Anonymous Routing in Structured Peer-to-Peer Overlays

by

Nikita Borisov

B.Math (University of Waterloo) 1998
M.S. (University of California, Berkeley) 2002

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Eric A. Brewer, Chair
Professor David A. Wagner
Professor David J. Aldous

Spring 2005

The dissertation of Nikita Borisov is approved:

Chair Date

Date

Date

University of California, Berkeley

Spring 2005

Anonymous Routing in Structured Peer-to-Peer Overlays

Nikita Borisov

Abstract

Anonymous Routing in Structured Peer-to-Peer Overlays

by

Nikita Borisov

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Eric A. Brewer, Chair

As more of our daily activities are carried out online, it becomes important to develop technologies to protect our online privacy. Anonymity is a key privacy technology, since it serves to hide patterns of communication that can often be as revealing as their contents. This motivates our study of the use of large scale peer-to-peer systems for building anonymous systems.

We first develop a novel methodology for studying the anonymity of peer-to-peer systems, based on an information-theoretic anonymity metric and simulation. We use simulations to sample a probability distribution modeling attacker knowledge under conservative assumptions and estimate the entropy-based anonymity metric using the sampled distribution. We then validate this approach against an analytic method for computing entropy. The use of sampling introduces some error, but it can be accurately bounded and therefore we can make rigorous statements about the success

of an entire class of attacks.

We next apply our methodology to perform the first rigorous analysis of Freenet, a peer-to-peer anonymous publishing system, and identify a number of weaknesses in its design. We show that a targeted attack on high-degree nodes can be very effective at reducing anonymity. We also consider a next generation routing algorithm proposed by the Freenet authors to improve performance and show that it has a significant negative impact on anonymity. Finally, even in the best case scenario, the anonymity levels provided by Freenet are highly variable and, in many cases, little or no anonymity is achieved.

To provide more uniform anonymity protection, we propose a new design for peer-to-peer anonymous systems based on structured overlays. We use random walks along the overlay to provide anonymity. We compare the mixing times of random walks on different graph structures and find that de Bruijn graphs are superior to other structures such as the hypercube or butterfly. Using our simulation methodology, we analyze the anonymity achieved by our design running on top of Koorde, a structured overlay based on de Bruijn graphs. We show that it provides anonymity competitive with Freenet in the average case, while ensuring that worst-case anonymity remains at an acceptable level. We also maintain logarithmic guarantees on routing performance.

Professor Eric A. Brewer
Dissertation Committee Chair

Contents

List of Figures

Acknowledgments

This thesis marks the end of a long educational path and I would like to thank all the people who have helped me along the way. I would like to start with my parents, who instilled in me the love of learning and intellectual pursuits, provided me with the right opportunities, and have been a great source of encouragement and support both during my childhood and in my adult years.

Next, I would like to thank my friends, in Moscow, Ottawa, Waterloo, and Berkeley. You provided me with distractions that may not have always accelerated the pace of my educational progress, but they were much needed, and your love and company is much appreciated.

My fondest memories of Waterloo involve the Computer Science and Pure Math clubs, where I could combine the educational and social aspects of my life; thanks to everyone there for all the good times. During my stay in Berkeley, I have had more than my fair share of wonderful officemates. I am especially grateful to Steve Gribble, who has taught me more about being a graduate student than anyone else, David Wagner, who has at times served as colleague, collaborator, teacher, and friend, and Ian Goldberg, who has always been a true friend and without whom I would not have come to Berkeley in the first place. And everyone else — Barbara, Emre, Yatin, Mike, Rob, Josh, Mihut, Kevin, Mike, Bowei, Rabin, Sergiu, and Mike — I am glad for the times we have had together.

I would also like to thank members of the Berkeley security reading group for

valuable discussions, excellent ideas, and helpful comments on paper drafts and presentations. Jason Waddle has made the most direct contribution through his role as collaborator during the early states of this work.

I have learned a lot from countless colleagues outside Berkeley as well. People who have helped deepen my understanding of anonymity include George Danezis, Roger Dingledine, Bryce Wilcox, and Matthew Wright. Roger deserves special thanks for compiling an excellent anonymity bibliography [27], which has been an invaluable resource.

My advisor, Eric Brewer, has helped me in more ways than I can count. Eric has always managed to point out my insights that I did not know I had, while contributing many additional insights of his own. He has also been a great mentor in all aspects of my academic career. And I really appreciate his encouragement and motivation, as well as the freedom he gave to develop my own pace and interests.

Finally, I would like to thank Lenore, who has given me much love and support and helped see me through these often-stressful times.

Chapter 1

Introduction

1.1 Privacy

Privacy, some say, is a recent paradigm. Earlier times saw humans living in small, tightly-knit social circles, where detailed knowledge about their lives was available to everyone. Only by moving into densely populated cities did people finally acquire a chance to have privacy in their lives, losing themselves among the crowd and interacting with anonymous strangers who knew nothing of their history.

Today, in the 21st century, we are quickly seeing this notion of privacy vanish. More and more of the details of our daily lives are handled by computers, which can (and do) maintain a detailed and accurate record of everything they process. The same computers, linked by intricate networks, share and aggregate this data, and mine it for patterns of activities that can be incredibly revealing. Such mining is performed for many reasons: law enforcement and intelligence agencies searching for

evidence of criminal activity, private investigative companies providing background reports to potential employers, or market research firms looking for better ways to sell a product. No matter the purpose, though, this information is increasingly detailed — and increasingly available. We are quickly moving towards a world where someone you've never met, at the click of a button, can know more about you than your own parents might.

So does this mean that privacy is a short-lived fad, destined to be a footnote in the history books? A closer look shows that the concept of privacy has long been around, and will remain with us for quite some time. Even in close-knit societies of old, there were such things as closed-door meetings, whispered conversations, and secret notes. And even today, with a proliferation of surveillance, logging, and data mining technologies, there are spaces, such as one's home, where one can remain private for at least the near future.

What has been changing is not whether privacy exists or not, but the *costs* of remaining private. Privacy has always had a cost, since one has to alter one's activities in some way in order to remain private.[1] The existence of cities was able to reduce this cost, as it was easy to keep your actions private when surrounded by millions, most of whom did not take an interest in your activities. On the early Internet, the cost was similarly cheap, as summed up by the classic New Yorker cartoon: "On the Internet, nobody knows you are a dog." But today, the balance has been shifting,

[1]Though perhaps it had never before been so neatly quantified as by the discount cards offered by today's grocery stores, offering a savings of 10 cents off your purchase in return for revealing your identity.

and not only can someone tell whether you are a dog, but it's often easy to find out whether you *have* a dog, and if so, what is its name!

So there are two negative side effects to the ever increasing lack of online privacy. The first will be a continuation of the privacy violations we see today, as the expectations of privacy of Internet users don't match the ever-shifting realities of online privacy, or lack thereof. But a second effect will be people responding to the growing privacy costs of being online by simply refusing to participate. Already, we are seeing parents guarding their children from using the Internet — a marvelous educational resource — for fear that they will reveal too much information to potential stalkers or predators. Government agencies, and indeed companies, are often shirking the use of email because the information therein can easily be discovered through public inquiry, and privacy protections that we have come to expect from conversations do not apply in the online realm. And more and more people will make the choice that the benefits of being connected to, and interacting with, millions of strangers do not outweigh the amount of personal information revealed to those strangers.

This concern motivates the need for privacy enhancing technologies, which let one participate in the Internet while not revealing as much privacy-compromising information. Even though these technologies come at a cost, they introduce a different, and often better, tradeoff between the benefits of participation and the costs to one's privacy. As such, they stand to improve the overall usefulness of the Internet. And this is where we draw motivation for our work on one such privacy preserving technology: anonymity.

1.2 Anonymity and P2P Systems

Anonymity is a way to dissociate actions from identities. It is a key privacy technology, since the value of private information is greatly diminished if it cannot be tied to a particular identity. The reason cities are conducive to privacy is not that it is easier to hide your actions, but rather than the people who may observe you do not know who you are. Moreover, there are some kinds of private information that can only be protected with anonymity technologies, as the actions themselves cannot be otherwise hidden, but the association with the actors is sensitive information. For example, your web browsing habits might reveal a lot of information about you, even if the contents of your connection with each web site are fully protected by some encryption scheme such as TLS [26].

Technological support for anonymity has been the subject of much research, starting with Chaum's seminal paper [14]. (A review of the different anonymity designs will follow in Section 3.1.) Recently, much of the research has focused on peer-to-peer (p2p) anonymous systems. "Peer-to-peer," in this context, means a dynamic, decentralized network comprising of a large number (thousands to millions) of users, each of whom both provides services for others in the network and uses the services provided by others.

The interest in p2p anonymous systems is motivated by several factors. The anonymity community has been long concerned about central points of failure, and research into mix networks, starting with Chaum's original paper, has aimed to de-

fend against attacks on a particular server. The decentralized nature of p2p systems provides a mechanism to distribute trust among a very large population. At the same time, p2p systems are designed to scale to very large numbers of users, in part by using scalable algorithms for network maintenance, and in part by exploiting the capacity scaling that comes from users also providing service to others. Anonymity systems greatly benefit from large user populations, since users can hide their actions among a larger crowd of potential actors. Finally, there is an extra level of deniability that can be achieved by contributing to an anonymity system, rather than being just a user [72]. Therefore, users have an incentive to provide services to other users, mitigating the free rider problem of p2p networks [2, 66].

Existing p2p systems can be divided into two categories: structured and unstructured. Structured systems have received a lot of attention in the research community, whereas most of the deployed systems are unstructured (with some exceptions on both sides). The key difference between the two designs is that structured systems maintain a mathematical invariant by using some formula that determines which nodes must connect to which other nodes. Unstructured systems, on the other hand, allow connections to occur between an arbitrary pair of nodes. An important question that we aim to answer is which approach is best for anonymity.

1.3 Hypothesis

In this work, we propose to to use a structured peer-to-peer network to build an anonymous routing system. The context is that we wish to be able to route a message from one node in the network to another while hiding the identity of the source node. This will allow us to render anonymous a large number of applications that use this generic routing primitive. Our approach can also be extended to a generic anonymous forwarding infrastructure, by routing to a random node and using it as a proxy to forward traffic. As such, our solution can be useful for a very wide range of applications.

Our claim is that we can design such a system and achieve acceptable anonymity levels, under a set of attack assumptions outlined in Section 2.2. In particular, we want to show that a system based on structured networks is more effective at providing anonymity than an unstructured design. To prove this claim, we need to solve several research challenges. First of all, we must prevent attackers from exploiting the structure of the network to trace back a message and discover its originator. This is a difficult task, since the structure of a network will limit the routing choices, and furthermore, in order to function, the structure cannot be kept a secret. Our main approach will be to hide messages a node sends among those it forwards for others, and to ensure a wide enough diversity of paths through the network such that any message has a large number of plausible senders.

A second challenge is to analyze how effectively the system provides anonymity

against attackers, and compare our design relative to an unstructured one. One approach is to formulate attack strategies and evaluate their success. However, such an analysis would only cover the kind of attacks that we may come up with, and, like many past security systems, our design could be vulnerable to new, previously unknown attack strategies. Instead, we use an information-theoretic approach that can evaluate an entire class of attacks and compute a conservative bound on their success. An important contribution of our work is to extend this approach, previously applied only to small and simple systems, to the domain of large, complex, and dynamic p2p systems.

The third challenge is to make sure that the overhead in providing anonymous access is not too high. In order to have a large user base, it is essential that the overhead is such that not only users for whom privacy is a high priority use the system. Structured systems typically have a performance advantage over unstructured ones. However, we must make sure that our modifications, which will certainly come at some cost, do not introduce sufficient overhead to negate this advantage.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 contains some definitions and specifies the attack model that we will be using throughout our work. Chapter 3 discusses some related work. Chapter 4 explains the simulation-based methodology we use for studying anonymous systems. In Chapter 5 we apply this methodology to

analyze Freenet [18], a peer-to-peer anonymous publishing system. We find a number of flaws in the design of Freenet, and in Chapter 6, we propose an alternative design for anonymous peer-to-peer systems, based on structured overlays. In Chapter 7 we apply our methodology to analyze the anonymity of our alternative design and compare it with Freenet. Chapter 8 concludes with a summary of our contributions and a discussion of future research directions.

Chapter 2

Definitions and Assumptions

In this chapter, we review some of the definitions that we will be using in the rest of the thesis and outline some important assumptions underlying our work.

2.1 Definitions

2.1.1 Anonymity Definitions

anonymity The state of not being identifiable among some set of subjects.

sender anonymity Hiding the identity of the originator or sender of a message, from either the destination or a third party.

destination anonymity Hiding the identity of a destination of a message, from either the sender or a third party.

unlinkability Hiding the communication relationships between senders and destinations, potentially comprised of multiple messages.

We will be mainly concerned with sender anonymity. Namely, our aim will be to hide the identity of the sender of a message from both the destination and any third parties observing it. Our focus on sender anonymity is motivated by the fact that in structured peer-to-peer networks, it is difficult to hide the identity of a destination, since it is implied by the structure. However, the node corresponding to any destination address is usually assigned at random. Additionally, in many applications, all communication is performed by sending messages towards some randomly chosen key, and associations between clients are made by using the same key. For those applications, sender anonymity is all that is needed. Other applications may be recast into similar terms by the introduction of rendezvous servers [35].

Sender anonymity provides some amount of unlinkability, since not being able to tie messages to their senders makes it difficult to determine communication relationships. However, we will not perform analysis of unlinkability itself and leave it to future work to determine whether long-term communication relationships can be inferred despite the origins of each message being hidden.

2.1.2 Peer-to-peer Definitions

The term "peer-to-peer" has been used to describe a large number of systems; we will adopt a specific definition for our discussion.

peer-to-peer A p2p network has the following properties:

- Users of the network also provide services for others
- The network is designed to scale to large numbers of users (thousands to millions)
- The management of the network is decentralized
- The membership is dynamic and nodes may leave and join at any time

structured peer-to-peer A structured network uses a mathematical formula to determine which nodes should connect to which other nodes in order to maintain some structure invariant. Structured networks generally guarantee bounds of $O(\log N)$ hops to reach any node and either $O(\log N)$ or constant neighbors at each node. (Where N is the total number of nodes.)

2.2 Attack Model

We are going to evaluate our design with respect to a *passive logging* attack model [76]: some number c of nodes are compromised, and they record all messages going through them and collude by sharing information among each other. We consider static, rather than adaptive compromise: the adversary chooses which nodes to compromise before any messages are sent. The static compromise model is more realistic, as very few attackers have the capacity to compromise arbitrary nodes at will.

We will assume that attackers can control and monitor timing sufficiently well that they can detect when a message forwarded by one attacker node was received by another one. However, we will discount all other forms of active attacks and assume that the attackers follow the protocol faithfully. We will usually assume that the attackers have full knowledge of both the p2p algorithms and the connection topology of the network, though we will consider how this assumption can be relaxed in Section 5.5.

There are two major kinds of attack not covered by our model: monitoring (or affecting) links between two non-compromised nodes, and active attacks. Much of the attacks on the links between nodes can be stopped by using link-layer encryption and authentication. Someone monitoring a large number of links can still perform traffic analysis; this problem is not unique to our setup but a general concern for anonymity systems. There is still much ongoing research on effective defenses against traffic analysis, such as cover traffic. We believe that restricted topologies of p2p networks can make cover traffic more practical by using a bounded number of links per node [21], but we leave the topic of resisting traffic analysis in p2p designs for future research.

Active attacks can further be broken down into three categories.

- A malicious node may, upon receiving a message, deviate from the protocol and forward it incorrectly, forward a modified version, or not forward it at all. This attack is not very effective against the designs we consider, because the

destination of messages is included in the clear, so regardless of what the node does with the message, no new information about its origin will be revealed. (Although traffic analysis capability could be enhanced by selectively forwarding or otherwise altering the stream of messages being sent back to the sender.)

- Attackers may send out probes to learn information about the network. Since we make the conservative assumption that attackers know the entire topology of the network, information learned in this way would be subsumed by our attack model.

- Attackers may subvert the network topology and routing algorithms. The most important example of this is the Sybil attack [31], where a single attacker node pretends to have multiple identities. By introducing a large number of such ghost nodes, the attacker may control an arbitrary fraction of all nodes in the system. Other attacks include denial of service, which may be an effective way to reduce the anonymity of the system.

  For the purposes of our hypothesis, structured networks are better equipped to defend against denial of service and Sybil attacks than unstructured ones [13]. However, the only currently practical solution to the Sybil attack requires a centralized authority to issue membership certificates. In order to achieve truly distributed trust, an exploration of other alternatives is needed, which is out of scope of this thesis.

We consider attackers observing a single message or a single path through the p2p

network and trying to identify the origin based on this observation. This model is appropriate for applications such as file sharing or content distribution, where a node will typically only access any file once. For systems where long-term communication patterns exist, further study is needed to determine whether more information can be learned over time. We hope that our single-message analysis can serve as a foundation for this future research.

In summary, we believe that our assumptions are reasonable for many settings, and the problems we choose to overlook are not unique to our design but are common to other anonymity and peer-to-peer systems. Therefore, we hope to leverage solutions developed by future research in those domains to improve our design.

Chapter 3

Related Work

3.1 Anonymous Systems

3.1.1 Chaum-like Mixes

The seminal paper on anonymous systems was written by Chaum [14]. He proposed a system for anonymous email based on *mix networks*. A mix shuffles a batch of messages together and outputs them in a random order. The sender and the mix use public key cryptography in order to hide the correspondence between input and output messages: the message to the mix contains $Enc(K_m, \{D, M\})$, where K_m is the public key of the mix, D is the destination of the message, and M is the payload itself. The mix will decrypt and learn D and M and then forward M to the destination D.

Mixes in Chaum's design can be chained by using the encrypted message as the

payload of a message encoded for a second mix. Given mixes m and m' with public keys K_m and $K_{m'}$ respectively, a sender might encode a message as:

$$Enc(K_m, \{m', Enc(K_{m'}, \{D, M\})\})$$

and send this message to m, which will decrypt it and forward it to m'. In this case, m never learns the true destination, and m' does not learn the identity of the sender, so both mixes must be compromised in order to violate anonymity. Longer chains are possible, distributing the trust for safeguarding anonymity among a set of mixes, where the security of the chain depends on at least one mix being honest. Chaum also described a mechanism for creating anonymous reply addresses.

The mix network design, in one form or another, has been followed by many anonymous systems. The first widely used implementation of mix networks was the Type I cypherpunk anonymous remailers [36], using PGP [79] encryption in order to wrap email messages and deliver them anonymously. They were followed by Mix-Master [51], and then MixMinion [22], which use the same basic principles, but split messages into equal-sized chunks, sent along potentially different routes, in order to defeat traffic analysis.

The concept of mix networks was first translated into the domain of general IP traffic by Wei Dai, in his proposal for PipeNet [20]. PipeNet would build anonymous channels for low-latency, bi-directional communication, using layered encryption similar to Chaum's design. This layering suggested the title of "Onion Routing" for the first implementation of this kind of IP forwarding [72]. Other implementations

followed, including the commercial deployment of the (now defunct) Freedom Network [7], and the more recent effort behind Tor [29], a second-generation onion routing design.

One issue relevant to all of these systems is how the sequence of mixes is chosen for each message or path. Mix cascades [10] were designed to resist the $n-1$ attack: they guarantee that even if all but one mix are compromised, the mixing process remains secure. However, cascades require a fixed mix network membership and topology. With dynamic membership, the originator typically chooses a random subset of the current nodes for the route, and anonymity may therefore be compromised with probability $(c/n)^L$ where c is the number of compromised mixes out of the total n, and L is the path length. A pressing research question in these free route systems is how to maintain a directory of all the currently active servers, without the directory itself being a point of attack.

3.1.2 Alternative Anonymity Systems

Although the mix design has been quite influential, there are a number of notable alternatives. The first, once again due to Chaum, is the Dining Cryptographers Problem [15]. It creates an anonymous broadcast channel without using cryptography as such, but it nevertheless provides information-theoretic security guarantees. Unfortunately, for absolute security the channel requires $O(n^2)$ communication among n members, and must deal with complicated collision-avoidance and denial of service issues.

Another system that uses a different approach is Crowds [62], designed for anonymous web browsing. Briefly, Crowds nodes forward web requests to each other at random, executing a form of a random walk. At each step the random walk may probabilistically terminate and the current node then sends the request to the web server. Section 4.3 has a more detailed description of Crowds. The interesting feature of the system is that anonymity is achieved not only through having your messages forwarded by other honest nodes, but also through forwarding messages for other honest nodes and hiding your own among them. We will return to this detail in the next section.

Two popular deployed anonymizing services also did not follow the mix network design. `anon.penet.fi` was an anonymous remailer, popular in the early nineties. It operated simply by stripping off email headers and replacing the source or destination address with a long-lived pseudonym. It used no encryption, so it was vulnerable to traffic analysis or even passive monitoring of the link. However, its privacy was in fact compromised through the pseudonym list, which was the subject of a subpoena in 1996. After the subpoena, the remailer was shut down voluntarily by its operator [40]

Anonymizer.com is another service that operates on a similar model, but is used for web browsing instead of email. It uses TLS [26] between the client and the anonymizing proxy in order to foil passive attacks, although traffic analysis is still possible. Nevertheless, there have not been any publicized attacks during the nearly decade of its operation, and it remains the only successful commercial anonymity service.

3.1.3 Peer-to-Peer Systems

All of the systems mentioned above run into scaling issues once there are more than about 100 nodes. This is because all nodes need to know which other nodes are up and running, and with more than 100 nodes the update traffic overhead becomes significant. Static topologies likewise suffer scaling issues since node failures that disrupt the topology will be more likely in larger networks. This limitation places a capacity restriction on the entire system and prevents it from becoming a popular global service.

Peer-to-peer networks were designed explicitly with scaling in mind. And at the same time, the questionable legality of the dominant peer-to-peer application (file sharing) created a demand for anonymity. There was therefore a natural convergence of p2p and anonymity, and even the early designs of Gnutella [39] aimed to offer some anonymity protection. It was incomplete and ultimately ineffective [19], but other systems quickly followed suit.

In the file sharing (or publishing) domain, Freenet [18] was designed to provide anonymity to both publishers and readers. Freenet nodes forward queries to neighbors, who then follow heuristics to try to eventually reach a node that has a copy of a document. Details of the Freenet design are covered in Section 5.1. GNUNet [9] is similar, though it uses a different algorithm for deciding where to forward queries, and it splits files into blocks, each of which is replicated and queried individually. GNUNet and Freenet both share the property that, like in Crowds, nodes gain anonymity by

forwarding traffic for others. They do not, however, use probabilistic forwarding and use a time-to-live field (TTL) instead to limit query propagation.

Freenet and GNUNet also aim to provide censorship resistance in addition to anonymity. To achieve this, they replicate data and try to hide the identity of the nodes storing each copy. Another system with similar goals was Free Haven [28], but its design relied on a pre-existing anonymous communication channel rather than implementing anonymity mechanisms itself.

Other peer-to-peer networks aim to provide a generic anonymous communications infrastructure, similar to onion routing. MorphMix [63] constructs an onion routing path by recursively selecting neighbors of the current mix nodes. Each node knows about only 6 other nodes, and uses its neighbors to find other nodes to use as mix servers. To prevent a malicious node from suggesting colluding members and thus potentially violating anonymity, MorphMix uses a collusion detection scheme, ultimately based on IP addresses. The neighbor connections at each node are in constant flux in order to learn about other nodes in the network and prevent attacks exploiting network topology knowledge.

Tarzan [33] also creates a peer-to-peer onion routing system. Tarzan addresses the collusion problem by using a restricted topology, where connections are formed only between nodes in different *domains*, that is, with different IP prefixes. The rationale is that an attacker may be able to control hundreds or even thousands of IP addresses within the same subnet, but not hundreds of subnets. The restricted topology also has a small degree, and cover traffic is sent along each link in the topology.

Some recent designs have been built on top of structured peer-to-peer networks. Cashmere [78] uses the anycast [57] mechanism in Pastry [65] to ensure resilience of onion routes to individual node failure. However, it requires a centralized and sophisticated certificate infrastructure to distribute public keys of the mix servers. AP3 [49] does not require a centralized infrastructure, and its design is similar to our design sketched in Section 6.5. However, instead of random walks on the peer-to-peer overlay, it performs routing to a random destination, resulting in significantly longer hop counts. We compare AP3 to our own design in Section 7.3.

3.2 Attacks and Analysis

Mix networks were designed to resist a very powerful threat model: a global adversary who can at the least passively monitor all links, and potentially actively modify messages as well. Mix networks resist this kind of adversary quite well under the assumption that all messages are of the same size. Otherwise, an adversary can track messages based on size; this is known as a *traffic analysis* attack.

Most forms of traffic analysis, however, concern long-term communications, such as repeated correspondence or IP/TCP connections maintained through onion routing. In the latter case, correlations in the volume of traffic between mixes can give away which routes are going through which mix nodes. Cover traffic is the most common technique to defend against traffic analysis; however, it may not be practical in all settings. Recent work highlights the importance of traffic analysis by showing

that it may be practical even for a much more limited adversary [52]. Another type of attack available to a global adversary is the *intersection attack*, which shows that by intersecting the total sets of people using a mix network at each time that an output stream of interest is active, one can eventually narrow this set down to the true source [61].

Long-term communications anonymity can be a violated even with a more limited adversary model. For example, Wright *et al.* discuss the degradation to anonymity resulting from repeatedly choosing paths out of a set of mix servers, some fraction of which are compromised [75]. They show that anonymity will be compromised after $O((n/c)^2)$ path reformulations, where c is the number of compromised nodes out of a total of n. However, if timing attacks can be eliminated, this bound increases to $O((n/c)^l)$, where l is the path length. Their later work explores some potential countermeasures to such passive logging attacks [76].

The situation for Crowds [62] is significantly worse, where only $O(n/c)$ path reformulations are needed for compromise. This is due to the the *predecessor attack*: whenever a compromised node receives a message, its predecessor in the path is more likely to be the origin than any other node. The difference is too small to directly identify the origin, but repeated path reformulations allow for more precise statistical inference; in particular, any node that appears as a predecessor in two paths is very likely to be the origin.

Our own analysis also concerns the differences in probabilities assigned to different origins for an intercepted message. However, we consider only the single-message case

and evaluate how much information can be learned based on the routing algorithms and the restricted topology of p2p networks. We therefore use information-theoretic anonymity metrics [25, 67], rather than success of repeated attacks, to quantify anonymity. We leave the analysis of multiple message attacks to future work; see Section 8.2.3.

The work most closely related to ours is an analysis of anonymity in unmodified Chord [55]. The analysis assumes that there is but a single adversary, and shows that on average, the information revealed is minimal. Our analysis, on the other hand, considers multiple colluding adversaries, and we are concerned with anonymity in the worst as well as the average case. Another look at anonymity resulting from restricted routes is due to Danezis [21]. He considers the problem of designing mix networks of moderate size (several hundred nodes) with static, centrally-designed restricted connection topologies. In this scenario, he found that expander graphs result in fast mixing times and therefore provide better anonymity. Mixing times play a central role in our work as well, but our goal is to construct larger scale, dynamic, and decentralized peer-to-peer networks and his techniques are not directly applicable.

Chapter 4

Empirical Approach to Anonymity Measurement

4.1 Quantitative Anonymity Analysis

Anonymity is often thought of in binary terms: either one is anonymous or one is not. In an ideal anonymous system, an attacker, no matter how powerful, would have no information about who performed an action. Some schemes, such as the dining cryptographers network (DC-Net [15]), are in fact able to offer information theoretic guarantees of anonymity. However, such networks are currently practical only for small groups, and hence mere participation in them narrows down your identity to a small set of people.

In practice, anonymity systems don't achieve the ideal and the protection provided is always limited. Therefore, it makes sense to discuss not just whether a system

provides anonymity, but how well it does so. And such a discussion will be made more productive if we can quantify anonymity protection. This way we can determine which designs are better than others, as well as evaluate which systems are good enough given some desired level of privacy. Anonymity is defined as "the state of being not identifiable within a set of subjects, the *anonymity set*" [58]. A good starting point, then would be to measure the size of this set and and aim to make it as large as possible. However, this fails to capture the properties of most anonymous systems and a more detailed analysis is needed.

A metric that better represents the anonymity of a system is based on entropy [25, 67]. This metric has become a popular tool for analyzing anonymous systems [21, 24, 30]. However, computing the entropy requires a detailed model of the anonymity system and its attackers, and thus far the uses of the metric have been limited to systems with a simple underlying design and few nodes.

In this chapter, we will explain the metric, and then describe techniques, based on simulation, that can be used to apply the entropy metric to large, complicated systems and derive a measure of their anonymity. To make the discussion more concrete, we will demonstrate our techniques on the Crowds system [62]. Crowds has a simple enough design to be analyzed directly; using it as an example will give us a point of comparison between the standard techniques and the simulation approach.

4.2 Entropy Metric

4.2.1 Anonymity Sets and Their Problems

Anonymous systems have classically been analyzed in terms of the size of their anonymity sets [58]. For some action X, the anonymity set includes all of the actors who might have performed X. The anonymity set is always based on some view of the system. Someone who can directly observe the actor may be able to reduce the anonymity set to a singleton, while an observer with more limited information may have to contend with a larger set of possible actors. It is clear that it is desirable to make anonymity sets as large as possible. However, there are many examples of systems that achieve the largest possible anonymity set (i.e. the set includes all honest actors), but differ greatly in the levels of anonymity protection provided.

Consider a system where messages from n participants are shuffled together by a mix and cryptographically transformed so that an observer cannot tie the output messages to the input ones [14]. Assuming the mix is honest and its key is not compromised, an observer would not be able to determine the sender of any particular message and would have to consider the entire collection of n participants as the anonymity set.

Now imagine that the mix has an error which causes it to probabilistically select the identity permutation 99 times out of 100, and a uniformly distributed random permutation at other times. An attacker who realizes this error will know that the first output message corresponds to the first input with overwhelming probability:

$99/100 + 1/100n$. However, the anonymity set is still maximal, since there is a chance, however small, that the output corresponds to any other input.

This is an extreme example, but many real systems allow the attacker to assign higher probabilities to some members of the anonymity set than others. In this case, a metric that analyzes probabilities rather than enumerating possibilities is useful. Thus, recent analysis of anonymity systems has focused not just on the size of the set, but also on the probability distribution of all of its members. For comparison purposes, it is useful to take a functional over the distributions and reduce them to a single number.

4.2.2 Entropy Metric

In 2002, Serjantov *et al.* and Diaz *et al.* independently proposed using entropy as an anonymity metric [67, 25]. The metric is computed as follows: assign each participant of the system a distinct number between 1 and n. Let p_i be the attacker's estimate of the probability that participant i was responsible for some observed action. Then the entropy metric is calculated as:

$$H = -\sum_i p_i \log_2 p_i$$

This metric measures the amount of uncertainty an attacker has as to the identity of a participant. More precisely, it represents the number of bits of additional information necessary to correctly identify the participant. It takes on values between

0, achieved by a point distribution, and $\log_2 n$, achieved by a uniform distribution over all participants. Therefore, $\log_2 |A| \geq H$, where $|A|$ is the anonymity set size. Serjantov *et al.* define an *effective anonymity set size* following this logarithmic relationship: a system with entropy H has effective anonymity set of size 2^H. In other words, the system is equivalent, from an information-theoretic standpoint, to a perfect anonymous system with 2^H participants.

4.2.3 Min-Entropy

An alternative entropy formulation is min-entropy:

$$H_{\min} = -\log_2 \max_i p_i$$

An equivalent formulation, $\max_i p_i$, has been previously used to quantify anonymity [37]. We certainly want this metric to be large (equivalently, $\max_i p_i$ to be small), since otherwise the attacker would be able to identify the participant with high probability. However, when $H_{\min} > 1$ (Reiter and Rubin call this case *probable anonymity*), the true identity of the participant will likely not be the one assigned the maximal probability $\max_i p_i$. Therefore, we prefer to use the Shannon entropy formulation H because it captures the shape of the full probability distribution, rather than just the maximum value.

4.3 Example: Crowds

To illustrate the entropy metric, we will use the example of Crowds [62]. Crowds is an anonymous web browsing system where a set of cooperating nodes pass requests for web pages among themselves before contacting the web server and retrieving the page. The forwarding is performed according to the following algorithm:

The original node A selects a random node, B, and forwards its request to it. The recipient node, B, flips a biased coin. With probability p_f, B selects another random node C and forwards the request to it. (C may be the same as A.) Otherwise, B contacts the web server to request the web page. If the request is forwarded to C, it repeats the same process of probabilistically forwarding or satisfying the request.

The request effectively performs a random walk along the set of nodes before being sent to the web. The length of the walk is probabilistic, with the expected length of $1/(1-p_f)$. This probabilistic method makes it impossible for an intermediate node to determine whether it received a web request from the original node or from another intermediate.

Therefore, in a Crowds network with n participants, an node has to consider the anonymity set of maximal size $n-1$ when trying to determine the origin of a message it forwards (it can exclude itself). However, the message is more likely to have been originated at the node's immediate predecessor than any other node. The probability that the predecessor is the origin can be shown to be $1-p_f(n-2)/n$, while probability of any other node originating the message is p_f/n [62].

This calculation can be extended to the case where c out of n participants are colluding attackers and can therefore exclude each other from consideration. In this case, the predecessor probability is $1 - p_f(n-c-1)/n$, with the probability of other nodes remaining the same. (This probability gap is the basis of the predecessor attack [75].) We can now compute the entropy of Crowds in this case:

$$H = -\left(1 - \frac{p_f(n-c-1)}{n}\right)\log_2\left(1 - \frac{p_f(n-c-1)}{n}\right) - \frac{n-c-1}{n}\frac{p_f}{n}\log_2\frac{p_f}{n}$$

For example, a Crowds network with 100 nodes, 10 of which are corrupt, and using $p_f = 0.75$ will have entropy $H \approx 5.24$ bits. The optimal entropy for such a system is $\log_2 90 \approx 6.49$ bits, so a slightly over one bit of information is revealed to the attackers. Reasoning about the effective anonymity set size, it is reduced from the optimum of 90 to about 38.

4.3.1 Conditional Entropy

In fact, this is a slightly pessimistic view of Crowds, because this entropy is obtained when one of the corrupt nodes intercepts a message. But some messages will be sent to the web after traversing only honest nodes; a message is only intercepted with probability $c/(n - p_f(n-c))$. In all other cases, the entropy of the system can be seen to be $\log_2(n-c)$, since the corrupt nodes make no observation. Diaz *et al.* suggest using a weighted average of the metrics in this case [25]:

$$H_0 = \frac{c}{n - p_f(n-c)} H + \left(1 - \frac{c}{n - p_f(n-c)}\right) \log_2(n-c)$$

Using this calculation for the above parameters of $n = 100$, $c = 10$, and $p_f = 0.75$, we get $H_0 = 6.11$, much closer to the optimal value. (The effective anonymity set size is about 69.) One interpretation of this is that H_0 describes the average anonymity provided by Crowds.

Another interpretation comes from information theory. Let us introduce two random variables: X, modeling the senders, and Y, modeling the predecessor observed by the colluding nodes. The domain of X is $\{1, \ldots, n-c\}$, while the domain of Y is $\{1, \ldots, n-c, \emptyset\}$, with $\emptyset$ representing the case where no predecessor is observed. We can use Crowds to define the joint distribution X, Y, showing how the probability of any combination of senders and observed predecessors.

We can see that the probability distribution computed in the previous section is really the distribution of X conditioned on a particular observation y. So the entropy metric evaluates $H(X|Y = y)$ for some y. And the weighted average H_0 can be restated as:

$$H_0 = \sum_y Pr[Y = y] H(X|Y = y) = E_y H(X|Y = y)$$

(We use E_y to denote expectation with respect to y.) This quantity is known as the *conditional entropy* of X with respect to Y, denoted as $H(X|Y)$. It represents the uncertainty about X given knowledge of Y. Since X models the senders and

Y models the observations of attackers, this is a natural quantity to use to model anonymity.

Because conditional entropy is an expectation measure, it is a good model for the long-term anonymity behavior of the system, involving many instances of messages and observations. We will therefore compute the conditional entropy of all the systems we analyze below. However, the averages involved do not fully capture the potential exposure relevant to a user of the system. For example, even though a request sent by a user of Crowds is observed with a probability less than 1, users might be concerned with how exposed they would be *given* that their message is intercepted. For other systems we analyze, there is a non-negligible chance that the anonymity provided to the users is significantly less than the conditional entropy; in some cases, the entropy may in fact be 0!

Users who value their privacy will be reluctant to use a system with such risks of exposure. Therefore, we will also look at the worst-case exposure a user may have to contend with:

$$H_{\text{worst}}(X, Y) = \min_{y} H(X|Y = y)$$

For our Crowds example, H_{worst} is the 5.24 value computed above.

4.4 Calculating Probability Distributions

The entropy metric is a useful tool for analyzing anonymous systems. However, before it can be applied, we must be able to calculate the probability distribution of senders given a certain attack scenario. In the above example of Crowds, this calculation is simple because of the fully connected structure. More complicated systems will introduce dependencies that must be analyzed. For small-scale systems, these dependencies can be addressed by a case analysis; for example, see the analysis of composition of mixes in [67].

The case analysis may be automated by constructing a probabilistic model of the system and using model checking tools such as Prism [45]. However, probabilistic modeling suffers from a state explosion problem and has thus far been successful only for systems with fewer than 20 nodes [30, 68].

In contrast, the systems we plan to analyze contain thousands of nodes, connected by dynamically constructed networks with various degrees of regularity, so all of the above methods quickly become intractable. On one hand, this can be seen as an advantage, since attackers must also figure out how to compute the probability distributions over possible senders in order to derive information from their observations. A large, dynamic system presents a complicated puzzle to solve, and it is likely that any realistic attacker will not be able to gain full knowledge of the topology, let alone be able to fully analyze it.

On the other hand, this is a dangerous property on which to base the anony-

mity of a system. It should be clear (and we will demonstrate this below) that the observations of the attackers have *some* information content. The hope is that the attackers will not have sufficient knowledge about the network or powerful enough analysis methods to make full use of this content. However, it is difficult to put a bound on the amount of network knowledge attackers will have. Even when the view of the network presented by the protocol is limited, there are many side channels that reveal communication patterns about other nodes. And barring complexity theoretic bounds, it is similarly difficult to impose reasonable limits on the attackers' analysis capacity.

So it could easily be the case that, although the best concrete attack we come up with today is ineffective at reducing the anonymity of the system, future attack strategies may be much more successful, making at least partial use of the information content in the system. Therefore, we will focus on techniques to measure the information content itself, with a view of perfect knowledge of the system. These techniques may not tell us how anonymous a system is under realistic assumptions, but they will provide a *bound* on the success of a class of attacks that rely on partial information.

4.5 Simulation-based Analysis

The key tool we will use is simulation of the system. Running a system in simulation both endows us with a complete knowledge of the network and enables us

to *measure* the information content of the system. Our approach can scale to quite large system sizes, and does not rely on being able to form a mathematical model of the system, allowing us to analyze complicated systems lacking a simple theoretical foundation.

We start by building a simulator of a system starting in some snapshot state. The snapshot includes all of the participants, all of their internal state, as well as connections between them. The snapshot also includes a model of an attack strategy, which describes how simulation events map to attacker observations. Generally, we will model attacks as a compromise of some of the nodes of the system, allowing attackers to observe (and potentially modify) all traffic going through those nodes. But this is not fundamental to the techniques, and other kinds of observations are possible.

Then we simulate communication events starting from that snapshot. Recall that to calculate entropy, we need to compute the joint distribution X, Y for X taking values among some domain of participants and Y taking values among a domain of observations. In our simulations, we pick a random value x for X and simulate a communication event with the corresponding participant. During simulations, we record the observations made by attackers according to our model. We will assume that every communication event maps to only one observation y; if this is not the case, it is easy to switch to the power set of the domain of Y (in most systems, only a small subset of the power set 2^Y will actually occur). We keep a counter $C_{x,y}$ for each pair of x and y, and increment the one corresponding to this simulation run.

We then rewind the simulation back to the snapshot, pick a different random value for x and repeat, incrementing the correct counter $C_{x,y}$. We repeat this process K times (the choice of K will be discussed below). What is happening here is that we are sampling the joint distribution of X, Y. The sampled value of the distribution at a point x, y is $q_{x,y} = C_{x,y}/K$ (we use q here for the sampled distribution to distinguish it from p, which will denote the true distribution). Note that this joint distribution will be affected not just by the snapshot, but also by the prior distribution we choose for X. We will tend to use a uniform prior distribution, but this is also not fundamental to our process.

Given the sampled distribution, we can compute the entropy metrics $\tilde{H}(X|Y = y)$ and $\tilde{H}(X|Y)$ on it. This will give us an estimate of the entropy of the system and hence the anonymity it provides. But there are several sources of error that may be present in this estimate. The first is that the sampled distribution may deviate from the true distribution, and hence produce inaccurate values for the entropy estimates. We will discuss sampling error in the next section. However, even if the entropy estimate is accurate, it may be the case that the snapshot we chose is not representative of real-world systems.

This is a standard problem in simulation, and we can take standard approaches to mitigate it. First, we can take measurements of actual systems to supply simulation parameters. Second, we can vary choices of parameters across some space and measure the reaction of the anonymity to the changes. This analysis will be a useful tool for better understanding the system, as it will illuminate which parameters are important

for anonymity. Finally, we can compute the variance in our measurements caused by random choices made in creating the snapshot. (Note that this will be *inter*-snapshot variance, as opposed to *intra*-snapshot variance discussed in the next section.) A low variance will give us greater confidence in our results. A high variance is cause for concern, since it indicates that anonymity is highly sensitive to the parameters of the system.

A final concern is that the simulation will not be a faithful representation of the real system. Once again, we can use standard techniques of validation against the real implementation. In fact, it may be possible to use the actual implementation of the system as a basis of for simulation. We can simulate the network of nodes using the real code for the implementation and generate communication events, sufficiently tagged so that we may accurately record observations and their corresponding participants at the intermediate point. We have not done this in our work due to scaling concerns, however, Section 8.2.5 discusses how efficient simulation using the real code base may be performed.

4.6 Sampling Accuracy

The sampled distribution q is only an estimate of the true distribution p. An important question is how closely does the distribution approximate the true one, and how much is the entropy estimate affected by the discrepancy between the two distributions.

The problem of estimating entropy has been studied in depth, first by statisticians and information theorists, and more recently by neural scientists, who use entropy and mutual information estimates to study the information content of neural signals [6, 8, 56, 71]. In this section, we will state without proof some relevant results; for more details, we refer the reader to [56].

Consider a sampled distribution q and the entropy computed over that distribution. This entropy estimate is called the *plug-in* estimator or the *maximum likelihood* estimator, or H_{MLE}. The first result is that this estimator is normally distributed, i.e. $H_{MLE} \sim N(\mu, \sigma)$ for a given mean μ and variance σ^2.

The mean and variance both depend on the distribution and the number of samples. The variance depends on how uniform the probability distribution is, as well as the number of samples K:

$$\sigma^2 = \frac{\sum_i p_i(-\log_2 p_i - H(X))^2}{K}$$

So while the variance adds a source of error, the error will tend towards 0 as $K \to \infty$. In addition, the variance can be bounded by $\sigma^2 \leq (\log_2 m)^2/K$, where m is the number of non-zero entries in the distribution p.

However, another (and often larger) source of error is the mean. The plug-in estimator can be shown to be negatively biased, i.e.:

$$\mu = H(X) + B, B \leq 0$$

The bias means that even averaging a large number of entropy estimates will produce

an incorrect result. Fortunately, this bias can be bounded by the following formula:

$$-\log_2\left(1+\frac{m-1}{K}\right) \leq B \leq 0 \tag{4.1}$$

Therefore, the bias also tends towards 0 as the number of samples grows. For any $\epsilon > 0$, we can find K such that $\sigma, B < \epsilon$ ($\sigma = \sqrt{\sigma^2}$ is the standard deviation.) In such a case, we can obtain a 95% confidence interval of size 3ϵ.

4.7 Crowds Simulation

To illustrate our simulation-based approach to estimating entropy, we apply it to Crowds. This will allow us to check the validity of our analysis as to the accuracy of the estimator by ensuring that the true entropy value always falls within the predicted confidence interval.

We built a simulator for the Crowds system. The Crowds algorithm is simple enough that our simulator is fewer than 100 lines of Python [74], including code to compute entropy. The simulator is parameterized by the number of nodes n, the number of corrupt nodes c, and the probability of forwarding p_f. To simulate a request in Crowds, we pick a random honest node x as the origin and forward the request according to the Crowds algorithm until it either is sent to the web server or is intercepted by a corrupt node. In the former case, we write down the observation $y = \emptyset$, while in the latter case we set $y = \mathit{pred}$, the number of the predecessor node. In both cases, we increment $C_{x,y}$.

Note that a more naive way to record the observations would be to include both the predecessor node and the choice of intermediate attacker node. However, a quick analysis shows that the latter is independent of the origin, and hence can be safely eliminated, reducing the size of the sampled probability distribution.

After K queries, we can compute $q_{x,y} = C_{x,y}/K$. We will begin by estimating the entropy given that the first node is seen as the predecessor, i.e. $H(X|Y = 1)$. Since only $K_1 = \sum_x C_{x,y}$ samples are relevant to this conditioned distribution, we use K_1 instead of K. Figure 4.1 shows the estimator results for different sizes of K_1. For each size, we compute 100 estimates of the entropy using H_{MLE} and graph the mean and the 95% confidence interval based on the sample variance only. The variance is quite small, even for small sample sizes, and the effects of bias are apparent: the estimate is monotonically increasing, and the confidence interval based on sample variance does not always include the true entropy value. Figure 4.2 shows the same data, but with the 95% confidence intervals computed according to the bounds from the previous section. These intervals are much larger, because they accurately reflect the error introduced by the estimator. Note that in all cases, the true entropy falls within the intervals shown.

The confidence intervals are asymmetric, since error due to bias is strictly negative. The variance estimates used in the two figures are also different: in the first figure, we use the sample variance, while in the second we use the variance bound of $(\log m)^2/K$. This bound is not tight, as can be seen by comparing the two figures. However, it has the advantage that it can be computed after a single run of K samples, while

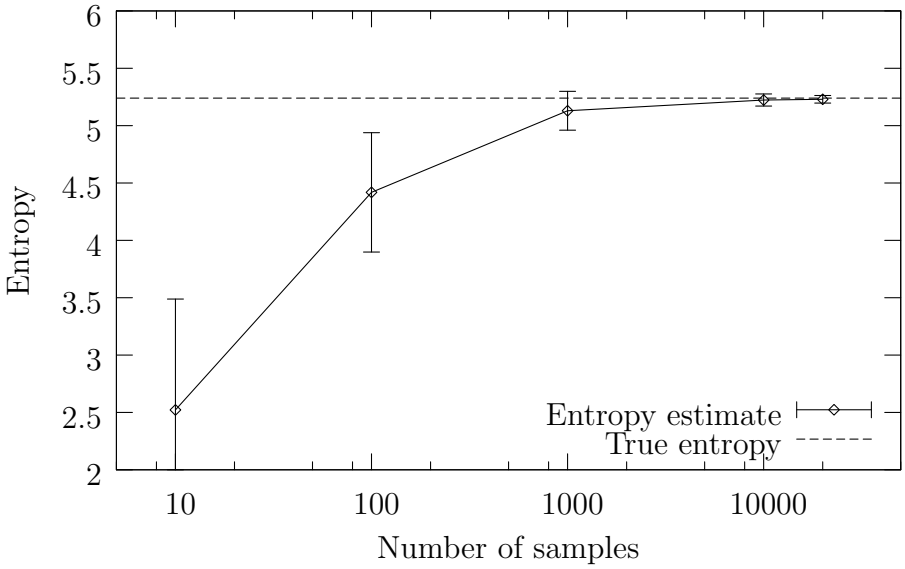

Figure 4.1: Crowds entropy estimator, with confidence intervals based on sample variance.

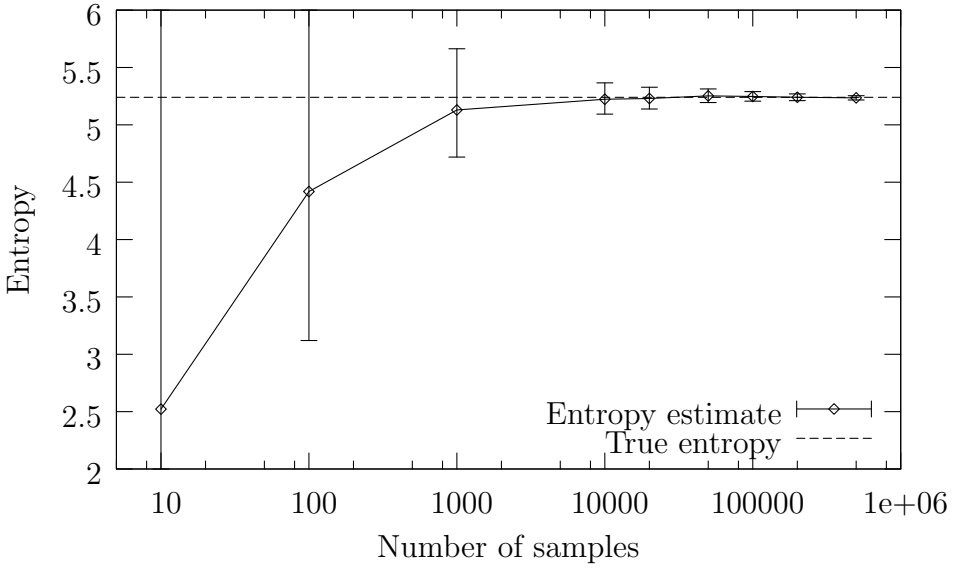

Figure 4.2: Crowds entropy estimator, with confidence intervals taking bias into account.

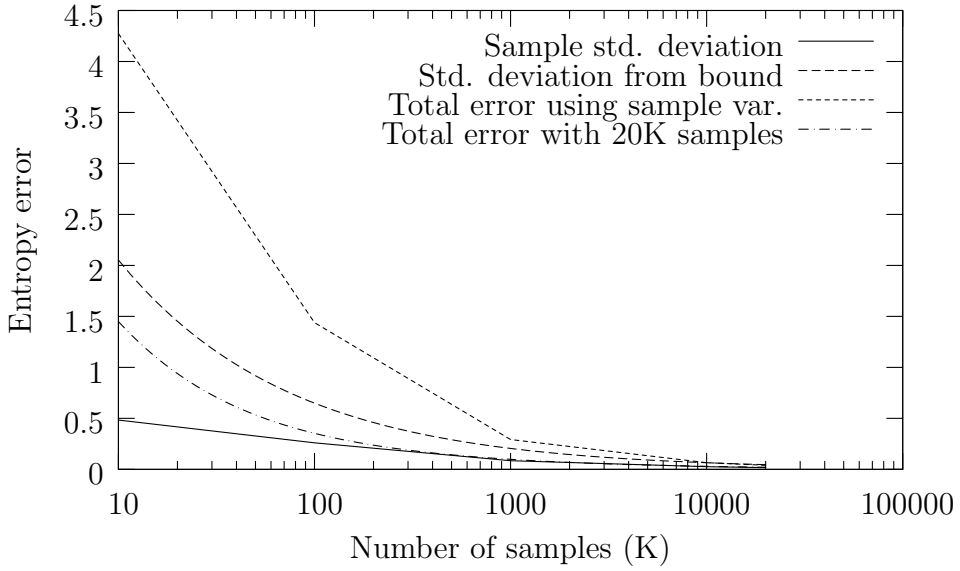

Figure 4.3: Comparison of errors using sample variance and analytical bounds.

the sample variance requires many repeated runs. Even if we used, say, 20 runs to estimate the sample variance, instead of the 100 above, we would get better estimates by using a single run of $20K$ samples and using the variance bound instead. (Indeed, we were able to compute the estimate with larger numbers of samples in Figure 4.2.) Figure 4.3 shows the difference in the total error produced by the two methods.

4.7.1 Conditional Entropy

We can use a similar process to estimate conditional entropy. Recall that the conditional entropy is the weighted average of individual entropies:

$$H(X|Y) = \sum_y p_y H(X|Y = y)$$

Using the sampled values of the probability distribution, and the entropy estimate derived as per the last section, we can compute:

$$\tilde{H}(X|Y) = \sum_y q_y \tilde{H}(X|y=y), \quad \text{where} \quad q_y = \sum_x q_{x,y}$$

Note that in this case we want to consider all the samples we collected, so we set $q_{x,y} = C_{x,y}/K$ and not $C_{x,y}/K_1$. The question is then how to bound the error for this estimate. We can bound error of each individual term, but there could be additional error introduced because q_y is a sampled value, rather than the true weight. We turn to an identity about conditional entropy:

$$H(X|Y) = H(X,Y) - H(Y)$$

where $H(X,Y)$ is the entropy of the joint distribution X, Y, and can be estimated the same way as in the previous section:

$$\tilde{H}(X,Y) = \sum_{x,y} q_{x,y} \log_2 q_{x,y}$$

The variance and bias of this function can also be computed according to the analytic bounds. This entropy estimate involves more samples (K instead of K_1), but it also involves a larger probability space:

$$m = |X \times Y| = 90 \cdot 91 = 8190$$

Similarly, we can estimate:

$$\tilde{H}(Y) = \sum_y q_y \log_2 q_y$$

And we can compute the bounds on the bias and variance for this distribution, using $m = |Y| = 91$ (recall that Y also includes the null observation). Let us call the bias bounds for $\tilde{H}(X,Y)$ and $\tilde{H}(Y)$ b_0 and b_1 respectively, and let σ_0^2 and σ_1^2 be their respective variances. Then, for the quantity:

$$\tilde{H}(X|Y) = \tilde{H}(X,Y) - \tilde{H}(Y)$$

the error due to bias must lie in the range $(-b_0, b_1)$. The variance is:

$$\sigma^2 = \sigma_0^2 + \sigma_1^2 - 2r\sigma_0\sigma_1$$

where r is the correlation coefficient between $\tilde{H}(X,Y)$ and $\tilde{H}(Y)$ (since the estimates are based on the same observations, they are not independent). $-1 \leq r \leq 1$, so:

$$\sigma^2 \leq \sigma_0^2 + \sigma_1^2 + 2\sigma_0\sigma_1$$

Figure 4.4 shows the conditional entropy, along with the 95% confidence intervals obtained using the methods below. We see that in this case, the variance and bias are significantly overestimated by the bounds. This is largely due to the nature of the probability distributions involved in Crowds. For example, $H(Y) = \log_2 |Y|$ because, averaging over all sources, all observations are equally likely. Similarly, the joint distribution X, Y is nearly uniform: for all $y \neq x$ and $y \neq \emptyset$, $p_{x,y}$ is the same. This means that sampling is in fact much more effective at estimating the entropy of these distributions than the bounds, which assume the worst-case. However, we believe that for more complicated systems and corresponding probability distributions, the

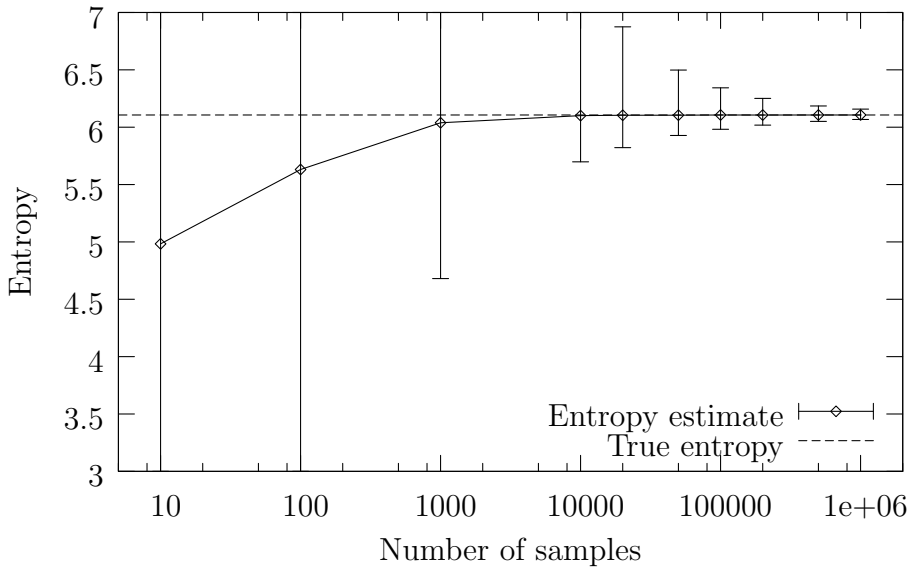

Figure 4.4: Crowds conditional entropy estimate.

errors due to bias and variance will be larger. Therefore, our approach is less sample-inefficient than it seems given these graphs. And, the availability of error bounds, however conservative, will be important for computing the confidence of our estimates in cases where we cannot discover the entropy through other means.

4.8 Summary

We have presented a method for studying anonymous systems using simulations and the entropy metric. We argued that although measurements according to this method do not correspond to any concrete attack, they provide a bound on the success of an entire class of attacks and are therefore useful to gain confidence in the performance of an anonymous system. We discussed sources of error introduced in

our method and showed how to bound them. Having applied our method to a simple example, we are now ready to use it on more complicated systems that have thus far resisted any analysis using existing methods.

Chapter 5

Freenet Analysis

5.1 Overview of Freenet

Freenet [18] is a peer-to-peer anonymous publishing system. In Freenet, a network of nodes is collectively responsible for storing and serving a set of documents. Each documents is stored on a subset of nodes in the network and is indexed by the hash of its contents, called the *content-hash key*. This key is used to locate the document in the network. Freenet also has a separate directory service to map names from a certain namespace to content-hash keys. We will focus our analysis on the mechanism to find documents based on content-hash keys.

Each node maintains a routing table with references to several other nodes. The routing table includes a list of keys that each neighbor nodes is likely to serve. To find a document, a node will send a query to the neighbor with the closest matching key. If that node does not have the document, it will recursively forward the query

to the node in its own routing table with the closest key, and so on. If a routing loop is detected, the process backtracks and the query is forwarded to the node with the next closest key. A time-to-live field in the query is used to limit the extent of the search.

A successful query causes the routing tables of the nodes along the path to the data to be updated with a reference to the node that provided the data. Since the local storage on each node is limited, the routing table is limited in size; if it becomes full, the least recently used entry is removed. The document is sent back using the path of the query, and is usually cached along the return path. Caches are also updated by discarding least recently used items, resulting in popular documents being widely replicated in the network.

Because of the highly dynamic nature of a Freenet network, it is hard to say anything certain about the kind of topology it might have. The design intuition is that each node will specialize in storing and locating similarly valued keys. The larger structure of Freenet is expected to follow a power-law graph, with several high-degree nodes providing connectivity between nodes with smaller degree. Such a structure should support efficient routing even for large networks. Indeed, simulations show that Freenet networks can scale to many thousands of nodes and still maintain short median routes [18].

5.2 Simulation

To perform the entropy experiments on Freenet, we built a simulator for the Freenet routing algorithm. We based our simulator on the freely available Java source code for a Freenet node. We re-implemented the routing algorithm in C++ and we validated the correctness of our implementation by performing the experiments listed in the Freenet overview paper [18]. The resulting simulator is less than 1000 lines of code.

In our experiments, we select uniformly and independently from all nodes a fraction c to be attacker nodes. (This uniform choice does not necessarily represent the best attacker strategy, and we will explore other possibilities later, but it is a helpful starting point for analysis.) Attackers keep a record of all messages received, but otherwise honestly participate in the protocol. The observations recorded are of the form "node B forwarded a query for key K to attacker A."

In fact, we exclude the identity of A in the recorded observation, since the choice of the next node is independent of the origin. The attacker node also could include the TTL among the observations. Freenet nodes are supposed to occasionally randomize TTLs of forwarded messages to reduce the information that can be learned from the TTL value. However, there will probably be some correlation between the TTL and the distance from the origin node, even after these distortions. For simplicity, we make our attackers oblivious to the TTL values, and use a simple scheme of setting TTL to 100 at the query origin.

We build up a network by starting with a single node and then performing the join algorithm to add new nodes at a random point in the network. Interspersed with joins are inserts of random keys and queries for previously inserted keys. The inserts and queries ensure that the routing tables are similar to what they would be on an actual Freenet network; they also fill the caches at each node with copies of simulated documents. Attackers are dormant during the build phase and do not perform any logging. Once the network reaches the desired size, we take a snapshot of the network and proceed to the measurement phase.

5.3 Entropy Measurements

Once we have the snapshot, we are ready to compute the entropy. We pick a random key and successively perform a query for it starting at each honest node. Whenever a message arrives at an attacker, an observation is recorded. Note that Freenet routing is deterministic, hence we only need one message from each node to fully sample the distribution. Once an attacker is encountered, we stop the query routing and continue to the next source; this is done to simulate collaborating attackers. After the first attacker sees a message, forwarding it further reveals no new information about the source to the attackers. In a real attack scenario, attackers would have to forward the message to avoid detection; however, we assume that they would be mark forwarded messages and discard observations about them when they are seen again. With each query, we roll back the state to the snapshot to accurately

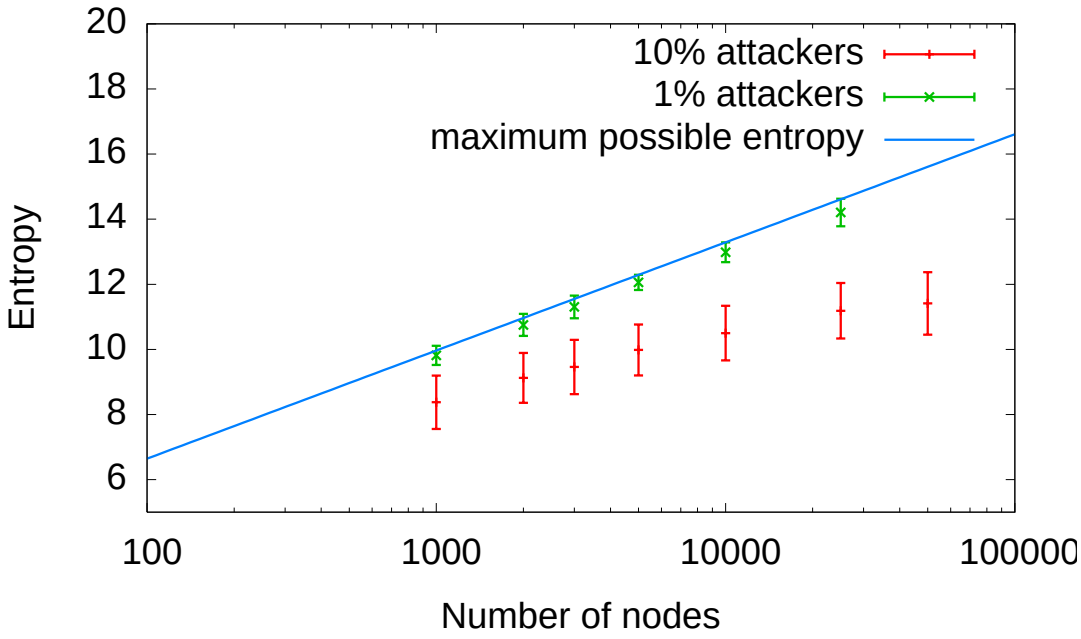

Figure 5.1: Entropy metric applied to Freenet.

model the attackers' perspective of a non-deterministic choice of sender.

Figure 5.1 shows the conditional entropy computed for several choices of c and N. For example, for a network with 1000 nodes and 100 attackers, the average conditional entropy value is 8.3 bits. This average is computed over several random choices of generated networks, distributions of attackers, and queries. We can interpret this to mean that on average, the attacker has about 8.3 bits of uncertainty about the origin of a message; in other words, the source can be narrowed down to about $2^{8.3} \approx 300$ nodes. If we increase the network size to 10000 and scale up the number of attackers correspondingly to 1000, the average entropy now becomes 10.5 bits, equivalent to about 1500 nodes. Notice that since entropy is a logarithmic value, a small increase

corresponds to a large difference in anonymity.

As we add more nodes, the entropy continues to grow, but not nearly as quickly as the network; that is, doubling the size of the network does not generally result in an extra bit of entropy. One reason for this effect is that messages tend to require more hops in larger networks while the fraction of attackers stays constant, so message is more likely to be intercepted. Recall that the conditional entropy calculation includes those cases when the query is not observed at all. For a network with 1000 nodes, the mean query path length is only 3 hops and thus only 18% of all requests get observed when there are 10% attackers. With 10000 nodes, the path length increases to 5 hops, and the number of observed requests to 28%.

With only 1% attackers, the number of observed requests stays small; at 10000 nodes, only 3.2% of the requests are observed. Therefore, the conditional entropy with 1% of attackers is close to the maximum possible. However, when attackers *do* observe the query, the entropy corresponding to that observation is quite low, as the anonymity set excludes all nodes that are able to avoid observation entirely, which is most of them.

To compute the variance depicted in the graphs, we performed 500 simulations, each with a different configuration. We also experimented with using the same network configuration and varying the choice of attackers, or keeping the same choice of attackers and varying the choice of query key only. We wanted to see if the variance was caused by particular network configurations that resulted in worse anonymity. However, this was not the case as the variance was roughly the same even when

the configuration was kept static. Note that in this case, there is no bias or variance introduced due to sampling, because the default routing algorithm of Freenet is deterministic, hence our simulation explores the full space of possible paths in the network.

In addition to the conditional entropy figures, we were interested in the distribution of entropy numbers between various source nodes. Different source nodes will cause different observations, with differing corresponding entropies. The conditional entropy represents the average over all source nodes, but Figure 5.2 shows the cumulative distribution function of entropies with 10000 nodes and 10% attackers. Since only 28% of all queries get observed, the remaining 72% result in the null observation and corresponding maximal entropy. However, for most of the remaining queries, the entropies are quite a bit smaller, with 7% of queries resulting in 0 entropy, i.e. with the nodes being directly identified. These nodes are likely the low-degree nodes in the power-law distribution, forwarding their queries directly to attacker nodes.

The bottom 20% of queries have entropy below 5, which means that the effective anonymity set size (32) is many orders of magnitude smaller than the number of nodes in the system. In short, whenever a query is observed, the resulting entropy is highly likely to be very small, and the main cause of the reasonably large conditional entropy is the prevalence of the null observation. This is a significant concern for any Freenet users sensitive about their privacy, since, in this particular case, they would have a one-fifth chance of receiving little or no anonymity protection from Freenet. Furthermore, since the probability of a query being observed depends on the routing

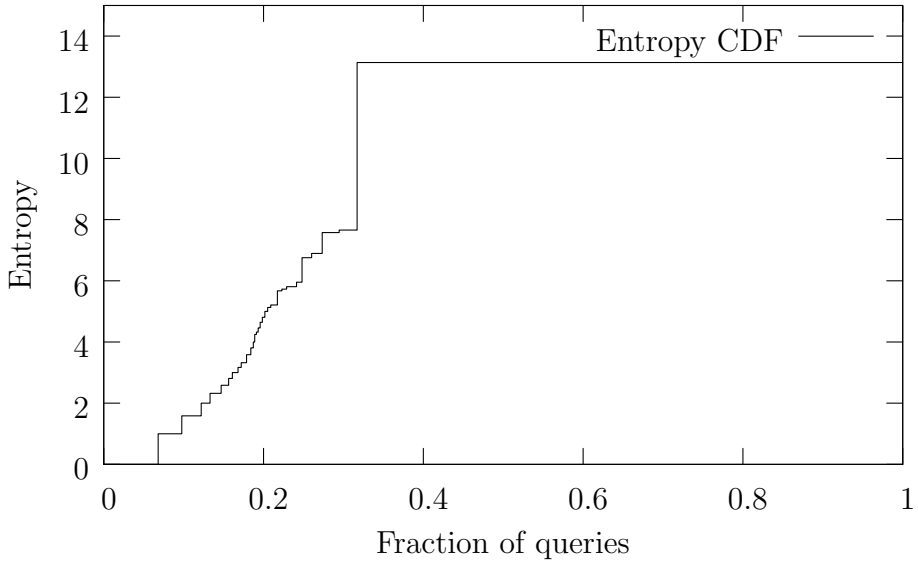

Figure 5.2: Cumulative distribution function for Freenet entropy with 10000 nodes, 10% attackers.

tables of other nodes and the placement of attackers, the users would have no way to control, or even measure, the level of anonymity protection they receive, and instead would have to leave the matter to chance.

5.4 Targeted Attacks

The degree of nodes in the Freenet network can be shown to follow a power-law distribution. This means that some nodes will have a much higher degree than others. This has been shown to have an effect on the resilience of Freenet to denial of service attacks [17]: if the high-degree nodes are attacked, the network is likely to become partitioned. In this section we examine the effects of the power law distribution on anonymity.

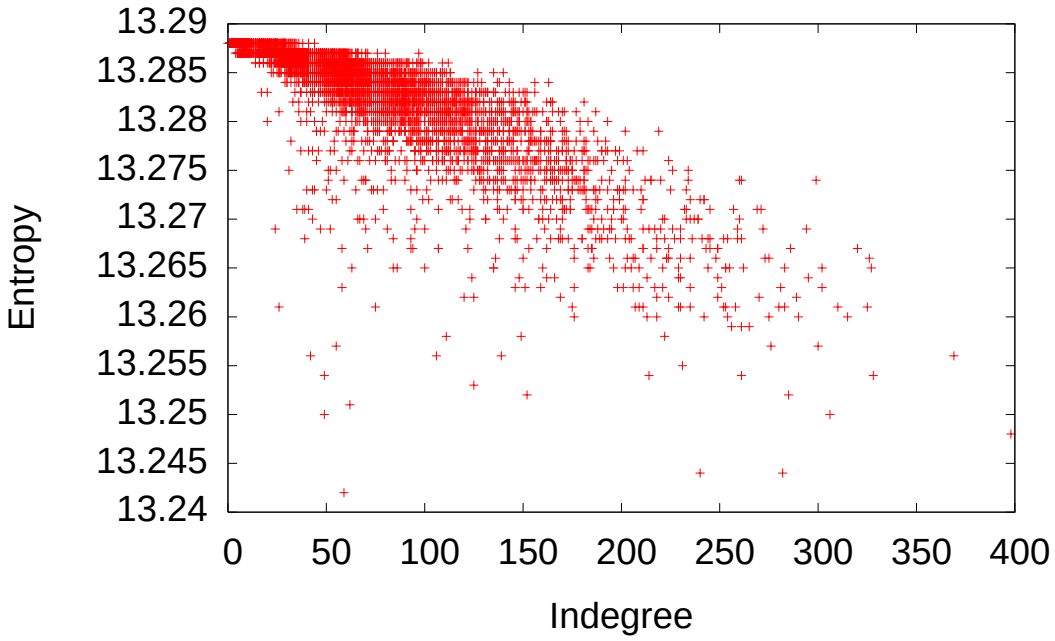

Figure 5.3: Usefulness of nodes by indegree.

High degree attacker nodes will lie on many more paths and thus will intercept more messages than low degree ones. On the other hand, the paths passing through them will be used by a large number of nodes, so each observation might carry less information than in a low degree node. To examine this tradeoff, we computed how much information each node learns individually. For each node, we computed the entropy it would observe if it were the *only* attacker in the system. Figure 5.3 shows a scatter plot of these entropies versus the indegree of a node, for a network of 10000 nodes. (Note that the vertical axis has a very small range, as a single attacker, even with a high indegree, will still see only a small fraction of queries.) Although the indegree is not the only factor determining the effectiveness of a node, there is a clear

correlation.

Attackers may exploit this fact, either through actively compromising nodes that have a high indegree, or by attempting to acquire a high indegree for existing nodes. The latter may be accomplished by acting as a high-performance node. Nodes in Freenet form neighbor links to nodes that successfully answer queries; by increasing the sizes of its neighbor and key caches, an attacker node will be able to answer more queries successfully and thus it will acquire more inbound links.

Figure 5.4 shows the conditional entropy metric computed when the attackers are chosen out of the nodes of the highest indegree, rather than randomly. The entropy is dramatically lower than in Figure 5.1. When the top 10% of 10000 nodes are attackers, the conditional entropy is only about 4.7 bits, which means that a node's identity can be effectively narrowed down to one of 25 nodes.

The aim of Freenet was to create a decentralized network with no single point of trust or failure. However, the power-law degree distribution of nodes means that the high-degree nodes can mount effective attacks against the network, impacting both resilience and anonymity; in essence, these nodes must become *de facto* trusted nodes. This situation is even worse than a centralized solution, where the trusted nodes are at least selected from high-assurance machines. To ensure a network where trust and responsibility is truly distributed, the design of Freenet must be changed to avoid such points of concentration.

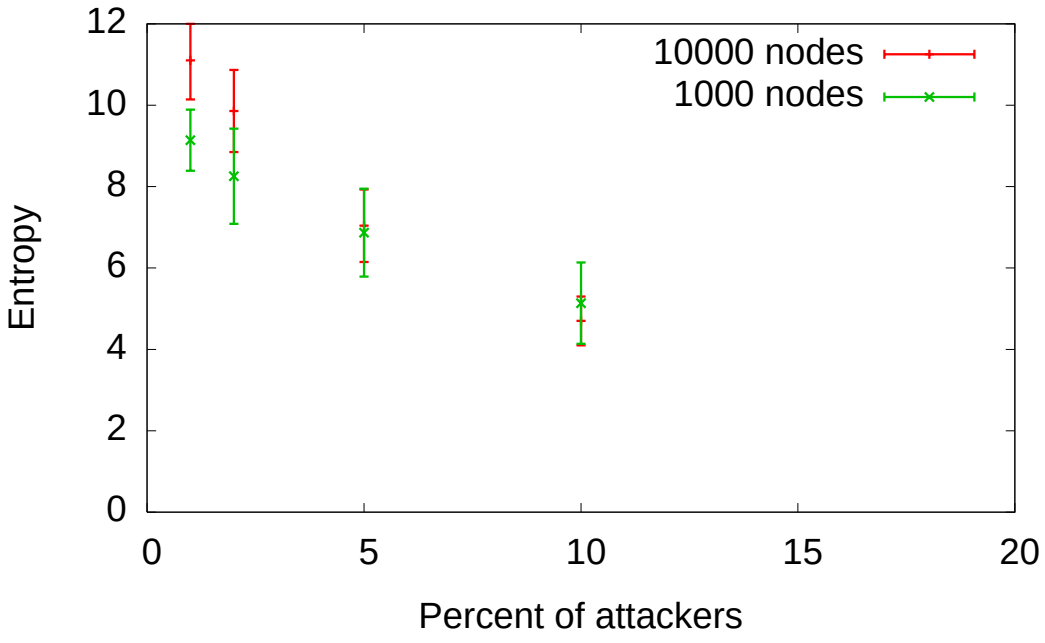

Figure 5.4: Targeted attacker strategy.

5.5 Time Evolution

Our model of an adversary who knows the entire routing tables at each node is not very realistic. To model a more limited adversary, we modify our strategy. Starting at the snapshot, we simulate a sequence of D random insertions and queries, modeling the evolution of the system state over D time steps without recording any observations. After these steps, we perform a query for some key K from each honest node, and record attacker observations as above. Then we roll the simulation back to the snapshot and simulate D time steps again, making new random choices, and then record the observations after D steps. We repeat this process a total of T times.

Our original analysis, by holding the topology static, effectively conditioned the

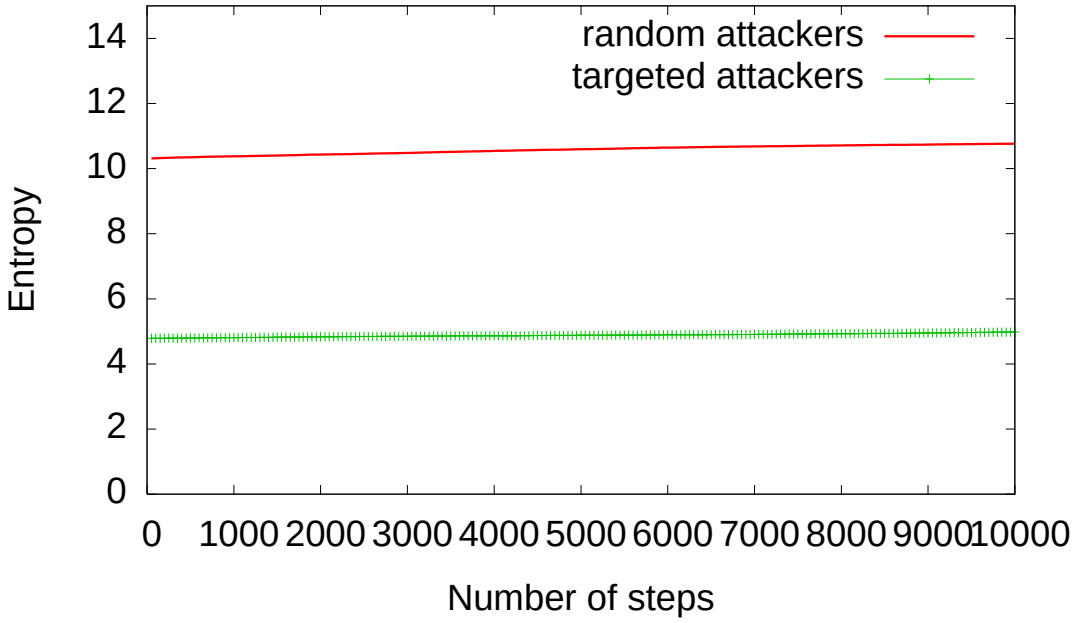

Figure 5.5: Entropy metric with limited knowledge.

probability distribution of sources and observations on a given network topology. By introducing these random changes on each of the T runs, we are relaxing this condition and instead conditioning on the state of the system D steps ago, with random changes in between. Therefore, this simulates an attacker with complete knowledge of a system D steps ago, and no knowledge of any the changes since. We use this more limited attacker model to simulate some knowledge of the *static structure* of the network, but not its transient properties.

Figure 5.5 shows entropy values for a 10000-node network with 10% attackers for increasing values of D. The amount of uncertainty an attacker has to deal with indeed increases with time. However, even after 10000 random insertions and queries, the

attacker's uncertainty has increases by less than half a bit. This suggests that the routing structure of the network does not change drastically over time, which means that attackers may be able to infer a large part of the graph structure using repeated probing.

We must be careful to account for bias and variance introduced by the sampling inherent in this approach (the random choices introduced in modifying the network structure mean that our simulations are no longer deterministic). In this case, we were able to collect a large enough number of samples to reduce these error contribution to be about 0.1 bits of entropy, according to the bounds from Section 4.6. ($m = 484093$ and $K = 4500000$, so bias is at most about -0.10 and the standard deviation is at most 0.0062.)

An important consequence of this analysis is that results obtained through simulation of perfect knowledge of the network topology, as was carried out in the previous sections, is a good approximation of the anonymity of the more dynamic network. This allows us to analyze larger systems with more precision because we can use deterministic analysis. To confirm this with an example, Figure 5.5 shows the entropy of values for a targeted 10% of attackers compromising the nodes with the highest indegree. After 10,000 steps, the entropy value remains under 5 bits, and thus this attack strategy remains quite effective.

Figure 5.6 shows the cumulative distribution function of the entropies corresponding to different queries after 10000 steps of time evolution. In this case, the entropy increases more smoothly than in Figure 5.2, and many fewer queries result in a zero-

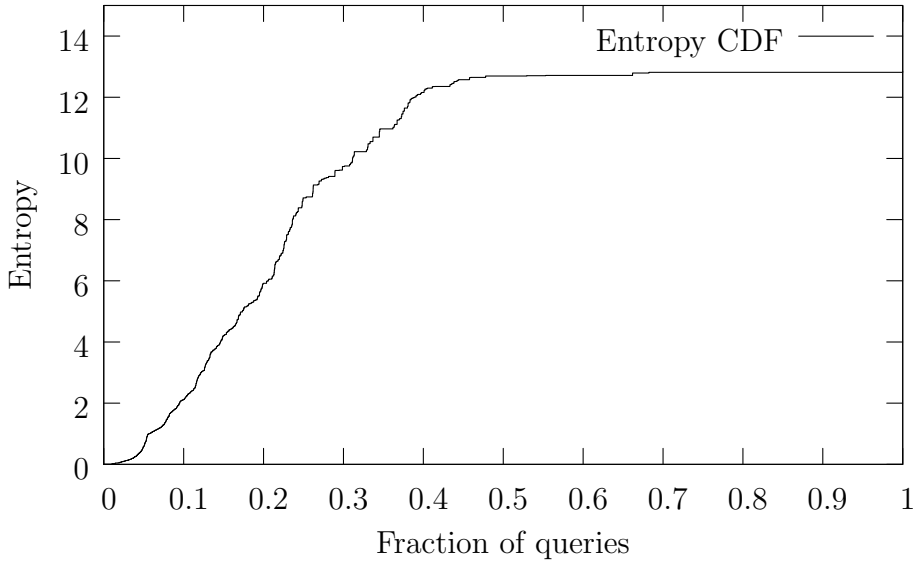

Figure 5.6: CDF of entropy after 10000 steps of time evolution, with 10000 nodes and 10% attackers.

entropy observation. Nonetheless, the entropy of the bottom 20% of queries is still no more than 5, meaning that even with a limited view of the network topology, the attackers can effectively isolate the origin of one-fifth of all queries to within a small set of nodes.

5.6 Next-Generation Routing

Freenet is undergoing continued development; one of the current projects is an improved "next-generation" routing algorithm [16]. The goal of next-generation routing is to make the Freenet network adapt to network conditions and help it pick paths with better end-to-end latency. We will omit the full details of the algorithm, but the key idea is to replace the list of keys in the routing tables with an estimator of

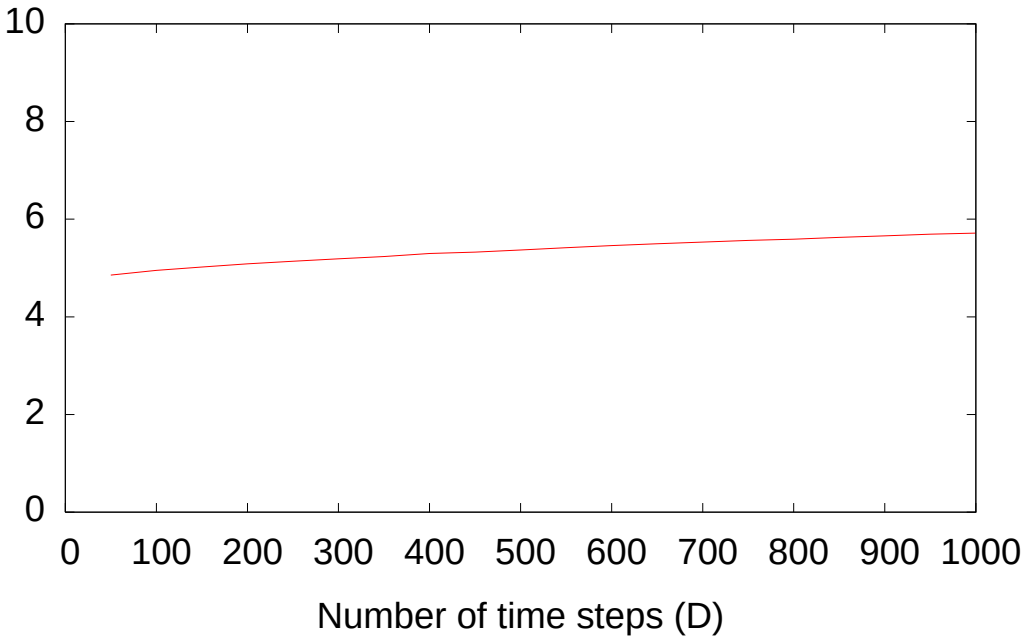

Figure 5.7: Next-generation routing.

how long a node is likely to take to answer the query. This is done by way of a piecewise-linear estimator of query time based on the key value, updated whenever a query is performed.

The exact details of the algorithm are still in flux as the estimator is tuned to achieve better performance. One question we are interested in is what effect the new routing algorithm has on anonymity. In order to study this, we implemented a simplified version of the algorithm inside our simulator and used it to measure entropy.

We use the approach from the previous section to model a more limited attacker. Figure 5.7 shows the analysis of a 1000-node network with 10% attackers, with an

increasing number of time steps. Since the next-generation algorithm is more involved, we were unable to run our simulations for as many time steps. But it is easy to see that the latency-aware routing introduces more structure to the routing: there are several fewer bits of entropy than with the regular routing, and the low entropy is preserved even after updates to the network. Furthermore, this structure is likely to be influenced by the characteristics of the underlying network layer, which can readily be measured by attackers.

This should be a cause for concern for using the new algorithm. The reduction in anonymity is perhaps not surprising, as a low latency for queries will limit the anonymity set to a small network neighborhood. However, the current version of Freenet introduces latencies that are several times the diameter of the network; it remains an open research question how much can latency be improved without sacrificing anonymity. Our empirical methodology can serve as a useful tool in such research.

5.7 Summary

The methodology developed in the previous chapter has allowed us to develop the first rigorous analysis of anonymity in Freenet. We have found that, although the average anonymity achieved by Freenet may lie within an acceptable range, there is a lack of uniformity in the level anonymity provided to different nodes and queries and a significant fraction of all queries will receive little or no anonymity protection. This should make privacy sensitive users cautious about using such a system, since

their anonymity depends on factors beyond their control (and in general, ones they can't even measure).

In general, Freenet queries receive anonymity largely only when they are not observed by attackers. This is particularly a concern since, as we have shown, targeting high-degree nodes can help attackers see many queries after compromising only a small fraction of nodes. Our analysis also shows that the next-generation algorithm, aiming to improve Freenet performance, results in significantly lower anonymity levels. Using time evolution, we demonstrate that these results remain true even with a more relaxed model of attacker knowledge.

In the following chapter, we will develop a new design for anonymous peer-to-peer networks to address all of these issues, by providing near uniform levels of anonymity and guaranteeing acceptable worst-case performance, eliminating the possibility of targeted attacks, and providing improved performance without sacrificing anonymity.

Chapter 6

Structured Anonymous Peer-to-Peer Overlays

As we saw in the previous chapter, the design of Freenet [18] resulted in highly variable in anonymity, with a significant chance of providing little or no anonymity to its users. We conjecture that this feature is a consequence of the ad-hoc network construction used by Freenet, and therefore consider a *structured* approach to anonymity. After discussing some advantages of structured networks, we lay out a basic design for building structured anonymous p2p overlays. Then we go into some specifics, in particular, what kinds of structured networks are best suited for anonymity.

6.1 Structured Overlay Basics

A structured overlay network is one where connections between nodes are made according to some algorithm. There are many structured designs, such as [70, 60, 65, 77, 41], but they all share a few common features. They all use a large identifier space, usually either $\mathbb{Z}_{2^{128}}$ or $\mathbb{Z}_{2^{160}}$, and partition this space among the overlay nodes. Connection between nodes are made according to a formula based on the regions of the identifier space they control, and are designed to approximate some regular graph structure, such as the butterfly or hypercube. The graph structure is designed to be able to efficiently route to a node responsible for any particular identifier in the space.

The structured networks all guarantee that each node has links to either a constant number of other nodes, or to $O(\log N)$ nodes, where N is the total number of nodes in the overlay. Similarly, routing to any point in the identifier space can be accomplished in $O(\log N)$ steps across the overlay.

6.2 Why Structured Overlays?

There are several factors that make structured overlays an attractive platform for anonymous systems. To start, they are more efficient than their unstructured counterparts. The routing algorithms in unstructured networks do not even guarantee reachability, let alone some performance bound. More efficient networks will be more attractive to users, and therefore gather a large user base. Further, there is a trade-off between anonymity techniques and efficiency; starting with a more efficient baseline

will make it more likely that users will be willing to pay the performance penalty for anonymity.

Structured networks are efficient not only in terms of routing performance, but also in terms of state. A node following the normal protocol will know only a constant or logarithmic number of nodes. This limited view gives attackers less information to work with while trying to discover the identities tied to a particular communication event. Of course, deviating from the protocol may result in a broader picture of the network, and in our analysis we do not rely on attackers having a limited view.

The regular structure of the underlying connection graphs has several important implications for anonymity. Regularity implies symmetric node roles, with no node being a more attractive attack target than any other node. The regular structure also means that we should expect most properties of the system to be more uniform. In particular, there is a hope that if we can achieve good anonymity on average, a similar level of anonymity will be provided in most cases, avoiding the pitfalls of variable anonymity present in networks such as Freenet.

Finally, structured networks are easier to analyze. Easy analysis does have the potential to help both network designers and attackers. However, as we will see, we are able to provide anonymity guarantees that are higher in confidence and better in terms of entropy than the unstructured design of Freenet. Thus, although topology-based attacks are more realistic against structured overlays, we are better able to bound their success. We argue that this is preferable to the uncertainty present in unstructured networks, where the topology reveals *more* information, and users rely

only upon hope that most of the topology will remains unknown.

6.3 Anonymous Distributed Hash Table

Our goal in the remainder of this chapter will be to present a design for an anonymous *distributed hash table* (DHT). The DHT is a primitive implemented by all of the structured p2p overlays. It allows `put` and `get` operations to be performed on the overlay, keyed by an index into the identifier space. The keys are usually computed by a hash function, such as MD5 [64] or SHA1 [54], motivating the use of an ID space of size 2^{128} or 2^{160}. For example, the DHT may store documents, indexed by the hash of their contents. Such an application would be similar to Freenet, though with much better performance guarantees. There are many other applications, in some of which the DHT provides not a storage but rather a location service.

We aim to augment the DHT design to provide anonymity to the users of the DHT API. In other words, we want users to perform the `put` and `get` requests while not revealing their identity. This will allow to directly anonymize many of the applications designed to run on top of DHTs. In the context of the DHT, our goal can be seen as *sender* anonymity, since we protect the node sending a `put` or `get` request, but not the node receiving and serving the request. However, for many applications, the anonymous DHT design will provide *publisher* anonymity as well, because publishers will interact with the DHT by issuing `put` requests, while the nodes servicing the requests for a particular key will be random and not have any relationship with the

data or service involved. For other applications, rendezvous servers [35] may be a way to achieve destination anonymity.

6.4 Recursive Routing

There are two ways of routing queries inside a DHT [69]. In *iterative* routing, a node will contact its neighbors and find out the address of the next peer it should contact in order to reach the destination. Then it contacts that peer and finds out the address of the next hop in the route, and repeats this process until the destination is found. The query is then sent directly to the destination node. In *recursive* routing, on the other hand, the query itself is sent to a neighbor of the origin node, who forwards it to the next peer recursively, until it reaches the destination. Responses are then passed back along the recursive path, though it is possible to shortcut this step by including the address of the originator in the query.

Recursive routing offers some measure of anonymity, as intermediate nodes do not know where a query originated. In this way, it is similar to the routing of queries in Freenet. Earlier work by us and others [11, 55, 44] explores the viability of using recursive routing, or a simple modification thereof, to achieve anonymity in the structured Chord [70] overlay. However, a fundamental difficulty with this approach is that routing in Chord follows a well-defined distance metric, with queries always being forwarded to nodes that are closer to the destination, according to this metric. This means that any intermediate node need only consider those nodes that

are further away than itself as the possible origin. It also need not consider nodes between itself and its immediate predecessor, since they would have had to send a query "backwards" in order to follow that path.

This presents a problem for nodes that are located far away from the destination. If the first node in the chain is an attacker, it can identify the origin with high probability. In fact, details of Chord routing render about half of all nodes easily identifiable during their first hop. Several randomization strategies explored in [11] reduce the extent problem, and may in fact achieve an acceptable average level of anonymity. However, the fundamental difficulty experienced by nodes far away from the destination remains.

The concept of a distance metric is not unique to Chord, but rather is shared among all the structured p2p overlays (it is used to prove guaranteed termination of routing.) Therefore, recursive routing will always leave some nodes exposed. We will need some other mechanism to protect these nodes, which we will discuss in the next section. However, we will still use the recursive routing primitive, since its mechanism of hiding the origin is useful, if not sufficient, for providing anonymity.

6.5 Random Walk Design

To provide an extra level of anonymity protection, we split query routing into two phases. For the first phase, we borrow an idea from Crowds [62] and execute a random walk. This phase will be the key to achieving anonymity, by sending a query

to a random point in the network. In the second phase, the node at the terminus of the random walk will send the query to the destination using recursive routing.

A random walk serves to hide the origin of the query. A sufficiently long random walk will arrive at a point in the network that is nearly independent of the starting point. This can be seen as a consequence of some facts about the *stationary distribution* of random walks on graphs.

For a given graph, we can construct a discrete Markov process modeling random walks on that graph. The states of the process correspond to the vertices of the graph, and the transition probabilities are based on picking a random edge out of a given state. Let (X_t) be the sequence of random variables modeling the Markov process at time step t. Given some probability on the states p, we call $P_p(X_t)$ the probability distribution of the Markov process started with distribution p after t steps. Similarly, $P_i(X_t)$ will be the probability of distribution of a Markov process started in state i after t steps.

For any strongly connected, non-bipartite graph the corresponding Markov process has a unique stationary distribution p_s, such that $P_{p_s}(X_1) = p_s$. Given any initial distribution of p, it is the case that:

$$\lim_{t \to \infty} ||P_p(X_t) - p_s|| = 0$$

where $||X - Y||$ is the variation distance between two distributions, defined as:

$$||X - Y|| = \max_i |x_i - y_i|$$

In other words, a random walk will always converge to the stationary distribution.

This is the key observation for our design. Given a sufficiently long random walk, the distribution of states will be arbitrarily close to the stationary distribution, and therefore will be independent of the starting point, to within a small margin of error. The number of steps that will be sufficient will be determined by by the *mixing time* of the Markov process, i.e. how quickly the distribution converges to stationary.

Similar to Crowds, we use random walks with variable lengths. During the random walk phase, each node will forward to a random neighbor and continue the walk with probability $(L-1)/L$, and switch to phase 2 recursive routing with probability $1/L$. The random walk will have expected length L, but it may be shorter or longer. This variability will have to be accounted for in our analysis. We cannot control the length of the random walk precisely through methods such as a time-to-live (TTL) field, since these methods would reveal how far along the random walk a node is and therefore compromise the identity of the origin to the first hop node.

6.6 Mixing Times

In this section, we will examine the mixing times of various distributed hash tables. A faster mixing time will mean that we can make phase 1 shorter and achieve the same mixing results, hence we will want to use a topology that provides the best mixing times.

A key tool in our analysis will be coupling [4]. Given a Markov process and two states i, j, we define a joint process $(X_t^{(i)}, X_t^{(j)})$, such that $X_t^{(i)}$ is distributed as the

chain started in state i, and $X_t^{(j)}$ as the chain started in state j. Let the *coupling time* T^{ij} be a random time such that:

$$X_t^{(i)} = X_t^{(j)}, T^{ij} \leq t < \infty$$

Then the coupling inequality states that:

$$||X_t^{(i)} - X_t^{(j)}|| \leq Pr[T^{ij} > t]$$

Therefore, by computing the coupling time, we can give a bound on the mixing time of the process.

6.6.1 De Bruijn Graphs

Several proposals for distributed hash tables are based on the de Bruijn graphs [46, 53, 32, 41]. De Bruijn graphs are attractive for p2p applications because they have a minimal diameter for any practical regular graph.

A de Bruijn graph [23] consist of nodes, labeled with integers from $\mathbb{Z}_{k^d}$. A given node x has k outgoing edges connecting it to $kx + j \pmod{k^d}$ for $j = 0, \ldots, k-1$. The node labels can be seen as d-digit strings in base k, with the connections defined by shifting the label to the left and shifting in a new digit from the right.

We will construct a coupling process for two arbitrary nodes x and y. Assign probabilities to the joint process:

$$Pr[(x_t, y_t) \rightarrow (kx_t + j, ky_t + j)] = \frac{1}{k} \qquad \text{for } j = 0, \ldots, k-1$$

$$Pr[(x_t, y_t) \rightarrow (x_{t+1}, y_{t+1})] = 0 \qquad \text{otherwise}$$

It is easy to see that $X_t^{(x)}$ and $X_t^{(y)}$ are distributed correctly to use the coupling inequality. Then we can write:

$$(x_t, y_t) = (j_t + k(j_{t-1} + k(\ldots + k(j_1 + kx)\ldots), (j_t + k(j_{t-1} + k(\ldots + k(j_1 + ky)\ldots))$$
$$= \left(\sum_{i=1}^{t} k^{t-i} j_t + k^t x, \sum_{i=1}^{t} k^{t-i} j_t + k^t y\right)$$

Therefore, for $t \geq d$, $x_t = y_t$ since $k^t x \equiv k^t y \equiv 0 \pmod{k^d}$. Applying the coupling inequality,

$$||X_t^{(i)} - X_t^{(j)}|| \leq Pr[T^{ij} > t] = 0 \quad \text{for } t \geq d$$

Therefore, after d steps, a random walk will converge precisely to the stationary distribution. It is easy to see that the bound derived by coupling is tight, since any random walk shorter than d steps will not be able to reach all of the k^d nodes. Intuitively, each step of the random walk shifts in a new random digit from the right, so after d steps the label will consist of d uniformly random digits.

6.6.2 Chord

Chord [70] is a popular distributed hash table. Its topology is similar to a hypercube with $\log_2 N$ dimensions. In Chord, nodes have identifiers in the space $\mathbb{Z}_{2^k}$, usually with $k = 128$ or 160. Given a node with identifier x, for each $j = 0, \ldots, k-1$, the node has a link to the next highest node (the *successor*) after $x + 2^j \pmod{2^k}$. Nodes will be usually assigned identifiers at random, and since the number of nodes will be much less than 2^k, most of the links with small j will point to the same node (the successor of x). To simplify our analysis, we will set $N = 2^{k'}$ for some smaller k',

and assign each node a unique identifier from $\mathbb{Z}_{2^{k'}}$. In this case, we can dispense with successors and connect each node x to nodes $x + 2^j \pmod{2^{k'}}$ for $j = 0, \ldots, k' - 1$. (This model doesn't capture the exact topology of Chord; we will return to this observation in Section 6.8.)

We have not been able to achieve a tight analytic bound on mixing in Chord. However, we can show experimentally that it is slower than in de Bruijn networks; refer to Figure 6.2. There we plot the variation distance between random walks started at two randomly picked states; we can see that although it tends towards 0 exponentially, it does not equal to 0 after any number of steps. To gain some intuition behind why this is, consider a Chord network with 4 nodes, i.e. $k' = 2$. A random walk starting at 0 will arrive at nodes 1 and 2 with probability 1/2 each. However, a 2-step random walk will result in the following distribution:

$$Pr[x_2 = 0] = \frac{1}{4}, Pr[x_2 = 1] = 0, Pr[x_2 = 2] = \frac{1}{4}, Pr[x_2 = 3] = \frac{1}{2}$$

The reason for this non-uniform distribution is that, at each step, 2^{j_t} is added to the identifier. A reordering of the sequence of j_t's produces the same end result; therefore, values resulting from sequence of different j_t's (3 in this case) will be more frequently reached than values from a sequence of identical j_t's (0 and 2).

6.6.3 PRR-trees

Another common topology used by structured p2p networks is PRR trees [59], used by Pastry [65] and Tapestry [77]. Identifiers in PRR trees are best thought of as elements of $\mathbb{Z}_d^k$ for some d and k. (A common choice is $d = 16$ and $k = 32$.) Nodes are assigned identifiers at random. Given a node with identifier $(x_1, \ldots, x_k)$, for each $j = 1, \ldots, k$ and $d' = 0, \ldots, d-1$, with $d' \neq x_j$, there is a link from $(x_1, \ldots, x_k)$ to some $(y_1, \ldots, y_k)$ such that:

$$y_i = x_i \qquad \text{for } i < j$$

$$y_i = d' \qquad \text{for } i = j$$

For $i > j$, the choice of y_i is arbitrary. This freedom of choice is exploited in Pastry and Tapestry to form connections that optimize some sort of metric, such as latency.

If no $(y_1, \ldots, y_k)$ satisfying the above conditions exists, no link is made for that choice of j and d'. In a network of N nodes, each node will have approximately $(d-1)\log_d N$ links. We will make two simplifying assumptions for our analysis. First, as with Chord, we will pick N and k' such that $N = d^{k'}$, with node identifiers distributed uniformly among elements of $\mathbb{Z}_d^{k'}$. In this case, a node will have exactly $(d-1)k'$ links out of each node. Second, when making a link for a given j and d', we will enforce that $y_i = x_i$ for $i \neq j$.

In this case, we will construct a coupling $(X_t^{(x)}, X_t^{(y)})$ for arbitrary nodes x and y. Call $[x/(j, d')]$ to be the node linked from x corresponding to j and d'. Then

$$Pr[(x_t, y_t) \to ([x_t/(j, d')], [y_t/(j, d')])] = \frac{1}{(d-1)k'}$$

$$\text{for } j = 1, \ldots, k',\ d' = 0, \ldots, d-1,\ d' \neq (x_t)_j,\ d' \neq (y_t)_j \tag{6.1}$$

$$Pr[(x_t, y_t) \to ([x_t/(j, d')], [y_t/(j, d'')])] = \frac{1}{(d-1)k'}$$

$$\text{for } j = 1, \ldots, k',\ d' = (y_t)_j,\ d'' = (x_t)_j \tag{6.2}$$

$$Pr[(x_t, y_t) \to (x_{t+1}, y_{t+1})] = 0 \quad \text{otherwise} \tag{6.3}$$

In this case, after a transition of type (6.1), $(x_t)_j = (y_t)_j$, and this will be preserved among future transitions. However, to ensure that $x_t = y_t$, we need to make such a transition for each j where x_0 and y_0 are different. Since each j is equally likely at each transition step, this can be seen as an instance of the coupon collector's problem. Classical analysis shows $t \sim k' \log k' + \theta k'$, $Pr[cover] \sim e^{-e^{-\theta}}$. [73] ($Pr[cover]$ is the probability that all j's have been covered at least once.) For example, for $Pr[cover] = 0.5$, we need $\theta \approx 0.363$. That means that the probability of coupling after $(k' \log k' + 0.363k')(d-2)/(d-1)$ steps is only 0.5. (We have to add an extra factor of $(d-2)/(d-1)$ because $1/(d-1)$ of the transitions will be of type (6.2).)

Figure 6.1 compares the coupling-derived bound with experimentally derived variation distances for a randomly chosen pair of initial states. The graph shows that the bound is not tight, but it also demonstrates a lower bound that shows that mixing in PRR trees is slower than in de Bruijn graphs. The intuition behind why this is so is apparent in the coupling proof and the relationship with the coupon collector's problem: for $t < k' \log k'$, it is likely that for several values of j, $(x_t)_j = (x_0)_j$, since

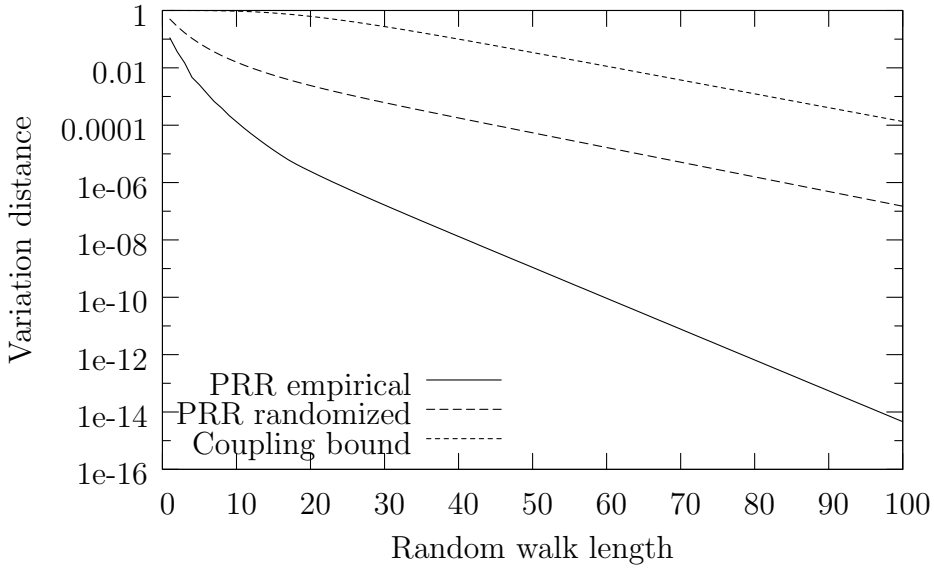

Figure 6.1: Random walks on PRR trees.

this relationship will remain true until a step of the form $x_t \to [x_t/(j, d')]$ is taken.

Figure 6.1 also shows the variation distance computed for a randomized PRR network, where we remove the requirement that $x_i = y_i$ for $i > j$ in a link $y = [x/(j, d')]$. In this case, the variation distance converges less quickly, showing that the randomization leads to less effective mixing. We expect that selecting these links based on a metric such as latency, rather than at random, would result in slower mixing still, since the random walk would tend to stay within a low-latency neighborhood of its origin.

6.6.4 CAN

The topology of the CAN network [60] is based on a k-dimensional torus. Each node occupies a section of the torus, and is connected to the sections adjacent to it. We can model a CAN topology with evenly distributed sections similar to a PRR tree, with node identifiers being points in $\mathbb{Z}_d^k$. A node $(x_1, \ldots, x_k)$ is connected to $(x_1, \ldots, x_i \pm 1, \ldots, x_k)$ for $i = 1, \ldots, k$.

We will not include a coupling analysis for CAN, but we will note that the same coupon collector problem behavior we saw in PRR trees applies here as well: to fully randomize the destination, a transition in each dimension is necessary. (In fact, CAN with $d = 2$ or 3 behaves identically to the idealized PRR scenario described in the previous section.) Figure 6.2 shows the experimentally derived mixing time for CAN.

6.6.5 Viceroy

The Viceroy network [47] is loosely based on the butterfly topology. Nodes in Viceroy have an identifier in a large space, $\mathbb{Z}_{2^k}$ for $k = 128$ or $k = 160$, as well as a *level*. A node with identifier x and level l has 5 links:

- *nextonlevel* points to node with the next largest identifier on level l (similar to the successor in Chord, but restricted by level).
- *prevonlevel* points to node with the next smallest identifier on level l.
- *left* points to the node with the next largest identifier on level $l + 1$.

- *right* points to the node with next largest identifier on level $l + 1$ following $x + 2^{k-l}$

- *up* points to the node with the next largest identifier on level $l - 1$.

Nodes get assigned to levels at random so that they are approximately evenly distributed among $O(\log N)$ levels. By convention, levels are ordered vertically, starting with level 1 at the top, and the largest level at the bottom.

The design is similar to the butterfly topology, where nodes are also assigned to levels, and connections between levels are made such that increasing levels cover exponentially decreasing distance. In some formulation, the butterfly network has good mixing properties: a random walk starting at level 1 and following only downward links to increasing levels will arrive at a uniformly distributed node at the lowest level. (In fact, there is a mapping of the butterfly graph to the de Bruijn graph under which this type of walk is equivalent to a random walk on the de Bruijn graph [5].) However, it is difficult to incorporate this type of random walk into our anonymous routing design, as it would need to include a phase of traveling up to a level 1 node first, and during this phase the origin could be easily identified.

An unbiased random walk on the Viceroy network, taking each of the five links with equal probability, mixes rather poorly, because it is unlikely to reach nodes at the upper levels: there is only one link for traveling up, and two for traveling down. Even if we correct this imbalance by taking up links with twice the probability of down links, there is still a problem in that only nodes at the upper levels have links

that move an identifier far away from the starting point. For example, a random walk starting at the lowest level will tend to stay in a small neighborhood of the starting point unless it reaches an upper level, which will happen with only a small probability.

We could further increase the probability of the up links to make them more likely than both the downlinks, but that will result in the opposite problem that lower level nodes are unlikely to be reached. In general, because the Viceroy network is designed for multiple phases of routing (first traveling up to level 1 and then down to find the destination), it is unlikely that its structure will support efficient random walks.

6.7 Empirical Results

Figure 6.2 shows the variation distance between random walks starting at two random states, for idealized versions of the various topologies. The topologies all have size 512; for most networks, the nodes have degree 9. We do not show de Bruijn networks on this graph, since after the first 3 steps of the random walk, the variation distance for de Bruijn is 0.

Since we compute the graphs for random walk starting at two random states, this provides a lower bound on the variation distance in question:

$$\max_{i,j} ||P_i(X_t) - P_j(X_t)||$$

We have found that different choices of starting states result in slightly different variation distances; however, the slope of the (log-scale) graph remains constant. In

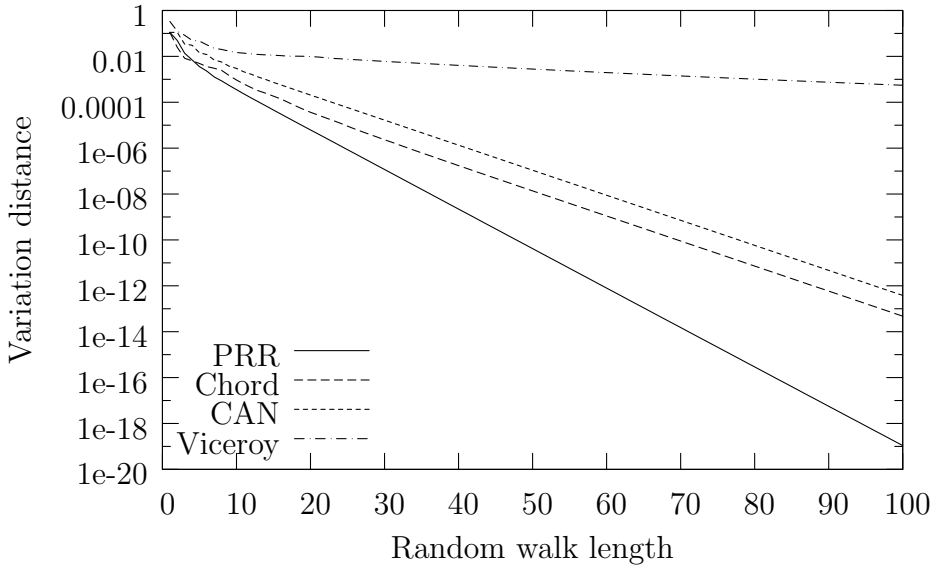

Figure 6.2: Random walks on ideal networks.

particular, we can relate the parameters of the networks with the convergence rates of the variation distance, though only by experimental means.

For a Chord network with 2^k nodes, the variation distances converges according to $V(t) \sim \left(\frac{k-2}{k}\right)^t$. For PRR trees with base d and identifier length k (with d^k nodes), the variation distance converges acccording to $V(t) \sim \left(\frac{(d-1)(b-1)-1}{d(b-1)}\right)^t$. CAN, with d dimensions and d^k nodes has a convergence rate of $V(t) \sim \left(\frac{2d+b-6}{2d}\right)^t$. Note that for all of these networks, the size of the network affects the rate of mixing. In other words, as we scale up the size of the network, the advantage of de Bruijn mixing over the other topologies will become even more significant.

6.8 Non-idealized topologies

Our analysis of the various peer-to-peer topologies made several simplifying assumptions. In particular, all graphs we studied were regular with a number of nodes expressed as d^k for some d and k. In practice, peer-to-peer systems have to construct the network topology dynamically and in a decentralized fashion, which results in some irregularities in the graphs, both due to the number of nodes not being a perfect power of d, and because the (usually) random assignment of nodes to locations in the topology results in a certain amount of imbalance.

For example, consider Chord. With N nodes, the average distance from a node to its successor will be $2^k/N$. The number of distinct outgoing links from any Chord node will then be $\log_2 N$, since links corresponding to $x + 2^{k-j}$ for $j \geq \log_2 N$ will all point to the successor. However, random assignment of identifiers will result in some nodes being further away from their successors than others, with a factor of $O(\log N)$ difference between the smallest and the largest. Therefore, the degree of nodes will similarly be different by $O(\log \log N)$.

To study how this affects the mixing times of the various peer-to-peer systems, we generated topologies through a random network construction, and then simulated random walks along these topologies. Figure 6.3 plots variation distance between a random walk started at two random points for Koorde [41] (based on the de Bruijn graph), Pastry [65] (based on PRR trees), Chord, and CAN, with $d = 2$ and $N = 512$, except for Koorde, where $d = 8$.

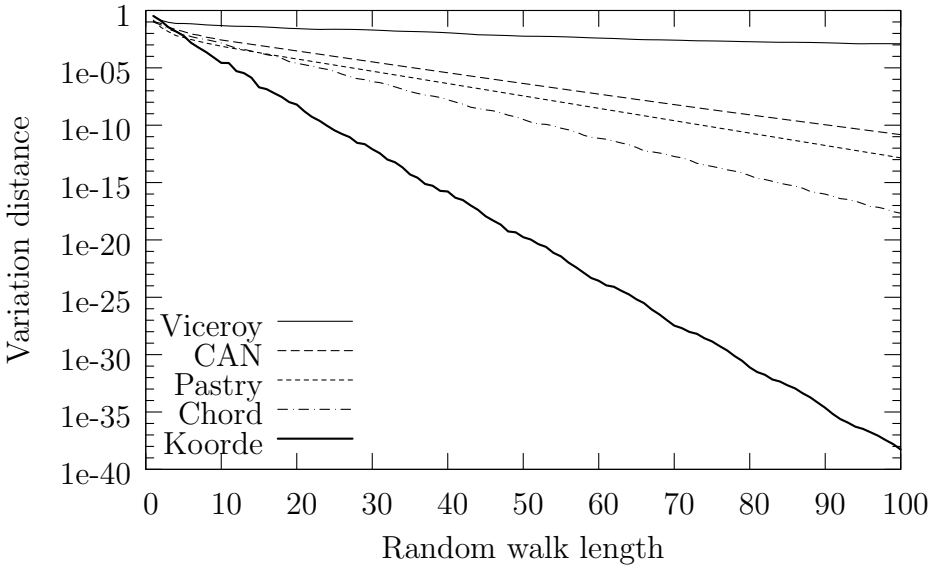

Figure 6.3: Random walks on non-ideal networks.

We can see that the irregularities in the network prevent Koorde from achieving perfect mixing. However, the convergence is still much faster than in the other network systems, motivating our use of Koorde for the anonymous routing design. We were also interested in how the mixing properties of the networks change with growing size; Figures 6.4 and 6.5 shows the variation distance plots as the size of the network is increased. Consistent with our observations in the previous section, the mixing rates of networks other than Koorde depend on the size of the network; whereas Koorde mixing remains the same regardless of the network size. Therefore, we expect that the benefits of using de Bruijn over other topologies will be even more relevant as the sizes of the networks scale to hundreds of thousands of nodes, which are beyond our capability to analyze mixing times through simulation.

Note that the de Bruijn graph only shows random walks up to length 45, because past this point the floating point precision of our simulations was insufficient to compute the variation distance. Also, we did not include scaling simulations for Viceroy, since it should be clear from the previous graphs that the Viceroy mixing times will be much poorer than the other topologies.

6.9 Summary

In this chapter, we have described a design for an anonymous peer-to-peer system based on structured overlays and random walks. We showed, through an analysis of mixing times, that the de Bruijn-based topology used in Koorde has greatly superior mixing properties as compared with other structured overlay topologies. In the next chapter, we will analyze the anonymity provided by such a design.

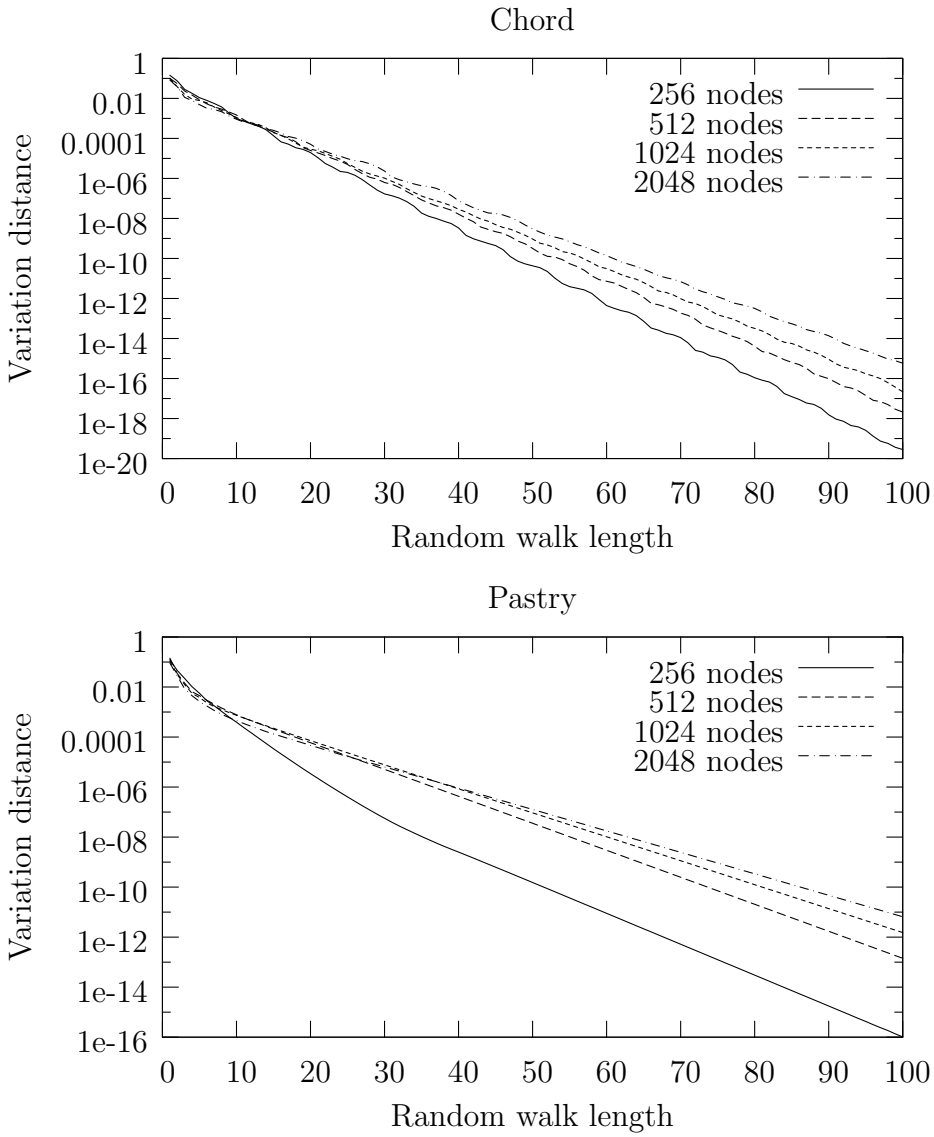

Figure 6.4: Random walks with increasing network sizes, Chord and Pastry.

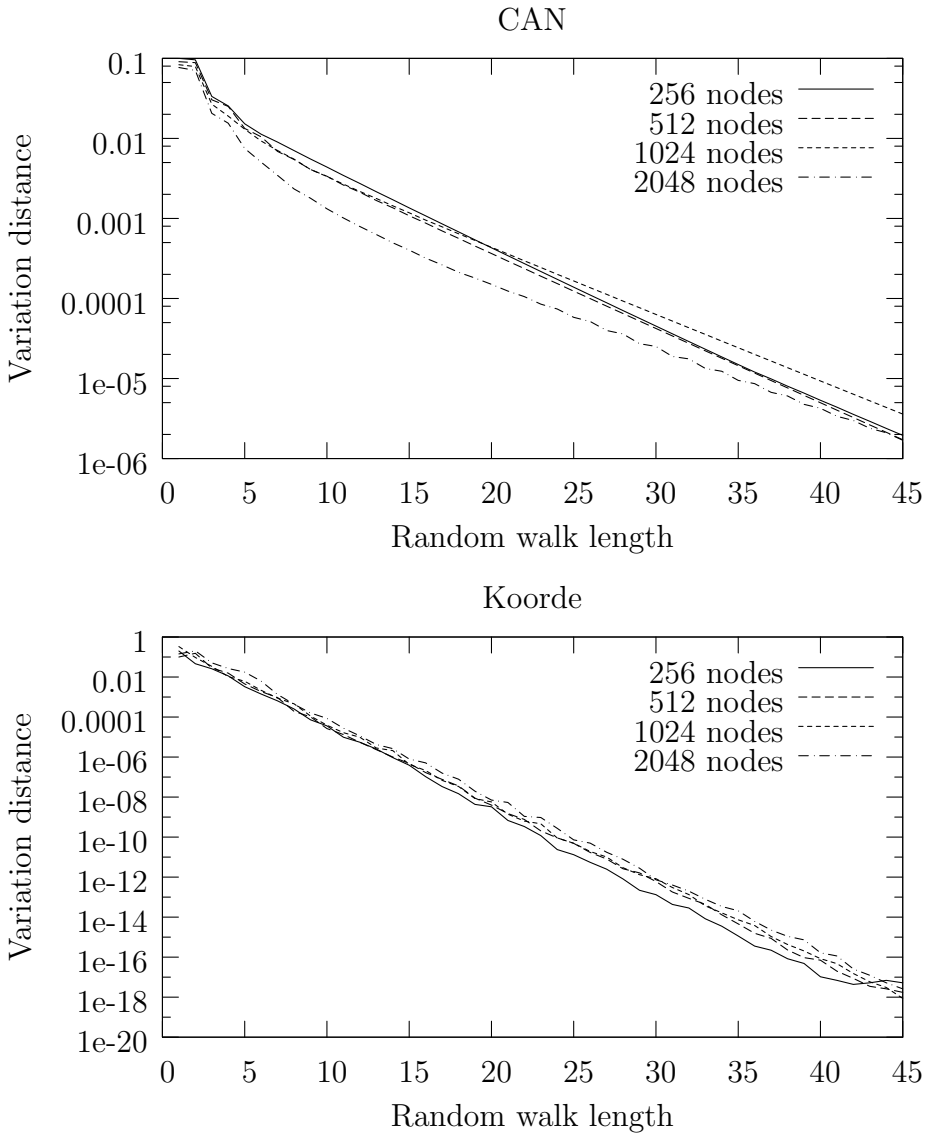

Figure 6.5: Random walks with increasing network sizes, CAN and Koorde.

Chapter 7

Analysis of Structured Anonymous Networks

In this chapter, we will analyze the anonymity achieved by the two-phase, random walk-based design proposed above. We have some analytical results from a simplified network model, but most of the analysis will be done with help of simulations. In addition, we compare our design to an alternative anonymity system based on structured p2p overlays [49].

7.1 Mixing Time versus Anonymity

The previous chapter analyzed the mixing time of random walks and showed that after a small number of steps the variation distance between random walks started at two different points can be made very small. For example, in an ideal de Bruijn

graph, if the attacker observes the terminus of a k step random walk (where k is the diameter of the graph), he would learn no information about the origin, since each source node would have an equal chance of reaching each other node after k random walk steps. Even with a dynamically generated topology, a small number of steps is sufficient to achieve a very small variation distance.

However, an attacker may observe the random walk in the middle, rather than at the end. The use of expected-length random walks prevents the attacker from knowing how far the walk has progressed from its origin point, however, he can still assign higher probabilities to nodes closer in network distance.

For example, suppose the attacker observes a message passing through some node x. What are the chances, for each possible source node, that an expected length L random walk would have passed through this node? If x is the origin, then the probability is clearly 1. For a node whose neighbor set includes x, the random walk has probability $1/d$ of picking x in the first step (where d is the degree). Similarly, for a node t links away from x, the chances of visiting x on the t-th step is $1/d^t$. Finally, for $t \geq k$, the odds of visiting x on the t-th step of a random walk are $1/N$ (where $N = d^k$) for any origin node. And the odds that a random walk will have at least t steps are $\left(\frac{L-1}{L}\right)^{t-1}$.

We can therefore compute the chances of *not* visiting x at any step of the walk. For a node that is t steps from x, it is:

$$p_t = \sum_{i=1}^{t-1} \left(\frac{L-1}{L}\right)^{i-1} \frac{1}{L} + \sum_{i=t}^{k-1} \left(\frac{L-1}{L}\right)^{i-1} \frac{1}{L}\left(1-\frac{1}{d^t}\right) + \sum_{i=k}^{\infty} \left(\frac{L-1}{L}\right)^{i-1} \frac{1}{L}\left(1-\frac{1}{d^t}\right)\left(1-\frac{1}{N}\right)^{i-k+1}$$

$$= \frac{\sum_{i=1}^{t-1}\left(\frac{L-1}{L}\right)^{i-1} + \left(1-\frac{1}{d^t}\right)\left(\sum_{i=t}^{k-1}\left(\frac{L-1}{L}\right)^{i-1} + \left(\frac{L-1}{L}\right)^{k-2}\sum_{i=1}^{\infty}\left(\frac{L-1}{L}\frac{N-1}{N}\right)^{i}\right)}{L}$$

$$= \frac{\sum_{i=1}^{t-1}\left(\frac{L-1}{L}\right)^{i-1} + \left(1-\frac{1}{d^t}\right)\left(\sum_{i=t}^{k-1}\left(\frac{L-1}{L}\right)^{i-1} + \left(\frac{L-1}{L}\right)^{k-2}\frac{1}{\frac{LN}{(L-1)(N-1)}-1}\right)}{L}$$

The probability of that a random walk *did* pass through x at some step, on the other hand, is $Pr[x|y_t] = 1 - p_t$. We can read this conditional as "probability that a random walk visits x, having started at node y_t at distance t from x". What we are interested in, however, is $Pr[y_t|x]$, i.e. the probability that a particular node was the origin given that a random walk visited x.

$$Pr[y_t|x] = \frac{Pr[y_t \wedge x]}{Pr[x]}$$

We can rewrite:

$$Pr[x] = \sum_y Pr[y \wedge x]$$

For $t < k$, there are d^t nodes at that distance from x. Therefore,

$$Pr[x] = \left(\sum_{t=0}^{k-1} d^t Pr[y_t \wedge x]\right) + \left(d^k - \sum_{t=0}^{k-1} d^t\right) Pr[y_k \wedge x]$$

But $Pr[y_t \wedge x] = Pr[x|y_t]Pr[y_t] = Pr[x|y_t]/N$, since we are assuming a uniform prior distribution over all sources and therefore $Pr[y_t] = 1/N$ for each y_t. We then have:

$$Pr[x] = \frac{\left(\sum_{t=0}^{k-1} d^t(1-p_t)\right) + \left(d^k - \sum_{t=0}^{k-1} d^t\right)(1-p_k)}{N}$$

Now we can calculate:

$$Pr[y_t|x] = \frac{Pr[y_t \wedge x]}{Pr[x]} = \frac{Pr[x|y_t]Pr[y_t]}{Pr[x]} = \frac{1-p_t}{NPr[x]}$$

And then we can compute the entropy associated with the observation of a message passing through a node x as follows:

$$H_x = \left(\sum_{t=0}^{k-1} d^t Pr[y_t|x] \log Pr[y_t|x]\right) + \left(d^k - \sum_{t_0}^{k-1} d^t\right) Pr[y_k|x] \log Pr[y_k|x]$$

Figure 7.1 plots the entropy computed this way for different choices of L in two networks with $d = 8$ and $k = 4$ and 6 respectively. (The networks therefore have 4096 and 262144 nodes respectively.) If we set L to be only as large as k, the entropy falls a bit short of optimal: 9 vs. 12 bits for the 4096 node network and 13 vs. 18 bits for the 262144 node one. Increasing L to $2k$, the entropy is significantly better: 10.5 bits and 15.5 bits respectively. Increasing L further achieves only marginal improvements.

The analysis does not take into account colluding attacker nodes; in practice, we may assume that attackers can recognize messages that another attacker has seen before. Therefore, we must eliminate all paths that visit an attacker before arriving at node x. We can adjust our model in the following way: if c out of N nodes are compromised, the probability that a random walk will chance upon an attacker at any given step is c/N. We can model the attacker shared knowledge as the termination of the random walk with probability c/N at each node. This probability is in addition to a $1/L$ probability of the walk being terminated when it arrives at a non-compromised node. Therefore, we can set:

$$L' = \frac{1}{\frac{N-c}{N}\frac{1}{L} + \frac{c}{N}}$$

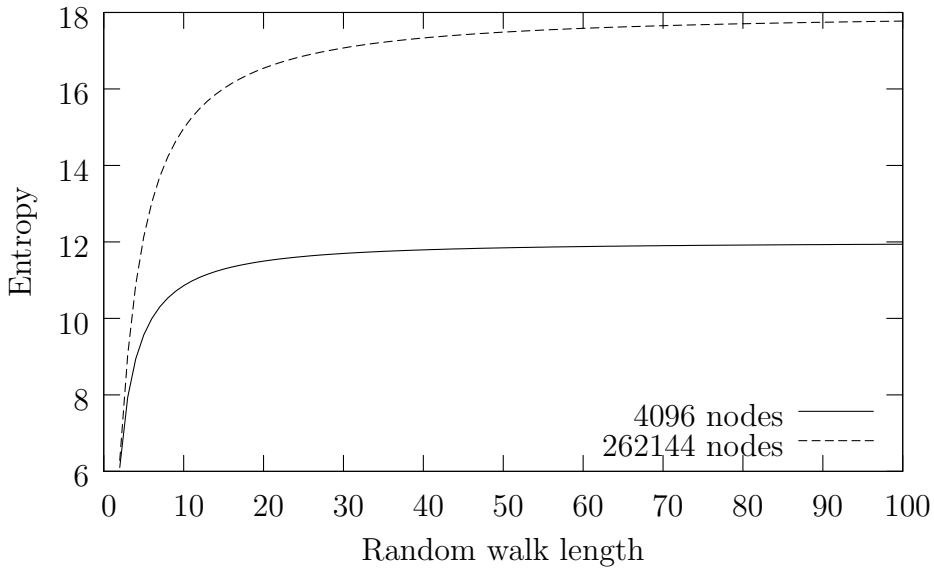

Figure 7.1: Entropy computed using analytical methods.

And use the above model with a random walk of expected length L'. For example, with 10% of all nodes being attackers and a real $L = 8$, we need to set $L' = 4.7$. Therefore, it does not make sense to set the expected length of the random walk to be too large, as the marginal gains of increasing the random walk will be small under such a heavy attack. This computation also demonstrates the limits of our approach: with many more than 10% of the nodes compromised, the random walk will likely arrive at a compromised node too soon to effectively hide the origin of the query.

There are two more details missing from the analysis. First, it ignores any observations made during phase 2 of the routing. This is not a significant shortcoming, since an observation of phase 2 will point to some probability distribution of phase 1 terminus nodes, each of which can then be translated to a phase 1 observation with

certain entropy. Second, the graph we use has an ideal de Bruijn structure, rather than a dynamically generated topology. We will address the second point with help of simulation.

7.2 Koorde Simulations

To analyze the anonymity behavior of Koorde, modified to include the random walk phase, we perform routing simulations, similar to the Freenet analysis. In this case, however, we can perform the simulations more efficiently if we model the network as a Markov process.

Starting with a randomly generated Koorde topology, we create two states in the Markov process for each node. For a node x, the two states — $(x, 1)$ and $(x, 2)$ — represent a query visiting that node during phase 1 and 2 respectively. We assign some fraction f of nodes to be compromised; for the rest, we assign transition probabilities to the states corresponding to each uncompromised node to model the routing behavior. For a state $(x, 1)$, we create a transition to $(x', 1)$ with probability $(L - 1)/(dL)$ for each of the d nodes x' that x links to. We also create a transition to $(x, 2)$ with probability $1/L$. For any (uncompromised) node $(x, 2)$ we create a probability 1 transition to the next node picked by the Koorde routing algorithm for a particular destination. We pick an arbitrary, fixed destination for creating the Markov process. The destination node x_d, in turn, will be a sink, i.e. it will have a probability 1 self transition in the state $(x_d, 2)$.

For the compromised nodes, we create sink states $(x, 3)$ and $(x, 4)$ for each node x that has a link to a compromised node y. We then change all transitions $(x, 1) \to (y, 1)$ to $(x, 1) \to (x, 3)$; similarly, $(x, 2) \to (y, 2)$ gets remapped to $(x, 2) \to (x, 4)$. The states are sinks so that we can model attacker collusion by not having the queries forwarded further once they are observed by the attacker. We label the sinks with the node identifiers of the predecessors, rather than the compromised nodes, since that way the attacker observation is clear from the sink state.

Given this process (X_t), we can compute the probability distribution $P_i(X_t)$ of the chain started in state i after t steps, by starting with a probability distribution vector $(0, \ldots, 0, 1, 0, \ldots)$ with a 1 in the i-th position, and repeatedly multiplying it by the transition matrix. We will be most interested in $P_{(x,1)}(X_t = (x', 3))$ and $P_{(x,1)}(X_t = (x', 4))$, since this represents the probability of a query starting at node x causing the observation corresponding to $(x', 3)$ or $(x', 4)$. Since we are concerned with the probability of the query causing the observation after any number of steps, we would like to consider $t = \infty$. However, the probability that the Markov process simulating the routing will not arrive at some sink state diminishes exponentially, and after a large enough number of steps this probability is small enough to be insignificant. Therefore, we can compute the conditional probability of an observation such as $(x', 3)$ given that the query started in state x, i.e. $Pr[Obs = (x', 3)|Orig = x]$. Since we are assuming a uniform prior distribution, i.e. $Pr[Orig = x] = 1/(1 - f)N$

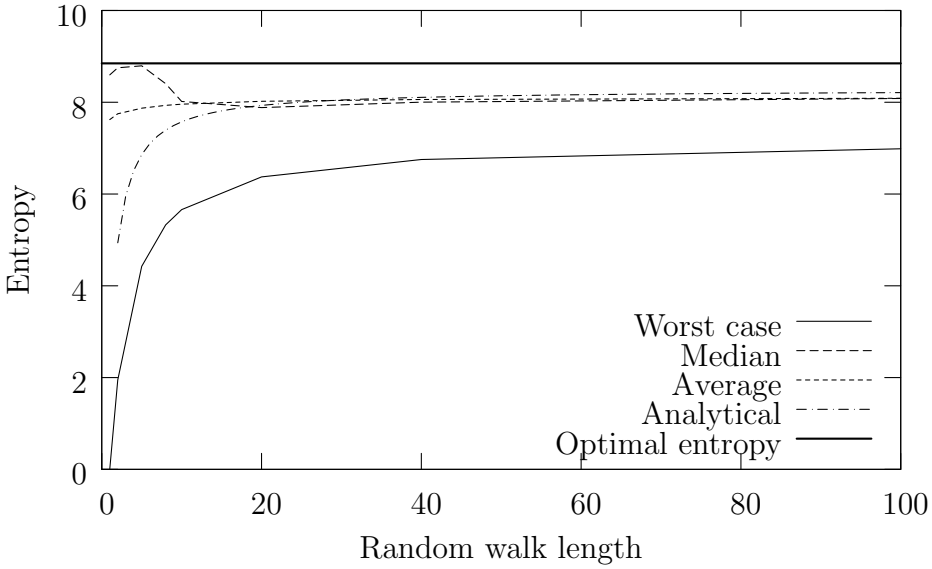

Figure 7.2: Entropy in a 512-node dynamically constructed network.

where N is the total number of nodes in the system, we have that:

$$Pr[Obs = (x', 3) \wedge Orig = x] = \frac{Pr[Obs = (x', 3)|Orig = x]}{(1 - f)N}$$

And:

$$Pr[Orig = x|Obs = (x', 3)] = \frac{Pr[Obs = (x', 3) \wedge Orig = x]}{\sum_y Pr[Orig = y \wedge Obs = (x', 3)]}$$

Therefore, having computed $Pr[Obs = (x', 3)|Orig = x]$ for all possible origins x (by simulating the Markov process starting in state $(x, 1)$), we can then compute $Pr[Orig = x|Obs = (x', 3)]$ for each x, and therefore determine $H(Orig|Obs = (x', 3))$. And repeating this process for each observation will allow us to compute the conditional entropy $H(Orig|Obs)$.

Figure 7.2 shows the entropy computed in this way for a Koorde network of 512 nodes, with $d = 8$ and 10% attacker nodes. We computed both the average

(i.e. conditional) entropy, as well as the median and worst-case entropy for different lengths of random walks. Even with a very short random walk, the conditional entropy remains within one bit of optimal. The median, in fact, is higher with a shorter random walk. This is because for a network of this size and a short random walk, a large number of queries don't get observed at all, resulting in higher average entropy. If over half the queries don't get observed, the median will equal the optimal entropy.

However, looking at the worst-case entropy, short random walks exhibit a similar behavior to Freenet: when queries *do* get observed, the entropy is low. Increasing the expected random walk length has a very noticeable effect of improving the worst-case anonymity; at 10 steps, the worst-case entropy reaches 5.6 bits. This is well below the optimal value we would like to achieve, but with such a large percentage of attackers, it may be an acceptable figure. Note that this entropy is achieved only in the worst case; for over half the queries, the entropy is within 1 bit of optimal, as shown by the median. A more privacy-sensitive application might set the expected random walk length to be higher still, however, there is a limit to how high the worst-case entropy will be — even with 100 steps, it is less than 7.

We also plot the analytical results for comparison. The worst-case entropy is somewhat lower than what is computed analytically for two reasons. First, the analytical results assume a regular de Bruijn topology, whereas the Koorde topology we use has imbalances due to the random dynamic construction. Second, the attacker model we used assume a random distribution of compromised nodes relative to the node being

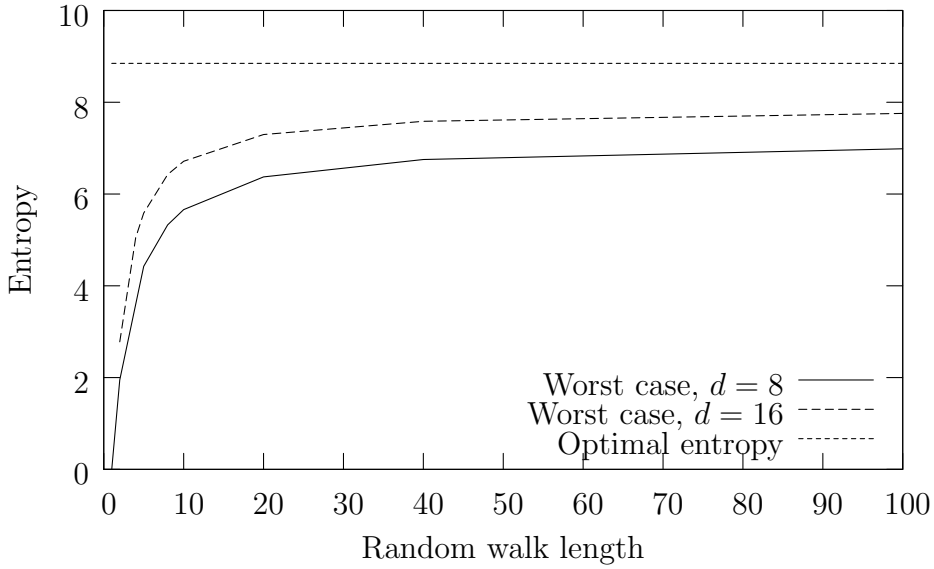

Figure 7.3: Comparison of worst-case entropies with $d = 8$ and $d = 16$.

observed. However, in the worst-case scenario, a large number of the nodes linking to x will be compromised. For example, if each out of 512 nodes picked 8 incoming neighbors at random, with 10% of the nodes being compromised, the worst-case node would be expected to have 5 out of its 8 neighbors be compromised nodes. In this case, the number of random walk paths going through this worst-case node that do not traverse any compromised node first is greatly reduced. This suggests that increasing the indegree of nodes will improve the worst-case anonymity; Figure 7.3 demonstrates that this is indeed the case.

A final question we wanted to answer is what percentage of the queries receive the worst-case entropy? To answer this question, we plotted the cumulative distribution function of the entropy of different queries in Figure 7.4. The figure used a network of

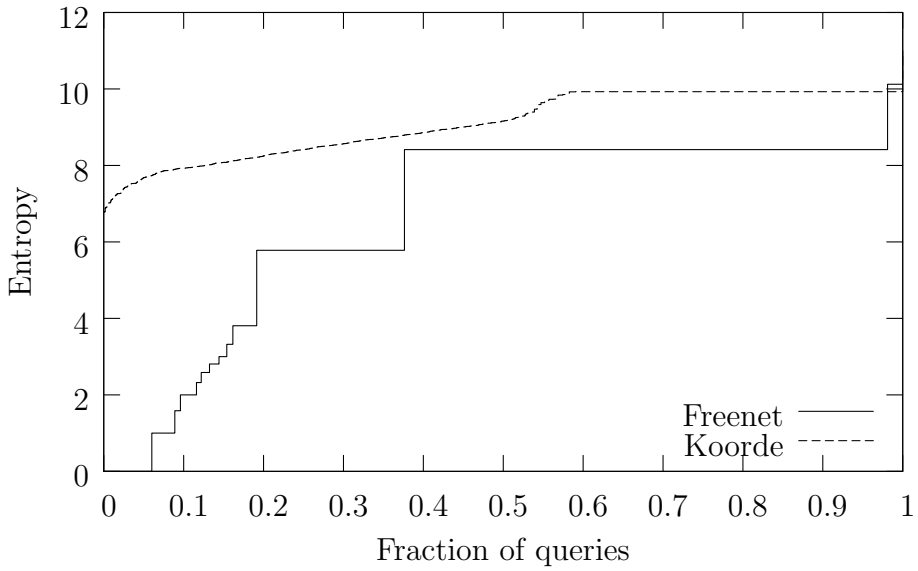

Figure 7.4: Comparison of Freenet and Koorde CDFs with 2048 nodes. (For Koorde, $d = 8$ and $L = 10$.)

2048 nodes, with 10% attackers, and an expected random walk length of 10. Although the worst-case entropy was 6.7 bits, 90% of all queries received entropy of 8 bits or above. We also compare this graph to a Freenet network of the same size. In Freenet, not only did the bottom 6% of queries receive no anonymity protection, 37% of all queries have lower entropy than the worst-case for the Koorde-based design.

7.3 Comparison with AP3

AP3 [49] is an anonymous communication system based on a structured peer-to-peer network. The design of AP3 is similar to ours, with a few key differences. AP3 is based on Pastry [65], rather than Koorde, although the designers suggest that other structured overlays could be used. More significantly, the random walk is performed

in a different fashion. At the outset, the source will pick a random destination and perform Pastry routing to send its message there. The node at that destination will choose, based on a weighted coin-flip, whether to deliver the message or repeat the process, choosing another random destination and routing towards it. Finally, AP3 is intended not as an anonymizing DHT but as a general anonymous communication system; therefore, phase 2 of our routing is replaced with some application-dependent message delivery mechanism.

We wanted to compare the design choices made in AP3 to our own design. We therefore implemented a simulator for a modified AP3. Our modifications were to use DHT routing as the message delivery mechanism; in other words, using the same phase 2 routing as our design. We also generated a Pastry topology using random choices for neighbors, rather than using latency-based measurements. We expect that choosing neighbors based on latency would result in worse anonymity, as Pastry routes would tend to stay within a small latency neighborhood of the starting point for most of the route.

Our observations are based on a fraction of compromised nodes. Note that AP3 does not reveal the ultimate destination to the intermediate nodes during Pastry routing, but only to the randomly selected forwarder. In our implementation, compromised intermediate nodes on the Pastry route *do* learn the destination. We have made this choice because the intermediate node can disrupt Pastry routing to ensure that the forwarder is chosen to be one of the colluding nodes. Techniques by Castro *et al.* [13] may make such attacks difficult, but they require expensive public key

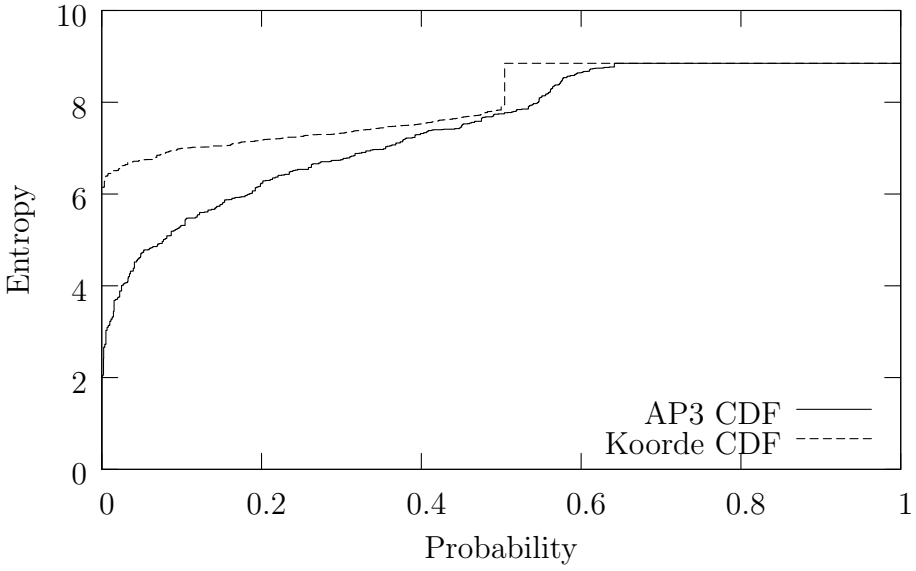

Figure 7.5: Comparison of AP3 to our Koorde-based design with 512 nodes.

operations at each routing step.

Figure 7.5 compares the anonymity achieved by the modified AP3 to our design. We used a network of 512 nodes, with 10% compromised. The Pastry network used a base of 2 and Koorde used a base of 8, resulting in average degrees of 9 and 8 respectively. For our design, we used $L = 10$. For AP3, the authors suggest that p_f, the probability of forwarding at each node should be between 0.5 and 0.9. We used 0.62 in our measurements because that results in an average path length of 14.5, comparable to the average path length of 14 for our design.

We measured anonymity by performing 5 million randomized routing samples in AP3. The anonymity levels achieved by AP3 are generally good, if somewhat worse than in our design. However, the first 20% of all queries have anonymity lower

than the worst-case of our design, and the bottom 10% provide very little anonymity protection. This is due to the fact that, for some Pastry routes, an intermediate node can easily determine the origin of the route, similar to our observations about Chord in Section 6.4. And since the average number of forwarders is only 1.6 with $p_f = 0.62$, the origin of the route is the original source with high probability.

7.4 Summary

We have analyzed the anonymity provided by our proposed design for anonymous routing in peer-to-peer systems. We have found that, on average, the anonymity achieved is close to optimal, even with a large number of attackers. Unlike Freenet, the worst-case anonymity provided to any query can be bounded at an acceptable level when the expected length of the random walk is about twice k, the diameter of the graph. Somewhat higher worst-case anonymity can be achieved with even longer random walks, but the gains beyond $2k$ are limited. Another way to improve the worst-case anonymity is to increase the degree of the nodes in the network.

Chapter 8

Conclusions

8.1 Summary of Contributions

We have developed a novel methodology for studying anonymous peer-to-peer networks. Our methodology is based on the information-theoretic anonymity metric [25, 67], but it uses simulation rather than direct analysis to calculate probability distributions. As such, we are able to apply it to complex systems with many nodes that had previously resisted analysis. The use of simulations allows us to incorporate the full complexities of the system design, rather than accept simplifications required by analytical models. Despite using techniques such as sampling, we are able to compute rigorous bounds on the errors resulting from sampling and derive accurate estimates for the information-theoretic entropy metric. We validated our methodology by comparing it to a direct entropy analysis of Crowds [62].

Using this methodology, we performed the first rigorous analysis of Freenet [18].

We have found that Freenet is subject to a targeted attack on high-degree nodes that is very effective at compromising anonymity. We also analyzed the proposed next-generation routing algorithm [16] and discovered that it dramatically lowers anonymity. In the normal case, we found that although Freenet provides acceptable average anonymity levels, there is a lot of variability and up to a quarter of the users will receive poor anonymity protection. Through use of time evolution and random changes to the network, we showed that these findings are true even with a more limited assumption about attacker knowledge of the network topology.

Motivated by these problems, we designed an alternative peer-to-peer anonymous system, using random walks on structured networks. We have analyzed the mixing times of random walks on different structured network topologies and found that networks based on de Bruijn graphs [23] mix much faster than other designs. We therefore designed based our system on Koorde [41], which uses a de Bruijn-like topology. We performed both analytical and empirical measurements of anonymity in our Koorde-based design and found that it provides acceptable anonymity, even in the worst-case, while maintaining logarithmic performance. Our analysis shows that structured networks are a viable platform for anonymous systems and that they are able to guarantee an acceptable minimum anonymity level, even under strong attacker knowledge assumptions.

8.2 Future Work

8.2.1 Structured P2P Advances

Recently, there have been new developments in how structured peer-to-peer networks are designed. One topic that has received a lot of attention has been reducing the latency of requests. The designers of Pastry [65] and Tapestry [77] incorporated distance measurements into the choices left available by the PRR algorithm. Similar kinds of choices can be made during neighbor selection in Chord [70]; in addition, Chord and CAN [60] can adjust their routing algorithms to choose paths with better latency [38].

An interesting question is whether it is possible to use these techniques to improve the latency of our anonymous p2p design. Biasing path selection for lower latency would clearly reduce anonymity, similar to what happened with the Freenet next-generation routing algorithm (Section 5.6). However, neighbor selection for lower latency could reduce the average delay in satisfying a query. The de Bruijn graph construction does not offer choices in selecting neighbor; however, the design could be extended to a multi-de Bruijn where such choice is possible. We expect that in such a construction, the number of random hops would need to be increased to achieve the same anonymity. However, the total delay may still be less because each hop would be much shorter in terms of network distance.

Another topic of study has been load balancing. A random distribution of nodes in an identifier space results in regions that differ by a factor of $O(\log N)$ in size.

Algorithms that assign a region to a node after considering 2 choices can reduce this factor to $O(\log \log N)$ [12] and others that sample $O(\log N)$ choices can lower it further to $O(1)$ [1, 3, 42, 48, 53]. As we saw in Chapter 6, mixing times on perfect graph structures are much faster than in approximate graphs. Load balancing can help reduce the gap from the ideal mixing times and therefore produce systems that are more efficient, and potentially offer better anonymity protection.

8.2.2 Applications

We have designed a generic anonymous DHT mechanism, allowing us to render anonymous applications built on top of the DHT primitive. Some applications where anonymity might be useful include instant messaging [43], email [50], and content distribution [34]. One challenge in making anonymous versions of these applications is to make sure the application-level protocols do not reveal identifying information. Note that for applications like instant messaging and email, our anonymous design may be useful even if end-to-end anonymity is not desired. That is, even if the people corresponding know each other's identities, they may still want to hide the fact they are communicating from others.

For applications like the Coral content distribution network [34], our design would need to be adapted to support their "sloppy" extension to the DHT primitive. A more significant extension would be to use the random walk design for a generic TCP or IP forwarding infrastructure. The nodes along the random walk in phase 1 would forward traffic in both directions, while phase 2 would be replaced with a direct TCP

connection to the destination. Connections with a large number of hops would be unreliable, but perhaps resilience techniques similar to those in Cashmere [78] could be used for longer-lived connections.

8.2.3 Multiple Message Attacks

Our analysis focuses on how much information can be learned from a single message. However, in the context of applications such as instant messaging or TCP forwarding, long-term communications is to be expected, which may enable more powerful attacks, such as an extension of the predecessor attack [75]. The entropy metric can give us some insight as to how these attacks might behave. If we model the anonymity system as a noisy channel, we can use mutual information between the origin and the observations to compute the expected data rate, giving us a bound on how many messages may be sent before the origin is revealed. Mutual information can be computed from conditional entropy:

$$I(X;Y) = H(X) - H(X|Y)$$

If observations caused by each message were independent, we would expect that the origin could be identified after $I(X;Y)/H(X)$ observations.[1] However, if the observations are not independent, the information-theoretic bound will be overly conservative and we will need other tools to study the resilience of the system to multiple message attacks.

[1]Note that this quantity can be computed directly from the scaled version of the entropy metric suggested by Diaz *et al.* [25].

Another issue with long-term communication is the possibility of traffic analysis. Cover traffic may be more practical on a restricted peer-to-peer topology because of the low degree of each node, but the feasibility of cover traffic and its effectiveness at stopping traffic analysis is still an open research question.

8.2.4 Metric Extensions

The entropy metric was able to extract a single quantitative measure out of a probability distribution and it served us well in anonymity analysis. However, the systems we studied have many different events with different entropies. As we saw, averaging all those entropies to produce conditional entropy was insufficient to describe all the relevant characteristics of an anonymity system. We have relied on comparing the detailed entropy distributions with help of cumulative distribution function plots, but it would have been useful to have a metric that could quantify the anonymity behavior beyond the average case. Note that although it was useful for our purposes, worst-case entropy is not always the best choice for such a metric because the worst-case entropy may occur with an insignificantly small probability. In such a case, something like the first percentile entropy may be more descriptive.

8.2.5 Simulation Toolkit

Our simulation approach is quite general and can be applied to a number of systems. A generic toolkit for performing empirical entropy analysis would significantly

reduce the effort required to apply our methodology to new or existing systems. Such a toolkit might take a compact description of a network topology and routing algorithm choices and produce graphs similar to those found in our analysis.

One technique that has helped us in our analysis of our Koorde-based design has been non-deterministic simulation using Markov models (see Section 7.2). The language for specifying networks to be analyzed should therefore include a way to describe non-deterministic choices to enable building Markov models, perhaps similar to the language used in Prism [45].

Non-deterministic simulation may also be useful in cases where we want to instrument the real implementation of a peer-to-peer system in order to perform anonymity analysis. We can generate random messages from random nodes and tag messages with their true source, as discussed in Section 4.5; we can also add a probability field, initially 1.0, to the message. Then we can instrument the non-deterministic routing choices to generate several messages with a properly reduced probability tag. Effective batching and aggregation strategies for messages so tagged may help explore a large path space while reducing the number of actual messages sent between the peer-to-peer nodes.

Bibliography

[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi, and E. Pavlov. A generic scheme for building overlay networks in adversarial scenarios. In *Proceedings of the International Parallel and Distributed Processing Symposium*, April 2003.

[2] E. Adar and B.A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[3] M. Adler, E. Halperin, R.M. Karp, and V.V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, 2003.

[4] D. Aldous. Random walks of finite groups and rapidly mixing markov chains. In J. Azéma and M. Yor, editors, *Séminare de Probabilités XVII 1981/82*, volume 986 of *Lecture Notes in Mathematics*, pages 243–297. Springer-Verlag, 1983.

[5] F.S. Annexstein, M. Baumslag, and A.L. Rosenberg. Group action graphs and parallel architectures. *SIAM Journal of Computing 19*, 1990.

[6] A. Antos and I. Kontoyiannis. Convergence properties of functional estimates for discrete distributions. *Random Structures and Algorithms, 19*, pages 163–193, 2000.

[7] Adam Back, Ian Goldberg, and Adam Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[8] G. Basharin. On a statistical estimate for the entropy of a sequence of independent random variables. *Theory of Probability and Its Applications, 4*, pages 333–336, 1959.

[9] Krista Bennett and Christian Grothoff. GAP – practical anonymous networking. In Roger Dingledine, editor, *Proceedings of the Privacy Enhancing Technologies Workshop*. Springer-Verlag, LNCS 2760, March 2003.

[10] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[11] Nikita Borisov and Jason Waddle. Anonymity in structured peer-to-peer networks. Technical Report UCB//CSD-05-1390, UC Berkeley, May 2005.

[12] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In *Proceedings fo the 2nd Internation Workshop on Peer-to-Peer Systems*, February 2003.

[13] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, December 2002.

[14] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[15] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[16] Ian Clarke. Freenet's next generation routing protocol. http://freenet.sourceforge.net/index.php?page=ngrouting.

[17] Ian Clarke, Theodore W. Hong, Scott G. Miller, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[18] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[19] The Cleaner. Gnutella wall of shame. http://www.zeropaid.com/busted/.

[20] Wei Dai. Pipenet 1.1. Usenet post, August 1996.

[21] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop*. Springer-Verlag, LNCS 2760, March 2003.

[22] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III anonymous remailer protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2003.

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

[24] Claudia Díaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of the 9th European Symposium on Research in Computer Security*, LNCS, September 2004.

[25] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of the Privacy Enhancing Technologies Workshop*. Springer-Verlag, LNCS 2482, April 2002.

[26] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC2246, January 1999.

[27] Roger Dingledine. Anonymity bibliography. http://freehaven.net/anonbib/.

[28] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In H. Federrath, editor, *Design-*

ing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability. Springer-Verlag, LNCS 2009, July 2000.

[29] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[30] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of the Privacy Enhancing Technologies Workshop*, volume 3424 of *LNCS*, May 2004.

[31] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop*, March 2002.

[32] P. Fraigniaud and P. Gauron. An overview of the content-addressable network D2B. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*, 2003.

[33] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.

[34] M.J. Freedman, E. Freudenthal, and D. Maziéres. Democratizing content publication with Coral. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation*, 2004.

[35] Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, December 2000.

[36] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the Internet. In *Proceedings of the 42nd IEEE Spring COMPCON*. IEEE Computer Society Press, February 1997.

[37] Philip Golle and Ari Juels. Parallel mixing. In *Proceedings of the ACM Conference on Communications and Computer Security*, October 2004.

[38] Krishna P. Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[39] Ian Hall-Beyer et al. Gnutella. http://www.gnutella.com/.

[40] Johan Helsingius. Johan Helsingius closes his Internet remailer. http://www.fitug.de/news/1997/penet.html, August 1996.

[41] Frans Kaashoek and David Karger. Koorde: A degree-optimal distributed hash table. In *Proceedings of the 2nd International Workshop on Peer-to-peer Systems*, February 2003.

[42] D. R. Karger and M. Ruhl. New algorithms for load balancing in peer-to-peer systems. In *Proceedings of the IRIS Student Workshop*, August 2003.

[43] Brad Karp, Sylvia Ratnasamy, Sean Rhea, and Scott Shenker. Spurring adoption of DHTs with OpenHash, a public DHT service. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems*, February 2004.

[44] K. Jayanth Kumar and Matulya Bansal. Anonymity in Chord. http://www.cs.berkeley.edu/~kjk/chord-anon.ps, December 2002.

[45] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. Technical Report 760/2001, University of Dortmund, September 2001.

[46] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In *Proceedings of ACM SIGCOMM*, August 2003.

[47] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing*, 2002.

[48] G. S. Manku. Balanced binary trees for ID management and load balance in distributed hash tables. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing*, 2004.

[49] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the ACM SIGOPS European Workshop*, 2004.

[50] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S. Wallachi, Xavier Bonnairei, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra. POST: A secure, resilient, cooperative messaging system. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems*, May 2003.

[51] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003.

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

[53] M. Naor and U. Wieder. Novel architectures for P2P applications: The continuous-discrete approach. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, June 2003.

[54] National Institute of Standards and Technology. Secure hash standard (SHS). Federal Information Processing Standards Publication 180-1, April 1995.

[55] Charles O'Donnell and Vinod Vaikuntanathan. Information leak in the Chord lookup protocol. In *Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing*, 2004.

[56] Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253, 2003.

[57] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. RFC1546, November 1993.

[58] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, version 0.17, July 2000.

[59] C.G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, June 1997.

[60] S Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, August 2001.

[61] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[62] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[63] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[64] R. Rivest. The MD5 message-digest algorithm. RFC1321, April 1992.

[65] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, November 2001.

[66] S. Sariou, P.K. Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking*, 2002.

[67] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of the Privacy Enhancing Technologies Workshop*, San Diego, CA, April 2002. Springer-Verlag, LNCS 2482.

[68] Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004.

[69] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.

[70] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM*, August 2001.

[71] S. Strong, R. Koberle, R. de Ruyter van Steveninck R., and W. Bialek. Entropy

and information in neural spike trains. *Physical Review Letters, 80*, pages 197–202, 1998.

[72] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[73] R. von Mises. Uber aufteilungs- und besetzungswahrscheinlichkeiten. *Revue de la Faculté des Sciences de l'Université d'Istanbul, 4*, pages 145–163, 1939.

[74] G. von Rossum and The Python Consortium. The Python programming language. http://www.python.org/.

[75] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium*. IEEE, February 2002.

[76] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[77] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 2004.

[78] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings fo the 2nd Symposium on Networked Systems Design and Implementation*, May 2005.

[79] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.