V, representing people and edges E, representing trust relationships between people. We consider the friendship relationship to be an undirected edge in the graph G. Such an edge indicates that two nodes trust each other to not be part of a Sybil attack. Furthermore, we denote the friendship relationship between an attacker node and an honest node as an *attack edge* and the honest node connected to an attacker node as a *naive node* or *misguided node*. Different types of nodes are illustrated in Figure 5.1. These relationships must be understood by users as having security implications, to restrict the promiscuous behaviour often observed in current social networks, where users often flag strangers as their friends [107].

We build our Sybil defence around the following assumptions:

1. At least one honest node in the network is known. In practise, each node trying to detect Sybil nodes can use itself as the a priori honest

node. This assumption is necessary to break symmetry: otherwise an attacker could simply mirror the honest social structure, and any detector would not be able to distinguish which of the two regions is the honest one.

2. Social networks are fast mixing: this means that a random walk on the social graph converges quickly to a node following the stationary distribution of the graph. Several authors have shown that real-life social networks are indeed fast mixing [58, 108].

3. A node knows the complete social network topology (G) : social network topologies are relatively static, and it is feasible to obtain a global snapshot of the network. Friendship relationships are already public data for popular social networks like Facebook [105] and Orkut [106]. This assumption can be relaxed to using sub-graphs, making SybilInfer applicable to decentralised settings.

Assumptions (1) and (2) are identical to those made by the SybilGuard and SybilInfer systems. Previously, the authors of SybilGuard [13] observed that when the adversary creates too many Sybil nodes, then the graph G has a small cut: a set of edges that together have small stationary probability and whose removal disconnects the graph into two large sub-graphs.

This intuition can be pushed much further to build superior Sybil defences. It has been shown [109] that the presence of a small cut in a graph results in slow mixing which means that fast mixing implies the absence of small cuts. Applied to social graphs this observation underpins the key intuition behind our Sybil defence mechanism: the *mixing between honest nodes in the social networks is fast, while the mixing between honest nodes and dishonest nodes is slow.* Thus, computing the set of honest nodes in the graph is related to computing the bottleneck cut of the graph.

One way of formalising the notion of a bottleneck cut, is in terms of *graph conductance (*Φ*)* [110], defined as:

$$\Phi = \min_{X \subset V : \pi(X) < 1/2} \Phi_X,$$

where Φ_X is defined as:

$$\Phi_X = \frac{\Sigma_{x \in X} \Sigma_{y \notin X} \pi(x) P_{xy}}{\pi(X)},$$

and $\pi(\cdot)$ is the stationary distribution of the graph. Intuitively for any subset of vertices $X \subset V$ its conductance Φ_X represents the probability of going from X to the rest of the graph, normalised by the probability weight of being on X. When the value is minimal the bottleneck cut in the graph is detected.

Note that performing a brute force search for this bottleneck cut is computationally infeasible (it is actually NP-Hard, given its relationship to the sparse-cut problem). Furthermore, finding the exact smallest cut is not as important as being able to judge how likely any cut is, to be dividing nodes into an honest and dishonest region. This probability is related to the deviation of the size of any cut from what we would expect in a natural, fast mixing, social network.

5.2.1 Inferring honest sets

In this paper, we propose a framework based on Bayesian inference to detect approximate cuts between honest and Sybil node regions in a social graph and use those to infer the labels of each node. A key strength of our approach is that it not only associates labels to each node, but also finds the correct probability of error that could be used by peer-to-peer or distributed applications to select nodes.

The first step of SybilInfer is the generation of a set of random walks on the social graph G. These walks are generated by performing a number s of random walks, starting from each node in the graph (i.e. a total of $s \cdot |V|$ walks.) A special probability transition matrix is used, defined as follows:

$$P_{ij} = \begin{cases} \min(\frac{1}{d_i}, \frac{1}{d_j}) & \text{if } i \to j \text{ is an edge in G} \\ 0 & \text{otherwise} \end{cases},$$

where d_i denotes the degree of vertex i in G.

This choice of transition probabilities ensures that the stationary distribution of the random walk is uniform over all vertices $|V|$. The length of the random walks is $l = O(\log |V|)$, which is rather short, while the num-

ber of random walks per node (denoted by s) is a tunable parameter of the model. Only the starting vertex and the ending vertex of each random walk are used by the algorithm, and we denote this set of vertex-pairs, also called the *traces*, by T.

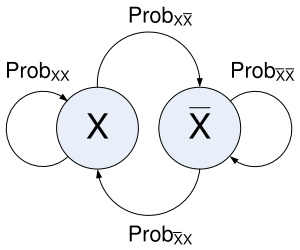
Now consider any cut $X \subset V$ of nodes in the graph, such that the a priori honest node is an element of X. We are interested in the probability that the vertices in set X are all honest nodes, given our set of traces T, i.e. $P(X = \text{Honest}|T)$. Through the application of Bayes' theorem we have an expression of this probability:

$$P(X = \text{Honest}|T) = \frac{P(T|X = \text{Honest}) \cdot P(X = \text{Honest})}{Z},$$

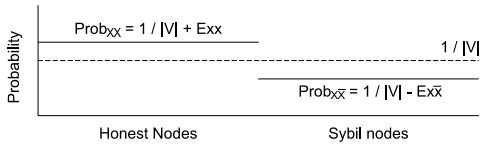
where Z is the normalization constant given by: $Z = \Sigma_{X \subset V} P(T|X = \text{Honest}) \cdot P(X = \text{Honest})$. Note that Z is difficult to compute because it involves the summation of an exponential number of terms in the size of $|V|$. Only being able to compute this probability up to a multiplicative constant Z is not an impediment. The a priori distribution $P(X = \text{Honest})$ can be used to encode any further knowledge about the honest nodes, or can simply be set to be uniform over all possible cuts.

Bayes' theorem has allowed us to reduce the initial problem of inferring the set of good nodes X from the set of traces T, to simply being able to assign a probability to each set of traces T given a set of honest nodes X, i.e. calculating $P(T|X = \text{Honest})$. Our only remaining theoretical task is deriving this probability, given our model's assumptions.

Note that since the set X is honest, we assume (by assumption (2)) fast mixing amongst its elements, meaning that a short random walk reaches any element of the subset X uniformly at random. On the other hand, a random walk starting in X is less likely to end up in the dishonest region $\bar{X}$, since there should be an abnormally small cut between them. (This intuition is illustrated in Figure 5.2(a).) Therefore we approximate the probability that a short random walk of length $l = O(\log |V|)$ starts in X and ends at a particular node in X is given by $\text{Prob}_{XX} = \Pi + E_{XX}$, where Π is the stationary distribution given by $1/|V|$, for some $E_{XX} > 0$. Similarly, we approximate the probability that a random walk starts in X and does not end in X is given by $\text{Prob}_{X\bar{X}} = \Pi - E_{X\bar{X}}$. Notice that $\text{Prob}_{XX} > \text{Prob}_{X\bar{X}}$, which captures the property that there is fast mixing amongst honest nodes



(a) A schematic representation of transition probabilities between honest X and dishonest $\bar{X}$ regions of the social network.



(b) The model of the probability a short random walk of length $O(\log |V|)$ starting at an honest node ends on a particular honest or dishonest (Sybil) node. If no Sybils are present the network is fast mixing and the probability converges to $1/|V|$, otherwise it is biased towards landing on an honest node.

Figure 5.2: Illustrations of the SybilInfer models.

and slow mixing between honest and dishonest nodes. The approximate probabilities Prob_{XX} and $\text{Prob}_{X\bar{X}}$ and their likely gap from the ideal $1/|V|$ are illustrated in Figure 5.2(b).

Let N_{XX} be the number of traces in T starting in the honest set X and ending in same honest set X. Let $N_{X\bar{X}}$ be the number of random walks that start at the honest set X and end in the dishonest set $\bar{X}$. $N_{\bar{X}\bar{X}}$ and $N_{\bar{X}X}$ are defined similarly. Given the approximate probabilities of transitions from one set to the other and the counts of such transitions we can ascribe a probability to the trace:

$$P(T|X = \text{Honest}) = (\text{Prob}_{XX})^{N_{XX}} \cdot (\text{Prob}_{X\bar{X}})^{N_{X\bar{X}}} \cdot (\text{Prob}_{\bar{X}\bar{X}})^{N_{\bar{X}\bar{X}}} \cdot (\text{Prob}_{\bar{X}X})^{N_{\bar{X}X}},$$

where $\text{Prob}_{\bar{X}\bar{X}}$ and $\text{Prob}_{\bar{X}X}$ are the probabilities a walk starting in the dishonest region ends in the dishonest or honest regions respectively.

The model described by $P(T|X = \text{Honest})$ is an approximation to reality that is suitable enough to perform Sybil detection. It is of course unlikely that a random walk starting at an honest node will have a uniform probability to land on all honest or dishonest nodes respectively. Yet this simple probabilistic model relating the starting and ending nodes of traces is rich enough to capture the "probability gap" between landing on an honest or dishonest node, as illustrated in Figure 5.2(b), and suitable for Sybil detection.

5.2.2 Approximating E_{XX}

We have reduced the problem of calculating $P(T|X = \text{Honest})$ to finding a suitable E_{XX}, representing the 'gap' between the case when the full graph is fast mixing (for $E_{XX} = 0$) and when there is a distinctive Sybil attack (in which case $E_{XX} >> 0$).

One approach could be to try inferring E_{XX} through a trivial modification of our analysis to co-estimate $P(X = \text{Honest}, E_{XX}|T)$. Another possibility is to approximate E_{XX} or Prob_{XX} directly, by choosing the most likely candidate value for each configuration of honest nodes X considered. This can be done through the conductance or through sampling random walks on the social graph.

Given the full graph G, Prob_{XX} can be approximated as $\text{Prob}_{XX} = \frac{\Sigma_{x \in X} \Sigma_{y \in X} \Pi(x) P^l_{xy}}{\Pi(X)}$, where P^l_{xy} is the probability that a random walk of length l starting at x ends in y. This approximation is very closely related to the conductance of the set X and $\bar{X}$. Yet computing this measure would require some effort.

Notice that Prob_{XX}, as calculated above, can also be approximated by performing many random walks of length l starting at X and computing the fraction of those walks that end in X. Interestingly our traces already contain random walks over the graph of exactly the appropriate length, and therefore we can reuse them to estimate a good Prob_{XX} and related probabilities. Given the counts $N_{XX}, N_{X\bar{X}}, N_{\bar{X}X}$ and $N_{\bar{X}\bar{X}}$:

$$\text{Prob}_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|}$$

and

$$\text{Prob}_{\bar{X}\bar{X}} = \frac{N_{\bar{X}\bar{X}}}{N_{\bar{X}\bar{X}} + N_{\bar{X}X}} \cdot \frac{1}{|\bar{X}|},$$

and $\text{Prob}_{X\bar{X}} = 1 - \text{Prob}_{XX}$ and $\text{Prob}_{\bar{X}X} = 1 - \text{Prob}_{\bar{X}\bar{X}}$.

Approximating Prob_{XX} through the traces T provides us with a simple expression for the sought probability, based simply on the number of walks

starting in one region and ending in another:

$$\begin{aligned} P(T|X = \text{Honest}) \quad = \quad & (\frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|})^{N_{XX}} \cdot \\ & (\frac{N_{X\bar{X}}}{N_{X\bar{X}} + N_{XX}} \cdot \frac{1}{|\bar{X}|})^{N_{X\bar{X}}} \cdot \\ & (\frac{N_{\bar{X}\bar{X}}}{N_{\bar{X}\bar{X}} + N_{\bar{X}X}} \cdot \frac{1}{|\bar{X}|})^{N_{\bar{X}\bar{X}}} \cdot \\ & (\frac{N_{\bar{X}X}}{N_{\bar{X}X} + N_{\bar{X}\bar{X}}} \cdot \frac{1}{|X|})^{N_{\bar{X}X}}. \end{aligned}$$

This expression concludes the definition of our probabilistic model, and contains only quantities that can be extracted from either the known set of nodes X, or the set of traces T that is assigned a probability. Note that we do not assume any prior knowledge of the size of the honest set, and it is simply a variable $|X|$ or $|\bar{X}|$ of the model. Next, we shall describe how to sample from the distribution $P(X = \text{Honest}|T)$ using the Metropolis-Hastings algorithm.

5.2.3 Sampling honest configurations

At the heart of our Sybil detection techniques lies a model that assigns a probability to each sub-set of nodes of being honest. This probability $P(X = \text{Honest}|T)$ can be calculated up to a constant multiplicative factor Z, that is not easily computable. Hence, instead of directly calculating this probability for any configuration of nodes X, we will attempt instead to sample configurations X_i following this distribution. Those samples are used to estimate the marginal probability that any specific node, or collections of nodes, are honest or Sybil attackers.

Our sampler for $P(X = \text{Honest}|T)$ is based on the established Metropolis-Hastings algorithm [111] (MH), which is an instance of a Markov Chain Monte Carlo sampler. In a nutshell, the MH algorithm holds at any point a sample X_0. Based on the X_0 sample a new candidate sample X' is proposed according to a probability distribution Q, with probability $Q(X'|X_0)$. The new sample X' is 'accepted' to replace X_0 with probability α:

$$\alpha = \min(\frac{P(X'|T) \cdot Q(X_0|X')}{P(X_0|T) \cdot Q(X'|X_0)}, 1);$$

otherwise, the original sample X_0 is retained. It can be shown that after multiple iterations this yields samples X according to the distribution $P(X|T)$ irrespective of the way new candidate sets X' are proposed or the initial state of the algorithm, i.e. a more likely state X will pop-out more frequently from the sampler, than less likely states.

A relatively naive strategy can be used to propose candidate states X' given X_0 for our problem. It relies on simply considering sets of nodes X' that are only different by a single member from X_0. Thus, with some probability p_{add} a random node $x \in \bar{X_0}$ is added to the set to form the candidate $X' = X_0 \cup x$. Alternatively, with probability p_{remove}, a member of X_0 is removed from the set of nodes, defining $X' = X_0 \cap x$ for $x \in X_0$. It is trivial to calculate the probabilities $Q(X'|X_0)$ and $Q(X'|X_0)$ based on p_{add}, p_{remove} and using a uniformly at random choice over nodes in X_0, $\bar{X_0}$, X' and $\bar{X'}$ when necessary.

A key issue when utilizing the MH algorithm is deciding how many iterations are necessary to get independent samples. Our rule of thumb is that $|V| \cdot \log |V|$ steps are likely to guarantee convergence to the target distribution P. After that number of steps the coupon collector's theorem states that each node in the graph would have been considered at least once by the sampler, and assigned to the honest or dishonest set. In practice, given very large traces T, the number of nodes that are difficult to categorise is very small, and a non-naive sampler requires few steps to produce good samples (after a certain burn-in period that allows it to detect the most likely honest region).

Finally, given a set of N samples $X_i \sim P(X|T)$ output by the MH algorithm it is possible to calculate the marginal probabilities any node is honest. This is key output of the SybilInfer algorithm: given a node i it is possible to associate a probability it is honest by calculating: $\Pr[i \text{ is honest}] = \frac{\sum_{j \in [0, N-1)} I(i \in X_j)}{N}$, where $I(i \in X_j)$ is an indicator variable taking value 1 if node i is in the honest sample X_j, and value zero otherwise. Enough samples can be extracted from the sampler to estimate this probability with an arbitrary degree of precision.

More sophisticated samplers would make use of a better strategy to propose candidate states X' for each iteration. The choice of X' can, for example, be biased towards adding or removing nodes according to how often random walks starting at the single honest node land on them. We expect nodes that are reached often by random walks starting in the honest region to be

honest, and the opposite to be true for dishonest nodes. In all cases this bias is simply an optimization for the sampling to take fewer iterations, and does not affect the correctness of the results.

5.3 Security Evaluation

In this section we discuss the security of SybilInfer when under Sybil attack. We show analytically that we can detect when a social network suffers from a Sybil attack, and correctly label the Sybil nodes. Our assumptions and full proposal are then tested experimentally on synthetic as well as real-world data sets, indicating that the theoretical guarantees hold.

5.3.1 Theoretical results

The security of our Sybil detection scheme hinges on two important results. First, we show that we can detect whether a network is under Sybil attack, based on the social graph. Second, we show that we are able to detect Sybil attackers connected to the honest social graph, and this *for any* attacker topology.

Our first result states that:

> THEOREM A. In the absence of any Sybil attack, the distribution of $P(X = \text{Honest}|T)$, for a given size $|X|$, is close to uniform, and all cuts are equally likely ($E_{XX} \approxeq 0$).

This result is based on our assumption that a random walk over a social network is fast mixing, meaning that, after $\log(|V|)$ steps, it visits nodes drawn from the stationary distribution of the graph. In our case the random walk is performed over a slightly modified version of the social graph, where the transition probability attached to each link ij is:

$$P_{ij} = \begin{cases} \min(\frac{1}{d_i}, \frac{1}{d_j}) & \text{if } i \to j \text{ is an edge in G} \\ 0 & \text{otherwise} \end{cases},$$

which guarantees that the stationary distribution is uniform over all nodes (i.e. $\Pi = \frac{1}{|V|}$). Therefore we expect that in the absence of an adversary the

short walks in T to end at a network node drawn at random amongst all nodes $|V|$. In turn this means that the number of end nodes in the set of traces T, that end in the honest set X is $N_{XX} = \lim_{|T_X|\to\infty} \frac{|X|}{|V|} \cdot |T_X|$, where T_X is the number of traces in T starting within the set $|X|$. Substituting this in the equations presented in Sections 5.2.1 and 5.2.2 we get:

$$\text{Prob}_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|} \Rightarrow \tag{5.1}$$

$$\Pi + E_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|} \Rightarrow \tag{5.2}$$

$$\frac{1}{|V|} + E_{XX} = \frac{(|X|/|V|) \cdot |T_X|}{|T_X|} \cdot \frac{1}{|X|} \Rightarrow \tag{5.3}$$

$$E_{XX} = 0. \tag{5.4}$$

As a result, by sufficiently increasing the number of random walks T performed on the social graph, we can get E_{XX} arbitrarily close to zero. In turn this means that our distribution $P(T|X = \text{Honest})$ is uniform for given sizes of $|X|$, given our uniform a priori $P(X = \text{Honest}|T)$.

In a nutshell by estimating E_{XX} for any sample X returned by the MH algorithm, and testing how close it is to zero we detect whether it corresponds to an attack (as we will see from theorem B) or a natural cut in the graph. We can increase the precision of the detector arbitrarily by increasing the number of walks T.

Our second results relates to the behaviour of the system under Sybil attack:

> THEOREM B. Connecting *any* additional Sybil nodes to the social network, through a set of corrupt nodes, lowers the dishonest sub-graph conductance to the honest region, leading to slow mixing, and hence we expect $E_{XX} > 0$.

First we define the dishonest set $\bar{X}_0$ comprising all dishonest nodes connected to honest nodes in the graph. The set $\bar{X}_S$ contains all dishonest nodes in the system, including nodes in $\bar{X}_0$ and the Sybil nodes attached to them. It must hold that $|\bar{X}_0| < |\bar{X}_S|$, in case there is a Sybil attack. Second we note that the probability of a transition between an honest node $i \in X$ and a dishonest node $j \in \bar{X}$ cannot increase through Sybil attacks, since it is equal to $P_{ij} = \min(\frac{1}{d_i}, \frac{1}{d_j})$. At worst the corrupt node will increase its degree by

connecting Sybils which has only the potential to decrease this probability. Therefore we have that $\sum_{x\in\bar{X}_S}\sum_{y\notin\bar{X}_S}P_{xy} \leq \sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0}P_{xy}$. Combining the two inequalities we get that:

$$\frac{\sum_{y\notin\bar{X}_S}P_{xy}}{|\bar{X}_S|} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0}P_{xy}}{|\bar{X}_0|} \Leftrightarrow \tag{5.5}$$

$$\frac{\sum_{y\notin\bar{X}_S}\frac{1}{|V|}P_{xy}}{|\bar{X}_S|\frac{1}{|V|}} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0}\frac{1}{|V|}P_{xy}}{|\bar{X}_0|\frac{1}{|V|}} \Leftrightarrow \tag{5.6}$$

$$\frac{\sum_{y\notin\bar{X}_S}\pi(x)P_{xy}}{\Pi(\bar{X}_S)} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0}\pi(x)P_{xy}}{\Pi(\bar{X}_0)} \Leftrightarrow \tag{5.7}$$

$$\Phi(\bar{X}_S) < \Phi(\bar{X}_0). \tag{5.8}$$

This result signifies that independently of the topology of the adversary region the conductance of a sub-graph containing Sybil nodes will be lower compared with the conductance of the sub-graph of nodes that are simply compromised and connected to the social network. Lower conductance in turn leads to slower mixing times between honest and dishonest regions [109] which means that $E_{XX} > 0$, even for very few Sybils. This deviation is subject to the sampling variation introduced by the trace T, but the error can be made arbitrarily small by sampling more random walks in T.

These two results are very strong: they indicate that, in theory, a set of compromised nodes connecting to honest nodes in a social network, would get no advantage by connecting any additional Sybil nodes, since that would lead to their detection. Sampling regions of the graph with abnormally small conductance, through the use of the random walks T, should lead to their discovery, which is the theoretical foundation of our technique. Furthermore we established that techniques based on detecting abnormalities in the value of E_{XX} are *strategy proof*, meaning that there is no attacker strategy (in terms of special adversary topology) to foil detection.

5.3.2 Practical considerations

Models and assumptions are always an approximation of the real world. As a result, careful evaluation is necessary to ensure that the theorems are robust to deviations from the ideal behaviour assumed so far.

The first practical issue concerns the fast mixing properties of social net-

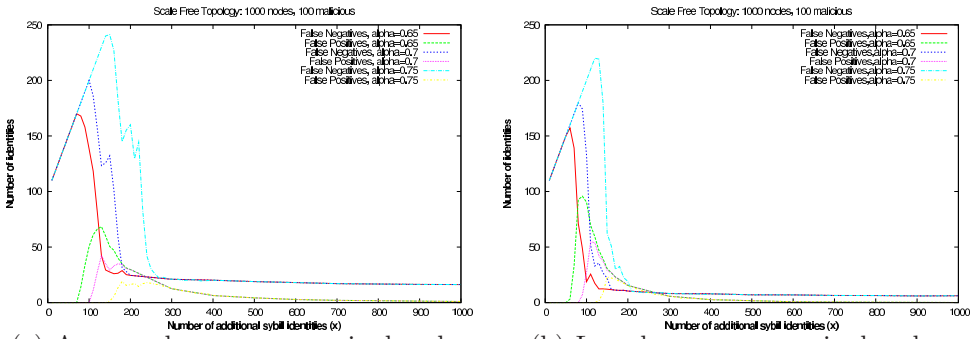
works. There is a lot of evidence that social networks exhibit this behaviour [58], and previous proposals relating to Sybil defence use and validate the same assumption [13, 73]. SybilInfer makes an further assumption, namely that the modified random walk over the social network, that yields a uniform distribution over all nodes, is also fast mixing for real social networks. The probability $P_{ij} = \min(\frac{1}{d_i}, \frac{1}{d_j})$, depends on the mutual degrees of the nodes i and j, and makes the transition to nodes of higher degree less likely. This effect has the potential to slow down mixing times in the honest case, particularly when there is a high variation in node degrees. This effect can be alleviated by removing random edges from high degree nodes to guarantee that the ratio of maximum and minimum node degree in the graph is bounded (an approach also used by SybilLimit).

The second consideration also relates to the fast mixing properties of networks. While in theory fast mixing networks should not exhibit any small cuts, or regions of abnormally low conductance, in practice they do. This is especially true for regions with new users that have not had the chance to connect to many others, as well as social networks that only contain users with particular characteristics (like interest, locality, or administrative groups). Those regions yield, even in the honest case, sample cuts that have the potential to be mistaken as attacks. This effect forces us to consider a threshold E_{XX} under which we consider cuts to be simply false positives. In turn this makes the guarantees of schemes weaker in practice than in theory, since the adversary can introduce Sybils into a region undetected, as long as the set threshold E_{XX} is not exceeded.

The threshold E_{XX} is chosen to be $\alpha \cdot E_{XXmax}$, where $E_{XXmax} = \frac{1}{|X|} - \frac{1}{|V|}$, and α is a constant between 0 and 1. Here α can be used to control the tradeoff between false positives and false negatives. A higher value of alpha enables the adversary to insert a larger number of sybils undetected but reduces the false positives. On the other hand, a smaller value of α reduces the number of Sybils that can be introduced undetected but at the cost of higher number of false positives.

Given these practical considerations, we can formulate a weaker security guarantee for SybilInfer:

> THEOREM C. Given a certain "natural" threshold value for E_{XX} in an honest social network, a dishonest region performing a Sybil attack



(a) Average degree compromised nodes (b) Low degree compromised nodes

Figure 5.3: Synthetic scale-free topology: SybilInfer evaluation as a function of additional Sybil identities (ψ) introduced by colluding entities. False negatives denote the total number of dishonest identities accepted by SybilInfer while false positives denote the number of honest nodes that are misclassified.

will exceed it after introducing a certain number of Sybil nodes.

This theorem is the result of Theorem B that demonstrates that the conductance keeps decreasing as the number of Sybils attached to a dishonest region increases. This in turn will slow down the mixing time between the honest and dishonest region, leading to an increasingly large E_{XX}.

Intuitively, as the attack becomes larger, the cut between honest and dishonest nodes becomes increasingly distinct, which makes Sybil detection easier. It is important to note that as more Sybils are introduced into the dishonest region, the probability of the whole region being detected as an attack increases, not only the new Sybil nodes. This provides strong disincentives to the adversary from performing larger Sybil attacks, since even previously undetected malicious nodes might be flagged as Sybils.

5.3.3 Experimental evaluation using synthetic data

We first experimentally demonstrate the validity of Theorem C using synthetic topologies. Our experiments consist of building synthetic social network topologies, injecting a variable number of Sybil nodes, and applying SybilInfer to establish how many of them are detected. A key issue we explore is the number of introduced Sybil nodes under which Sybil attacks are not detected.

Social networks exhibit a scale-free (or power law) node degree topology [112]. Our network synthesis algorithm replicates this structure through preferential attachment, following the methodology of Nagaraja [58]. We create m_0 initial nodes connected in a clique, and then for each new node v, we create m new edges to existing nodes, such that the probability of choosing any given node is proportional to the degree of that node; i.e.: $\Pr[(v,i)] = \frac{d_i}{\sum_j d_j}$, where d_i is the degree of node i. In our simulations, we use $m = 5$, giving an average node degree of 10.

In such a scale-free topology of 1000 nodes, we consider a fraction $f = 10\%$ of the nodes to be compromised by a single adversary. The compromised nodes are distributed uniformly at random in the topology. Compromised nodes introduce ψ additional Sybil nodes and establish a scale-free topology amongst themselves. We configure SybilInfer to use 20 samples for computing the marginal probabilities, and label as honest the set of nodes whose marginal probability of being honest is greater than 0.5. The experiment is repeated 100 times with different scale-free topologies.

Figure 5.3(a) illustrates the false positives and false negatives classifications returned by SybilInfer, for varying value of ψ, the number of additional Sybil nodes introduced. We observe that when $\psi < 100$, $\alpha = 0.7$, then all the malicious identities are classified as honest by SybilInfer. However, there is a threshold at $\psi = 100$, beyond which all of the Sybil identities, including the initially compromised entities are flagged as attackers. This is because beyond this point, the E_{XX} for the Sybil region exceeds the natural threshold leading to full detection, validating Theorem C. The value $\psi = 100$ is clearly the optimal attack strategy, in which the attacker can introduce the maximal number of Sybils without being detected. We also note that even in the worst case, the false positives are less than 5%. The false positive nodes have been misclassified because these nodes are closer to the Sybil region; SybilInfer is thus incentive compatible in the sense that nodes which have mostly honest friends are likely not to be misclassified.

We can also see the effect of varying the threshold E_{XX}. As α is increased from 0.65 to 0.7, the ψ for the optimal attacker strategy increases from 70 to 100. This is because an increase in the threshold E_{XX} allows the adversary to insert more Sybils undetected. The advantage of increasing α lies in reducing the worst case false positives. We can see that by increasing α from 0.7 to 0.75, the worst case false positives can be reduced from 5% to 2%.

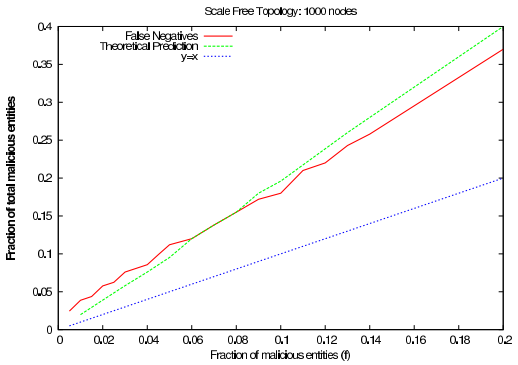


Figure 5.4: Scale-free topology: fraction of total malicious and Sybil identities as a function of real malicious entities.

Note that for the remainder of the paper, we shall use $\alpha = 0.7$.

We also wish to show that the security of our scheme depends primarily on the number of colluding malicious nodes and not on the number of attack edges. To this end, we chose the compromised nodes to have the lowest number of attack edges (instead of choosing them uniformly at random), and repeat the experiment. Figure 5.3(b) illustrates that the false positives and false negatives classifications returned by SybilInfer, where the average number of attack edges are 500. Note that these results are very similar to the previous case illustrated in Figure 5.3(a), where the number of attack edges is around 800. This analysis indicates that the security provided by SybilInfer primarily depends on the number of colluding entities. The implication is that the compromise of high degree nodes does not yield any significant advantage to the adversary. As we shall see, this is in contrast to SybilGuard and SybilLimit, which are extremely vulnerable when high degree nodes are compromised.

Our next experiment establishes the number of Sybil nodes that can be inserted into a network given different fractions of compromised nodes. We vary the fraction of compromised colluding nodes f, and for each value of f, we compute the optimal number of additional Sybil identities that the attackers can insert, as in the previous experiment.

Figure 5.4 presents a plot of the maximum Sybil identities as a function of the compromised fraction of nodes f. Note that our theoretical prediction (which is strategy-independent) matches closely with the attacker strategy of connecting Sybil nodes in a scale-free topology. The adversary is able to introduce roughly about 1 additional Sybil identity per real entity. For

instance, at $f = 0.2$, the total number of Sybil identities is 0.37. As we observe from the figure the ability of the adversary to just include about one additional Sybil identity per compromised node embedded in the social network remains constant, no matter the fraction f of compromised nodes in the network.

5.3.4 Experimental evaluation using real-world data

Next we validate the security guarantees provided by SybilInfer using a sampled LiveJournal topology. A variant of snowball [113] sampling was used to collect the full data set data, comprising over 100,000 nodes.

To perform our experiments we chose a random node and collect all nodes in its three hop neighbourhood. The resulting social network has about 50,000 nodes. We then perform some pre-processing step on the sub-graph:

- Nodes with degree less than 3 are removed, to filter out nodes that are too new to the social network, or inactive.
- If there is an edge between $A \to B$, but no edge between $B \to A$, then $A \to B$ is removed (to only keep the symmetric friendship relationships.)

We note that despite this pre-processing nodes all degrees can be found in the final dataset, since nodes with initial degree over 3 will have some edges removed reducing their degree to less than 3.

After pre-processing, the social sub-graph consists of about 33,000 nodes. First, we ran SybilInfer on this topology without introducing any artificial attack. We found a bottleneck cut diving off about $2,000$ Sybil nodes. It is impossible to establish whether these nodes are false positives (a rate of 6%) or a real-world Sybil attack present in the LiveJournal network. Since there is no way to establish ground truth, we do not label these nodes as either honest/dishonest.

Next, we consider a fraction f of the nodes to be compromised and compute the optimal attacker strategy, as in our experiments with synthetic data. Figure 5.5 shows the fraction of malicious identities accepted by SybilInfer as a function of fraction of malicious entitites in the system. The trend is similar to our observations on synthetic scale-free topologies. At $f = 0.2$, the fraction of Sybil identities accepted by SybilInfer is approximately 0.32.

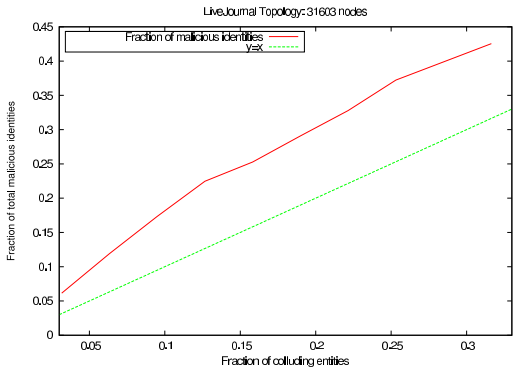


Figure 5.5: LiveJournal topology: fraction of total malicious identities as a function of real malicious entities.

5.3.5 Comparison with SybilLimit and SybilGuard

SybilGuard [13] and SybilLimit [73] are state of the art decentralized protocols that defend against Sybil attacks. Similar to SybilInfer, both protocols exploit the fact that a Sybil attack disrupts the fast mixing property of the social network topology, albeit in a heuristic fashion. A brief overview of the two systems can be found in the appendix, and their full descriptions is given in [13, 73].

Figure 5.6 compares the performance of SybilInfer with the performance of SybilLimit. First it is worth noting that the fraction of compromised nodes that SybilLimit tolerates is only a small fraction of the range within which SybilInfer provide its guarantees. SybilLimit tolerates up to $f = 0.02$ compromised nodes when the degree of attackers is low (about degree 5 – green line), while we have already shown the performance of SybilLimit for compromised fractions up to $f = 0.35$ in Figure 5.4. Within the interval SybilLimit is applicable, our system systematically outperforms: when very few compromised nodes are present in the system ($f = 0.01$) our system only allows them to control less than 5% of the entities in the system, versus SybilLimit that allows them to control over 30% of entities (rendering insecure Byzantine fault tolerance mechanisms that require at least 2/3 honest nodes). At the limit of SybilLimit's applicability range when $f = 0.02$, our approach caps the number of dishonest entities in the system to fewer than 8%, while SybilLimit allows about 50% dishonest entities. (This large fraction renders leader election or other voting systems ineffective.)

An important difference between SybilInfer and SybilLimit is that the former is not sensitive to the degree of the attacker nodes. SybilLimit provides

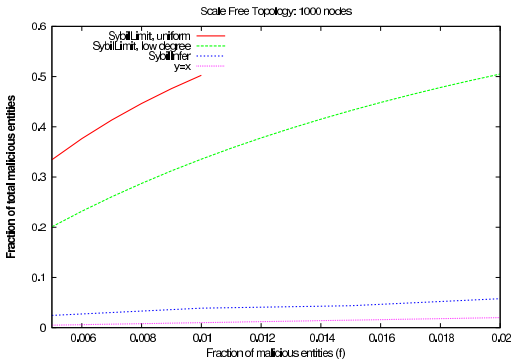


Figure 5.6: Comparison with related work.

very weak guarantees when high degree (e.g. degree 10 – red line) nodes are compromised, and can protect the system only for $f < 0.01$. In this case SybilInfer allows for 5% total malicious entities, while SybilLimit allows for over 50%.

This is an illustration that SybilInfer performs an order of magnitude better than the state of the art both in terms of range of applicability and performance within that range (SybilGuard's performance is strictly worse than SybilLimit's performance, and is not illustrated). An obvious question is: Why does SybilInfer perform so much better than SybilGuard and SybilLimit? It is particularly pertinent since all three systems are making use of the same assumptions, and a similar intuition, that there should be a "gap" between the honest and Sybil regions of a social network. The reason SybilLimit and SybilGuard provide weaker guarantees is that they interpret these assumptions in a very sensitive way: they assume that *an overwhelming majority* of random walks staring in the honest region will stay in the honest region, and then bound the number of walks originating from the Sybil region via the number of corrupt edges. As a result they are very sensitive to the length of those walks, and can only provide strong guarantees for a very small number of corrupt edges. Furthermore the validation procedure relies on collisions between honest nodes, via the birthday paradox, which adds a further layer of inefficiency to estimating good from bad regions.

SybilInfer, on the other hand, interprets the disruption in fast-mixing between the honest and dishonest region simply as a faint bias in the last node of a short random walk (as illustrated in Figures 5.2(a) and 5.2(b)). In our experiments, as well as in theory, we observe a very large fraction of the T walks crossing between the honest and dishonest regions. Yet the faint

difference in the probability of landing on nodes in the honest and dishonest regions is present, and the sampler makes use of it to get good cuts between the honest and dishonest nodes.

5.3.6 Computational and time complexity

Two implementations of the SybilInfer sampler were build in Python and C++, of about 1KLOC each. The Python implementation can handle 10K node networks, while the C++ implementation has handled up to 30K node networks, returning results in seconds.

The implementation strategy for both samplers has favoured a low time complexity over storage costs. The critical loop performs $O(|V| \cdot \log |V|)$ Metropolis Hastings iterations per sample returned, each only requiring about $O(\log |V|)$ operations. Two copies of the full state are stored, as well as associated data that allows for fast updating of the state, which requires $O(|V|)$ storage. The transcript of evidence traces T is also stored, as well as an index over it, which dominates the storage required and makes it order $O(|V| \cdot \log |V|)$.

There is a serious time complexity penalty associated with implementing non-naive sampling strategies. Our Python implementation biases the candidate moves towards nodes that are more or less likely to be part of the honest set. Yet exactly sampling nodes from this known probability distribution, naively may raise the cost of each iteration to be $O(|V|)$. Depending on the differential between the highest and lowest probabilities, faster sampling techniques like rejection sampling [114] can be used to bring the cost down. The Python implementation uses a variant of Metropolis-Hastings to implement selection of candidate nodes for the next move, at a computation cost of $O(\log |V|)$. The C++ implementation uses naive sampling from the honest or dishonest sets, and has a very low cost per iteration of order $O(1)$.

The Markov chain sampling techniques used consider sequences of states that are very close to each other, differing at most by a single node. This enables a key optimization, where the counts $N_{XX}, N_{X\bar{X}}, N_{\bar{X}X}$ and $N_{\bar{X}\bar{X}}$ are stored for each state and updated when the state changes. This simple variant of self-adjusting computation [115], allows for very fast computations of the probabilities associated with each state. Updating the counts, and associated

information is an order $O(\log |V|)$ operation. The alternative of recounting these quantities from T would cost $O(|V| \log |V|)$ for every iteration, leading to a total computational complexity for our algorithm of $O((|V| \log |V|)^2)$. Hence implementing it is vital to getting results fast.

Finally our implementations use a zero-copy strategy for the state. Two states and all associated information are maintained at any time, the current state and the candidate state. Operations on the candidate state can be done and undone in $O(\log |V|)$ per operation. Accepted moves can be committed to the current state at the same cost. These operations can be used to maintain the two states synchronised for use by the Metropolis-Hastings sampler. The naive strategy of re-writing the full state would cost $O(|V|)$ per iteration, making the overall complexity of the scheme $O(|V|^2 \log |V|)$.

5.4 Deployment Strategies

So far we presented an overview of the SybilInfer algorithm, as well as a theoretical and empirical evaluation of its performance when it comes to detecting Sybil nodes. The core of the algorithm outperforms SybilGuard and SybilLimit, and is applicable in settings beyond which the two systems provide no security guarantees whatsoever. Yet a key difference between the previous systems and SybilInfer is the latter's reliance on the full friendship graph to perform the random walks that drive the inference engine. In this section we discuss how this constraint still allows SybilInfer to be used for important classes of applications, as well as how it can be relaxed to accommodate peer-to-peer systems with limited resources per client.

5.4.1 Full social graph knowledge

The most straightforward way of applying SybilInfer is using the full graph of a social network to infer which nodes are honest and which nodes are Sybils, given a known honest seed node. This is applicable to centralised online services, like free email hosting services, blogging sites, and discussion forums that want to deter spammers. Today those systems use a mixture of CAPTCHA [116] and network based intrusion detection to eliminate mass attacks. SybilInfer could be used to either complement those mechanisms

and provide additional information as to which identities are suspicious, or replace those systems when they are expensive and error prone. One of the first social network based Sybil defence systems, Advogato [102], worked in such a centralized fashion.

The need to know the social graph does not preclude the use of SybilInfer in distributed or even peer-to-peer systems. Social networks, once mature, are generally stable and do not change much over time. Their rate of change is by no means comparable to the churn of nodes in the network, and as a result the structure of the social network could be stored and used to perform inference on multiple nodes in a network, along with a mechanisms to share occasional updates. The storage overhead for storing large social networks is surprisingly low: a large social network with 10 billion nodes (roughly the population of planet earth) with each node having about 1000 friends, can be stored in about 187Gb of disk space uncompressed. In such settings it is likely that SybilInfer computation will be the bottleneck, rather than storage of the graph, for the foreseeable future.

A key application of Sybil defences is to ensure that volunteer relays in anonymous communication networks belong to independent entities, and are not controlled by a single adversary. Practical systems like Mixmaster [16], Mixminion [15] and Tor [3] operate such a volunteer based anonymity infrastructure, that are very susceptible to Sybil attacks. Extending such an infrastructure to use SybilInfer is an easy task: each relay in the system would have to indicate to the central directory services which other nodes it considers honest and non-colluding. The graph of nodes and mutual trust relations can be used to run SybilInfer centrally by the directory service, or by each individual node that wishes to use the anonymizing service. Currently, the largest of those services, the Tor network has about 2000 nodes, which is well within the computation capabilities of our implementations.

5.4.2 Partial social graph knowledge

SybilInfer can be used to detect and prevent Sybil attacks, using only a partial view of the social graph. In the context of a distributed or peer-to-peer system, each user discovers only a fixed diameter sub-graph around them. For example a user may choose to retrieve and store all other users

two or three hops away in the social network graph, or discover a certain threshold of nodes in a breadth first manner. SybilInfer is then applied on the extracted sub-graph to detect potential Sybil regions. This allows the user to prune its social neighbourhood from any Sybil attacks, and is sufficient for selecting a set of honest nodes when sampling from the full network is not required. Distributed backup and storage, and all friend and friend-of-friend based sharing protocols, can benefit from such protection. The storage and communication cost of this scheme is constant and relative to the number of nodes in the chosen neighbourhood.

In cases where nodes can afford to know a larger fraction of the social graph, they could choose to discover $O(c \cdot \sqrt{|V|})$ nodes in their neighbourhood, for some small integer c. This increases the chances two arbitrary nodes have to know a common node, that can perform the SybilInfer protocol and act as an introduction point for the nodes. In this protocol Alice and Bob want to ensure the other party is not a Sybil. They find a node C that is in the $c \cdot \sqrt{|V}$ neighbourhood of both of them, and each make sure that with high probability it is honest. They then contact node C that attests to both of them, given its local run of the SybilInfer engine, that they are not Sybil nodes (with C as the honest seed). This protocol introduces a single layer of transitive trust, and therefore it is necessary for Alice and Bob to be quite certain that C is indeed honest. Its storage and communication cost is $O(\sqrt{|V|})$, which is the same order of magnitude as SybilLimit and SybilGuard. Modifying this simple minded protocol into a fully fledged one-hop distributed hash table [117] is an interesting challenge for future work.

SybilInfer can also be applied to specific on-line communities. In such cases a set of nodes belonging to a certain community of interest (a social club, a committee, a town, etc.) can be extracted to form a sub-graph. SybilInfer can then be applied on this partial view of the graph, to detect nodes that are less well integrated than others in the group. There is an important distinction between using SybilInfer in this mode or using it with the full graph: while the results using the full graph output an "absolute" probability for each node being a Sybil, applying SybilInfer to a partial view of the network only provides a "relative" probability the node is honest *in that context.* It is likely that nodes are tagged as Sybils, because they do not have many contacts within the select group, which given the full graph would be classified as honest. Before applying SybilInfer in this mode it is

important to assess, at least, whether the subgroup is fast-mixing or not.

5.4.3 Using SybilInfer output optimally

Unlike previous systems the output of the SybilInfer algorithm is a probabilistic statement, or even more generally, a set of samples that allows probabilistic statements to be estimated. So far in the work, we discussed how to make inferences about the marginal probability that specific nodes are honest of dishonest by using the returned samples to compute Pr[i is honest] for all nodes i. In our experiments we applied a 0.5 threshold to the probability to classify nodes as honest or dishonest. This is a rather limited use of the richer output that SybilInfer provides.

Distributed system applications can, instead of using marginal probabilities of individual nodes, estimate the probability that the particular security guarantees they require hold. High latency anonymous communication systems, for example, require a set of different nodes such that with high probability at least one of them is honest. Path selection is also subject to other constraints (like latency). In this case the samples returned by SybilInfer can be used to calculate exactly the sought probability, i.e. the probability a single node in the chosen path is honest. Onion routing based systems, on the other hand are secure as long as the first and last hop of the relayed communication is honest. As before, the samples returned by SybilInfer can be used to choose a path that has a high probability to exhibit this characteristic.

Other distributed applications, like peer-to-peer storage and retrieval, have similar needs, but also tunable parameters that depend on the probability of a node being dishonest. Storage systems like OceanStore use Rabin's information dispersion algorithm to divide files into chunks stored and retrieved to reconstruct a file. The degree of redundancy required crucially depends on the probability nodes are compromised. Such algorithms can use SybilInfer to foil Sybil attacks, and calculate the probability that the set of nodes to be used to store particular files contains certain fractions of honest nodes. This probability can in turn inform the choice of parameters to maximise the survivability of the files.

Finally, a note of warning should accompany any Sybil prevention scheme: it is not the goal of SybilInfer (or any other such scheme) to ensure that

all adversary nodes are filtered out of the network. The job of SybilInfer is to ensure that a certain fraction of existing adversary nodes cannot significantly increase its control of the system by introducing 'fake' Sybil identities. As illustrated by the examples on anonymous communications and storage, system-specific mechanisms are still crucial to ensure that a minority of adversary entities cannot compromise any security properties. SybilInfer can only ensure that this minority remains a minority and cannot artificially increase its share of the network.

Sybil defence schemes are also bound to contain false-positives, namely, honest nodes labeled as Sybils. For this reason other mechanisms need to be in place to ensure that those users can seek a remedy to the automatic classification they suffered from the system, potentially by making some additional effort. Proofs-of-work, social introduction services, or even payment targeting those users could be a way of ensuring SybilInfer is not turned into an automated social exclusion mechanism.

5.5 Summary

We presented SybilInfer, an algorithm aimed at detecting Sybil attacks against peer-to-peer networks or open services, and labeling which nodes are honest and which are dishonest. Its applicability and performance in this task are an order of magnitude better than previous systems making similar assumptions, like SybilGuard and SybilLimit, even though it requires nodes to know a substantial part of the social structure within which honest nodes are embedded.

CHAPTER 6

X-VINE: SECURE AND PSEUDONYMOUS ROUTING USING SOCIAL NETWORKS

Securing DHTs has always been a challenging task [62, 63, 65], especially in the face of a Sybil attack [8], where one node can pretend to have multiple identities and thus interfere with routing operations. Traditional solutions to this attack require participants to obtain certificates [65], prove possession of a unique IP address [9,118], or perform some computation [71]. This creates a barrier to participation, limiting the growth of the P2P user base, and at the same time does not fully address the problem of Sybil attacks.

To address this, recent research proposes to use social network trust relationships to mitigate Sybil attacks [13,73,119]. However, these systems share some key shortcomings:

High control overhead: These systems rely on flooding or large numbers of repeated lookups to maintain state. For example, Whanau [1] is the state-of-art design that secures routing in DHTs, but it is built upon a *one-hop* DHT routing mechanism, and has high overheads: state and control overhead increases with $O(\sqrt{n}\log n)$, where n is the number of participants in the social network. As networked systems become increasingly deployed at scale (e.g., in the wide area, across service providers), in high-churn environments (e.g., developing regions, wireless, mobile social networks [120]), and for applications with stronger demands on correctness and availability (e.g., online storage, content voting, reputation systems), the problem of high overhead in existing works stands to become increasingly serious; multi-hop DHT routing mechanisms are going to be necessary.

Lack of privacy: These systems require a user to reveal social contact information (friend lists). Some of these schemes require global distribution of this contact information. This is unfortunate, as social contacts are considered to be private information: leading real-world systems like Facebook [105] and LiveJournal [121] provide users with a functionality to limit access to this information. Forcing users to reveal this private information could greatly

hinder the adoption of these technologies.

A second privacy concern, common to both traditional DHTs and those that use social networking information, is that users must communicate directly with random peers, revealing their IP addresses. This provides an opportunity for the attacker to perform traffic analysis and compromise user privacy [122, 123]. Prior work [12, 67]has demonstrated that a colluding adversary can associate a DHT lookup with its lookup initiator, and thus infer the activities of a user. A *pseudonymous* routing mechanism can defend against such attacks, and would be especially beneficial for privacy sensitive DHT applications [20, 118].

To address these shortcomings, we propose X-Vine, a protection mechanism for large-scale distributed systems that leverages social network trust relationships. X-Vine has several unique properties. X-Vine protects *privacy* of social relationships by ensuring that a user's relationship information is revealed only to the user's immediate friends. At the same time, X-Vine also protects *correctness* of DHT routing, by mitigating Sybil attacks while requiring only logarithmic state and control overhead. To the best of our knowledge, X-Vine is the first system to provide both properties, which may serve to make it a useful building block in constructing the next generation of social network based distributed systems. Finally, X-Vine also provides a basis for pseudonymous communication; a user's IP address is revealed only to his/her trusted social network contacts.

X-Vine achieves these properties by incorporating social network trust relationships in the DHT design. Unlike traditional DHTs, which route directly between overlay participants (e.g., [1]), X-Vine embeds the DHT directly into the social fabric, allowing communication through the DHT to leverage trust relationships implied by social network links. This is done by using mechanisms similar to network layer DHTs like VRR [14]. We leverage this structure for two purposes. First, communication in X-Vine is carried out entirely across social-network links. The use of social network links enables pseudonymous communication; while the recipient may know the opaque identifier (pseudonym) for the source, the IP address of the source is revealed only to his/her friends. Second, recent work has shown that social networks can be used to detect Sybil attacks by identifying a bottleneck cut that connects the Sybil identities to the rest of the network [13, 73, 119]. X-Vine enables comparable Sybil resilience by bounding the number of DHT

relationships that can traverse a particular edge. With this multi-hop approach, we can limit the number of Sybil identities per attack edge (attack edges illustrated in Figure 6.1) to logarithmic in the size of the network with logarithmic control and routing state, a dramatic reduction from previous Sybil defense approaches. This allows X-Vine to scale to large user bases and high-churn environments.

We evaluate X-Vine both analytically and experimentally using large scale real-world social network topologies. Since recent work [124, 125] has advocated the use of interaction networks as a more secure realization of social trust, we also demonstrate the performance of X-Vine on interaction graphs. From our evaluation, we find that X-Vine is able to route using 10–15 hops (comparable to other DHTs) in topologies with 100 000 nodes while using only $O(\log n)$ routing state. In particular, we show that the overhead of X-Vine is two orders of magnitude smaller than Whanau. With respect to Sybil resistance, we found that honest nodes are able to securely route to each other with probability greater than 0.98 as long as the number of attack edges is $g \in o(n/(\log n))$. Using an implementation on PlanetLab, we estimate the median lookup latency in a 100 000 node topology to be less than 1.2 seconds. Even when 20% of the nodes fail simultaneously, the lookups still succeed with a probability greater than 95%. Finally, we also implement a plugin for DHT designers that can enable them to easily integrate social network contacts with a DHT by leveraging existing online social networks like Facebook.

Our proposed techniques can be applied in a wide variety of scenarios that leverage DHTs:

- Large scale P2P networks like Vuze/Kad/Overnet are popularly used for file sharing and content distribution. However, these networks are vulnerable to attacks on the routing protocol [126] and do not protect the privacy of the user [12]. X-Vine protects against attacks that target the DHT mechanisms and provides a basis for pseudonymous communication. Moreover, X-Vine is also robust to the high churn prevalent in these networks.

- Applications like Coral [127], Adeona [128], and Vanish [129] are built on top of DHTs. The security properties of these applications can often be compromised by exploiting vulnerabilities in the DHT. As an

example, the security of Vanish was recently compromised by a low-cost Sybil attack on the Vuze network [130]. Our proposed techniques protect these applications by bounding the number of Sybil identities in the DHT.

- Decentralized P2P anonymous communication systems like Tarzan [55], Salsa [9] and ShadowWalker [118] assume an external Sybil defense mechanism. X-Vine is particularly suitable for designing Sybil-resilient P2P anonymous communication systems, since it provides secure as well as pseudonymous routing.

- Freenet [20] is a widely used censorship resistant overlay network, but its routing algorithm has been shown to be extremely vulnerable in presence of even a few malicious nodes [84]. X-Vine can enable peers to resist censorship by securely and pseudonymously retrieving data objects from the Freenet network.

- Membership concealing overlay networks (MCONs) [85] hide the identities of the peers participating in a network (different from pseudonymity). Our proposed techniques can provide a substrate for designing fully decentralized membership concealing networks.

This chapter describes and evaluates X-Vine. We start by giving a high-level overview of the problem we address and our approach (Section 6.1), followed by a detailed description of our routing algorithm (Section 6.2) and its security mechanisms (Section 6.3). We then describe our experimental results (Section 6.4). Finally, we discuss X-Vine's limitations (Section 6.5), and summarize (Section 6.6).

6.1 X-Vine Overview

6.1.1 Design goals

We start by defining the goals for our design.

1. Secure routing: if an honest node X performs a lookup for an identifier ID, then the lookup mechanism must return the global successor of ID

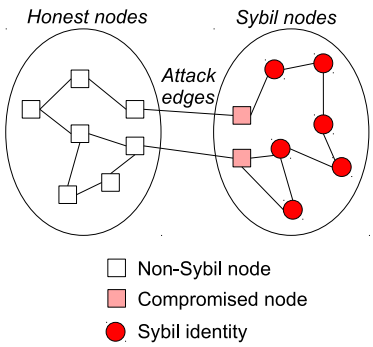


Figure 6.1: Illustration of honest nodes, Sybil nodes, and attack edges between them.

(present in the routing tables of honest nodes).

2. Pseudonymous communication: an adversary should not be able to determine the IP address corresponding to a user.

3. Privacy of user relationships: an adversary should not be able to infer a user's social contacts.

4. Low control overhead: the control overhead of the system should be small to enable a scalable design. This excludes flooding-based and single-hop mechanisms.

5. Low latency: the length of the path used to route to an arbitrary identifier should be small, in order to minimize lookup latency.

6. Churn resilience: even when a significant fraction of nodes fail simultaneously, lookup queries should still succeed.

7. Fully decentralized design: we target a fully decentralized architecture without any central points of trust/failure.

We note that requirements 2, 3 and 4 distinguish us from prior work—state-of-the-art approaches do not provide pseudonymous routing, do not preserve privacy of user relationships, and have high control overhead.

6.1.2 Threat model and assumptions

We assume that a fraction of real users are compromised and colluding. Recent work [13, 73, 119] has leveraged the insight that it is costly for an attacker to establish many trust relationships. Following this reasoning, we assume that the number of attack edges, denoted by g, is bounded. Similar

to prior work, we assume that the attack edges are not specially chosen.

Recent work has challenged the assumption that it is costly for an attacker to create attack edges with honest nodes in friendship graphs [124, 131–133], and proposed the use of interaction graphs as a more secure realization of real world social trust. In this work, we will evaluate X-Vine with both traditional friendship graphs as well as topologies based on interaction graphs. Other mechanisms to infer the strength of ties between user [134] may also be helpful in creating resilient social graphs, but are not the focus of this paper.

We also assume that the set of colluding compromised nodes is a Byzantine adversary, and can deviate from the protocol in arbitrary ways by launching active attacks on the routing protocol. In particular, the set of compromised nodes can launch a Sybil attack by trying to insert multiple fake identities in the system. The key assumption we make about the adversary is that Sybil identities are distributed randomly in the DHT identifier space. We note that this assumption is a limitation of the X-Vine protocol, as discussed in Section 6.5. An exploration of defenses against adversaries who concentrate their nodes in a particular region of the overlay is beyond the scope of this paper.

6.1.3 Solution overview

We start by describing our algorithm in the context of an abstract, static network. Suppose we have a graph G, where nodes correspond to users of the social network, and edges correspond to social relationships between them. Our approach runs a DHT-based routing scheme over the graph that embeds path information in the network. We first describe how routing is performed, and then describe how the path information it creates can be used to mitigate Sybil attackers.

Pseudonymous routing in the social network: We construct a DHT on top of the social network, using mechanisms similar to network layer DHTs [14]. Each node in the network selects a random numeric identifier, and maintains paths (*trails*) to its neighbors in the identifier space in a DHT-like fashion. To join the network, a node performs a discovery process to determine a path to its *successors* in the DHT. Then, the node embeds trails in the network that point back to the joining node's identifier. To route

messages, packets are forwarded along these trails. By maintaining trails to each of the node's successors, a node can forward a message to any point in the namespace. Users that are directly connected by a social network link simply communicate via the IP layer. All communication performed by a node is done only with its friends, and this communication is done in a manner that does not reveal the node's local topology, preventing leakage of friendship list information to non-friends. Routing over social links also enables a user to communicate pseudonymous with respect to non-friends.

Protecting against Sybils: The scheme described above does not mitigate the Sybil attack, as a malicious node can repeatedly join with different identifiers, and thereby "take over" a large portion of the identifier space. Malicious nodes can in fact pretend that there is an entire network of Sybil nodes behind themselves (Figure 6.1). To protect against the Sybil attack, we constrain the number of paths between honest nodes and malicious nodes. Since Sybil nodes by their very nature are likely to be behind a small "cut" in the graph, by constraining the number of paths that may be set up, we can constrain the impact that a Sybil node can have on the entire network. In particular, honest nodes rate-limit the number of paths that are allowed to be constructed over their adjacent links, thereby limiting the ability of Sybil nodes to join the routing scheme, and hence participate in the network. When a joining node attempts to establish a trail over an edge that has reached its limit, the node adjacent to the full link sends the joining node a message indicating failure of the join request. This limits Sybil nodes from constructing very many paths into the network. Since Sybil nodes cannot construct many trails, they cannot place many identifiers into the DHT. Hence, an honest node can send traffic to another honest node by forwarding traffic over the DHT, as trails are unlikely to traverse Sybil-generated regions of the network.

6.2 X-Vine Protocol

The key feature of our design is that all DHT communication happens over social network links.[1] By ensuring that all communication takes place over

[1]Applications such as Vuze may optionally choose to benefit only from X-Vine's Sybil resilience, and can forgo pseudonymity by directly transferring files between overlay nodes

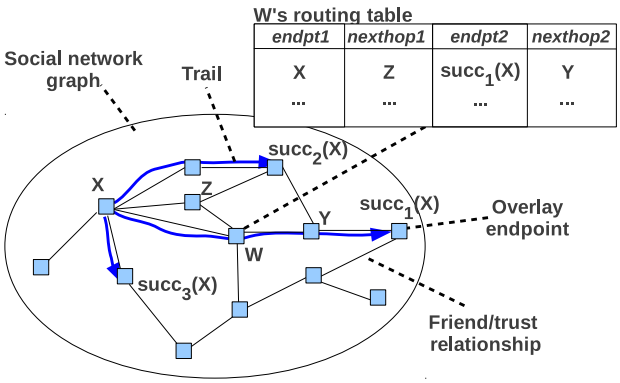


Figure 6.2: Overview of X-Vine.

social network links, we can leverage the trust relationships in the social network topology to enforce security properties. A trivial security property of our design is that an adversary needs to be connected to honest users via a series of social network links to communicate with them. Moreover, the IP address of the nodes only needs to be revealed to their contacts, enhancing privacy for users. Most importantly, our design is able to effectively resist Sybil attacks even when the number of attack edges is large.

6.2.1 Routing over social networks

Figure 6.2 illustrates the design of X-Vine. Our design uses a VRR-like [14] protocol to construct and maintain state at the overlay layer. Here, we first describe the state maintained by each node, and then describe how that state is constructed and maintained over time.

State maintained by each node: X-Vine constructs a *social overlay* on top of the social network, where a node has direct links to friends, but also maintains "overlay links" to remote nodes. These remote nodes (*overlay endpoints*) are selected in a manner similar to Chord [60]: each node is assigned an identifier from a ring namespace, and forms overlay links to nodes that are successors (immediately adjacent in the namespace), and (optionally) fingers (spaced exponentially around the ring). Unlike Chord however, a node is not allowed to directly send packets to its overlay neighbor: for security reasons, nodes only receive packets from their social network links. Hence, to forward a packet from a node to one of its overlay endpoints, the packet will have to

after the lookup operation.

traverse a *path* in the social network graph. To achieve this, corresponding to each of its overlay endpoints, a node maintains a *trail* through the social network. Each node along the trail locally stores a *record* consisting of four fields: the identifiers of the two endpoints of the trail, and the IP addresses of the next and previous hops along the trail. Using this information, a node can send a packet to its endpoints, by handing the packet off to the first node along the trail, which looks up the next hop along the trail using its trail records, and so on. Furthermore, using a Chord-like routing algorithm, a node can route to any other node in the namespace, by (upon reaching an endpoint) selecting the next overlay hop that maximizes namespace progress to the destination (without overshooting). As an optimization, instead of waiting until the endpoint is reached to determine the next overlay hop, intermediate nodes along the path may "*shortcut*" by scanning all their trail records, and choosing the endpoint that maximizes progress in the namespace (see Algorithm 1 in Appendix B). If the intermediate node discovers an endpoint that makes more namespace progress to the destination than the current next overlay hop, the intermediate node may choose to forward the packet towards this new endpoint, to speed its progress (while explicitly maintaining the next overlay hop in the packet is not strictly necessary for routing, we do so to simplify parts of our design described later).

State construction and maintenance: Since nodes can route, we can perform other DHT operations by simply performing routing atop this structure. For example, we can execute a Chord-like join: upon arriving at the network, a node can route a *join request* towards its own identifier, and the node that receives it can return back the identifiers which should be the joining node's successors. However, there are two key changes we need to make. First, when a node initially arrives, it does not yet have any trail state and hence cannot forward packets. To address this, the joining node randomly selects one of its friends in the social network to act as a *bootstrap* node. The joining node sends its join request using the bootstrap node as a proxy. Second, the joining node also needs to build trails to each of its endpoints (e.g., its successors). To do this, for each endpoint, it sends a *trail construction request* to the identifier of that endpoint. As the request is routed, each intermediate node along the path locally stores a record corresponding to the trail. Finally, when these steps are completed, the joining node can route to

any node in the network (by forwarding packets through its endpoints), and it can receive packets from any node in the network (by accepting packets through its endpoints). To maintain this state, we need to achieve two things. First, we would like to correctly maintain the set of records along each trail in the presence of churn, so each node can reach the trail endpoint. This is done in a manner similar to AODV [135]: each node along the path locally probes its neighbors and removes trail records (sending *teardown* messages upstream if necessary) corresponding to failed trails. Second, we would like to make sure each trail points to the corresponding globally correct successor/finger. To do this, we leverage the stabilization mechanisms from Chord and VRR [14, 60].

6.2.2 Balancing routing state

Temporal correlation: while the scheme above is correct, it performs poorly in practice. The reason for this is due to *temporal correlation*—since trails are constructed using other trails, social network links that are initially chosen to be part of a trail become increasingly likely to be part of later trails. Because of this, nodes that join the network early tend to accumulate more state over time. To illustrate this problem, we describe an example. Suppose a node X has d friends $a_1, a_2, .., a_d$. Suppose also that there is a trail from X to Y for which the next hop is node a_d. Next, suppose node X is an intermediate node in a new overlay path that is being setup from node a_1 (which is also the previous hop). With probability $2/d$, the next hop of the overlay path will be a_d. Similarly, in the future, the probability of a_d being chosen as the next hop in an overlay path increases to $3/(d+1)$, and then to $4/(d+2)$, and so on. This example illustrates that a social network link that was initially chosen as part of a trail has an increasing chance of being chosen in trails that are set up in the future. Consequently nodes that join the social network early tend to be part of many trails. This is not desirable from both a security perspective or a performance perspective.

Stabilization algorithms: To address the problem of temporal correlation, we propose two modifications to the core X-Vine algorithms: The first algorithm leverages the social connections of new users to reduce the path lengths of existing trails. When a new node joins the system, its social con-

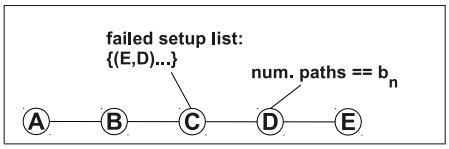


Figure 6.3: Example: backtracking.

tacts that are already part of the X-Vine system consider all trails in their routing tables that have a path length greater than a threshold thr_1 (set to the upper quartile of trail path path lengths). Corresponding to each such trail, the social contacts check if modifying the trail via the new node would reduce the path length, and if so, a teardown message is sent to the old trail and another trail via the new node is setup. The threshold on the path length helps to avoid needless communication for trails that are already short, and are thus unlikely to benefit much from new edges in the social graph topology. The second algorithm helps to load balance the routing state at nodes, and also leads to a reduction in the path lengths of trails. This algorithm is run by all nodes whose routing state is greater than a threshold thr_2. Such nodes consider all trails in their routing tables whose path length is greater than a threshold thr_1 (similar to the previous algorithm), and send messages to the overlay end points to check if alternate trails can be established, and if their path length is shorter than the current path length. If a shorter alternate trail exists, then it replaces the existing trail. This helps reduce the routing state size at congested nodes, while simultaneously reducing the trail path lengths.

6.2.3 Bounding state with local policies

We have seen that the shortcut-based routing protocol described in Section 6.2.1 faces the problem of temporal correlation, leading to unbounded growth in routing state. To complement our stabilization algorithms, we propose a mechanism by which nodes can set a hard bound on their routing state size using local routing policies. These policies can be set to account for heterogeneity across nodes, users' desired degree of participation in the network, and to limit the worst-case state overhead at any particular node. Our architecture allows users to set two types of policies pertaining to state maintained at their local node:*Bounding routes per link:* If the number of

trails traversing an adjacent social network link reaches a threshold b_l, then the user's node refuses to set up any more trails traversing that link. *Bounding routes per node:* If the number of trails traversing the user's node reaches a threshold value b_n, then the node refuses to set up any more trails via itself. Due to these routing policies, it is possible that a request to set up a trail may be unable to make forward progress in the overlay namespace. To address this, we introduce a technique called *backtracking* that explores alternate social network paths in case an intermediate node refuses to process a path setup request. To do this, each node maintains a *failed setup list*, containing a list of trails that have failed to set up. When a node attempts to set up a trail via a next hop, and receives a *rejection* message indicating that the next hop is full, the node inserts an entry into its failed setup list. Each record in the list contains the identifier of the destination overlay endpoint that the packet was traversing towards, and the identifier of the next hop in the social network that rejected the path setup. When forwarding a message to a particular destination endpoint, a node removes from consideration next hops contained in the failed setup list corresponding to that endpoint (see Algorithm 2 in Appendix B). The failed setup list is periodically garbage collected by discarding entries after a timeout.

For example (Figure 6.3), suppose node A wishes to establish a path to E, and determines B is the best next overlay hop. A places E into the *next overlay hop* field in the message, and forwards the message to B. Similarly, B forwards the message to C. Suppose D is congested (has more than b_n paths traversing it). In this case, C sends the path setup message to D, but D responds back with a rejection message. C then stores the entry (E, D) in its failed setup list, to indicate that establishing a path via D to reach E was unsuccessful. C then attempts to select an alternate next hop that makes progress in the namespace (either a route to the current next overlay hop, or a "shortcut" route that makes more progress than the current next overlay hop). If C does not have such a route, it sends a rejection message back to B, which inserts the entry (E, C) in its failed setup list. This process repeats until a path is discovered, or a time-to-live (TTL) contained in the packet is exceeded. When the TTL is exceeded, the path setup fails, and the source node must attempt to rejoin to establish the path.

6.3 Securing X-Vine

The previous section described our approach to perform routing atop the social network. In this section, we describe how to extend and tune the design in the previous section to improve its resilience to attack. We start by providing an overview of attacks on our design(Section 6.3.1), and then propose extensions to improve resilience to them (Section 6.3.2).

6.3.1 Attacks on the routing protocol

We investigate defenses to the following attacks on DHTs:

Sybil attack [8]: The attacker can insert a large number of Sybil identities in the DHT, and set up paths with their successors and predecessors. The attack results in honest nodes' routing tables being populated by malicious Sybil identities. This increases the probability that lookup queries will traverse some malicious nodes, which can then drop or misroute the lookup queries. Observe that to minimize resources, it suffices for Sybil identities to maintain paths with only nodes in the predecessor list, since paths to the nodes in the successor list will result in a shortcut to the honest successor nodes.

Attacks on routing table maintenance: In addition to the Sybil attack, the adversary could also manipulate the routing table maintenance protocols to increase the probability of malicious nodes being present in honest nodes' routing tables. *Intercepting trails:* During churn, malicious nodes can become part of a large number of trail paths between nodes, in order to attract lookup traffic (for example, by refusing to send trail teardown messages). *Attacking trail construction:* The attacker could prevent honest nodes from finding a trail path to their correct successor. This could be done by dropping or misrouting the trail setup messages. *Attacks on message integrity:* Malicious nodes that forward control traffic could modify the content of the messages, to disrupt trail setup (for example, by creating routing loops). *Forgery attacks:* The malicious nodes could spoof source identifiers in messages sent to honest nodes (for example, to give the appearance that the message came from the honest node's friends).

Attacks on lookups: Once the attacker is able to intercept a lookup query, it can either drop the packet or misroute it. Such attacks can prevent the honest nodes from either discovering their correct successor in the ring, or discovering a malicious successor list set respectively. By advertising malicious nodes as the successors of an honest joining node, a significant fraction of the honest joining node's traffic would traverse malicious nodes. Note that attacks on both overlay construction and overlay routing are captured by this attack, since in a DHT, both bootstrap and routing are accomplished by the same operation: a lookup.

6.3.2 Proposed defenses

We note that it is imperative to secure *both* the routing table maintenance and lookup forwarding. If the routing table maintenance protocol were insecure, then the adversary could manipulate the routing table entries of honest nodes to point to malicious nodes, and routing to honest nodes would not be successful. However, even if the routing table maintenance mechanisms are secure, the adversary still has the opportunity to drop lookup packets or misroute them.

Mitigating the Sybil attack: To limit the impact of the Sybil attack, we propose that nodes implement a routing policy that bounds the number of trails that traverse a social network edge. We denote the bound parameter as b_l. Since the attacker has limited attack edges, this bounds the number of overlay paths between the honest subgraph and the Sybil subgraph *regardless of the attacker strategy*. Thus, we limit the number of Sybil identities that are part of the honest node's routing table. The key challenge in this approach is to determine the bound b_l that enables most honest nodes to set up trails with each other while hindering the ability of Sybil nodes to join the DHT. Our analytic and experimental results suggest that a bound of $b_l \in \Theta(\log n)$ works quite well. Similar to Yu et al. [73], we assume that the bound b_l is a system wide constant known to all honest nodes. Honest nodes are able to set up trails with each other even though there is a bound on the number of trails per social network link because of the fast-mixing nature of the social network. On the other hand, a Sybil attack gives rise to a sparse cut in the social network topology, and we use this sparse cut to limit the impact of the

Sybil identities. The number of overlay paths between the honest and Sybil subgraphs is bounded to $g \cdot b_l$. The adversary could choose to allocate each overlay path to a different Sybil identity, resulting in $g \cdot b_l$ Sybil identities in the DHT (in the routing tables of honest nodes). We can further limit the number of Sybil identities in the routing tables of honest nodes by ensuring that the adversary must allocate at least a threshold t number of overlay paths per Sybil identity. This would bound the number of Sybil identities in honest nodes routing tables to $g \cdot b_l/t$. Note that the number of overlay paths between the honest and Sybil regions does not change. We propose the following mechanism to ensure that the adversary sets up trails with at least a threshold t overlay neighbors. Nodes periodically probe their overlay neighbors to check if each successor in their routing table has set up a trail with at least t other nodes in the overlay neighborhood. Note that the check is performed by directly querying the overlay neighbors. The threshold t is set to $t < 2 \cdot num\_successors$ to account for malicious overlay nodes returning incorrect replies. If the adversary does not allocate t trails per Sybil identity (set up with its successors and predecessors), the honest nodes can detect this via probing and can teardown the trails to the malicious Sybil identity. Note that the adversary cannot game the probing mechanism unless it has a large number of Sybil identities in the overlay neighborhood of a node. Since the Sybil identities are distributed at random in the overlay namespace, this is unlikely to happen unless the adversary has a large number of attack edges ($g \in \Omega(n/(\log n))$).

Securing routing table maintenance: We provide the following defenses to attacks on routing table maintenance:

Trail interception attacks: Observe that our mechanism to defend against Sybil attacks, i.e., bounding the number of trails that traverse a social network link, also defends against malicious nodes that attempt to be a part of a large number of trails. Specifically, the adversary has a *quota* of $g \cdot b_l$ trails between honest nodes and itself, and it can choose to utilize this quota either by inserting Sybil identities in the DHT or by being part of trails between two honest nodes. Either way, the effect of this attack is limited by the bound b_l.

Trail construction attacks: Suppose that a node X is trying to set up a trail with its overlay neighbor Y. To circumvent the scenario where a malicious intermediate node M simply drops X's path set up request to Y, we propose

that upon path setup the end point Y sends an acknowledgment along the reverse path back to X. If after a timeout, the node X does not receive an acknowledgment from Y, then it can retry sending the trail set up request over a different route. Again, the fast-mixing nature of the social network topology guarantees that two nodes are very likely to have multiple paths between each other.

Message integrity and forgery attacks: To provide message integrity is the use of self-certifying identifiers [136–138]. Nodes can append their public keys to the message and produce a digital signature of the message along with the appended public key. The self-certifying nature of identifiers ensures that the public key for a specified node identifier cannot be forged; this enables us to provide both message integrity as well as authentication.

Securing the lookup protocol: Even if the routing table maintenance protocol is secure, the adversary can still drop or misroute lookup requests that traverse itself. We secure the lookup protocol using redundant routing, similar to Castro et al. [65]. Instead of a single lookup, a node can choose to perform r lookups for the destination (where r is the redundancy parameter) using r diverse trusted links in the social network topology. Redundant routing increases the probability that at least one lookup will traverse only honest nodes and find the correct successor. If the lookup is performed during route table maintenance, the correct successor can be identified since it will be impossible to set up a trail to an incorrect one; if the lookup is searching for a particular node or data item, then self-certifying techniques can be used to identify incorrect responses.

6.3.3 Privacy protection

All communication in X-Vine happens over social network links; while a user's IP address is revealed to his/her social contacts, it is not exposed to random peers in the network. Therefore as long as a user's social contacts are trusted, he/she can communicate pseudonymously. Moreover, observe that X-Vine's mechanisms do not require a user to expose his/her social contacts. This is in sharp contrast to prior work [1], wherein this information is revealed as part of protocol operations to everyone in the network. Note that in the absence of a mapping from a DHT ID to an IP address, the adversary

cannot perform traffic analysis to infer social contacts. The only source of information leakage is when the adversary can map DHT IDs of two users to their respective IP addresses (for example, by virtue of being their trusted contacts); in this case the adversary can perform traffic analysis attacks to infer whether the two users have a trust relationship or not. In X-Vine, the privacy risk is with respect to social contacts, rather than random peers in the network. Note that in this paper, we are only concerned with overlay level adversaries; adversaries which operate at the ISP level, or have external sources of information [139] are outside the scope of our threat model.

6.4 Experiments and Analysis

We evaluate X-Vine with theoretical analysis, experiments using real-world social network topologies, and a prototype implementation. We measure routing state size, lookup path lengths, security against Sybil attacks, resilience against churn, and lookup latency. We also developed a Facebook application to facilitate the use of our design.

Simulation environment: We constructed an in-house event-driven simulator. As done in [14], we bootstrap X-Vine by selecting a random node in the social network as the first node, and the social network neighbors of that node then become candidates to join the X-Vine network. Next, one of these neighbors is selected to join, with a probability proportional to the number of trust relationships it has with nodes that are already a part of the X-Vine network. This process is then repeated. Note that some nodes may not be successful in joining because of the bound on number of trails per link (as discussed in detail later).

Data sets: Recent work has proposed the use of interaction graphs [124, 125] as a better indicator of real world trust than friendship graphs. Hence we consider both traditional social network graph topologies as well as interaction graph topologies in our evaluation. The datasets that we use have been summarized in Table 6.1.

Facebook friendship graph from the New Orleans regional network [125]: The original dataset consists of 60 290 nodes and 772 843 edges. We processed the dataset in a manner similar to the evaluation done in SybilLimit [73] and SybilInfer [119], by imposing a lower bound of 3 and an upper bound of 100

on the node degree (see [73, 119] for details). [2] After processing, we are left with 50 150 nodes and 661 850 edges.

Facebook wall post interaction graph from the New Orleans regional network [125]: The original dataset consists of 60 290 users. After processing, we are left with 29 140 users and 161 969 edges. Note that links in this dataset are directed, and we consider an edge between users only if there were interactions in both directions.

Facebook interaction graph from a moderate-sized regional [3] *network [124]:* The dataset consists of millions of nodes and edges, but our experiments are memory limited and do not scale to millions of nodes. Instead, we first truncate the dataset by considering only a four hop neighborhood from a seed node. After processing, we are left with 103 840 nodes and 961 418 edges.

Synthetic scale-free graphs: Social networks exhibit a scale-free node degree topology [112]. Our network synthesis algorithm replicates this structure through preferential attachment, following the methodology of Nagaraja [58]. The use of synthetic scale free topologies enables us evaluate X-Vine while varying the number of nodes in the network.

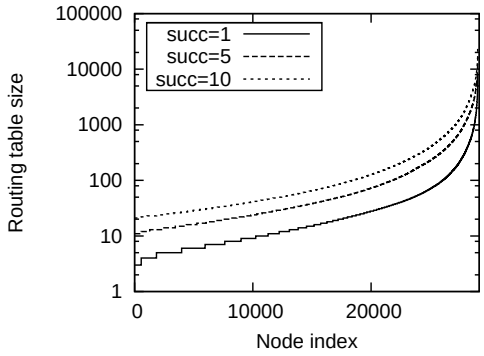
Table 6.1: Topologies

Dataset	**Nodes**	**Edges**	**Mean Degree**
New Orleans Facebook Friendship graph	50 150	661 850	26.39
New Orleans Facebook Interaction graph	29 140	161 969	11.11
Anonymous Facebook Interaction graph	103 840	961 418	18.51

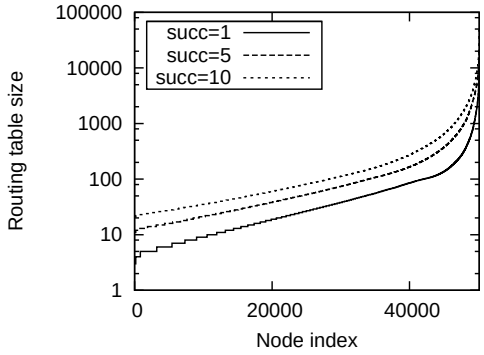
Overhead: Figure 6.4 plots the routing table size for different successor list sizes. We can see the temporal correlation effect here, as the distribution of state shows super-exponential growth. Temporal correlation is highly undesirable both from a performance and a security standpoint. If the few nodes with very large state become unavailable due to churn, the network could get

[2] Recent work by Mohaisen et al. [140] shows that social networks may not be as fast mixing as previously believed. However, we note that their results do not directly apply to X-Vine since they did not consider node degree bounds in their analysis. X-Vine excludes users having few friends from participating in the routing protocol, though such users could use their trusted friends to lookup keys.

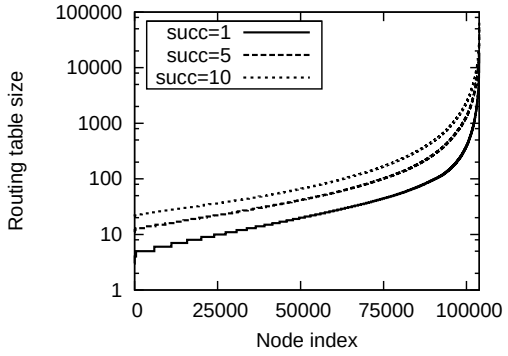
[3] Because of privacy reasons, the name of the regional network has been left anonymous by the authors of [124].



(a) New Orleans Interaction graph

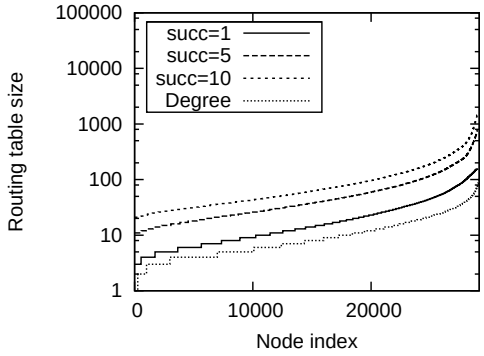


(b) New Orleans Friendship graph

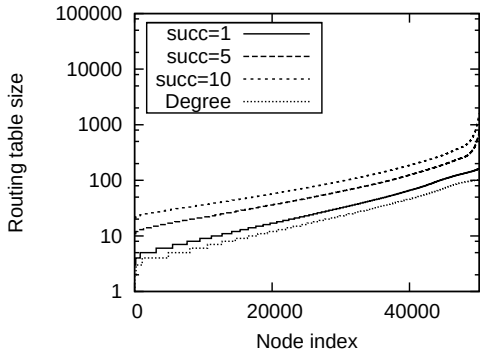


(c) Anonymous Interaction graph

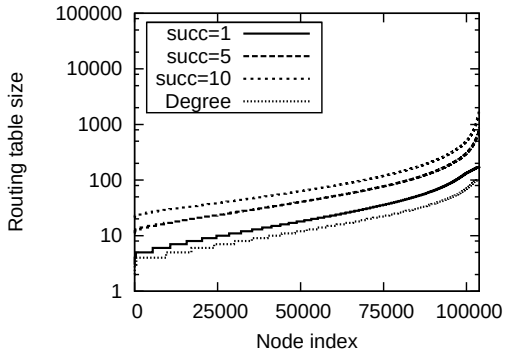
Figure 6.4: *Routing state, with no bounds on state:* Due to temporal correlation, some nodes exhibit high state requirements.



(a) New Orleans Interaction graph



(b) New Orleans Friendship graph



(c) Anonymous Interaction graph

Figure 6.5: *Routing state, with node and edge bounds:* Bounding state significantly reduces state requirements. Using a successor list of size 5, the average routing state for the three topologies is 67, 81, and 76 records respectively. X-Vine requires orders of magnitude less state than Whanau [1].

destabilized. Moreover, if one of these nodes is malicious, it could easily intercept a large number of lookups and drop them. To address this, we enable the routing policy that bounds the number of paths traversing nodes and links. Based on our analytic model in Appendix A, we propose the following bound on the number of paths per link: $b_l = \alpha \cdot 2 \cdot num\_successors \cdot \log(n)$, where α is a small fixed constant. The bound per link ensures that if a node has degree d, then its routing table size will never exceed $d \cdot b_l \in O(\log n)$. We can see that the routing state does not undergo an exponential increase as in previous plots. Moreover, routing state increases with node degrees, which is desirable. Based on these routing table sizes, we can estimate the communication overhead of X-Vine by computing the cost of sending heartbeat traffic for all records in the routing table. Considering the routing table size to be 125 records, UDP ping size to be 40 bytes, and a heartbeat interval of 1 s, the estimated mean communication overhead is only 4.8 KBps.

Comparison with Whanau [1]: Routing state in Whanau depends on the number of objects stored in the DHT. Routing tables in Whanau are of size $\Theta(\sqrt{n_o} \log n_o)$, where n_o is the number of objects in the DHT. If there are too many objects stored in the DHT, Whanau resorts to maintaining information about all the nodes and edges in the social network (increasing state/overhead to $\Theta(n)$). If there are too few objects in the DHT, Whanau resorts to flooding to find objects [1]. We note that such properties make Whanau unsuitable for many common applications. Even if we consider the case where each node in the DHT stores only tens of objects, the average routing table size in Whanau for the 103 840 node anonymous interaction graph is about 20 000 records—an increase of more than two orders of magnitude as compared with X-Vine. If we consider a heartbeat interval of 1 second in Whanau (in order to accurately maintain object states for common DHT applications), the resulting communication overhead is about 800 KBps. This difference increases further with an increase in the number of objects in the DHT or the size of the network. For instance, we scaled up our experiments to a larger 613 164 node anonymous interaction graph topology using a machine with 128 GB RAM, and found that the average routing state in X-Vine using a successor list size of 10 was only 195 records, as compared with more than 50 000 records in Whanau. Note that routing state in X-Vine is independent of the number of objects in the DHT.

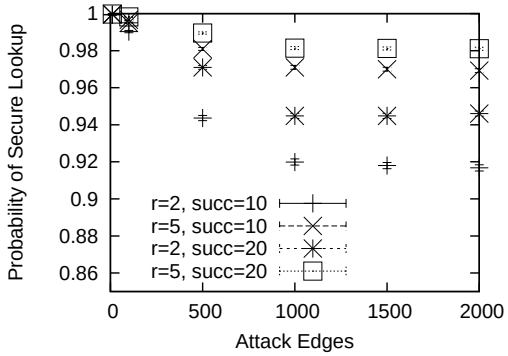
Table 6.2: Mean Lookup Path Length

# Succ	New Orleans interaction			New Orleans friendship			Anonymous interaction		
	$r = 1$	$r = 5$	$r = 10$	$r = 1$	$r = 5$	$r = 10$	$r = 1$	$r = 5$	$r = 10$
1	97.9	57.7	51.7	103.6	57.5	48.1	166.7	96.3	81.0
5	30.0	18.2	16.8	34.8	19.3	16.7	48.9	25.5	21.7
10	20.2	13.0	12.16	23.1	13.7	12.1	29.9	16.9	14.8
20	15.4	10.3	9.6	17.0	10.7	9.45	21.0	12.8	11.3

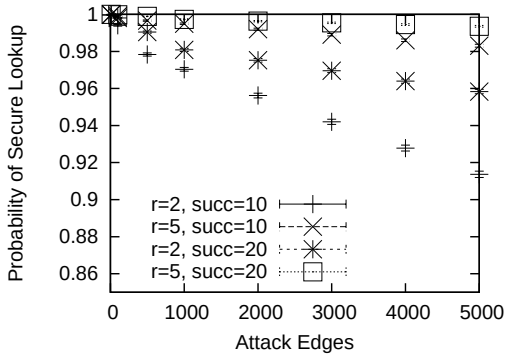
False Positive Analysis: Next, we consider the impact of link/node path bounds on honest node's ability to join the DHT. We found that most honest nodes were able to join the DHT due to the fast mixing nature of honest social networks. In fact, for all our experimental scenarios, the false-positive rate was less than 0.5%, which is comparable to the state-of-the-art systems [73, 119]. By tuning the parameter b_l, it is possible to trade off the false-positive rate for Sybil resilience: b_l will reduce the false-positive rate at the cost of increasing the number of Sybil identities in the system. For the remainder of the paper, we shall use $\alpha = 1, \beta = 5$.

Path Length Analysis: Table 6.2 depicts the mean lookup path lengths for the real world datasets with varying successor list sizes and varying redundancy parameter. We first observe that lookup performance improves with increasing successor list sizes. For example, in the New Orleans interaction graph, the mean lookup path length decreases from 97.9 to 15.4 when the successor list size increases from 1 to 20 (using $r = 1$). Further improvements in performance can be realized by performing redundant lookups as described in Section 6.3 and caching the lookup with the smallest path length. We can see that in the same dataset, mean lookup path length decreases from 15.4 to 10.3 when the redundancy parameter is increased from $r = 1$ to $r = 5$ (using successor list of size 20). Further increases in redundancy show diminishing returns. Observe that when the successor list size is at least 10, and the redundancy parameter is at least 10, then the mean lookup path lengths for all datasets are less than 15 hops. Increasing the successor list size to 20 (and keeping $r = 10$) reduces this value to less than 11.5 for all datasets.

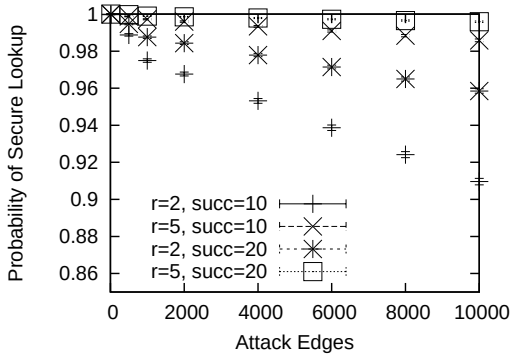
Security under Sybil Attack: Recall that if the adversary has g attack edges, then the number of trails between the honest and the Sybil subgraph is bounded by $g \cdot b_l$ (regardless of the attacker strategy). Our attack methodology is as follows: we randomly select a set of compromised nodes until the adversary has the desired number of attack edges. The compromised



(a) New Orleans interaction graph



(b) New Orleans friendship graph



(c) Anonymous Interaction graph

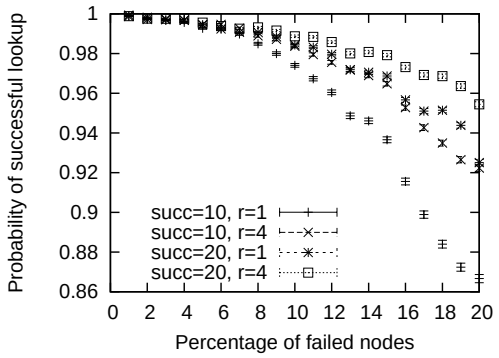
Figure 6.6: Probability of secure lookup as a function of number of attack edges.

nodes then launch a Sybil attack, and set up trails between Sybil identities and their overlay neighbors. If the trail set up request starting from a Sybil node gets shortcutted back to the Sybil identities, the request is backtracked. This ensures that the adversary uses only a single attack edge per trail. Node identifiers of Sybil identities are chosen at random with the adversarial goal of intercepting as many lookups as possible. All lookups traversing compromised/Sybil nodes are considered unsuccessful.

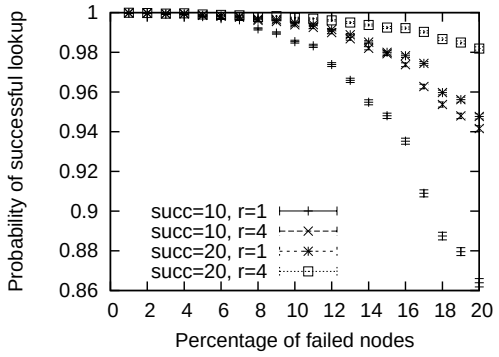
Figure 6.6 plots the probability of a secure lookup as a function of number of attack edges, redundancy parameter, and size of successor list. We find that the probability of secure lookup increases as the redundancy parameter is increased. This is because as the number of redundant lookups increases, there is a greater chance that a lookup will traverse only honest nodes and return the correct result. We also find that the probability of secure lookup also increases when the size of the successor list increases. This is because increasing successor list size reduces the mean lookup path length, reducing the probability that an adversary can intercept the lookup query. As long as $g \in o(n/(\log n))$, the probability of secure lookup can be made arbitrarily high by increasing the redundancy parameter and the successor list size. Finally, reducing b_l would further limit the impact of Sybil identities, at the cost of increased false positives.

Churn Analysis: Next, we evaluate the performance of X-Vine under churn. We are interested in the *static resilience* of X-Vine, i.e., the probability of lookup success after a fraction of the nodes in the system fail simultaneously. To account for churn, we modified the lookup algorithm to *backtrack* whenever it cannot make forward progress in the overlay namespace. Figure 6.7 depicts the mean probability of lookup success as a function of the fraction of nodes that fail simultaneously, averaged over 100 000 lookups. Similar to the analysis of lookup security, we can see that an increase in either the redundancy parameter or the successor list size result in improved resilience against churn. We can also see that as the fraction of failed nodes increases, the probability of lookup success decreases, but is still greater than 0.95 for all scenarios using $r = 4$ and $succ = 20$.

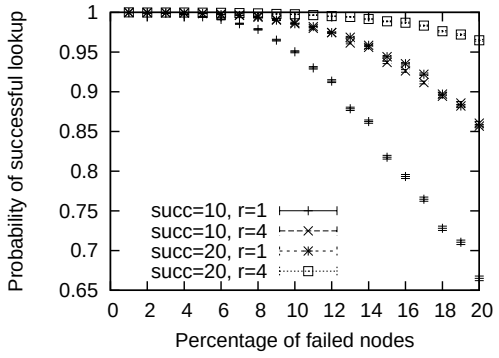
PlanetLab Implementation: To validate our design and evaluate lookup latency in real-world environments, we implemented the X-Vine lookup protocol in C++ as a single-threaded program using 3 000 LOC. We used libasync



(a) New Orleans Interaction graph



(b) New Orleans Friendship graph



(c) Anonymous Interaction graph

Figure 6.7: Lookup resilience against churn.

[141, 142] and Tame [143] to implement non-blocking socket functionality (UDP) in an event-based fashion. We ran our implementation over 100 ran-

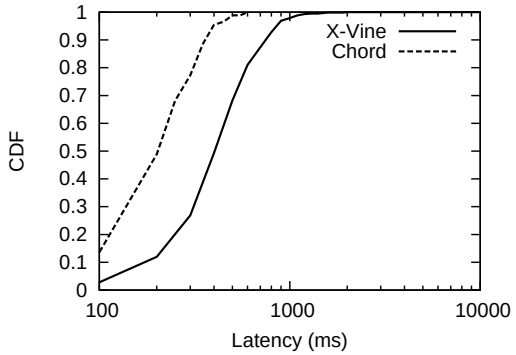


Figure 6.8: Lookup latency.

domly selected nodes in the PlanetLab network. We used a synthetic scale-free graph as the social network topology. The duration of the experiment was set to 1 hour, and nodes performed lookups every 1 second. Figure 6.8 depicts the CDF of observed one-way lookup latencies. We can see that the median lookup latency was only 400 ms (as compared to 200 ms in Chord), for the mean lookup path length of 5 hops (not shown in the figure). Using these values, we can estimate the median lookup latency for mean lookup path lengths of 10 hops and 15 hops (that were observed in our experiments over real world social network topologies in Table 6.2) to be about 800 ms and 1200 ms respectively. We see some outliers in Figure 6.8 due to the presence of a few slow/unresponsive nodes in PlanetLab. For this experiment, we mapped vertices in the social network topology to random PlanetLab nodes (possibly in different geographic locations). Thus, our analysis is conservative; accounting for locality of social network contacts would likely improve the lookup performance.

Facebook Application: To bootstrap a X-Vine node, its user needs to input the IP addresses of his/her friends. Since this can be a cumbersome for a user, we implemented a Facebook application (available at `http://apps.facebook.com/x--vine`) that automates this process and improves the usability of our design. The work flow of the application is as follows: (i) When a user visits the Facebook application URL, Facebook checks the credentials of the user, the user authorizes the application, and then the request gets redirected to the application hosting server. (ii) The

application server authenticates itself, and is then able to query Facebook for user information. The application server records the user information along with the user IP address. (iii) The application server then queries Facebook for a list of user's friends, and returns their previously recorded IP addresses (if available) to the user.

This list of IP addresses could then be used by the DHT software to bootstrap its operations. Our implementation demonstrates that a user's social contacts can be integrated into the DHT protocol using only a few hundred lines of glue code. Keeping in spirit with our fully decentralized design goal, in future, our application could be implemented on a decentralized platform like Diaspora [144] such that the app server is not a central point of trust or failure.

6.5 Limitations

We now discuss some limitations of our design. First, X-Vine requires a user's social contacts to be part of the overlay; the DHT needs to be bootstrapped from a single contiguous trust network. Next, X-Vine assumes that Sybil identities are distributed randomly in the DHT identifier space. We emphasize that this assumption is shared by prior systems [74], and that defending multi-hop DHTs against targeted clustering attacks is currently an open problem. In future work, we will investigate the possibility of adapting the cuckoo hashing mechanism [117] proposed by Lesniewski-Laas (for one-hop DHTs) in the context of securing multi-hop DHTs. X-Vine also does not defend against attackers who target users by compromising nodes close to them in the social network topology. Finally, applications using X-Vine experience higher than usual latencies since all communications are pseudonymous and traverse multiple social network links.

6.6 Summary

We described X-Vine, a protection mechanism for DHTs that operates entirely by communicating over social network links. X-Vine requires $O(\log n)$ state, two orders of magnitude less in practical settings as compared with ex-

isting techniques, making it particularly suitable for large-scale and dynamic environments. X-Vine also enhances privacy by not revealing social relationship information and by providing a basis for pseudonymous communication.

CHAPTER 7

PISCES: TRUSTWORTHY AND SCALABLE ANONYMOUS COMMUNICATION

To our knowledge, no proposed system securely leverages trust information in a scalable anonymity system. ShadowWalker is secure and scalable, but incorporating trust information is very difficult due to the rules governing communication links. Johnson et al. propose a method to incorporate trust into a Tor-like system, in which all proxies are known to all users [145, 146], but it cannot be easily applied to P2P systems in which users only know about a small set of all proxy nodes. Both Nagaraja [58] and Danezis et al. [59] describe a compelling vision for incorporating trust into P2P anonymity by building circuits over edges in a social network graph. Unfortunately, social links cannot be fully trusted, as users can be manipulated into adding links [131,132]. This fact makes both approaches vulnerable to *route capture attacks*, in which the entire circuit is comprised of attacker-controlled nodes.

We propose to bring together the benefits of trust relations, in the form of a social network, with the advantages of secure random walks in a P2P environment. Simply put, we construct the random walks on a social network topology to select circuits in such a way that they cannot be manipulated by the attacker. This provides substantial protection to users, even when they add a few social links to malicious peers. The main technique that we leverage for our random walks is the *reciprocal neighborhood policy (RNP)*. The RNP states that all links in the graph must be bi-directional; for social networks, this means that social relationships must be mutual (friends, not followers). By providing techniques that enforce this policy, we ensure that a random walk on the topology is truly random. Further, to prevent an attacker from benefiting by creating a large clique of malicious peers in the graph, we bias random walks away from peers with many friends.

Using the RNP, we present the design of Pisces, a P2P anonymity system that uses secure random walks on social networks to take advantage of trust without being exposed to circuit manipulation. A core contribution of our

work is a technique for enforcing the RNP in a fully decentralized fashion. We efficiently distribute each node's current list of contacts so that those contacts can verify periodically that they are in the list. A contact that should be in the list, but is not, can remove the node permanently from its contacts. Further, the list is signed by the node, so any mismatched lists for the same time period constitute proof that the node is cheating.[1] First we demonstrate through a combination of analysis, simulation, and experiments, that our RNP provides good deterrence against active attacks. Next, we also show that our distributed design provides robust enforcement of the RNP, with manageable overhead for distributing and checking contact lists.

Based on the RNP, we also present a circuit construction algorithm for Pisces. Using a real world social network topology and a reasonable set of trust assumptions, we find that Pisces significantly outperforms ShadowWalker, and provides up to 6 bits higher entropy in a single communication round. Also, compared with the naive strategy of using conventional random walks over social networks (as in the Drac system), Pisces provides twice the amount of entropy over 100 communication rounds.

This chapter is organized as follows. We present an overview of our system and the Pisces protocol in Section 7.1. We evaluate Pisces in Section 7.2, Section 7.3, and Section 7.4. We present a discussion in Section 7.5, and summarize in Section 7.6.

7.1 Pisces Protocol

7.1.1 Design goals

1. Scalable anonymity: we are interested in the design of anonymous communication systems that can scale to millions of users and relays, with low communication overhead. Since anonymity is defined as the state of being unidentifiable in a group [147], architectures that can support millions of users provide the additional benefit of increasing the overall system anonymity.

2. Trustworthy anonymity: we target an architecture that is able to leverage

[1] In astrology, though not according to any source we can still find, it is said that a smart Pisces is good at detecting liars; hence the name of our system.

a user's social trust relationships to improve the security of anonymous communication. Current mechanisms for scalable anonymous communication are based on structured peer-to-peer topologies [9, 25, 66, 118], and are unable to leverage trust represented in unstructured social network graphs (G).
3. Decentralized design: the design should not have any central entities. Central entities are attractive targets for attackers, in addition to being a single point of failure for the entire system.

7.1.2 Threat model

In this work, we consider a colluding adversary who can launch Byzantine attacks against the anonymity system. The adversary can perform passive attacks such as logging information for end-to-end timing analysis [32], as well as active attacks such as deviating from the protocol and selectively denying service to some circuits [54].

We assume the existence of mechanisms to defend against the Sybil attack [73, 119]. In particular, we consider two defense models: (a) an ideal Sybil defense which does not allow the insertion of any Sybil identity and (b) a realistic Sybil defense which allows the insertion of a bounded number of Sybil identities in the system. The latter model also requires the number of attack edges to be bounded by $g = \frac{h}{\log h}$, where h is the number of honest nodes in the system.

7.1.3 System model and assumptions

Each node generates a local public-private key pair. Pisces is a fully decentralized protocol and does not assume any PKI infrastructure. A node's identity in the system refers to its public key. Existing Sybil defense mechanisms can be used to validate node identities (public keys). We assume that the identities in the system can be blacklisted; i.e., the adversary cannot whitewash its identities by rejoining the system with a different public key. This is a reasonable assumption, since (a) mechanisms such as SybilLimit only allow the insertion of a bounded number of Sybil identities, and (b) replacing deleted attack edges is expensive for the attacker, particularly in a social network graph based on interactions. Finally, we assume loose

time synchronization amongst nodes. Existing services such as NTP [148] can provide time synchronization on the order of hundreds of milliseconds in wide area networks [149].

7.1.4 Problem overview

Random walks are an integral part of many anonymity systems. In a random walk based circuit construction, an initiator I of the random walk first selects a random node A from its neighbors in some topology (in our case, the social network graph). The initiator sets up an onion routing circuit with node A, and uses the circuit to download a list of node A's neighbors (containing the IP addresses and public keys of neighbors). Node I can then select a random node B from the downloaded list of node A's neighbors, and extend the onion routing circuit to node B. This process can be repeated to set up a circuit of length l.

Random walks are vulnerable to active route capture attacks that enable an adversary to bias the peer discovery process towards colluding malicious nodes. First, malicious nodes can exclude honest nodes from their neighbor list to bias the peer discovery process. Second, malicious nodes can modify the public keys of honest nodes in their neighbor list. When an initiator of the random walk extends a circuit from a malicious node to a neighboring honest node, the malicious node can simply emulate the honest neighbor. The malicious node can repeat this process for further circuit extensions as well. Finally, the malicious nodes can add more edges between each other in the social network topology to increase the percentage of malicious nodes in their neighbor lists.

To secure the random walk process, we first propose a policy which allows users to blacklist their neighboring node(s) if the neighbor is not advertising their information correctly. Second, we propose a protocol that securely realizes the above policy by detecting instances of route capture attacks by malicious nodes.

7.1.5 Reciprocal neighborhood policy

We present a new primitive for securing random walks, which we call *reciprocal neighbor policy*. Our main idea is to consider undirected versions of structured or unstructured topologies, and then entangle the routing tables of neighboring nodes with each other; i.e., if a malicious node X does not correctly advertise an honest node Y in its neighbor list, then Y also excludes X from its neighbor list (tit-for-tat).

The reciprocal neighborhood policy ensures that route capture attacks based on incorrect advertisement of honest nodes during random walks serves to partially isolate malicious nodes behind a small cut in the topology, reducing the probability that they will be selected in a random walk. In particular, this policy mitigates the first two types of route capture attacks described above, namely the exclusion of malicious nodes, as well as the public key modification attack. However, the adversary can still bias the peer discovery process by continuing to keep all honest nodes in its neighbor list while inserting a large number of malicious nodes therein. Thus, as described so far, such a reciprocal neighborhood policy would only be effective for topologies where node degrees are bounded as well as homogeneous, such as structured peer-to-peer topologies like Chord [60] and Pastry [61]. However, node degrees in unstructured social network topologies are highly heterogeneous, presenting an avenue for attack.

Handling the node degree attack

Addition of edges amongst colluding malicious nodes in a topology increases the probability that a malicious node is selected in a random walk. To prevent such increasing node degree attacks, we propose to perform random walks using the Metropolis-Hastings modification [111, 150] — the transition matrix used for our random walks is as follows:

$$P_{ij} = \begin{cases} \min(\frac{1}{d_i}, \frac{1}{d_j}) & \text{if } i \to j \text{ is an edge in G} \\ 0 & \text{otherwise} \end{cases}, \tag{7.1}$$

where d_i denotes the degree of vertex i in G. Since the transition probabilities to neighbors may not always sum upto 1, nodes add a self loop to the transition probabilities to address this. The Metropolis-Hastings modi-

fication ensures that attempts to add malicious nodes in the neighbor table decrease the probability of malicious nodes being selected in a random walk.

We will show that the Metropolis-Hastings modification along with reciprocal neighborhood policy is surprisingly effective at mitigating active attacks on random walks. A malicious node's attempts to bias the random walk process by launching route capture attacks reduce its own probability of getting selected as an intermediate node in future random walks, nullifying the effect of the attack.

7.1.6 Securing reciprocal neighbor policy

We now present our protocol for securely implementing the reciprocal neighborhood policy.

Intuition: Our key idea is to keep each node's neighbor list *static* for the duration of a time interval (t), *regardless* of churn events in the network, such as node joins or leaves. This proposal is a departure from conventional networks where neighbor lists are updated as soon as a node learns about churn events. The main advantage in having a static neighbor list for the full duration of a time interval is that presence of *conflicting* neighbor lists issued by a malicious node becomes a clear indication of malicious behavior, and can be used by nodes to set up blacklists. On the other hand, if malicious nodes advertise a single neighbor list for the entire time slot, then honest neighbors can directly query their neighbors to check if they have been included in the neighbor lists. To handle churn, all nodes update their neighbor lists after every time interval in a synchronized fashion (based on our assumption of loose time synchronization). The duration of the time interval for which the policy remains static determines the tradeoff between the communication overhead for securing the reciprocal neighborhood policy and the unreliability of circuit construction due to churn.

Setting up static neighbor list certificates: A short time prior to the beginning of a new time interval, each node sets up a new neighbor list that it will use in the next time interval:

1. *Liveness check*: In the first round, nodes exchange messages with their trusted neighbors to check for liveness and reciprocal trust. A reciprocal exchange of messages ensures that both neighbors are interested

in advertising each other in the next time interval (and are not in each other's local blacklists). Nodes wait for a time duration to receive these messages from all neighbors, and after the timeout, construct a preliminary version of their next neighbor list, comprising node identities of all nodes which responded in the first communication round.

2. *Degree exchange*: Next, the nodes broadcast the length of their preliminary neighbor list to all the neighbors. This step is important since Metropolis-Hastings random walks require node degrees of neighboring nodes to determine their transition probabilities.
3. *Final list*: After receiving these broadcasts from all the neighbors, a node creates a final neighbor list and digitally signs it with its private key. The final list includes the IP address, public key, and node degree of each neighbor, as well as the time interval for the validity of the list. Note that a neighbor may become offline between the first and second step, before the node has a chance to learn its node degree, in which case it can simply be omitted from the final list.
4. *Local integrity checks*: At the beginning of every new time interval, each node queries all its neighbors and downloads their signed neighbor lists. When a node A receives a neighbor list from B, it performs local integrity checks, verifying that B's neighbor entry for A contains the correct IP address, public key, and node degree. Additionally, it verifies that the length of the neighbor list is at most as long as was broadcast in phase 2. (Note that intentionally broadcasting a higher node degree is disadvantageous to a B, as it will reduce the transition probability of it being chosen by a random walk.) If any local integrity checks fails, A places B in its permanent local blacklist, severing its social trust relationship with B and refusing all further communication. If all the checks succeed, then these neighbor lists serve as a cryptographic commitment from these nodes–the presence of any conflicting neighbor lists for the same time interval issued by the same node is a clear evidence of misbehavior.

   If B's neighbor list omits A entirely, or if B simply refuses to send its neighbor list to A, B is placed on a temporary blacklist, and A will refuse further communication with B for the duration of the current time period, preventing any circuits from being extended from A to B. (Effectively, A performs a selective denial-of-service against B; see

Section 7.4.5 for more discussion of this.) The blacklist only lasts for the duration of the current round, since the omission could have resulted of a temporary communication failure.

Duplicate detection: Next, we need to ensure that B uses the same neighbor list during random walks as it presented to its neighbors. Our approach is to use the Whanau DHT to check for the presence of several conflicting neighbor lists signed by the same node for the same time period. After performing the local checks, A will store a copy of B's signed neighbor list in the Whanau, using B's identity (namely, its public key) as the DHT key. Then, when another node C performs a random walk that passes through B, it will receive a signed neighbor list from B. It will then perform a lookup in the DHT for any stored neighbor lists under B's key. If it discovers a different list for the same period with a valid signature, then it can notify B's neighbors about the misbehavior, causing them to immediately blacklist B.

One challenge is that the Whanau lookups are not anonymous and may reveal to external observers the fact that C is performing a random walk through B. This information leak, linking C and B, can then be used to break C's anonymity. To address this problem, we introduce *testing* random walks that are not actually used for anonymous communication but are otherwise indistinguishable from *regular* random walks. Whanau lookups to check for misbehavior are performed during[2] testing random walks only, since information leaks in that case will not reveal private information, whereas during regular random walks no lookups are performed. If each node performs a small number of testing walks within a each time period, any misbehavior will be detected with high probability.

Blacklisting: When C detects a conflicting neighbor list issued by B, it immediately notifies all of B's neighbors (as listed in the neighbor list stored in the DHT), presenting the two lists as evidence of misbehavior. B's neighbors will thereafter terminate their social relationships with B, blacklisting it. Note, however, that the two conflicting lists form incontrovertible evidence that B was behaving maliciously, since honest nodes *never* issue two neighbor lists in a single time interval. This evidence can be broadcast globally

[2]More precisely, the lookups must happen *after* the testing walk is complete, to ensure that information leaks from lookups cannot be used to distinguish it from a regular walk.

to ensure that *all* nodes blacklist B, as any node can verify the signatures on the two lists, and thus B will not be able to form connections with any honest nodes in the system. Moreover, honest nodes will know not to select B in any random walk, effectively removing it from the social graph entirely.

Proactive vs. reactive security: Our system relies on detecting malicious behavior and blacklisting nodes. Thus, as described so far, Pisces provides reactive security. To further strengthen random walk security in the scenario when the adversary is performing route capture for the first time, we propose an extension to Pisces that aims to provide proactive security. We propose a *discover but wait* strategy, in which users build onion routing circuits for anonymous communication, but impose a delay between building a circuit and actually using it for anonymous communication. If misbehavior is detected by a testing random walk within the delay period, the circuit will be terminated as B's neighbors blacklist it; otherwise, if a circuit survives some timeout duration, then it can be used for anonymous communication.

Performance optimization: Using all hops of a random walk for anonymous communication has significant performance limitations. First, the latency experienced by the user scales linearly with the random walk length. Second, long circuit lengths reduce the overall throughput that a system can offer to a user. Inspired by prior work [118], we propose the following performance optimization. Instead of using all hops of a random walk for anonymous communication, the initiator can use the random walk as a peer discovery process, and leverage the kth hop and the last hop to build a two-hop circuit for anonymous communication. In our evaluation, we find that values of k that are close to half the random walk length provide a good tradeoff between anonymity and performance.

7.2 Evaluation: Reciprocal Neighborhood Policy

In this section, we evaluate Pisces with theoretical analysis as well as experiments using real-world social network topologies. In particular, we (a) show the security benefits provided by the RNP, (b) evaluate the security, performance, and overhead of our protocol that implements the RNP and (c) evaluating the overall anonymity provided by Pisces.

We consider five datasets for our experiments, which were processed in a

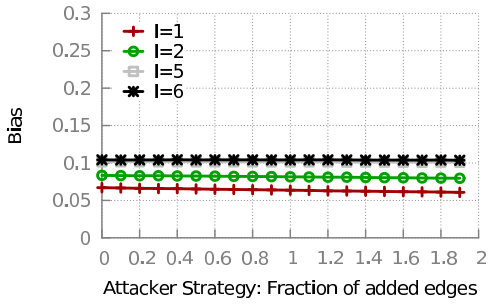
manner similar to evaluation done in SybilLimit [73] and SybilInfer [119]: (i) *a Facebook friendship graph from the New Orleans regional network [125]*, containing 50 150 nodes and 772 843 edges; (ii) *a Facebook wall post interaction graph from the New Orleans regional network [125]*, containing 29 140 users and 161 969 edges; (iii) *a Facebook interaction graph from a moderate-sized regional network [124]*, containing about 380 564 nodes and about 3.24 million edges; (iv) *a Facebook friendship graph from a moderate-sized regional network [124]*, containing 1 033 805 nodes and about 13.7 million edges; and (v) *Synthetic scale-free graphs:* Social networks exhibit a scale-free node degree topology [112]. Our network synthesis algorithm replicates this structure through preferential attachment [58].

To demonstrate the effectiveness of the RNP primitive for implementing trust-based anonymity, let us assume for now that there is a mechanism to securely achieve the RNP, i.e., that if a node X does not advertise a node Y in its fingertable, then Y also excludes X. In this scenario, we are interested in characterizing the probability distribution of random walks terminating at malicious nodes. In our analysis, we consider two attack models: (a) an ideal Sybil defense mechanism that does not permit any Sybil attacks and (b) a realistic (imperfect) Sybil defense that tolerates $\frac{h}{\log h}$ attack edges and admits 10 Sybils [73] per attack edge. First, we consider the security of reciprocal neighborhood policy under an ideal Sybil defense.

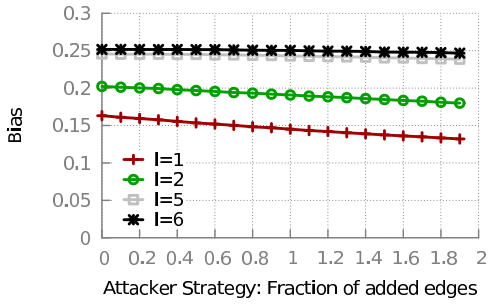
Theorem 1. Node degree attack*: Given h honest nodes and m malicious nodes that have g edges (attack edges) amongst each other in a connected social network, the adversary cannot bias the stationary distribution of an honest random walk by adding edges amongst malicious nodes.*

Proof. Let us denote π_i as the stationary probability for node i, and let P_{ij} denote the transition probability from node i to node j. Let n denote the total number of nodes in the social network ($n = h+m$). Since the transition probabilities between nodes are symmetric ($P_{ij} = P_{ji} = \min\left(\frac{1}{degree(i)}, \frac{1}{degree(j)}\right)$), observe that $\forall z, \pi_z = \frac{1}{n}$ is solution to the equation $\pi_i \cdot P_{ij} = \pi_j \cdot P_{ji}$. Since social networks are non-bipartite as well as undirected graphs, the solution to the above equation ($\pi = \frac{1}{n}$) must be the unique stationary distribution for the random walk. Thus the introduction of new edges amongst malicious nodes does not change the uniform stationary distribution of the random walk. □

(a) Without attack



(b) g = 1000 attack edges



(c) g = 2000 attack edges

Figure 7.1: *Probability of the l'th hop being compromised (Sampling Bias), under an increasing node degree attack:* For short random walks, this is a losing strategy for the adversary. For longer random walks, the adversary does not gain any advantage.

Next, we present simulation results using a synthetic scale-free topology with 1000 nodes. Figure 7.1(a) depicts the probability of a Pisces random walk terminating at malicious nodes as a function of random walk length for $g = 1000$ (100 malicious nodes) and $g = 2000$ (250 malicious nodes). We can see that the random walk quickly reaches its stationary distribution, and at the stationary distribution, the probability of a random walk terminating at one of the malicious nodes is 0.1 and 0.25 respectively. Figure 7.1(b) and (c) depict the probability of a random walk terminating at one of the malicious nodes under the node degree attack, for $g = 1000$ and $g = 2000$ respectively. We can see that adding edges amongst malicious nodes does not help the adversary for *any* random walk length.

Theorem 2. Local blacklisting*: Suppose that an adversary sacrifices $y \leq g$ attack edges. While the stationary distribution of the random walk remains uniform, the transient distribution of the random walk terminating at malicious nodes becomes smaller than without attack.*

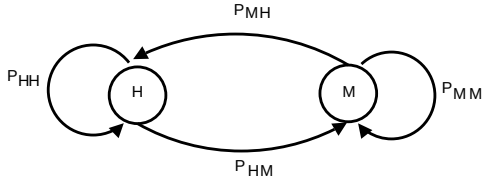


Figure 7.2: Attack model.

Proof. To characterize the transient distribution of the random walk, we model the process as a Markov chain (shown in Figure 7.2). Let us denote the honest set of nodes by H, and the set of malicious nodes by M.

The probability of an l hop random walk ending in the malicious region ($P(l)$) is given by:

$$P(l) = P(l-1) \cdot P_{MM} + (1 - P(l-1)) \cdot P_{HM}. \tag{7.2}$$

The terminating condition for the recursion is $P(0) = 1$, which reflects that the initiator is honest.

We can estimate the probabilities P_{HM} and P_{MH} as the forward and backward conductance [110] between the honest and the malicious nodes, denoted by ϕ_F and ϕ_B respectively. Thus we have that:

$$P(l) = P(l-1) \cdot (1 - \phi_B) + (1 - P(l-1)) \cdot \phi_F$$
$$= P(l-1) \cdot (1 - \phi_B - \phi_F) + \phi_F. \tag{7.3}$$

$$P(l) = \phi_F \cdot [1 + (1 - \phi_B - \phi_F) + (1 - \phi_B - \phi_F)^2$$
$$\ldots + (1 - \phi_B - \phi_F)^{l-1}]. \tag{7.4}$$

With g edges between honest and malicious nodes, we can estimate the forward conductance ϕ_F as follows:

$$\phi_F = \frac{\Sigma_{x \in H} \Sigma_{y \in M} \pi_x \cdot P_{xy}}{\pi_H}$$
$$= \frac{\Sigma_{x \in H} \Sigma_{y \in M} \cdot P_{xy}}{|H|} = O\left(\frac{g}{h}\right). \tag{7.5}$$

Similarly, with g edges between honest and malicious nodes, the backward conductance ϕ_B is estimated as:

$$\phi_B = \frac{\phi_F \cdot |H|}{|M|} = \frac{O(\frac{g}{h}) \cdot h}{m} = O\left(\frac{g}{m}\right). \tag{7.6}$$

Thus, we have that $\phi_F = O(\frac{g}{h})$, and $\phi_B = O(\frac{g}{m})$. If malicious nodes exclude y edges to honest nodes from their fingertables, application of the RNP ensures that the honest nodes also exclude the y edges from their fingertables (local blacklisting). Thus, route capture attacks result in deleting of attack edges which reduces both forward and backward transition probabilities.

Observe that the probability of the first hop being in the malicious region is equal to ϕ_F, which gets reduced under attack. We will now show this for a general value of l. Following Equation (7.4) and using $\Sigma_{i=0}^{i=m} x^i = \frac{1-x^{m+1}}{1-x}$ for $0 < x < 1$, we have that:

$$P(l) = \frac{\phi_F \cdot (1 - (1 - \phi_B - \phi_F)^l)}{1 - (1 - \phi_B - \phi_F)}$$
$$= \frac{\phi_F}{\phi_F + \phi_B} \cdot (1 - (1 - \phi_B + \phi_F)^l). \tag{7.7}$$

Using $\phi_B = \frac{h}{m} \cdot \phi_F$, we have that:

$$P(l) = \frac{m}{n} \cdot (1 - (1 - \phi_B + \phi_F)^l)$$
$$P(l) = \frac{m}{n} \cdot \left(1 - \left(1 - \frac{n}{m} \cdot \phi_F\right)^l\right). \tag{7.8}$$

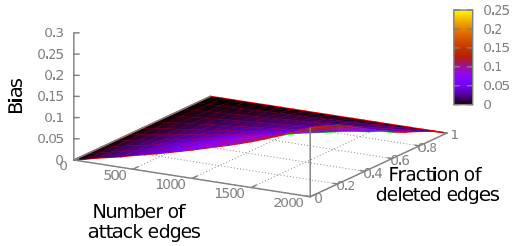
Differentiating $P(l)$ with respect to ϕ_F, we have that:

$$\frac{d}{d\phi_F}(P(l)) = \frac{m}{n} \cdot \left(l \cdot \left(1 - \frac{n}{m} \cdot \phi_F\right)^{l-1}\right). \tag{7.9}$$

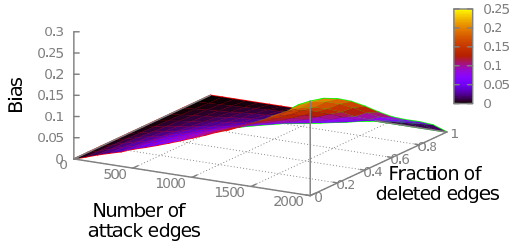
Note that $(1 - \frac{n}{m}\phi_F) = (1 - \phi_B - \phi_F) \geq 0$. This implies $\frac{d}{d\phi_F}P(l) \geq 0$. Thus, $P(l)$ is an increasing function of ϕ_F, and since the reduction of the number of attack edges reduces ϕ_F, it also leads to a reduction in the transient distribution of the random walk terminating at malicious nodes. □

Next, we validate our analysis using simulation. Figure 7.3 depicts the probability of a random walk terminating at one of the malicious nodes as a function of the number of attack edges as well as the fraction of sacrificed attack edges. We can see that in all scenarios, this bias is a decreasing function of the fraction of deleted edges. Moreover, it is notable that the decrease is larger for shorter random walks than for longer random walks. This reflects the fact that the stationary distribution under local blacklisting being unchanged, which we address next.

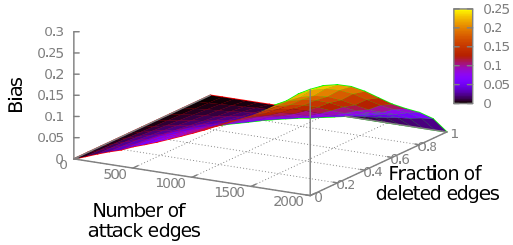
Theorem 3. Global blacklisting*: suppose that $x \leq m$ malicious nodes sacrifice $y_1 \leq g$ attack edges. Also suppose that these x malicious nodes originally had $y_2 \leq g$ attack edges. The stationary probability of random walk terminating at malicious nodes gets reduced proportional to x. The transient distribution of stationary walks terminating at malicious nodes is reduced as a function of y_2.*



(a) $l = 1$



(b) $l = 3$



(c) $l = 6$

Figure 7.3: *Probability of the l'th hop being compromised (Sampling Bias) under a route capture attack.* As more edges to the honest nodes are removed, the attacker's loss is higher. Note that the impact is very high on small length random walks, but gets smaller for longer length random walks.

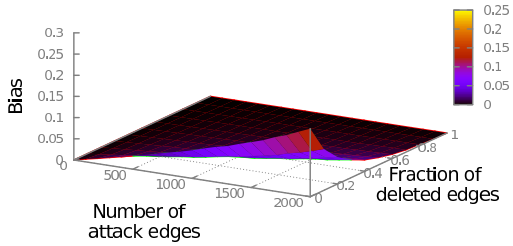
Proof. If x malicious nodes perform the route capture attack and are globally blacklisted, these nodes become disconnected from the social trust graph. It follows from our analysis of Theorem 1 that the stationary distribution of the random walk is uniform for all *connected* nodes in the graph. Thus, the stationary distribution of random walks terminating at malicious nodes gets reduced from $\frac{m}{m+h}$ to $\frac{m-x}{m-x+h}$. Furthermore, it follows from our proof of Theorem 2 that the transient distribution of random walks terminating at malicious nodes is an increasing function of the number of remaining attack edges, i.e, an increasing function of y_2. □

Figure 7.4 depicts the probability of random walks terminating at malicious nodes as a function of number of attack edges as well as the fraction of deleted edges when honest nodes use a global blacklisting policy. Again, we can see that sacrificing attack edges so as to perform route capture attacks is a losing strategy for the attacker. Moreover, the decrease is similar for all random walk lengths; this is because even the stationary distribution of the random walk terminating at malicious nodes is reduced.

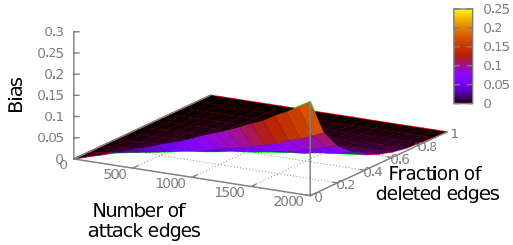
7.2.1 Anonymity implication

To de-anonymize the user without the help of the destination node (e.g. the website to which the user connects anonymously), both the first hop and the last hop of the random walk need to be malicious to observe the connecting user and her destinations, respectively. End-to-end timing analysis [151, 152] makes it so that controlling these two nodes is sufficient for de-anonymization. Figure 7.5 depicts the probability of such an attack being successful as a function of the number of attack edges and the fraction of deleted edges using the local blacklisting policy. While it is expected that the probability of attack would be a decreasing function of the fraction of deleted edges, we note that the decrease is larger than the reduction in previously depicted transient/ stationary probabilities. Figure 7.6 depicts the probability of end-to-end timing analysis for a global blacklisting policy. We see that, for attempting to undermine anonymous communications, route capture attacks are clearly a losing strategy against our approach.

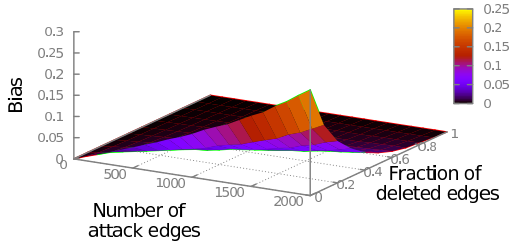
So far, we presented theoretical and simulation results assuming an ideal Sybil defense. Our theorems and analysis also hold for a more realistic Sybil



(a) $l = 1$



(b) $l = 3$



(c) $l = 6$

Figure 7.4: *Probability of l'th hop being compromised (Sampling Bias) under route capture attack with global blacklisting.* As more edges to the honest nodes are removed, the attacker's loss is higher. Note that the impact is the same for all random walk lengths.

defense that permits a bounded number of Sybils per attack edge. The key difference between the analysis of these two models is that, for a bounded number of Sybils, we model the random walk process as a three state Markov chain, with states for honest nodes, compromised nodes, and Sybil identities. This takes into account that random walks cannot directly transition from Sybil states to honest states.

7.3 Evaluation: Securing Reciprocal Neighborhood Policy

We now discuss the security and performance of our protocol that implements the reciprocal neighborhood policy.

7.3.1 Security proof sketch

Let us suppose that a malicious node A aims to exclude an honest node B from its fingertable. To pass node B's local integrity checks, node A has to return a neighborlist to node B that correctly advertises node B. Since random walks for anonymous communication are indistinguishable from testing random walks, there is a probability that the adversary will advertise a conflicting neighbor list that does not include node B to an initiator of the testing random walk. The initiator of the testing random walk will insert the malicious neighbor list into the Whanau DHT, and node B can perform a robust lookup for node A's key, and obtain the conflicting neighborlist. Since Whanau only provides availability, node B can check for integrity of the results by verifying node A's signature. Since honest nodes never advertise two conflicting lists within a time interval, node B can infer that node A is malicious (local blacklist) and also prove this to any third party (global blacklist).

7.3.2 Performance evaluation

First, we analyze the number of testing random walks that each node must perform to achieve a high probability of detecting a malicious node that

attempts to perform a route capture attack. Nodes must perform enough testing walks such that a high percentage of compromised nodes (which are connected to honest nodes) have been probed in a single time slot. First, we consider a defense strategy where honest nodes only insert the terminal hop of the testing random walks in Whanau (*Strategy 1*). Intuitively, from the coupon collectors problem, we have that $\log n$ walks per node should suffice to catch a malicious node with high probability. Indeed, from Figure 7.7, we can see that six testing walks per time interval suffice to catch a malicious node performing route capture attacks with high probability. The honest nodes can also utilize all hops of the testing random walks to check for conflicts (*Strategy 2*), in which case only two or three testing walks are required per time interval (at the cost of increased communication overhead for the DHT operations).

Next, we address the question of how to choose the duration of the time interval (t). The duration of the time slot governs the tradeoff between communication overhead and reliability of circuit construction. A large value of the time slot interval results in a smaller communication overhead but higher unreliability in circuit construction, since nodes selected in the random walk are less likely to still be online. On the other hand, a smaller value of the time interval provides higher reliability in higher circuit construction, but also has increased communication overhead, since a fixed number of testing walks must be performed within the duration of the timeslot. We can see this tradeoff in Figure 7.8. We consider two churn models for our analysis: (a) nodes have a mean lifetime of 24 hours (reflecting behavior of Tor relays), and (b) nodes have a mean lifetime of 1 hour (reflecting behavior of conventional P2P networks). For the two scenarios, using a time slot duration of 3 hours and 5 minutes, respectively, results in a 2-3% probability of getting an unreliable random walk for up to 25 hops.

7.3.3 Overhead

There are three main sources of communication overhead in our system. First is the overhead due to setting up the neighbor lists; which requires about d^2 KB of communication, where d is the node degree. The second source of overhead is the testing random walks, where nodes are required to perform

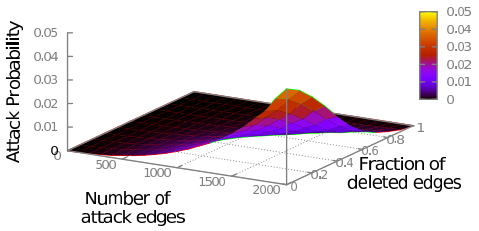


Figure 7.5: Probability of end-to-end timing analysis under route capture attack.

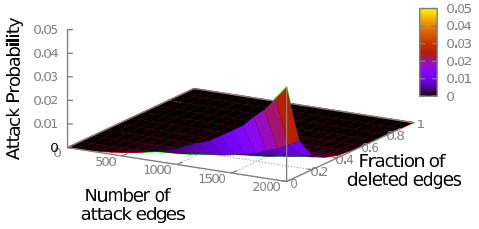


Figure 7.6: Probability of end-to-end timing analysis under route capture attack with global blacklisting.

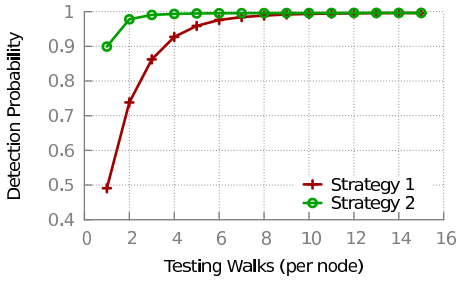


Figure 7.7: Probability of detecting a route capture [Facebook wall post interaction graph]. Attack model includes 10 Sybils per attack edge.

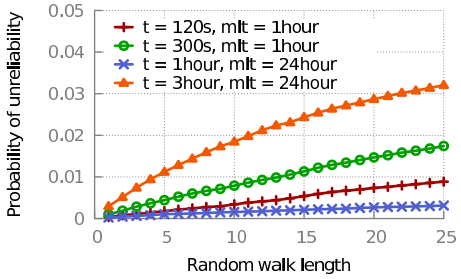


Figure 7.8: Unreliability in circuit construction [Facebook wall post interaction graph].

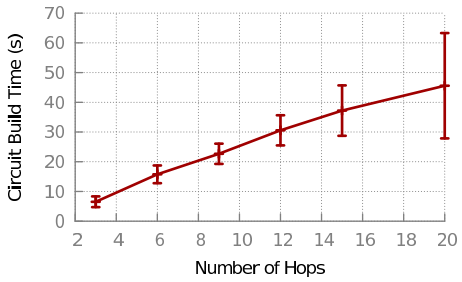


Figure 7.9: Circuit build times in Tor as a function of circuit length.

about six such walks of length 25 (see Section 7.3.2). The third source of overhead comes from participation in the Whanau DHT. Typically, key churn is a significant source of overhead in Whanau, requiring all of its routing tables to be rebuilt. However, in our scenario, only the values corresponding to the keys change quickly, but not the keys themselves, requiring only a modest amount of heartbeat traffic [1]. Considering the Facebook wall post topology, we estimate the mean communication overhead *per time interval* to be only about 6 MB.

We also evaluate the latency of constructing long onion routing circuits, using experiments over the live Tor network. We used the Torflow utility to build Tor circuits with varying circuit lengths; Figure 7.9 depicts our experimental results. Using this analysis, we estimate that 25 hop circuits would take about 1 minute to construct.

7.4 Anonymity

Earlier, we considered the probability of end-to-end timing analysis as our metric for anonymity. This metric considers the scenario where the adversary has exactly de-anonymized the user. However, in random walk based anonymous communication, the adversary may sometimes have probabilistic knowledge of the initiator. To quantify all possible sources of information leaks, we now consider the entropy metric to quantify anonymity [92, 93]. The entropy metric considers the *probability distribution* of nodes being possible initiators, as computed by the attackers. In this paper, we will restrict our analysis to Shannon entropy, since it is the most widely used mechanism for analyzing anonymity. There are other ways of computing entropy, such

as guessing entropy [153] and min entropy [154], which we will consider in the full version of this work. Shannon entropy is computed as:

$$H(I) = \sum_{i=0}^{i=n} -p_i \cdot \log_2(p_i), \tag{7.10}$$

where p_i is the probability assigned to node i of being the initiator of a circuit. Given a particular observation o, the adversary can first compute the probability distribution of nodes being potential initiators of circuits $p_i|o$ and then the corresponding conditional entropy $H(I|o)$. We can model the entropy distribution of the system as a whole by considering the weighted average of entropy for each possible observation, including the null observation.

$$H(I|O) = \sum_{o \in O} P(o) \cdot H(I|o). \tag{7.11}$$

Now, we first consider the scenario where an initiator performs an l hop random walk to communicate with a malicious destination, and the nodes in the random walk are all honest, i.e., the adversary is external to the system. For this scenario, we will analyze the expected initiator anonymityunder the conservative assumption that the adversary is aware of the complete social network graph.

7.4.1 Malicious destination: expected entropy

A naive way to compute initiator entropy for this scenario is to consider the set of nodes that are reachable from the terminus of the random walk in exactly l hops (the adversary's observation), and assign a uniform probability to all nodes in that set of being potential initiators. However, such an approach does not consider the mixing characteristics of the random walk; l hop random walks starting at different nodes may in fact have heterogeneous probabilities of terminating at a given node.

We now outline a strategy that explicitly considers the mixing characteristics of the trust topology. Let the terminus of an l hop random walk be node j. The goal of the adversary is to compute probabilities p_i of nodes being

potential initiators of the circuit. Observe that:

$$p_i = \frac{P^l_{ij}}{\sum_x P^l_{xj}}, \tag{7.12}$$

where P^l denotes the l hop transition matrix for the random walk process. Note that even for moderate size social graphs, the explicit computation of P^l is infeasible in terms of both memory and computational constraints. This is because even though P is a sparse matrix, iterative multiplication of P by itself results in a matrix that is no longer sparse. To make the problem feasible, we propose to leverage the *time reversibility* of our random walk process. We have previously modeled the random walk process as a Markov chain. Markov chains which satisfy the following property are known as time reversible Markov chains [155].

$$\pi_i \cdot P_{ij} = \pi_j \cdot P_{ji}. \tag{7.13}$$

It is trivial to see that both the conventional random walk and the Metropolis-Hastings random walk satisfy the above property and are thus time reversible Markov chains. It follows from time reversibility [155], that:

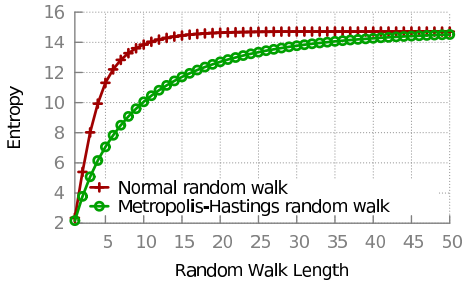
$$\pi_i \cdot P^l_{ij} = \pi_j \cdot P^l_{ji} \implies P^l_{ij} = \frac{\pi_j}{\pi_i} \cdot P^l_{ji}. \tag{7.14}$$

Thus it is possible to compute P^l_{ij} using P^l_{ji}. Let V_j be the initial probability vector starting at node j. Then the probability of an l hop random walk starting at node j and ending at node i can be computed as the i'th element of the vector $V_j \cdot P^l$. Observe that $V_j \cdot P^l$ can be computed without computing the matrix P^l as follows:

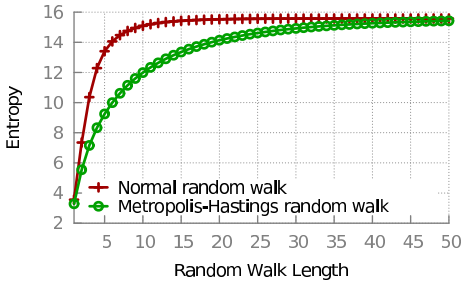
$$V_j \cdot P^l = (V_j \cdot P) \cdot P^{l-1}. \tag{7.15}$$

Since P is a sparse matrix, $V_j \cdot P$ can be computed in O(n) time, and $V_j \cdot P^l$ can be computed in $O(nl)$ time. Finally, we can compute the probabilities

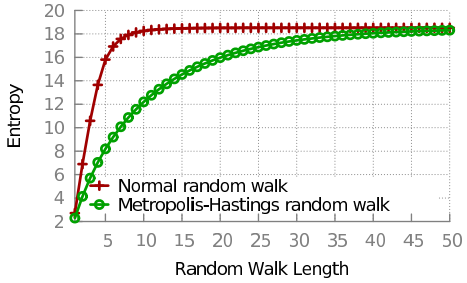
of nodes being potential initiators of circuits using Equation (7.12), and the corresponding entropy gives us the initiator anonymity under this scenario. We average the resulting entropy over 100 random choices of the terminal node j to compute the expected anonymity.



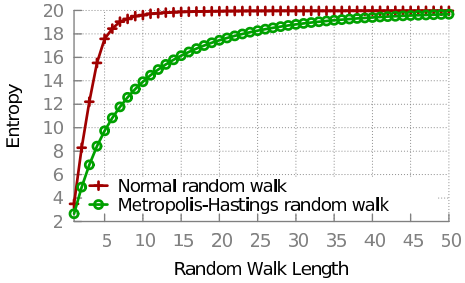
(a) Facebook wall post graph



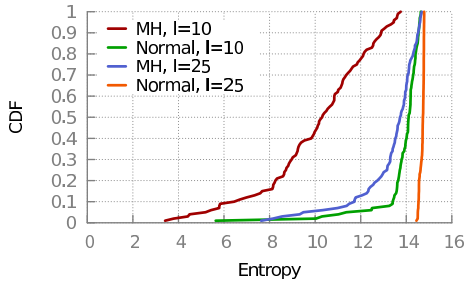
(b) Facebook link graph



(c) Anonymous Interaction graph



(d) Anonymous link graph



(e) CDF of entropy for Facebook wall graph

Figure 7.10: *Expected entropy as a function of random walk length.* We can see that the entropy of the Metropolis-Hastings random walk is less than the conventional random walk due to its slower mixing properties. However, even the Metropolis-Hastings random walks quickly converge to the stationary distribution. From the CDF, we also note that an overwhelming fraction of users can expect a high level of anonymity.

Figure 7.10(a)-(d) depicts the expected initiator anonymity as a function of random walk length for different social network topologies. We can see that longer random walks result in an increase in anonymity. This is because

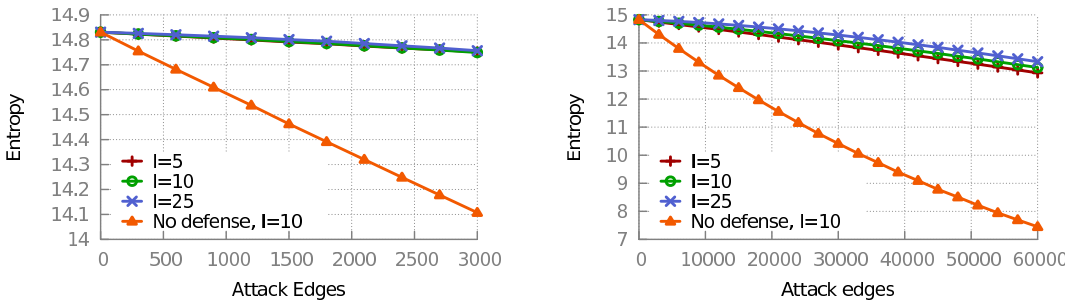
for short random walks of length l in restricted topologies such as trust networks, not every node can reach the terminus of the random walk in l hops. Secondly, even for nodes that can reach the terminus of the random walk in l hops, the probabilities of such a scenario happening can be highly heterogeneous. Furthermore, we can see that conventional random walks converge to optimal entropy in about 10 hops for all four topologies. In contrast, the Metropolis-Hastings random walks used in Pisces take longer to converge. This is because random walks in Pisces have slower mixing properties than conventional random walks. However, we can see that even the Metropolis-Hastings random walk starts to converge after 25 hops in all scenarios.

To get an understanding of the distribution of the entropy, we plot the CDF of entropy over 100 random walk samples in Figure 7.10(e). We can see that the typical anonymity offered by moderately long random walks is quite high. For example, using a Metropolis-Hastings random walk of length 25, 95% of users get an entropy of at least 11 bits.

So far, we observed the effect of mixing time of social network topologies on initiator anonymity. We found that Metropolis-Hastings random walks need to be longer than conventional random walks for equivalent level of anonymity against a malicious destination. Next, we will see the benefit of using Metropolis-Hastings random walks in Pisces, since these walks can be secured against an insider adversary.

7.4.2 Insider adversary

We now analyze the anonymity of the system with respect to an overlay level insider adversary. We first consider an adversary which has g attack edges to honest nodes, with $g = O(\frac{h}{\log h})$, and 10 sybils per attack edge. When both the first and the last hop of a random walk are compromised, then initiator entropy is 0 due to end-to-end timing analysis. Let M_i be the event where the first compromised node is at the ith hop and the last hop is also compromised. Suppose that the previous hop of the first compromised node is node A. Under this scenario, the adversary can localize the initiator to the set of nodes that can reach the node A in $i-1$ hops. If we denote the initiator anonymity under this scenario as $H(I|M_i)$, then from Equation (7.11), it



(a) (b)

Figure 7.11: *Entropy as a function of fraction of attack edges using* (a) realistic model of an imperfect Sybil defense (10 Sybils per attack edge) and (b) perfect Sybil defense for Facebook wall post interaction graph.

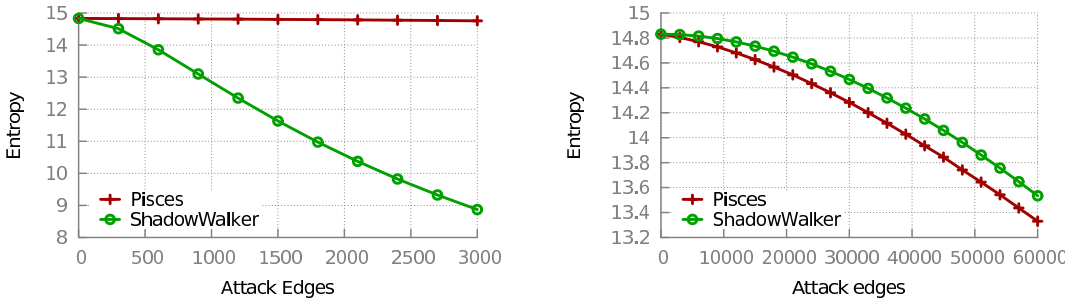
follows that the overall system anonymity is:

$$H(I|O) = \sum_{i=1}^{i=l} P(M_i) \cdot H(I|M_i) + (1 - \sum_{i=1}^{i=l} P(M_i)) \cdot \log_2 n. \qquad (7.16)$$

We compute $P(M_i)$ using simulations, and $H(I|M_i)$, using the expected anonymity computations discussed above. Figure 7.11(a) depicts the expected entropy as a function of the number of attack edges. We find that Pisces provides close to optimal anonymity. Moreover, as the length of the random walk increases, the anonymity does not degrade. In contrast, without any defense, the anonymity decreases with an increase in the random walk length (not shown in the figure), since at every step in the random walk, there is a chance of the random walk being captured by the adversary. At $g = 3000$, the anonymity provided by a conventional 10 hop random walk without any defenses is 14.1 bits, while Pisces provides close to optimal anonymity at 14.76 bit (anonymity set increases by a factor of 1.6).

It is also interesting to see that the advantage of using Pisces increases as the number of attack edge increases. To better investigate this, we consider the attack model with perfect Sybil defense and vary the number of attack edges. Figure 7.11(b) depicts the anonymity as a function of the number of attack edges. We can see that at 60 000 attack edges, the expected anonymity without defenses is 7.5 bits, as compared to more than 13 bits with Pisces (anonymity set increases by a factor of 45).

7.4.3 Comparison with ShadowWalker



(a) (b)

Figure 7.12: *Comparison with ShadowWalker. Entropy as a function of fraction of attack edges using* (a) realistic model of an imperfect Sybil defense (10 Sybils per attack edge) and (b) perfect Sybil defense for Facebook wall post interaction graph.

ShadowWalker [118] is a state of art approach for scalable anonymous communication that organizes nodes into a structured topology such as DeBruijn, and performs secure random walks on such topologies. We now compare our approach with ShadowWalker.

To compute the anonymity provided by ShadowWalker, we use the fraction f of malicious nodes in the system as an input to the analytic model of ShadowWalker [118], and use ShadowWalker parameters that provide maximum security. Figure 7.12(a) depicts the comparative results between Pisces (using $l = 25$) and ShadowWalker. We can see that Pisces significantly outperforms ShadowWalker. At $g = 1000$ attack edges, Pisces provides about 2 bits higher entropy than ShadowWalker, and this difference increases to 6 bits at $g = 3000$ attack edges.[3] This difference arises because Pisces directly performs random walks on the social network topology, limiting the impact of Sybil attackers, while ShadowWalker is designed to secure random walks only on structured topologies. Arranging nodes in a structured topology loses information about trust relationships between users, resulting in poor anonymity for ShadowWalker.

For comparison, we also consider the attack model with perfect Sybil defense and vary the number of attack edges. Figure 7.12(b) depicts the results

[3]At such high attack edges, ShadowWalker may even have difficulty in securely maintaining its topology, which could further lower anonymity.

for this scenario. We can see that even in this scenario where trust relationships lose meaning since the adversary is randomly distributed, Pisces continues to provides comparable anonymity to ShadowWalker. Pisces entropy is slightly lower since social networks are slower mixing than structured networks, requiring longer length random walks than ShadowWalker, which gives more observation points to the adversary.

7.4.4 Performance optimization

We now analyze the anonymity provided by our two hop optimization, which uses the kth hop and last hop of the random walk for anonymous communication. To analyze anonymity in this scenario, let us redefine M_i $(i \neq k)$ as the event when the first compromised node is at the ith hop, the last node is also compromised, but the kth node is honest. Let M_k be the event where the kth hop and the last hop is compromised (regardless of whether other nodes are compromised or not) and the definition of M_l remains the same as before, i.e., only the last hop is compromised. We can compute system anonymity as:

$$H(I|O) = \sum_{i=1}^{i=k-1} P(M_i) \cdot H(I|M_i) + \sum_{i=k+1}^{l} P(M_i) \cdot H(I|M_k) + (1 - \sum_{i=1}^{i=l} P(M_i)) \cdot \log_2 n. \quad (7.17)$$

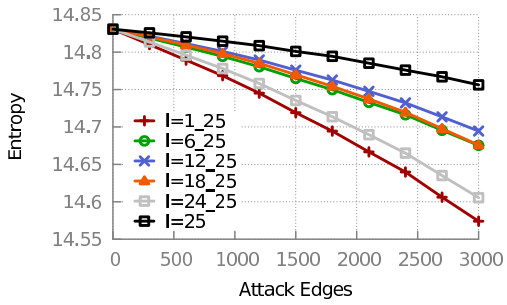


Figure 7.13: Anonymity using the two hop performance optimization, Facebook wall post interaction graph. k=12 results in provides a good tradeoff between anonymity and performance.

Figure 7.13 depicts the anonymity for our two hop optimization for different choices of k. We see a very interesting tradeoff here. Small values of k are not optimal, since even though the first hop is more likely to be honest, when the last hop is compromised, then the initiator is easily localized. On the other hand, large values of k are also not optimal, since these nodes are far away from the initiator in the trust graph and are less trusted. We find that optimal tradeoff points are in the middle, with $k = 12$ providing the best anonymity for our optimization. We also note that the anonymity provided by our two hop optimization is close to the anonymity provided by using all 25 hops of the random walk for anonymous communication.

7.4.5 Selective denial of service

Next, we evaluate Pisces anonymity against the selective DoS attack [54]. In this attack, an adversary can cause a circuit to selectively fail whenever he or she is unable to learn the initiator identity. This forces the user to construct another circuit, which results in a degradation of anonymity. We found that the degradation in initiator anonymity under this attack is less than 1%.

7.4.6 Anonymity over multiple communication rounds

So far, we had limited our analysis to a single communication round. Next, we analyze system anonymity over multiple communication rounds. Let us suppose that in communication rounds $1 \dots z$, the adversaries observation is $O_1 \dots O_z$. Let us denote a given node's probability of being the potential initiators after z communication rounds by $P(I = i|O_1, \dots, O_z)$. Now, after communication round $z + 1$, we are interested in computing the probability $P(I = i|O_1, \dots, O_{z+1})$. Using Bayes' theorem, we have that:

$$P(I = i|O_1, \dots, O_{z+1}) = \frac{P(O_1, \dots, O_{z+1}|I = i) \cdot P(I = i)}{P(O_1, \dots, O_{z+1})}. \tag{7.18}$$

The key advantage of this formulation is that we can now leverage the observations $O_1, \dots O_{z+1}$ being independent given a choice of initiator. Thus we have that:

$$P(I=i|O_1,\ldots,O_{z+1}) = \frac{\Pi_{j=1}^{j=z+1} P(O_j|I=i)\cdot P(I=i)}{P(O_1,\ldots,O_{z+1})}$$
$$= \frac{\Pi_{j=1}^{j=z+1} P(O_j|I=i)\cdot P(I=i)}{\sum_{p=1}^{p=h} P(O_1,\ldots,O_{z+1}|I=p)\cdot P(I=p)}$$
$$= \frac{\Pi_{j=1}^{j=z+1} P(O_j|I=i)\cdot P(I=i)}{\sum_{p=1}^{p=h} \Pi_{j=1}^{j=z+1} P(O_j|I=p)\cdot P(I=p)}. \quad (7.19)$$

Finally, assuming a uniform prior over all possible initiators, we have that:

$$P(I=i|O_1,\ldots,O_{z+1}) = \frac{\Pi_{j=1}^{j=z+1} P(O_j|I=i)}{\sum_{p=1}^{p=h} \Pi_{j=1}^{j=z+1} P(O_j|I=p)}. \quad (7.20)$$

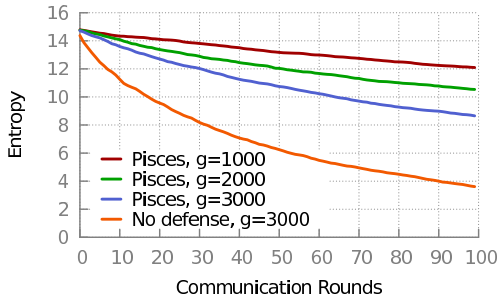


Figure 7.14: Anonymity degradation over multiple communication rounds, Facebook wall post interaction graph.

Figure 7.14 depicts the expected anonymity as a function of number of communication rounds. We can see that the entropy provided by Pisces outperforms conventional random walks by more than a factor of 2 after 100 communication rounds.

7.5 Discussion

Integration with Sybil defenses: We outline two strategies for integrating Pisces with Sybil defense. The first approach is to leverage mechanisms that require the whole social graph as input for Sybil defense, such as SybilInfer. This has the downside of communication overhead for reliably maintaining the social graph accurately, in presence of social graph churn such as new

users and new trust relationships. The upside of this approach is that *while* performing random walks, both for anonymous communication and for testing, no further communication is required to validate identities for Sybil defense. The second approach is to leverage decentralized mechanisms like SybilLimit that do not require users to maintain global information about the social graph. However, in this scenario, while performing random walks, each hop of the random walk must be validated for Sybil defense. A key challenge in validating nodes while performing random walks is to prevent other entities in the network from learning the nodes involved in the random walk performed by an initiator. Towards this end, we propose that the node being validated return its list of SybilLimit *tails* to the initiator using the partial onion routing circuit, who can then perform a privacy preserving set intersection protocol with its tails to perform Sybil defense.

Limitations: While Pisces is the first design that can scalably leverage social network trust relationships while mitigating route capture attacks, its architecture has several limitations. First, circuit establishment in Pisces has higher latency than existing systems, since random walks in Pisces tend to be longer. However, we note that circuits can be established pre-emptively, such that this latency does not affect the user. In fact, deployed systems such as Tor already build circuits preemptively. Second, some users that are not well connected in the social network topology may not benefit from using Pisces since reasonable length random walks starting from those nodes may not converge to the stationary probability distribution quickly. Third, Pisces currently does not support important constraints such as bandwidth based load balancing and exit policies. The focus of our architecture was to secure the peer discovery process in unstructured social network topologies, and we will consider the incorporation of these constraints in future work.

7.6 Summary

We proposed a mechanism for scalable anonymous communication that can securely leverage a user's trust relationships. Our key insight is that appearance of nodes in each other's neighbor lists can be made reciprocal. Using theoretical analysis and experiments on real world social network topologies, we demonstrate that Pisces substantially reduces the probability of active

attacks on circuit constructions.

CHAPTER 8

CONCLUSION

We have shown that lookup mechanisms in DHTs are not anonymous, and reveal information about the lookup initiator. To secure the lookup mechanism itself, redundant messages are used which enables a relatively small fraction of attackers to observe a large number of lookups in the network. We have shown how attacks based on information leaks from lookups can be used to break the anonymity guarantees of low-latency P2P anonymous communication systems like Salsa. Salsa was previously reported to tolerate up to 20% compromised nodes, but our results show that, with information leaks taken into account, over a quarter of all circuits are compromised. Our results demonstrate that information leaks are an important part of anonymity analysis of a system. In followup work [67], we showed that even newly proposed designs such as NISAN [66] and Torsk [25] are vulnerable to information leaks. An important question for future work is the design of analysis tools that can automaticall quantify information leaks in a system and aid the system designers towards the design of more secure systems.

We proposed ShadowWalker, a new design for low-latency P2P anonymous communication. ShadowWalker is able to effectively defend against common attacks on peer-to-peer systems (including the information leak attack discussed above) and achieve levels of anonymity superior to the state of the art in P2P anonymous communication. In particular, when 20% of all nodes are compromised, ShadowWalker provides 4.5 bits more entropy than Salsa. Moreover, the probability of end-to-end timing analysis in this case is less than 5%, which is close to the ideal scenario as in Tor, where the probability of end-to-end timing analysis is 4%. Our system presents several tradeoffs between anonymity and performance overhead. We have demonstrated points along these tradeoffs that have manageable computation and communication overheads while providing strong anonymity guarantees. ShadowWalker is also able to handle moderate churn in the network. As such, it presents a

promising new direction for P2P anonymous communication. In future work, we will also investigate the benefits of ShadowWalker's redundant structured topology design beyond anonymity systems.

We presented SybilInfer, an algorithm aimed at detecting Sybil attacks against peer-to-peer networks or open services, and label which nodes are honest and which are dishonest. Its applicability and performance in this task is an order of magnitude better than previous systems making similar assumptions, like SybilGuard and SybilLimit, even though it requires nodes to know a substantial part of the social structure within which honest nodes are embedded. SybilInfer illustrates how robust Sybil defences can be bootstrapped from distributed trust judgements, instead of a centralised identity scheme. This is a key enabler for secure peer-to-peer architectures as well as collaborative web 2.0 applications. SybilInfer is also significant due to the use of machine learning techniques and their careful application to a security problem. Cross-disciplinary designs are a challenge, and applying probabilistic techniques to system defence should not be at the expense of strength of protection, and strategy-proof designs. Our ability to demonstrate that the underlying mechanisms behind SybilInfer is not susceptible to gaming by an adversary arranging its Sybil nodes in a particular topology is, in this aspect, a very important part of the SybilInfer security design.

We describe X-Vine, a protection mechanism for DHTs that is resilient to the Sybil attack. X-Vine operates entirely by communicating over social network links, using network layer DHT techniques. X-Vine requires $O(\log n)$ state, two orders of magnitude less in practical settings as compared with existing techniques like Whanau [1], making it particularly suitable for large-scale and dynamic environments. X-Vine also enhances privacy by not revealing social relationship information and by providing a basis for pseudonymous communication. A key limitation of X-Vine is that it assumes that malicious identities are uniformly distributed in the DHT namespace; mechanisms that enforce this assumption would be an important direction for future work.

The final component of this dissertation is Pisces, a mechanism for P2P anonymous communication that can securely leverage a user's trust relationships. Pisces leverages social network based Sybil defense mechanisms such as SybilInfer and X-Vine. Our key insight is that appearance of nodes in each other's neighbor lists can be made reciprocal, in order to defend against

active attacks on the system. Using real world social network topologies and a reasonable set of trust assumptions, we find that Pisces significantly outperforms ShadowWalker, and provides up to 6 bits higher entropy in a single communication round. Also, compared with the naive strategy of using conventional random walks over social networks (as in the Drac system [59]), Pisces provides twice the amount of entropy over 100 communication rounds. In future work, we will investigate the open problem of incorporating the issues of heterogeneous node bandwidth and exit policies.

APPENDIX A

MATHEMATICAL ANALYSIS OF X-VINE:

As a first step in formally modeling X-Vine security, we develop an analytic model for routing in X-Vine. The model enhances our understanding of the relationship between operational parameters of X-Vine, and can serve as a stepping stone for a complete formal model to analyze X-Vine's security against Sybil identities.

Let there be N nodes in the system with node identifiers ranging from $0..N-1$. Let $L(0,w)$ be the expected lookup path length between the node with identifier 0 and w. Let us suppose that node maintain trails with a single successor. Without loss of generality, the average lookup path length can be computed as follows:

$$E(L) = \frac{\sum_{w=0}^{w=N-1} L(0,w)}{N}. \tag{A.1}$$

In the simplest case, $L(0,0) = 0$. Let us first compute $L(0,1)$. Note that node 0 and node 1 have a *trail* between them because they are overlay neighbors. Let d be the average node degree in the underlying topology. We assume that the length of the trail between overlay neighbors is close to their shortest path in the social network topology (approximately $\log_d(N)$). The lookup will proceed from node 0 along the trail to node 1. Thus we have that:

$$L(0,1) = \text{Expected trail length} \tag{A.2a}$$

$$L(0,1) = \log_d(N). \tag{A.2b}$$

Notice that there cannot be any shortcutting in the intermediate nodes on the trail path from node 0 to node 1 because we have assumed the trail to be

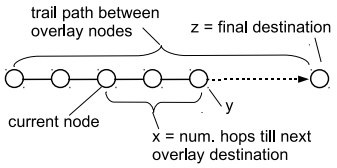


Figure A.1: X-Vine lookup.

the shortest path in the social network topology. Let us now compute $L(0,2)$. There are two possible scenarios. In the first case, there may be a trail with an end point at node 2 *going through* node 0. In this case, the packet is routed along the trail to node 2. Again, there cannot be any shortcutting along this trail because it is the shortest path. The mean path length in this case is $\frac{\log_d N}{2}$. In the second case, the packet will be routed towards overlay node 1 (successor of node 0). Thus we have that:

$$L(0,2) = P(\text{trail}) \cdot \frac{\log_d N}{2} + (1 - P(\text{trail})) \cdot (1 + l((\log_d N) - 1, 1, 2)), \quad \text{(A.3a)}$$

where $l(x, y, z)$ is defined as the expected path length when z is the destination identifier, y is the next overlay hop in the lookup, and x is the physical distance along a trail between the current node and y (Figure A.1). This means that $l((\log_d N) - 1, 1, 2)$ is the expected path length when the destination identifier is 2, the next overlay hop is 1, and the next overlay hop is $\log_d N$ hops away from the current node.

Note that each node maintains a trail to its successor, and the mean length of the trails is $\log_d(N)$. This means that the average routing state maintained by a node in the system is $\log_d(N)$. Since each routing table entry specifies two end points for a trail, the probability that a node has a trail with a particular end point going through it is $\frac{2 \log_d N}{N}$. Thus:

$$L(0,2) = \frac{2 \log_d N}{N} \cdot \frac{\log_d N}{2} + \left(1 - \frac{2 \log_d N}{N}\right) \cdot (1 + l((log_d N) - 1, 1, 2)). \quad \text{(A.3b)}$$

We now need to compute $l(x, 1, 2)$. Similar to our computation of $L(0, 2)$, again, there are three possible scenarios. In the first case, the current node (say A) is a friend of node 2. Then the path length is 1. In the second case, there is a trail with an end point at node 2 going through node A. In this case, the mean path length is $\frac{\log_d(N)}{2}$. In the third case, the packet continues along the current trail to node 1.

$$l(x,1,2) = \frac{2\log_d N}{N} \cdot \left(\frac{\log_d N}{2}\right) + \left(1 - \frac{2\log_d N}{N}\right) \cdot (1 + l(x-1,1,2)). \quad \text{(A.4)}$$

Here, the boundary conditions for terminating the recursion are as follows:

$$l(x,1,1) = x \text{ if } 0 \le x \le \log_d N \quad \text{(A.5a)}$$
$$l(x,z,z) = x \text{ if } 0 \le x \le \log_d N, 1 \le z \le N-1 \quad \text{(A.5b)}$$
$$l(0,y,z) = L(y,z) = L(0,(z-y)) \text{ if } 1 \le y \le z \le N-1. \quad \text{(A.5c)}$$

Let us now compute $L(0, w)$. Consider the following two scenarios. In the first case, let the closest preceding node in node 0's routing table be node i (shortcut to $i \neq 1$). Now node i may either be a friend of node 0, in which case, the path length is $1 + L(i, w)$, or node i may be the end point of a trail going through node 0, in which case, the path length is $1 + l\left(\frac{\log_d N}{2} - 1, i, w\right)$. In the second case, there is no shortcutting, and the lookup proceeds towards the next overlay hop node 1. Thus, we have that:

$$L(0,w) = \sum_{i=2}^{w} P(\text{shortcut to i}) \cdot P(\text{shortcut via friend}) \cdot (1 + L(i,w)) + \sum_{i=2}^{w} P(\text{shortcut to i}) \cdot P(\text{shortcut via trail}) \cdot \left(1 + l\left(\frac{\log_d N}{2} - 1, i, w\right)\right) + P(\text{no shortcut}) \cdot (1 + l((\log_d N) - 1, 1, w)). \quad \text{(A.6)}$$

Let us now compute the probability of shortcutting to a node i. The probability of shortcutting to node w is simply $\frac{d+2\log_d N}{N}$. The probability of shortcutting to node $w-1$ can be computed as $P(\text{no shortcut to } w) \cdot P(\text{shortcut to } w-1 | \text{ no shortcut to } w)$. This is equal to $\left(1 - \frac{d+2\log_d N}{N}\right) \cdot \frac{d+2\log_d N}{N-1}$. Similarly, we can compute the probability of shortcutting to node i as:

$$P(\text{shortcut to i}) = P(\text{no shortcut to w..i+1}) \cdot \frac{d+2\log_d N}{N-(w-i)} \tag{A.7a}$$

$$P(\text{no shortcut to w..j}) = P(\text{no shortcut from w..j+1}) \cdot \left(1 - \frac{d+2\log_d N}{N-(w-j)}\right). \tag{A.7b}$$

Now, given that the lookup shortcuts towards overlay hop node i, it may do so because of a friendship entry in the routing table, or a trail in the routing table. The probability that the shortcut happened via a friend entry, $P(\text{shortcut via friend}) = \frac{d}{d+2\log_d(N)}$. The probability that the shortcut happened because of a X-Vine entry is
$P(\text{shortcut via trail}) = \frac{2\log_d(N)}{d+2\log_d(N)}$. Thus, we can rewrite Equation (A.6) as

$$\begin{aligned} L(0,w) = &\sum_{i=2}^{w} P(\text{shortcut to i}) \cdot \frac{d}{d+2\log_d N} \cdot (1 + L(i,w)) \\ &+ \sum_{i=2}^{w} P(\text{shortcut to i}) \cdot \frac{2\log_d N}{d+2\log_d N} \cdot \left(1 + l\left(\frac{\log_d N}{2} - 1, i, w\right)\right) \\ &+ P(\text{no shortcutting}) \cdot (1 + l((\log_d N) - 1, 1, w)). \end{aligned} \tag{A.8}$$

Similar to the above analysis, we can compute $l(x,i,w)$ as follows:

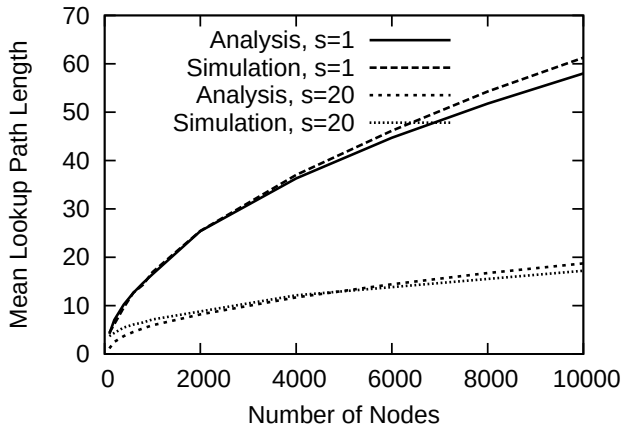


Figure A.2: Validation of analytic model using $d = 10$.

$$l(x,j,w) = \sum_{i=2}^{j+1} P(\text{shortcut to i}) \cdot \frac{d}{d + 2\log_d N} \cdot (1 + L(i,w)) + \sum_{i=2}^{j+1} P(\text{shortcut to i}) \cdot \frac{2\log_d N}{d + 2\log_d(N)} \cdot \left(1 + l\left(\frac{\log_d N}{2} - 1, i, w\right)\right) + P(\text{no shortcutting}) \cdot (1 + l(x-1,j,w)). \quad \text{(A.9)}$$

The boundary conditions for the termination of recursion are the same as in Equation (A.5).

Validation of analytic model: Figure A.2 plots the mean lookup path length as a function of number of nodes for a synthetic scale-free topology with average degree $d = 10$ using a redundancy parameter of $r = 1$. We can see that the results of simulation are a very close match with our analytic model, increasing confidence in our results. We note that our analytic model has implications for modeling network layer DHTs like VRR.

APPENDIX B

X-VINE PSEUDOCODE

We now describe pseudocode for X-Vine's lookup and trail path setup mechanisms.

Algorithm 1 *Fwd_lookup(identifier myid, message M)*: Determines next hop for a lookup message.

```
bestroute=0
foreach element E in RoutingTable
  if distance(E.endpoint,M.dest)<
              distance(bestroute,M.dest)
    bestroute=E
endfor
return bestroute
```

Algorithm 2 *Fwd_trailsetup(identifier myid, message M):* Determines next hop for trail path setup message.

```
bestroutes=∅
/* select all routes that make progress */
foreach element E in RoutingTable
  if distance(E.endpoint,M.dest)<distance(myid,M.dest)
    bestroutes.insert(E)
endfor
/* of these, discard (a) backtracked routes, (b) routes that have reached
bounds, (c) routes that don't make namespace progress compared to
M.nextovlhop*/
foreach element E in bestroutes
  if failed_set.contains(E.endpoint,E.nexthop) or
                (E.nexthop.numtrails > b_n) or
                (numtrailsto(E.nexthop) > b_l) or
                (distance(E.endpoint,M.dest) <
                  distance(M.nextovlhop,M.dest)
    bestroutes.remove(E)
endfor
/* if no remaining options, backtrack */
if bestroutes == ∅
  send_reject_to(M.prevhop)
  return
/* of remaining routes, select route with maximum namespace progress */
routetouse=0
foreach element E in bestroutes
  if distance(E.endpoint,M.dest)<
              distance(routetouse,M.dest)
    routetouse=E
endfor
return routetouse
```

REFERENCES

[1] C. Lesniewski-Laas and M. F. Kaashoek, "Whanau: a sybil-proof distributed hash table," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855711.1855719 pp. 8–8.

[2] C. Williams, "BT admits misleading customers over Phorm experiments," *The Register*, March 17 2008.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251375.1251396 pp. 21–21.

[4] K. Loesing, "Measuring the Tor network: Evaluation of client requests to the directories," The Tor project, Tech. Rep., 2009.

[5] The Tor Project, "Tor metrics portal, April 2011," http://metrics.torproject.org/.

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

[7] "Tor network status," http://torstatus.blutmagie.de/, accessed April 2011.

[8] J. R. Douceur, "The Sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687813 pp. 251–260.

[9] A. Nambiar and M. Wright, "Salsa: a structured approach to large-scale anonymity," in *Proceedings of the 13th ACM conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180409 pp. 17–26.

[10] G. Danezis and R. Clayton, "Route fingerprinting in anonymous communications," in *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1157740.1158240 pp. 69–72.

[11] G. Danezis and P. Syverson, "Bridging and fingerprinting: Epistemic attacks on route selection," in *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, ser. PETS '08. Berlin, Heidelberg: Springer-Verlag, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70630-4_10 pp. 151–166.

[12] P. Mittal and N. Borisov, "Information leaks in structured peer-to-peer anonymous communication systems," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455805 pp. 267–278.

[13] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against Sybil attacks via social networks," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1159913.1159945 pp. 267–278.

[14] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by DHTs," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1159913.1159954 pp. 351–362.

[15] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type iii anonymous remailer protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, ser. SP '03. Washington, DC, USA: IEEE Computer Society, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=829515.830555 pp. 2–.

[16] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol — Version 2," IETF Internet Draft, July 2003.

[17] J. Boyan, "The anonymizer: Protecting user privacy on the web," *Computer-Mediated Communication Magazine*, vol. 4, no. 9, 1997.

[18] I2P, "I2P anonymous network," http://www.i2p2.de/index.html, 2003.

[19] P. Boucher, A. Shostack, and I. Goldberg, "Freedom systems 2.0 architecture," Zero Knowledge Systems, Inc., White Paper, December 2000.

[20] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: a distributed anonymous information storage and retrieval system," in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. New York, NY, USA: Springer-Verlag New York, Inc., 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=371931.371977 pp. 46–66.

[21] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Proceedings of the First International Workshop on Information Hiding*. London, UK: Springer-Verlag, 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=647594.731526 pp. 137–150.

[22] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, ser. SP '97. Washington, DC, USA: IEEE Computer Society, 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=882493.884368 pp. 44–.

[23] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Commun. ACM*, vol. 42, pp. 39–41, February 1999. [Online]. Available: http://doi.acm.org/10.1145/293411.293443

[24] H. Federrath, "Project: An.on - anonymity.online: Protection of privacy on the Internet," http://anon.inf.tu-dresden.de/index_en.html.

[25] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable onion routing with Torsk," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653733 pp. 590–599.

[26] "Tor blog," https://blog.torproject.org/blog/tor-project-infrastructure-updates.

[27] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg, "PIR-Tor: scalable anonymous communication using private information retrieval," in *Proceedings of the 20th USENIX conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028098 pp. 31–31.

[28] J.-F. Raymond, "Traffic analysis: protocols, attacks, design issues, and open problems," in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. New York, NY, USA: Springer-Verlag New York, Inc., 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=371931.371972 pp. 10–29.

[29] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004, pp. 207–225.

[30] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *Proceedings of ESORICS 2006*, September 2006.

[31] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. New York, NY, USA: Springer-Verlag New York, Inc., 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=371931.371981 pp. 96–114.

[32] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC '04)*, A. Juels, Ed. Springer-Verlag, LNCS 3110, February 2004, pp. 251–265.

[33] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, pp. 66–92, November 1998. [Online]. Available: http://doi.acm.org/10.1145/290163.290168

[34] M. Wright, M. Adler, B. N. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols," in *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.

[35] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, ser. SP '03. Washington, DC, USA: IEEE Computer Society, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=829515.830556 pp. 28–.

[36] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous

communications systems," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 489–522, November 2004. [Online]. Available: http://doi.acm.org/10.1145/1042031.1042032

[37] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "Passive-logging attacks against anonymous communications systems," *ACM Trans. Inf. Syst. Secur.*, vol. 11, pp. 3:1–3:34, May 2008. [Online]. Available: http://doi.acm.org/10.1145/1330332.1330335

[38] O. Berthold, H. Federrath, and M. Köhntopp, "Project ȧnonymity and unobservability in the Internet," in *Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions*, ser. CFP '00. New York, NY, USA: ACM, 2000. [Online]. Available: http://doi.acm.org/10.1145/332186.332211 pp. 57–65.

[39] D. Kesdogan, D. Agrawal, and S. Penz, "Limits of anonymity in open environments," in *Revised Papers from the 5th International Workshop on Information Hiding*, ser. IH '02. London, UK, UK: Springer-Verlag, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=647598.731881 pp. 53–69.

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

[41] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," in *Proceedings of Privacy Enhancing Technologies Workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004, pp. 17–34.

[42] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1058433.1059390 pp. 183–195.

[43] A. Back, U. Möller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Proceedings of the 4th International Workshop on Information Hiding*, ser. IHW '01. London, UK, UK: Springer-Verlag, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=647597.731866 pp. 245–257.

[44] N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on Tor using long paths," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09.

Berkeley, CA, USA: USENIX Association, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855768.1855771 pp. 33–50.

[45] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Proceedings of the 15th European Conference on Research in Computer Security*, ser. ESORICS'10. Berlin, Heidelberg: Springer-Verlag, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888881.1888901 pp. 249–267.

[46] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046732 pp. 215–226.

[47] S. J. Murdoch, "Hot or not: revealing hidden services by their clock skew," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180410 pp. 27–36.

[48] S. Zander and S. J. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," in *Proceedings of the 17th Conference on Security Symposium*. Berkeley, CA, USA: USENIX Association, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496711.1496726 pp. 211–225.

[49] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315257 pp. 82–91.

[50] N. Hopper, E. Y. Vasserman, and E. Chan-TIN, "How much anonymity does network latency leak?" *ACM Trans. Inf. Syst. Secur.*, vol. 13, pp. 13:1–13:28, March 2010. [Online]. Available: http://doi.acm.org/10.1145/1698750.1698753

[51] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '04. New York, NY, USA: ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/1029179.1029199 pp. 66–76.

[52] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, ser. PET'07. Berlin, Heidelberg: Springer-Verlag, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1779330.1779341 pp. 167–183.

[53] M. Edman and P. Syverson, "As-awareness in Tor path selection," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653708 pp. 380–389.

[54] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security?" in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315258 pp. 92–102.

[55] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: ACM, 2002. [Online]. Available: http://doi.acm.org/10.1145/586110.586137 pp. 193–206.

[56] M. Rennhard and B. Plattner, "Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection," in *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '02. New York, NY, USA: ACM, 2002. [Online]. Available: http://doi.acm.org/10.1145/644527.644537 pp. 91–102.

[57] P. Tabriz and N. Borisov, "Breaking the collusion detection mechanism of MorphMix," in *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*, G. Danezis and P. Golle, Eds. Cambridge, UK: Springer, June 2006, pp. 368–384.

[58] S. Nagaraja, "Anonymity in the wild: mixes on unstructured networks," in *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, ser. PET'07. Berlin, Heidelberg: Springer-Verlag, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1779330.1779346 pp. 254–271.

[59] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie, "Drac: an architecture for anonymous low-volume communications," in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1881151.1881163 pp. 202–219.

[60] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001. [Online]. Available: http://doi.acm.org/10.1145/383059.383071 pp. 149–160.

[61] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware '01. London, UK: Springer-Verlag, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=646591.697650 pp. 329–350.

[62] D. S. Wallach, "A survey of peer-to-peer security issues," in *Proceedings of the 2002 Mext-NSF-JSPS International Conference on Software Security: Theories and Systems*, ser. ISSS'02. Berlin, Heidelberg: Springer-Verlag, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1765533.1765539 pp. 42–57.

[63] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687810 pp. 261–269.

[64] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: cooperative, decentralized anonymous communication," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. EW 11. New York, NY, USA: ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/1133572.1133578

[65] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, ser. OSDI '02. New York, NY, USA: ACM, 2002. [Online]. Available: http://doi.acm.org/10.1145/1060289.1060317 pp. 299–314.

[66] A. Panchenko, S. Richter, and A. Rache, "Nisan: network information service for anonymization networks," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653681 pp. 141–150.

[67] Q. Wang, P. Mittal, and N. Borisov, "In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866343 pp. 308–318.

[68] N. Borisov, "Anonymous routing in structured peer-to-peer overlays," Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, USA, 2005, chair-Eric A. Brewer.

[69] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[70] G. Ciaccio, "Improving sender anonymity in a structured overlay with imprecise routing," in *PETS*, June 2006.

[71] N. Borisov, "Computational puzzles as Sybil defenses," in *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*. Washington, DC, USA: IEEE Computer Society, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1157740.1158254 pp. 171–176.

[72] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta, "Limiting Sybil attacks in structured P2P networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, may 2007, pp. 2596 –2600.

[73] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against Sybil attacks," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1397759.1398053 pp. 3–17.

[74] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant dht routing," in *In ESORICS*. Milan, Italy: Springer, September 2005, pp. 305–318.

[75] N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'09. Berkeley, CA, USA: USENIX Association, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558977.1558979 pp. 15–28.

[76] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi, "Ostra: leveraging trust to thwart unwanted communication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387591 pp. 15–30.

[77] Y. Sovran, J. Li, and L. Subramanian, "Unblocking the Internet: Social networks foil censors," NYU, Tech. Rep., 2008.

[78] S. Marti, P. Ganesan, and H. Garcia-Molina, "SPROUT: P2P routing with social networks," in *Proceedings of the 1st International Workshop on Peer-to-Peer Computing and Databases*, 2004, pp. 425–435.

[79] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving P2P data sharing with oneswarm," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851198 pp. 111–122.

[80] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips, "Tribler: A social-based peer-to-peer system," Delft University of Technology, Tech. Rep., Feb 2006. [Online]. Available: http://www.pds.ewi.tudelft.nl/reports/2006/PDS-2006-002/PDS-2006-002.pdf

[81] H. Chen, X. Li, and J. Han, "Maze: a social peer-to-peer network," in *In of CEC-East*, 2004.

[82] J. Frankel, "http://waste.sourceforge.net."

[83] B. Popescu, B. Crispo, and A. S. Tanenbaum, "Safe and private data sharing with Turtle: Friends team-up and beat the system," in *12'th Cambridge International Workshop on Security Protocols*, April 2004.

[84] N. S. Evans, C. GauthierDickey, and C. Grothoff, "Routing in the dark: Pitch black," *ACSAC*, 2007.

[85] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim, "Membership-concealing overlay networks," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653709 pp. 390–399.

[86] A. Nambiar and M. Wright, "The Salsa simulator," http://ranger.uta.edu/~mwright/code/salsa-sims.zip, accessed October 2008.

[87] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders, "Möbius: An extensible tool for performance and dependability modeling," in *Computer Performance Evaluation: Modelling Techniques and Tools*, B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, Eds., vol. 1786. Schaumburg, IL: Springer, Mar. 2000, pp. 332–336.

[88] G. Danezis, "Mix-networks with restricted routes," in *Proceedings of Privacy Enhancing Technologies Workshop (PET 2003)*, R. Dingledine, Ed. Springer-Verlag, LNCS 2760, March 2003, pp. 1–17.

[89] N. de Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie van Wetenschappen*, vol. 49, 1946.

[90] R. Merkle, "Protocols for public key cryptosystems," in *IEEE Symposium on Security and Privacy*, 1980, pp. 122–133.

[91] A. Kapadia and N. Triandopoulos, "Halo: High-Assurance Locate for Distributed Hash Tables," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2008, pp. 61–79.

[92] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, ser. PET'02. Berlin, Heidelberg: Springer-Verlag, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1765299.1765304 pp. 54–68.

[93] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, ser. PET'02. Berlin, Heidelberg: Springer-Verlag, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1765299.1765303 pp. 41–53.

[94] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: estimating latency between arbitrary internet end hosts," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 11–11, July 2002. [Online]. Available: http://doi.acm.org/10.1145/571697.571700

[95] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," *International Workshop on Peer-to-Peer Systems*, pp. 256–267, 2003.

[96] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, 2002.

[97] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001. [Online]. Available: http://doi.acm.org/10.1145/502034.502054 pp. 202–215.

[98] "Wikipedia," www.wikipedia.org.

[99] "del.icio.us," delicious.com.

[100] D. J. Phillips, "Defending the boundaries: Identifying and countering threats in a usenet newsgroup," *Inf. Soc.*, vol. 12, no. 1, 1996.

[101] "Secondlife," secondlife.com.

[102] R. Levien and A. Aiken, "Attack-resistant trust metrics for public key certification," in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*. Berkeley, CA, USA: USENIX Association, 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267549.1267567 pp. 18–18.

[103] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary)," in *STOC*. ACM, 1987, pp. 230–240.

[104] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[105] "Facebook," www.facebook.com.

[106] "Orkut," www.orkut.com.

[107] Sophos, "Sophos Facebook id probe shows 41% of users happy to reveal all to potential identity thieves," August 14 2007.

[108] S. Milgram, "The small world problem," *Psychology Today*, vol. 2, pp. 60–67, 1967.

[109] D. Randall, "Rapidly mixing Markov chains with applications in computer science and physics," *Computing in Science and Engineering*, vol. 8, no. 2, pp. 30–41, 2006.

[110] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, no. 3, pp. 497–515, 2004.

[111] W. K. Hastings, "Monte carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, April 1970. [Online]. Available: http://dx.doi.org/10.1093/biomet/57.1.97

[112] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the gnutella network," *IEEE Internet Computing*, vol. 6, pp. 50–57, January 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=613352.613670

[113] L. Goodman, "Snowball sampling," *Annals of Mathematical Statistics*, vol. 32, no. 1, pp. 148–170, 1961.

[114] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002.

[115] U. A. Acar, "Self-adjusting computation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 2005, co-Chair-Guy Blelloch and Co-Chair-Robert Harper.

[116] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1766171.1766196 pp. 294–311.

[117] C. Lesniewski-Laas, "A Sybil-proof one-hop dht," in *Proceedings of the 1st Workshop on Social Network Systems*, ser. SocialNets '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1435497.1435501 pp. 19–24.

[118] P. Mittal and N. Borisov, "Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653683 pp. 161–172.

[119] G. Danezis and P. Mittal, "Sybilinfer: Detecting Sybil nodes using social networks," in *Proceedings of the Network and Distributed System Security Symposium,*. The Internet Society, 2009.

[120] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460445 pp. 337–350.

[121] "Livejournal," www.livejournal.com.

[122] K. Bauer, D. Mccoy, D. Grunwald, and D. Sicker, "Bitstalker: Accurately and efficiently monitoring bittorent traffic," in *Proceedings of the International Workshop on Information Forensics and Security*, 2009.

[123] M. Liberatore, B. N. Levine, and C. Shields, "Strengthening forensic investigations of child pornography on P2P networks," in *Proceedings of the 6th International Conference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1921168.1921193 pp. 19:1–19:12.

[124] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proceedings of the 4th ACM European Conference on Computer systems*, ser. EuroSys '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1519065.1519089 pp. 205–218.

[125] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in Facebook," in *Proceedings of the 2nd ACM Workshop on Online social networks*, ser. WOSN '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1592665.1592675 pp. 37–42.

[126] P. Wang, J. Tyra, E. Chan-Tin, T. Malchow, D. F. Kune, N. Hopper, and Y. Kim, "Attacking the kad network," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, ser. SecureComm '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1460877.1460907 pp. 23:1–23:10.

[127] M. J. Freedman, E. Freudenthal, and D. Mazières, "Democratizing content publication with coral," in *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*. Berkeley, CA, USA: USENIX Association, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251193 pp. 18–18.

[128] T. Ristenpart, G. Maganis, A. Krishnamurthy, and T. Kohno, "Privacy-preserving location tracking of lost or stolen devices: cryptographic techniques and replacing trusted third parties with DHTs," in *Proceedings of the 17th Conference on USENIX Security symposium*. Berkeley, CA, USA: USENIX Association, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496711.1496730 pp. 275–290.

[129] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, "Vanish: increasing data privacy with self-destructing data," in *Proceedings of*

the 18th Conference on USENIX Security Symposium, ser. SSYM'09. Berkeley, CA, USA: USENIX Association, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855768.1855787 pp. 299–316.

[130] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, "Defeating vanish with low-cost sybil attacks against large DHTs," in *NDSS*, 2010.

[131] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076746 pp. 93–102.

[132] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1526709.1526784 pp. 551–560.

[133] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu, "Reverse social engineering attacks in online social networks," in *Proceedings of the 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. DIMVA'11. Berlin, Heidelberg: Springer-Verlag, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2026647.2026653 pp. 55–74.

[134] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518736 pp. 211–220.

[135] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," United States, 2003.

[136] D. Mazieres, "Self-certifying file system," Ph.D. dissertation, MIT, 2000, supervisor-Kaashoek, M. Frans.

[137] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "Rofl: routing on flat labels," in *Proceedings of the 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1159913.1159955 pp. 363–374.

[138] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable internet protocol (aip)," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402997 pp. 339–350.

[139] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1607723.1608132 pp. 173–187.

[140] A. Mohaisen, A. Yun, and Y. Kim, "Measuring the mixing time of social graphs," in *Proceedings of the 10th Annual Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879191 pp. 383–389.

[141] "Sfslite," http://www.okws.org/doku.php?id=sfslite.

[142] "Using libasync," http://pdos.csail.mit.edu/6.824-2004/async/.

[143] M. Krohn, E. Kohler, and M. F. Kaashoek, "Events can make sense," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364385.1364392 pp. 7:1–7:14.

[144] "Diaspora," wwww.joindiaspora.com/.

[145] A. Johnson and P. Syverson, "More anonymous onion routing through trust," in *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*. Washington, DC, USA: IEEE Computer Society, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1602936.1603616 pp. 3–12.

[146] A. M. Johnson, P. Syverson, R. Dingledine, and N. Mathewson, "Trust-based anonymous communication: adversary models and routing algorithms," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046729 pp. 175–186.

[147] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, Aug. 2010, v0.34.

[Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[148] D. L. Mills, *Computer Network Time Synchronization: The Network Time Protocol*. CRC Press, 2006.

[149] C.-Y. Hong, C.-C. Lin, and M. Caesar, "Clockscalpel: understanding root causes of internet clock synchronization inaccuracy," in *Proceedings of the 12th International Conference on Passive and Active measurement*, ser. PAM'11. Berlin, Heidelberg: Springer-Verlag, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1987510.1987531 pp. 204–213.

[150] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. [Online]. Available: http://link.aip.org/link/?JCP/21/1087/1

[151] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, ser. CCS '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1102120.1102133 pp. 81–91.

[152] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A robust and invisible non-blind watermark for network flows," in *NDSS*, 2009.

[153] J. L. Massey, "Guessing and entropy," in *IEEE ISIT*, 1994.

[154] V. Shmatikov and M.-H. Wang, "Measuring relationship anonymity in mix networks," in *Proceedings of the 5th ACM Workshop on Privacy in eElectronic Society*, ser. WPES '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1179601.1179611 pp. 59–62.

[155] D. Aldous and J. A. Fill, "Reversible Markov chains and random walks on graphs," http://www.stat.berkeley.edu/~aldous/RWG/book.html, accessed February 2012.