2008

Design and analysis of anonymous communications for emerging applications

Souvik Ray
Iowa State University

Recommended Citation

Ray, Souvik, "Design and analysis of anonymous communications for emerging applications" (2008). *Graduate Theses and Dissertations*. 11189.
https://lib.dr.iastate.edu/etd/11189

Design and analysis of anonymous communications for emerging applications

by

Souvik Ray

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Zhao Zhang, Major Professor
Arun Somani
Srikantha Tirthapura
Thomas Daniels
Clifford Bergman
Ying Cai

Iowa State University

Ames, Iowa

2008

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Acknowledgements

First of all, I would like to thank my advisor, Dr. Zhao Zhang for his able guidance, support and patience all throughout my academic life here at Iowa State University. It has been a privilege working under his guidance and I have been fortunate enough to learn quite a few research tricks under him which I hope to carry forward to my future endeavors. In the same breath, I must mention that he has helped me grow as a person as well; always extending a helping hand and motivating me when I felt down and out during my PhD days. I would like to thank Dr. Arun Somani, Dr. Srikanta Tirthapura and Dr. Giora Slutzki for their support, guidance, and thoughtful insight into the problems that I have tried to solve in my PhD dissertation. I would also like to thank my other committee members, Dr Thomas Daniels, Dr Clifford Bergman and Dr Ying Cai for their valuable suggestions during my prelims and final oral examinations.

I would like to thank my colleagues, Jiang Lin, Yong Joon Park, Rohit Gupta, Veerendra Allada, Satyadev Nandakumar, Toby Xu and many others for their valuable suggestions, comments and insights. I would also like to acknowledge the support and help of the late Dr. Anirban Chakrabarti who graduated from Iowa State University. I must also mention that without the help and support of my friends, faculty, and the support staff in the Electrical and Computer Engineering dept at ISU, I would not have been able to achieve this milestone.

Finally, I would like to thank my wife, Krishnalekha, for her able support, patience and encouragement during the final stages of my PhD. I would also like to acknowledge the support, patience and words of encouragement from my parents and sisters.

CHAPTER 1. Introduction

Anonymity refers to the state that an entity is not identified in the communications with others. As discussed in [62], anonymous communication may have one or more of the following properties: sender anonymity, receiver anonymity, and unlinkability. Sender anonymity means that when a message is observed, the sender cannot be identified; receiver anonymity means that the receiver cannot be identified. Unlinkability means that the relationship between the sender and the receiver in the communication cannot be identified, even if sender anonymity or receiver anonymity cannot be guaranteed. An anonymity mechanism may provide anonymity against one type of threat but not against another type. For example, using a proxy between senders and receivers may provide sender anonymity against the receiver and vice versa, but cannot provide any anonymity against an eavesdropper who can observe all messages from and to the proxy. Many Internet-based applications, such as web browsing, e-mail processing and peer-to-peer file sharing, have included anonymity mechanisms as part of their security features. For example, a simple proxy has been used in Anonymizer [7]. A mix [20] reorders outgoing messages to further provide unlinkability against an eavesdropper. Onion routing [93] uses a set of Onion routers, roughly a set of mixes, which increase rerouting path and thus enhance anonymity against eavesdropping. Crowds [73] uses rerouting of random length and makes user nodes be proxies for each other. Because messages are forwarded multiple times before reaching the destination, an adversary who observes a message can hardly tell whether the message sender is the initiator of the communication or not.

There are two primary factors which influence the design of any anonymity system: (1) type of available infrastructure and (2) type of application. The aforementioned systems either rely on trusted infrastructure or use the services of peer-to-peer based forwarding systems. In both types of anonymity systems, the infrastructure plays a crucial role. Thus unlike encryption and like routing, it is not enough just for the two end points (initiator and responder) to participate; the entire infrastructure must participate for successful anonymous communications. In some anonymity systems, the path is established by a centralized trusted entity and therefore the role of an intermediate node is restricted to forwarding. In such systems, the centralized server makes the routing decisions. On the other hand, some peer-to-peer based systems use the services of untrusted peers for forwarding traffic. In such cases, the final path formed from the initiator to responder is a result of independent routing decisions taken by intermediate forwarders. For example, in Crowds like forwarding, the traffic is routed by the intermediate forwarders with a certain forwarding probability and both the path length and the nature of the routing path is dependent on the intermediate forwarders. Moreover, peer-to-peer systems are characterized by frequent joins and leaves of peers resulting in significant amount of churn. Besides degrading the quality of service, churn also makes these systems vulnerable to frequent path reconstructions which in turn affects the anonymity of the system. Therefore, the routing primitives should be adaptive to the inherent churn in such systems. By contrast, traditional mix-based systems (based on the original Chaum mixes) rely on the services of static mixes which are always required to be a part of the system.

The type of application for which anonymous infrastructure is needed also influences the design of the anonymous forwarding and routing protocol. Web browsing is one of the foremost applications for which anonymous communication systems like Crowds, Onion routing and Mixes were designed. Since it is a low latency application, an inherent requirement of such anonymous protocols is a service of high quality with low communication overhead. Consequently, neither a long path length nor multi-routing based protocol designs are suitable for web-browsing. On the other hand, rerouting based systems are not ideally suited for applications which involve large data transfers . Consider the case of grid computing [94]. The existing approaches, when applied to grid computing environment, will incur overheads that are too high to be acceptable. This is because most grid computing applications involve transfers of very large amounts of data, and because system efficiency is much more important for grid applications than for the others. For example, a computing task in GriPhyN [43] may request data files of multiple gigabytes from storage nodes. If a small number of proxies were used, the proxies would quickly become the bottleneck as the number of communications increases. Using rerouting of random length could avoid having the bottlenecks, but will greatly increase network traffic and the processing times at intermediate nodes, and thus significantly reduce the efficiency of the whole grid.

1.1 Emerging Research Challenges

Research in the area of traditional anonymous communication systems has been well studied. However, emerging applications like distributed storage, Peer-to-Peer based distributed applications and grid-based applications have different requirements and privacy challenges which cannot be directly solved using traditional anonymity protocols. While applications drive the anonymity requirements, the type of available infrastructure influences the forwarding and routing characteristics at the intermediate forwarders. For example, rerouting based approaches are not directly applicable to data grids which involve large amounts of data transfer. This is due to the high overhead incurred. Again P2P-based anonymity systems must account for the free-riding [33] problem inherent in such systems. Free-riding results in a very dynamic forwarding infrastructure for such systems which affects availability and thereby anonymity. While incentive mechanisms [5] can generally address such issues, the design of an incentive mechanism for anonymity systems poses additional challenges. Finally, emerging distributed storage applications use a distributed hash table abstraction [91] for distributed storage of documents or files. Such applications are vulnerable to adversarial attacks on the storage nodes due to information leak from compromised routing tables and therefore require censorship resistance. A detailed quantitative analysis of privacy leak from such distributed data structures can give us insights into the design issues which impact privacy and enable us to design better structures. Our research goal is to make efficient and reliable anonymous communications available to emerging applications. In the next section, we outline the motivation and the research challenges addressed in this thesis.

1.2 Contributions of the Thesis

This thesis presents the design, analysis and experimental results of our proposed protocols/designs. Parts of this thesis were previously published as [67, 68, 69, 65]. Our contributions address the challenges mentioned in the previous section and can be summarized as follows:

1.2.1 Anonymous Data-transfer for Large Scale Data-intensive Applications

As mentioned earlier, rerouting based anonymity systems are not ideally suitable for data-intensive applications due to the huge overhead that might be incurred. One such application is transfer of data in Data-grids [36]. We have developed a novel lightweight 2-hop forwarding protocol which achieves anonymity in grid transactions while incurring minimal overhead. To the best of our knowledge, our work is one of the first that addresses anonymity for grid transactions. We also compare the performance of the protocol with traditional multihop protocols in peer-to-peer systems and analyze the degree of anonymity that can be achieved for various grid site configurations.

1.2.2 Enhancing Anonymity through Incentives in P2P Systems

Peer-to-peer anonymity systems like Crowds [73] and Tarzan [37] which use re-routing to forward the payload are vulnerable to two factors which directly affect the anonymity of the system: (1) free riding of peers [33] and (2) random forwarding leading to frequent path reformations in dynamic systems [99]. Free riding is a significant problem in peer-to-peer systems whereby some nodes leave the system after using its resources and do not provide anything in return. In an anonymity system, it affects the availability of forwarders and also influences the system size which provides a measure for the anonymity set. Moreover, it results in frequent joins and leaves by peers (churn), which leads to a significant number of path reformations when random forwarding is used by the intermediate forwarders. This makes the system vulnerable to certain types of anonymity attacks, e.g. intersection attacks [99]. While the issue of peer availability for anonymity systems has been addressed in previous research [35], we are not aware of any work which addresses the two issues, i.e. routing compliance and availability, within a single framework. We develop an efficient incentive mechanism for peer participation and routing compliance in P2P-based anonymity systems. We use techniques from game theory to develop favorable routing strategies corresponding to nash equilibria points and outline the implementation of a payment mechanism which can be integrated with the incentive mechanism.

1.2.3 Privacy-leak in Distributed Hash Tables

Distributed Hash Tables [91, 64] have recently been proposed as efficient data structures for search and storage systems. However, the structured nature of the DHTs makes them vulnerable to censorship attacks on the storage nodes. For example, the routing table at a compromised node in Chord gives information about the IP addresses of its fingers and their approximate range on the identifier circle. In comparison, unstructured systems like Gnutella [40] are highly resistant to censorship attacks because the routing tables do not maintain any state information. Although privacy-preserving variants of distributed hash tables have been proposed [87, 24], there has been no comprehensive study comparing the leak of privacy from routing tables for these distributed data structures. We have developed an analytical model to compare different DHT designs on the basis of their vulnerability to privacy leaks. Our model uses the information theoretic concept of Entropy to quantify information leak from routing tables.

1.2.4 Enforcing Location Privacy through Access Control

P2P based distributed storage applications use indexes to improve search performance [58, 47, 95]. These indexes also make the systems vulnerable to location privacy leaks about the file servers because they give a direct mapping between a file and its location. We propose a cryptography-based access control mechanism to increase the reliability of search indexes, such that only authorized users can generate the location. Our mechanism is different from previous approaches [89] in that we address both reliability and location privacy.

1.3 Dissertation Outline

The rest of the thesis is structured as follows. Chapter 2 outlines the basics of anonymity, including definitions, anonymity metrics and measure of anonymity (Anonymity Set, Entropy). We also include different attack scenarios and types of adversary model. This is followed by Related work in Chapter 3. Chapter 4 outlines the design of a 2-hop fowarding protocol for grid computing. We validate the light-weight nature of the protocol through experiments, analyze the degree of anonymity for different adversarial scenarios and compare its performance with other traditional multi-hop protocols designed for achieving end to end anonymity. Chapter 5 outlines an incentive based anonymity protocol for peer-to-peer networks. We develop a game theoretic based model for inducing peers to participate in forwarding packets for other nodes, generate nash equilibria points and show its feasibility for an anonymity system. We also outline the implementation of a payment mechanism which can be integrated with the protocol. Chapter 6 outlines an analytical model for comparing different DHT designs on the basis of their vulnerability to information leak from routing tables. The model uses the information theoretic metric of Entropy to quantify information leak. We compare our approach with other existing work and conclude that under different scenarios, different designs have better resistance to privacy leaks. Chapter 7 outlines the design of a cryptographic protocol to enforce access control on file location in distributed Peer to Peer storage systems. The protocol ensures availability and reliability of search indexes which are used in DHT based systems. We analyze the effectiveness of the protocol through simulations and also through a prototype implementation on the PlanetLab testbed. Finally, in Chapter 8, we conclude the thesis and outline future directions.

CHAPTER 2. Definitions and Assumptions

2.1 Preliminaries

In this section we introduce the reader to some preliminary concepts of anonymity. We first define anonymity and then describe the different types of anonymity.

> **"Anonymity is the state of not being identifiable within a set of subjects, the anonymity set."**

1. **Sender Anonymity** This refers to the state when a particular message is not linkable to any sender and that to a particular sender, no message is linkable.

2. **Recipient Anonymity** This refers to the state when a particular message is not linkable to any recipient and that to a particular recipient, no message is linkable.

3. **Relationship Anonymity** Relationship Anonymity corresponds to the case when it is untraceable who communicates with whom. Therefore it is not possible to link the sender and receiver. Relationship Anonymity is therefore a weaker property than both Sender and Recipient Anonymity.

4. **Pseudonymity** Pseudonyms are identifiers of subjects (senders and recipients). The use of pseudonymity then refers to the fact that a subject is identified by a pseudonym. Pseudonymity can be extended to groups (group pseudonymity) or it can also be transferable (transferable pseudonym).

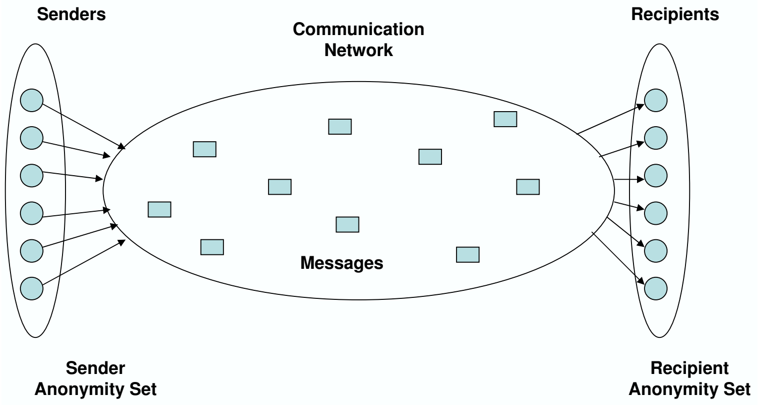

Figure 2.1. Anonymous Communication Infrastructure

2.2 Anonymity Metrics

2.2.1 Size of Anonymity Set

According to the definition of anonymity, the subjects who maybe related to an anonymous transaction constitute the anonymity set for that transaction. Figure 2.1 shows the **Sender anonymity set** and **Recipient Anonymity Set** for the anonymous communication infrastructure. Therefore larger the size of the anonymity set, better it is for Sender or Recipient anonymity. However, the metric of Anonymity Set size does not take into account the probability distribution of the subjects within the anonymity set, i.e. the probability that a particular subject is the sender or receiver. Therefore the use of entropy to calculate the expected size of the anonymity set was proposed in [81].

2.2.2 Entropy, Expected Size of Anonymity set and Degree of Anonymity

For a given distribution of probabilities, the concept of entropy provides a measure of information content of the distribution. Shannon entropy of a random variable X is then given by: $H(X) = -\sum_{x \in S} P(X = x) \log_2 P(X = x)$, where S is the set of possible values that X can take. $H(X)$ reaches it's maximum value when all states are equiprobable, that is if there is no indication whatsoever to assume that one state is more probable than another state.

Let N be the total number of subjects which can be linked to a transaction with a non-zero probability ($p_i > 0 \, \forall \, i = 1, \cdots, N$). The effective size of the anonymity set is then defined as the entropy $H(X)$ of the distribution X of probabilities that link the subjects of the anonymity set to the transaction.

We analyse the degree of anonymity achieved for different cases, where each case corresponds to the amount of information that the adversary possesses and derive conditions for zero and total anonymity. The degree of anonymity is defined as follows [90]:

$$d(\Omega) = 1 - \frac{H(X)_{max} - H(X)}{H(X)_{max}} = \frac{H(X)}{H(X)_{max}} = \frac{H(X)_{aposteriori}}{H(X)_{apriori}} \tag{2.1}$$

where Ω is the anonymity set.

An alternative entropy formulation is min-entropy:

$$H_{min} = -log_2 max_i p_i \tag{2.2}$$

However, Shannon entropy formulation better captures the probability distribution than min-entropy. We extensively use Shannon entropy formulation in our analytical models and experimental analysis.

2.3 Adversary Model and Attacks

The adversary attack model can be broadly classified as outlined by Raymond [71]:

- **Internal-External**: An adversary can compromise communication mediums (external) and mix nodes, recipients and senders (internal)

- **Passive-Active**: An active adversary can tamper messages, modify computations and actively control communication medium and other physical medium etc.; however, a passive adversary can just passively listen to communication medium and can try to correlate messages, but cannot tamper them. A passive adversary model is a much more realistic model than an active adversary model in most scenarios. However, even a passive adversary model can be very effective, especially in scenarios where it is easy to correlate traffic.

- **Static-Adaptive**: A static adversary selects and compromises resources before the start of a protocol; however, an adaptive adversary can dynamically select resources as the protocol progresses.

Some of the common anonymity attacks are **Intersection Attacks**, **Timing Attacks**, **Correlation Attacks**, **Predecessor Attacks** [100]. We briefly outline **Intersection Attack** which is the primary attack model addressed in our work [65]. In an Intersection attack, an attacker having information about other what users are active at any given time can, through observations, determine the endpoints of a connection. It is possible to get useful information about users by getting information about different anonymity sets which are active at different times and then generating an intersection of those sets.

CHAPTER 3. Related Work

3.1 Anonymity research

3.1.1 Chaums MIXes

Initial research in anonymity and untraceability can be attributed to the seminal work done by David Chaum. A number of protocols to thwart content correlation and causality correlation attacks, such as Web-MIXes [12], ISDN-MIXes [61], Stop-and-Go-MIXes [48], and many others have been based on Chaum's anonymous e-mail solution: a network of MIXes [20]. A mix shuffles a batch of messages together and outputs them in a random order. The sender and the mix use public key cryptography to hide the correspondence between input and output messages: the message to the mix is of the form, $\mathrm{Enc}\{K_m,\{D,M\}\}$, where K_m is the public key of the mix, D is the destination of the message and M is the payload. The mix decrypts the message, learns destination D and then sends the payload, M to D. Mixes can also be chained by using the encrypted message as the payload of a message encoded for a second mix. Each mix decrypts, delays and reorders the messages before relaying them. Mix-based approaches have high anonymity guarantees but suffer from high latency. The first widely used implementation of mix networks was anonymous remailers, using PGP encryption in order to wrap email messages and deliver them anonymously. They were followed by Mix-Master [3] and then MixMinion [27]. Chaum also proposed the Dining Cryptographers Problem [21]. It creates an anonymous broadcast channel without using cryptography as such, but it still provides information theoretic security guarantees. Systems like Anonymizer [7] and APFS Unicast [80] use an intermediate proxy to forward anonymous traffic. Since a high amount of trust is placed on the proxy, these systems are vulnerable to failure if the proxy is compromised.

3.1.2 Rerouting based Techniques

Onion Routing [93] and Crowds [73] are the most popular rerouting based alternatives and several prototypes have subsequently been deployed using these primitives. Both these approaches use single-path forwarding through intermediate routers to achieve anonymity. Onion routing uses a layered onion-like data structure to encrypt the payload and forwards it through known intermediate nodes. In Crowds, primarily designed for web-browsing, nodes forward web requests to each other at random, executing a form of random walk. At each step the random walk may probabilistically terminate and the current node then sends the request to the web server. Tor [31], Tarzan [37] and Center-Directing and Label-Switching [101] are systems built on top of Onion routing and Crowds routing primitives.

3.1.3 Peer to Peer Systems

Systems like MIXes and rerouting mechanisms like Onion Routing run into problems when the size of the network scales beyond 100 nodes; especially due to management of membership information about other nodes in the system. Peer to Peer networks were designed primarily with scaling in mind. In file sharing, Freenet [26] was designed to provide anonymity to both publishers and readers. Freenet nodes forward queries to neighbors, who then follow heuristics to try to reach a node with the desired file or data. GNUNet [11] is similar to Freenet though it uses a different forwarding algorithm. FreeHaven [28] was another similar mechanism but its design relied on a pre-existing anonymous communication channel rather than implementing anonymity mechanisms itself. Publius [97] and FreeHaven use similar strategies for achieving publisher anonymity. While Publius splits the symmetric key used to encrypt and decrypt a document into n shares, FreeHaven splits the document into n shares. Any k of the n peers must be available to reproduce the key (in the first case) and the document (in the second case).

Other Peer-to-Peer networks aim to provide a generic anonymous communication infrastructure, similar to onion routing. MorphMix [75] constructs an onion routing path by recursively selecting neighbors of the current mix nodes. Each node knows only about 6 other nodes, and uses its neighbors to find other nodes to use as mix servers. MorphMix uses a collusion detection scheme, which is based on IP addresses. Tarzan also creates a peer-to-peer onion routing system. Tarzan [37] addresses the collusion problem by using a restricted topology, where connections are formed only between nodes in different domains. Some recent designs have been built on top of structured peer-to-peer networks. Cashmere [104] uses anycast mechanism in Pastry [78] to ensure resilience of onion routes to individual node failures.

Any efficient implementation of an anonymity system is highly dependent on infrastructure support. It is not enough just for the two end points (initiator and responder) to participate; the entire infrastructure must participate for successful anonymous communications. Peer-to-Peer based anonymity systems are particularly vulnerable to non-availability of peers for forwarding traffic due to free-riding. Free-riding is an inherent problem in such dynamic systems whereby some nodes leave the system after using its services and do not provide anything in return. Incentive and reputation-based mechanisms have been proposed to address this issue in P2P anonymity systems.

3.2 Economics of Anonymity

The economic aspects of anonymity was first addressed in [4]. It outlined the reasons why anonymity systems are hard to deploy and enumerated the incentives to participate for both initiators and intermediate forwarders. Previous work in this area has treated protocol compliance [30, 29] and availability separately [35]. We address both the issues within the same framework. Reputation mechanisms were used to address the issue of compliance in MIX networks [29] and remailers [30] respectively. Reputation-based schemes are based on feedback about nodes in a system which are made through observations. As mentioned in [35], schemes based on system wide monitoring are not ideally suited for anonymity systems. Moreover, an inherent problem with a scoring or reputation mechanism is that nodes can collude with each other to increase their score or reputation and therefore increase their probability of being selected in the forwarding path. The work presented in [35] proposed the use of an incentive mechanism to ensure the availability of forwarders in a fixed length forwarding system. Although it addresses the issue of availability in forwarding-based anonymity systems, the proposed mechanism is limited to systems in which the identity of the intermediate nodes is known to the initiator. Incentive mechanisms for protocol compliant forwarding and routing have been proposed for both wired [5] and wireless [19, 6, 17] networks. The hidden-action problem in routing was addressed in [34] and an incentive mechanism was proposed to overcome the problem through the use of direct and recursive contracts. Micro-payment based schemes were proposed in [103] and [18] to stimulate cooperation in adhoc networks. However, these mechanisms are not ideally suited for anonymity systems in which the identity of the initator must be hidden from other peers.

3.3 Anonymity in Structured Peer-to-Peer Networks

There have been some attempts at providing anonymity for structured overlays. As mentioned earlier, AChord [44] attempts to improve recipient anonymity in Chord through the use of data lookup instead of address lookup. Thus, the IP address of the storage node is not revealed in the query reply. However, information leak from routing table entries cannot be prevented. Several other studies [59, 25, 13] have focused on the issue of sender-anonymity in Chord. We aim at analyzing the effect of leak of information from routing tables on recipient anonymity. An analytical framework for calculating information leak in the Chord protocol (with respect to the identity of the sender) is presented in [59]. Neblo [25] proposes the use of imprecise routing tables for enhancing recipient anonymity. While it highlights the importance of information leak from routing tables, the focus is on the Chord routing protocol and the design objective is obfuscating the information content of routing tables. We try to analyze the effect of the DHT routing geometry on the amount of information leak from routing tables. Anonymity in structured P2P networks was also studied in [13]. An empirical entropy-based metric was developed to measure source-anonymity in Chord. A routing extension was proposed which allows a tradeoff between anonymity and performance. Agyaat [87] attempts to provide recipient anonymity through the use of a two-tier hybrid organization in which the Chord structured overlay works together with a gnutella-like overlay to route messages. Gnutella-like clouds are connected with one another by means of a Chord ring.

3.4 Other Privacy-preserving Communication Systems

There are numerous other ways through which anonymity can be achieved and also other applications which require a certain degree of privacy. Since these protocols and applications are orthogonal to our interests to some extent, we just provide a brief outline here. In addition to unicast channels, anonymity can also be achieved through the use of group communications. P^5 [84] proposes a novel approach for mutual anonymity using broadcast channels. Hordes [85] provides initiator anonymity using multicasting. A multicast group is formed by all the initiator nodes. Initiators send requests to responders using Crowds or Onion routing, while the responder multicasts the response to the group of initiators. Epidemic protocols like MuON [9] have also been proposed to achieve anonymity in P2P systems where churn decreases the efficiency of the system. The MuON protocol achieves high resilience and appreciable anonymity through the use of an epidemic mechanism for disseminating information.

CHAPTER 4. Anonymity in Static and Closed Distributed Environments - An Anonymity Protocol for Grid Computing

In a traditional grid, entities trust each other as in a close community and thus do not make extra effort to hide their identity in communications. In fact, an entity must be authenticated and its identity be disclosed to its partners. As grid computing evolves, virtual organizations will be increasingly dynamic and complex. Grid computing will be integrated into other forms of distributed computing techniques such as peer-to-peer (P2P) networking [36]. For instance, in a commercial application enterprises may be accessing resources residing at third-part computing facilities, e.g. delegating a computation-intensive job or acquiring a large amount of data. Since the third party could have significant findings about an enterprise activity if the enterprise identity is disclosed, anonymous communication in such scenarios will be as important as other security issues.

Many Internet-based applications, such as web browsing, e-mail processing and peer-to-peer file sharing, have included anonymity mechanisms as part of their security features. For example, a simple proxy has been used in Anonymizer [7]. A mix [20] reorders outgoing messages to further provide unlinkability against an eavesdropper. Onion routing [93] uses a set of Onion routers, roughly a set of mixes, which increase rerouting path and thus enhance anonymity against eavesdropping. Crowds [74] uses rerouting of random length and makes user nodes be proxies for each other. Because messages are forwarded multiple times before reaching the destination, an adversary who observes a message can hardly tell whether the message sender is the initiator of the communication or not.

The existing approaches, when applied to grid computing environment, will incur overheads that are too high to be acceptable. This is because most grid computing applications involve

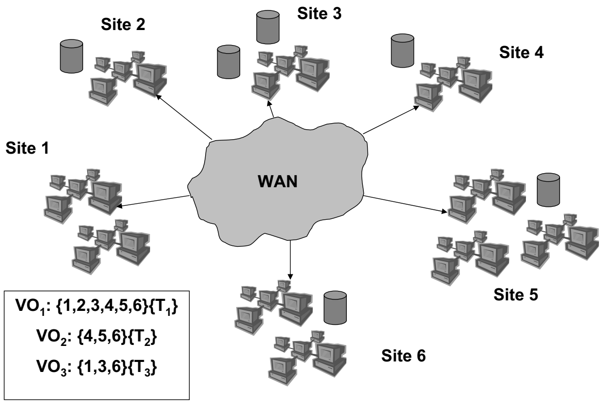

Figure 4.1. An example of a grid of multiple virtual organizations(VOs).

transfers of very large amounts of data, and because system efficiency is much more important for grid applications than for the others. For example, a computing task in GriPhyN may request data files of multiple gigabytes from storage nodes. If a small number of proxies were used, the proxies would quickly become the bottleneck as the number of communications increases. Using rerouting of random length could avoid having the bottlenecks, but will greatly increase network traffics and the processing times at intermediate nodes, and thus significantly reduce the efficiency of the whole grid.

We have found that *a distributed and highly efficient anonymity mechanism for grid computing can be designed if one takes use of the trust existing among certain sets of entities in a grid*. We call those sets of entities *trust sets*. In a grid a trust set may form if the entities have had close collaborations or if they belong to the same organization, which are typical in grid applications. In this chapter, we present the design of an anonymous forwarding protocol based on trust sets. In this protocol, trust sets are managed by grid administration nodes. When an entity requests an anonymous communication with an untrusted entity, another entity in the trust set of the requester will be selected as the forwarder for this communication. If only requester or responder anonymity is needed, the extra overhead incurred by the forwarding is roughly the cost of direct communication. If both types of anonymity are needed, the total overhead is roughly two times the cost of direct communication.

This anonymous protocol is different from previous ones in that it provides controlled anonymity,

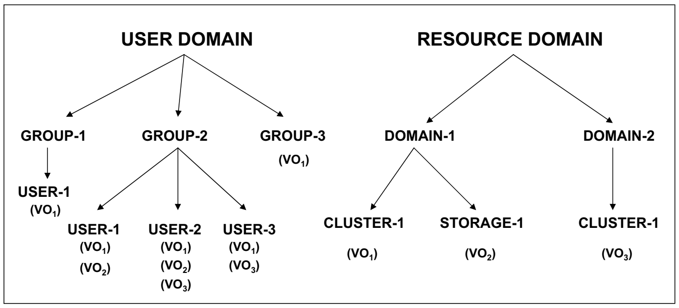

Figure 4.2. Resource sharing policy at site 6 in the example in Figure 1.

i.e. the identity of an entity in a communication is hidden from potential adversaries but not from some trustable peer entities. This type of anonymity cannot apply to many other Internet-based applications because the trust relationship does not exist at all. One would think that such a solution is trivial; however, it actually needs a careful design of trust management and an accurate analysis of the anonymity that it can provide. There are more complicated issues such as trade-offs between load balance and the anonymity in selecting a forwarder, or considering a small probability that a trustable entity might be taken over by adversary.

This protocol provides the best defense against the ill intentions of some entities involved in grid applications. It is not designed against eavesdropping and in that sense is not as strong as mixing or rerouting, but it still increases the difficulty of traffic analysis. In some cases a grid entity may not have trustable entities or only have a few to trust. Those cases are exceptional in grid applications and can be handled by reserving some proxies as forwarders for those entities.

4.1 Resource Sharing in Grids

Resource Sharing A grid can be viewed as a distributed collaborative environment where a group of producers and consumers participate in a virtual organization (VO) to achieve a common task. A virtual organization may consist of more than one site or physical organization. There can be a *user domain* and a *resource domain* associated with each site. A resource sharing policy may be enforced to determine how those sites share computing time, storage, and other resources among them. A single site or physical organization can participate in multiple VOs.

Consider the grid in Figure 4.2. Suppose that six sites decide to share their resources and

Privacy from site	Collaborative task	Privacy required?
1	T_1,T_2,T_3	No,Yes,No
2	T_1,T_2,T_3	No,Yes,Yes
3	T_1,T_2,T_3	No,Yes,No
4	T_1,T_2,T_3	No,No,Yes
5	T_1,T_2,T_3	No,No,Yes

	T_1	T_2	T_3
Site-1	1	0	1
Site-2	1	0	0
Site-3	1	0	1
Site-4	1	1	0
Site-5	1	1	0
Site-6	1	1	1

Figure 4.3. Privacy policy at site 6 and the VO membership matrix for the entire grid.

therefore form a virtual organization VO_1. Meanwhile, sites 4, 5 and 6 form another virtual organization VO_2 for achieving a different task, sharing another subset of their resources for that task. We represent each virtual organization as $VO_k = \{Entity\_set\}\{Task_k\}$. Thus, in this example $VO_2 = \{Site4, Site5, Site6\}\{T_2\}$. The resource sharing policy can be defined at the entity level. Figure 4.2 shows the entity level resource sharing policy for the user and resource domains at site 6.

Security, Privacy and Anonymity Security and privacy are critical to grid systems, of which the typical security features are authentication and encrypted communications. Anonymity is another important feature of privacy, and is desired by certain applications. For example, Internet users who are concerned with censorship would like to publish or receive files without disclosing their identity. In a commercial environment, anonymity may protect organizations from disclosing their activities to their opponents. Grid applications usually invoke both local and remote job executions. In case of remote executions, depending on the nature of the job and the relationship with the remote site, the mapping between the job submitted and the site from where the job is submitted may or may not be hidden from the remote side where the job is finally executed. Thus privacy is a policy issue and is site specific. Figure 4.3 shows the privacy policy of site 6 and the *VO membership matrix* of the entire grid. In the membership matrix M, $M[i][j] = \begin{cases} 0 & : \quad Site - i \notin VO_j \\ 1 & : \quad Site - i \in VO_j \end{cases}$

4.2 Anonymity and Trust Model for Grid Applications

Anonymity has not been provided in grid applications, but will be desirable in certain applications when their use extends. Consider the grid application of “Molecular Modeling for Drug Design”, in which molecules from a chemical database are screened for potential use as a drug. This is both data and compute intensive. A job (formulated as a parameter sweep application), when submitted to a grid, may involve data transfers from remote databases and execution at remote sites. In the highly competitive pharmaceutical industry, a company would want to hide its association with a particular drug design from its competitors to avoid restricted access to remote resources. Even for the more generic case of enterprise grids (P2P grids) which use platforms like XtremWeb [39] and Entropia [23], users submitting jobs have natural incentives for hiding their identity and remaining anonymous. Thus, while authorization (and therefore disclosure of identity) to some authority is required for submitting a job, strong incentives for remaining anonymous (and therefore hiding the identity of the user) to other grid entities also exist. In the long run, we believe that anonymity will be necessary for developing large scale and interconnected grids. The challenge is to efficiently handle anonymous communications in grid applications.

Trust is generally classified into **identity trust** and **behavior trust** [8]. Identity trust is concerned with verifying the identity or authentication of an entity. Behavior trust, on the other hand refers to the overall trustworthiness of an entity in a given context. For example, [8] uses the behavior trust model which is based on transactions in a grid computing environment. Moreover, a trust relationship can be one-to-one, one-to-many, many-to-one or many-to-many and can be symmetric or asymmetric ([42]).

The 2-hop forwarding protocol described in the next section assumes that the initiator site routes the job request through the forwarder site and once the job is executed at the remote site, the output data is sent back to the initiator through the forwarder site. Thus the initiator must have a high level of trust on the forwarder site. This trust is neither identity nor behavior trust, but is policy driven trust. Thus for example, if a site A has had a lot of remote job executions at B, it does not necessarily mean that it can trust B to forward a job request to C without revealing A’s identity to C.

We use the following trust model in our protocol. A grid essentially consists of various virtual organizations, which represent different trust domains. Thus a many-to-many and symmetric trust relationships exist among the members of a VO. We use the term *trust set* for the members of a virtual organization. In our trust model, we assume that if a site wants to send a job request to a remote site (such that job $\in T_k$), then it can trust any forwarder in VO which corresponds to T_k. Moreover, since both I and F belong to the same VO (w.r.t T_k), it is reasonable to assume that F would be aware of the type of job submissions at site I. We use this trust model in our analysis.

We use a simplified trust model in which each trust set is a virtual organizations and therefore each entity in a VO trusts every other entity. Other possible variants can include cases where each trust set can have entities from different virtual organizations.

4.3 Two-Hop Forwarding Protocol

We use a two-hop forwarding protocol (figure 4.4) in which the initiator routes the job request (e.g. input data, executables) through an intermediate forwarder and the job output is forwarded along the reverse path back to the initiator. We assume the existence of a trusted server called the Trust Set Maintenance (TSM) server, which maintains and periodically updates the VO memberships of the different entities in the grid. The TSM server generates the forwarding path for the initiator of the job request. This function may be integrated into some grid service nodes, such as job schedulers. The criterion for selecting the forwarder is described in details in the analysis section.

The procedure of creating a forwarding path is as follows:

- **Step 1:** The initiator contacts the TSM server S for forwarder selection. We assume that the initiator contacts a job scheduler to determine the identity of the responder.
- **Step 2:** The trusted server sends the forwarding path to I, which is encrypted with the public key of I. (We assume the existence of a public key infrastructure.) A randomly generated number n can be used to identify the transaction.

$$S \rightarrow I \quad : \quad \{n, F\}_{K_I}$$

- **Step 3:** I sends the job request to F and also signs the job request message using a certificate issued by a certificate authority for authentication. The job may consist of input data files,

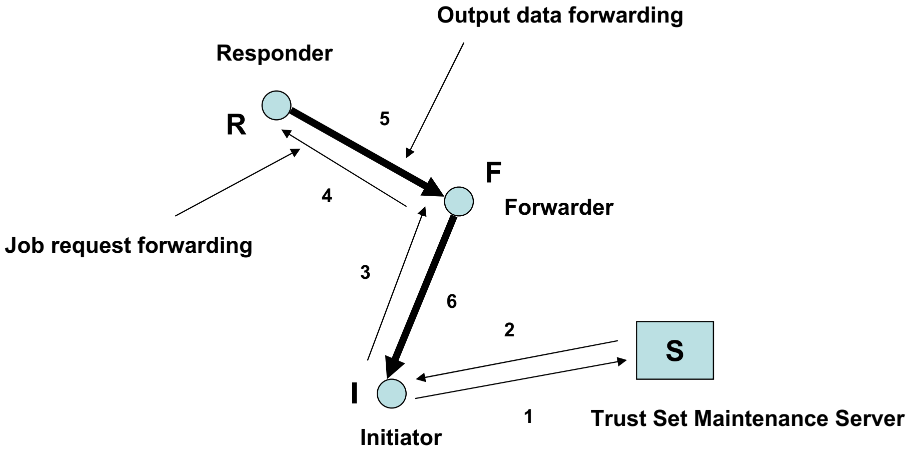

Figure 4.4. The 2-hop forwarding protocol.

executables etc.

$$I \rightarrow F \quad : \quad \{\{Job, n, R\}_{K_F}\}_{C_I}$$

- **Step 4:** F authenticates itself to R and forwards the job request on I's behalf.

$$F \rightarrow R \quad : \quad \{\{Job, n\}_{K_R}\}_{C_F}$$

- **Steps 5,6:** The output data is forwarded back to the initiator through the forwarder.

$$R \rightarrow F \quad : \quad \{Output, n\}_{K_F}$$

$$F \rightarrow I \quad : \quad \{Output, n\}_{K_I}$$

We will show that the protocol works effectively in a passive, internal [71] threat model, in which the adversary is from the responder site, e.g the administrator at the site or a intruder who has acquired the administrator privilege. For each incoming job request, the adversary will try to predict with a certain probability whether the site sending the request is the forwarder or the actual initiator (note that the forwarding protocol is used by the initiator only when it needs anonymity from the responder site). Moreover, the adversary may have partial or complete knowledge of the VO memberships of all the virtual organizations in the grid and can use this information to predict the Initiator anonymity set.

4.4 Analysis of the Degree of Anonymity

We use an approach based on information theory [83] to measure the **degree of anonymity** achieved for a given grid job transaction. For a given distribution of probabilities, the concept of entropy provides a measure of information content of the distribution. Shannon entropy of a random variable X is then given by: $H(X) = -\sum_{x \in S} P(X = x)\log_2 P(X = x)$, where S is the set of possible values that X can take. $H(X)$ reaches it's maximum value when all states are equiprobable, that is if there is no indication whatsoever to assume that one state is more probable than another state.

We analyse the degree of anonymity achieved for different cases, where each case corresponds to the amount of information that the adversary possesses and derive conditions for zero and total anonymity. The degree of anonymity is defined as follows [90]:

$$\begin{aligned} d(\Omega) &= 1 - \frac{H(X)_{max} - H(X)}{H(X)_{max}} \\ &= \frac{H(X)}{H(X)_{max}} \\ &= \frac{H(X)_{aposteriori}}{H(X)_{apriori}} \end{aligned} \tag{4.1}$$

where Ω is the anonymity set.

4.4.1 Definitions and Notations

We define S to be the set of grid sites. Considering a grid of size n, $S = \{s_1, s_2...., s_n\}$. Let T be the total number of collaborative tasks in the grid. This is also equal to the total number of VOs in the grid. We represent the VO membership of a site k as the set VO(k) for all tasks and VO(k,t) for task t. VO′(k) represents the set of entities which are members of exactly the same VO(s) as k. The VO membership matrix is represented as M, where $M[i][j] \in \{0, 1\}\ \forall\ 0 \leq i \leq n, 0 \leq j \leq T$. For example, in Figure 4.3, VO(3) = {1,2,4,5,6}, VO′(3) = {1} and $VO(3, t_3)$ = {1,6}. Ω is the set of possible initiators of a job request. We call this the *initiator anonymity set*. Ψ is the set of possible forwarders for a given initiator and a job request. We call this the Forwarder Set. We also use a random variable X, which denotes the initiator for a given job request.

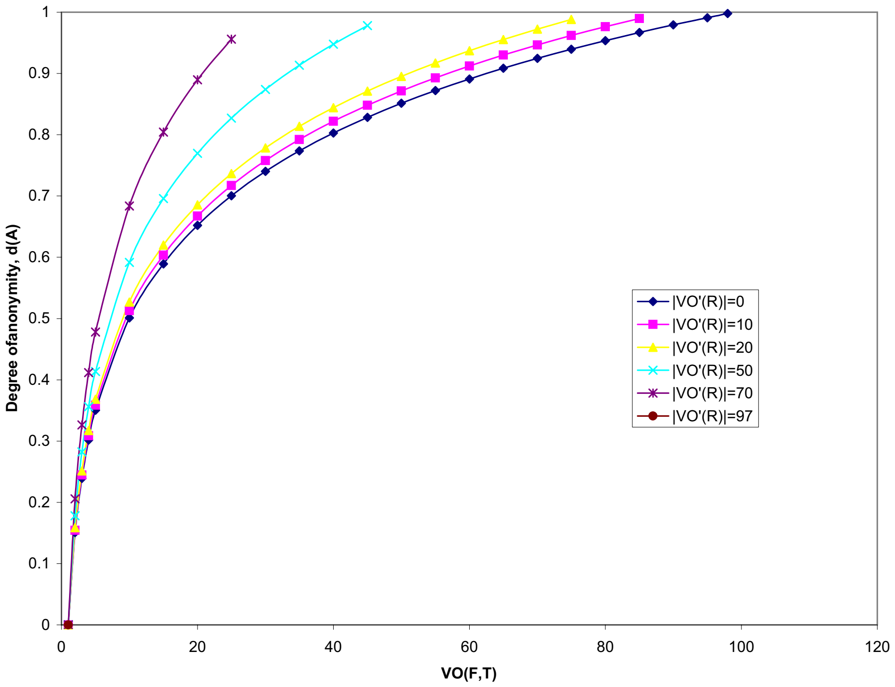

Figure 4.5. The variation of degree of anonymity with size of anonymity set (n=100).

4.4.2 A Priori and A Posteriori Entropy

We evaluate the entropies before and after the receipt of a job request at the responder site and analyse the degree of anonymity achieved for two extreme cases: (1) the responder site has information on only it's own VO membership; and (2) the responder site knows the entire VO membership, i.e. the matrix M. We also assume a uniform probability distribution over the anonymity set.

1. **Case 1: The responder site has the least information**

   In this case, the adversary (responder site) has membership information about the VOs to which it belongs, but it does not know the VO membership of other entities. According to the design of the protocol, a site uses a forwarder to route a job request only when it requires anonymity from the responder site. Moreover, all sites which are members of the same VO and collaborate for a common goal do not require anonymity from each other. Thus the responder can eliminate only itself from the anonymity set and therefore $|\Omega| = n - 1$, and

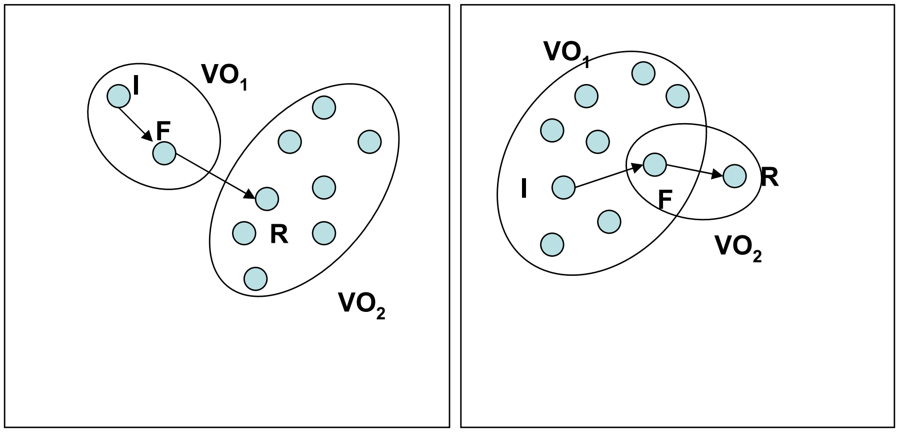

Figure 4.6. Minimum and maximum anonymity for a grid of size 10.

$P(X = s) = \frac{1}{n-1}$, and then

$$\begin{aligned} H(X)_{apriori} &= -\sum_{s\in\Omega} P(X = s)\log_2 P(X = s) \\ &= \log_2(n-1) \end{aligned} \tag{4.2}$$

After receiving the job request, the adversary gets to know the identity of the forwarder and can possibly predict the collaborative task to which the job belongs (without loss of generality, we can assume that the responder can predict the task from the executables and/or input data file sent by the initiator). Since the adversary has no information on the VO membership of F, it cannot predict the anonymity set and therefore for a large grid, a posteriori entropy is approximately equal to the a priori entropy and complete anonymity is achieved.

$$H(X)_{aposteriori} = \log_2(n-2) \tag{4.3}$$

$$d(\Omega) = \frac{H(X)_{aposteriori}}{H(X)_{apriori}} = \frac{\log_2(n-2)}{\log_2(n-1)} \approx 1 \tag{4.4}$$

2. **Case 2: The responder site has the maximum information**

In this case, the adversary knows the entire VO membership matrix M. Before observing the job request, the responder can eliminate only those entities from the anonymity set which belong to exactly the same VO(s) as the responder. Knowing the membership of F, it can derive the a posteriori anonymity set to be $VO(F, T_k)$, where T_k is the task to which the job belongs. Thus, a posteriori entropy is given by:

$$H(X)_{aposteriori} = log_2|VO(F, T_k)| \tag{4.5}$$

$$d(\Omega) = \frac{H(X)_{aposteriori}}{H(X)_{apriori}} = \frac{log_2|VO(F, T_k)|}{log_2(n - 1 - VO'(R))} \tag{4.6}$$

We analyse the variation of the degree of anonymity for different grid configurations for a given grid size. Figure 4.5 shows the variation of degree of anonymity with different sizes of anonymity set for a given size of the VO' membership set of R and a grid size of 100.
We have the following observations:

- The minimum degree of anonymity ($d(\Omega) = 0$) corresponds to the case when the initiator is a member of a virtual organization such that $|VO(I, T_k)| = 1$. This means that I has to use the only other member as the forwarder for collaborative task T_k.
- The maximum anonymity ($d(\Omega) = 1$) corresponds to the case when
  1. There are exactly two virtual organizations, one corresponding to T_k and the other corresponding to some other collaborative task;
  2. F is the only common member of both the VOs; and
  3. $|VO'(R)| = 0$.
- The ratio of the the size of the VO membership set of F, the VO' membership set of R and the number of virtual organizations influences the degree of anonymity that can be achieved. From Figure 4.5, we observe that a degree of anonymity in the range of (0.3-0.6) is achieved for a grid which has many VOs of small size ($|VO(F, T_k)| \leq 20$). For a given $|VO'(R)|$, the anonymity increases as $|VO(F, T_k)|$ increases. Intuitively we can say this observation is valid because if the initiator is a member of a VO which has many members, the anonymity set is larger and therefore a higher degree of anonymity can be achieved.

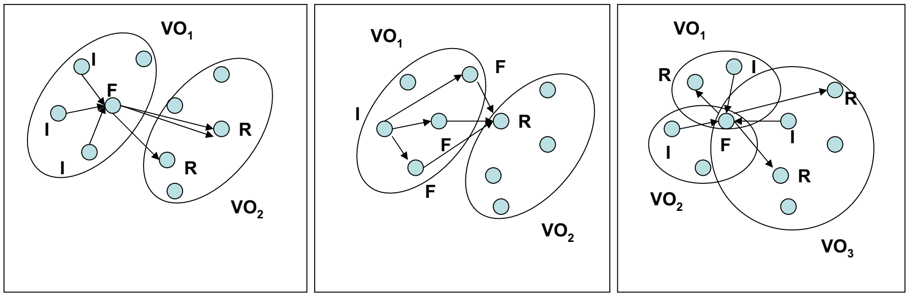

Figure 4.7. (a) Load imbalance due to selection of constant forwarder; (b) improved load balance but possibility of intersection attack; and (c) load imbalance due to selection of a forwarder which is a member of several VOs.

4.4.3 Selection of the Forwarder and Possible Attacks

From the analysis in the previous section, we observed that the size of the VO to which the initiator belongs w.r.t task T_k influences the degree of anonymity that can be achieved. Since in our trust model, we assume that the initiator trusts a forwarder only if it belongs to the same VO which corresponds to the collaborative task T_k, the selection of the forwarder is limited to the same VO. The selection of the same forwarder repeatedly for job request routing and data forwarding can lead to load imbalance and generation of hotspots. On the other hand, a random selection of the forwarder can lead to intersection attacks [84]. In an intersection attack, if the adversary knows that the initiator is in two different sets A and B, then the anonymity of the initiator is reduced to $A \cap B$. Moreover, a site which is a member of several VOs has a very high probability of being selected as the forwarder. Thus the selection of the forwarder is a tradeoff between load balance and higher degree of anonymity. Figure 4.7 shows the three different cases described above.

4.5 Performance Evaluation

We use a grid simulator (written in Java) to evaluate the performance of the 2-hop forwarding protocol. We use the BRITE [56] topology generator to generate a 2-tier (WAN,LAN) grid architecture containing 100 sites. A file size of 1 GB and link bandwidth varying between 0.1 and 10 Gbps is used. We use data transfer latency and bandwidth consumption as metrics for evaluating

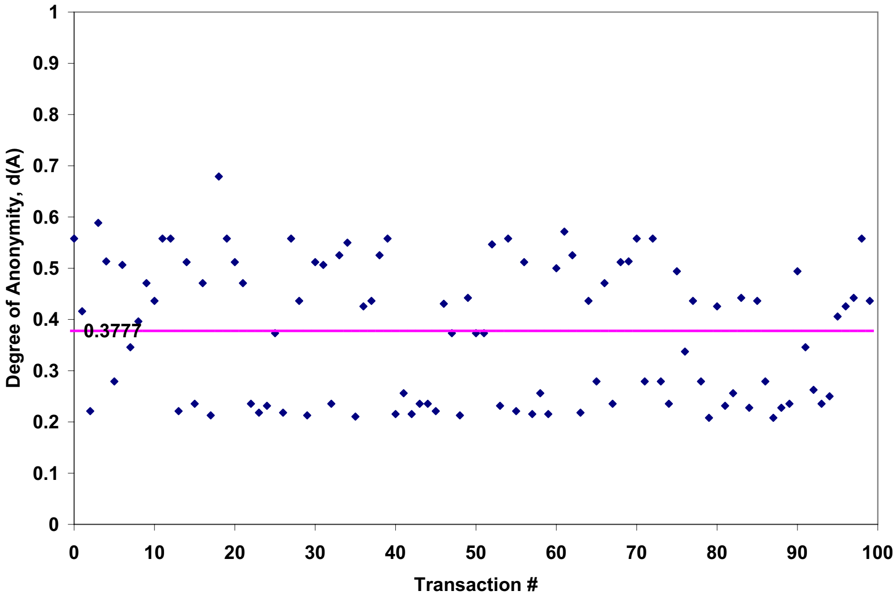

Figure 4.8. Degree of anonymity achieved on a per transaction basis.

the overhead associated with the protocol as compared to the case when initiator anonymity is not required. Note that system overhead at the forwarders is roughly proportional to the bandwidth consumption.

In our simulations we generate random transactions, where each transaction is the routing of a job request from the initiator to the responder through a forwarder and the transfer of output data from the responder to the initiator through the forwarder. For ease of implementation, we assume that a remote job is executed at the responder in it's entirety. For each transaction we measure the degree of anonymity, the file transfer latency and bandwidth consumption. We only consider the transfer of the output data from the responder to the initiator in our calculations.

4.5.1 Degree of Anonymity, Data Transfer Latency and Bandwidth Consumption

Figure 4.8 shows the degree of anonymity/transaction for a total of 100 transactions (grid configuration consists of 100 sites and 15 VOs, where each VO contains 3-8 sites) using the 2-hop forwarding protocol when the adversary has the *maximum information*. We observe that even when the adversary has the maximum information (this corresponds to the worst case for the initiator), $d(\Omega)$ lies in the range 0.2-0.6, with an average of 0.3777. Using information theory terminology, we

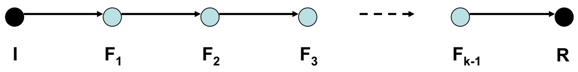

Figure 4.9. k-hop forwarding.

can say that on an average, even in the worst case, about 40% of the bits of the initiator's identity are hidden from the adversary (responder).

We observe that for output files of size 1 GB, both file transfer latency and bandwidth consumption increase by about a factor of 2. Thus this protocol is applicable for grid systems that require anonymity but can only accommodate modest overhead in achieving initiator anonymity.

4.5.2 Comparison with Multi-hop P2P Forwarding Protocol

In this section, we compare our protocol with a multi-hop forwarding protocol. We first outline some of the core functional differences between P2P and Grid, which influences the design of any forwarding protocol.

1. A grid consists of trust domains and therefore anonymity would normally be required from untrusted entities. This is different from traditional P2P where trust sets/domains do not exist and every peer node requires anonymity from every other peer node.

2. A peer to peer network is generally associated with file downloads where files are relatively small in size. Thus a file request does not provide any information about the initiator. In a grid, on the other hand, each virtual organization is associated with a collaborative task and therefore it might be possible to map a job request to a particular virtual organization, and thus this information can be used by the adversary in reducing the size of the anonymity set.

Figure 4.9 shows a k-hop forwarding protocol where $k - 1$ intermediate forwarders are used. We assume that an intermediate forwarder can be compromised with a certain probability. We compare our protocol with the Crowds [74] anonymity system.

In Crowds, the probability that I is the initiator (such that I occupies the 0th position and the first collaborator occupies any position ≥ 1) is given by $P(X = I) = \frac{n - p_f(n-c-1)}{n}$, where n is the

Table 4.1. Comparison of degree of anonymity.

No. of entities	Degree of Anonymity	
	2-hop	Crowds
40	0.99291	0.972287
80	0.997085	0.965869
120	0.998234	0.96222
160	0.998755	0.959724

total number of entities, p_f is the probability of forwarding by an intermediate forwarder and c is the total number of collaborators. Using (1), we calculate the degree of anonymity as:

$$\begin{aligned} d(\Omega) &= -[P(X=I)\log_2 P(X=I) \\ &\quad +(1-P(X=I))\log_2\frac{1-P(X=I)}{n-1}]/\log_2 n \\ &= [\frac{n-p_f(n-c-1)}{n}\log_2\frac{n}{n-p_f(n-c-1)} \\ &\quad +\frac{p_f(n-c-1)}{n}\log_2\frac{n(n-1)}{p_f(n-c-1)}]/\log_2 n \end{aligned}$$

Table 4.1 compares the degree of anonymity achieved in our 2-hop protocol with that of the Crowds protocol (we use c=0 and p_f=0.9) for the case when the adversary has the least information. As predicted, the anonymity of the 2-hop protocol is much stronger because the 2-hop protocol uses a trusted forwarder, which has a negligible probability of being compromised.

4.6 Summary

We have studied a new anonymity approach for grid computing and designed the 2-hop anonymity protocol. Based on the existing trust in grid application, this protocol is more efficient and stronger than the existing protocols. The degree of anonymity is quantitatively analyzed and the efficiency confirmed by our simulation. Nevertheless, the study is limited to cases that trust relationship can simply be derived from virtual organization membership. With a more practical and complicated trust model, which has not been well studied for grid computing, this protocol could be enhanced by considering varying trust levels, the possibility of attacks to trusted sites, and more trade-offs between anonymity and load balance.

CHAPTER 5. Dynamic Peer-to-Peer Infrastructures and their Effect on Anonymity

Many anonymity protocols and systems have been developed in recent years to support anonymous communications (initiator anonymity, responder anonymity or unlinkability). Most of those systems assume some forwarding insfrastructure, e.g. trusted forwarding servers or a large set of P2P forwarding nodes. Nevertheless, there lacks research on the insfrastructure itself. It is not a trivial issue: The high operational cost of trusted infrastructure has resulted in the commercial failure of such systems, e.g. the Freedom Network [41]. P2P-based forwarding systems [73, 37] are more commercially viable because they use unreliable and untrusted forwarding nodes, and has the advantage of using non-centralized forwarding nodes. However, P2P-based forwarding systems are affected by the churn problem, i.e. the frequent leaves and joins of nodes.

5.1 Effect of Churn on Anonymity

Churn is the continuous process of arrival and departure of peers in a dynamic P2P network. Existing studies of file-sharing systems use a node's **session time** and **lifetime** as the primary metrics of churn. A node's session time is the elapsed time between it joining the network and subsequently leaving it. A node's lifetime is the time between it entering the network for the first time and leaving the network permanently. It has been observed [76] that median session times are of the order of tens of minutes and median lifetimes of the order of days. It is common in P2P systems because of free riding [33], a scenario that many nodes join a P2P system for a short time to enjoy its benefit but not to provide the expected service. There are two negative consequences for P2P-based forwarding systems: It affects the availability of forwarding nodes, which reduces the size of anonymity set; and it forces frequent reformations of forwarding paths, which make the system vulnerable to intersection attacks [99].

5.2 Can Incentive Mechanisms Help?

To address the churn problem, incentive mechanism has been introduced to induce P2P nodes to provide stable service [35]. Nevertheless, an incentive mechanism for anonymous forwarding must consider the quality of anonymity beyond the stability of service. In a simple incentive mechanism, a forwarder may align its routing decisions to maximize its local interests; for example, to minimize communication costs. Additionally, the churn problem may only be alleviated – even with incentive, new nodes may continue to join and old nodes may leave in typical P2P systems. The problem of frequent forwarding path reformation still exists and should be considered in the design of the incentive mechanism.

In this chapter, we describe an incentive mechanism that induces the forwarders to make forwarding decisions aligned with the quality of initiator anonymity. The objective is achieved by binding the incentive received by a local node with the quality of initiator anonymity at the system level – a local node may maximize its interests by using a routing strategy aligned with the goal of anonymity. To have a sound foundation, we use game theory to design and analyze the forwarding strategy. We also propose payment-based incentive mechanism that keeps the anonymity of involved parties.

5.3 Incentive-based Forwarding and Routing Model

5.3.1 Motivation and Design Objectives

P2P and forwarding-based anonymity systems are primarily distinguished by their routing and forwarding infrastructure. In Onion routing [72] and MIX-based systems [20], the routing is done before the forwarding and therefore a forwarder merely performs the forwarding. In Crowds [73], a forwarder makes the routing decision. There is a hidden-action problem [34], i.e. the actions of the forwarders are hidden from the initiator and the quality of the path is decided by the decisions of the forwarders, which is not necessarily aligned with the quality of anonymity. We believe that a properly designed incentive mechanism is required to ensure appropriate forwarding and routing of packets in such systems. Designing such a mechanism for an anonymity system, however, is challenging because the mechanism itself cannot leak the identity information.

Intersection attack is a serious concern for anonymity systems that are used by applications with recurring activities; for example, those using HTTP, FTP, NNTP or raw sockets [98]. In those applications, an initiator usually makes repeated connections to a set of specific responders. The reformations of the forwarding path, which can be caused by frequent node joins and leaves, will increase the chance of exposing the initiator and the responder to the intersection attacks. In an intersection attck, the attacker observes the intersection of the sets of active nodes at different times and may find out the initiator or responder by reducing the intersection set.

The availability of forwarders has a strong impact on the success rate of intersection attack. In a P2P system, the availability of a peer node can be expressed as the ratio of the sum of its sessions times to its lifetime, where the lifetime is from the time of the initial entry of the peer node into the system to the time of its final departure, and a session time is the time between the arrival and the departure during a single session [76]. Consider the anonymous forwarding from an initiator I to a responder R. The higher the availability of a peer node, the higher the probability of it being selected as a forwarder. If a different set of forwarders are selected for each recurring connection between I and R, the probability of an successful intersection attack increases. In other words, if F_1,F_2,...., F_t are the set of all forwarders involved in the anonymous forwarding from I to R, then one should minimize this metric: $Q = |\bigcup_{i=1}^{t} F_i|$. Therefore, two conditions are desired in the systems we are concerned: (1) a relatively static set of intermediate nodes, and (2) stable selection of forwarders for all connections between I and R. The first condition is related to the availability of nodes, the second is concerned with the routing decision at each intermediate node.

Our goal is to design an incentive mechanism that not only induces peer nodes to provide stable forwarding service but also encourage them to make routing decisions aligned with the system objective of providing anonymity. We assume that the peer nodes are untrusted and unreliable in general. We quantify the problem as follows: Let $\pi = \{\pi^1, \pi^2 \cdots \pi^k\}$ be the set of k recurring connections between I and R and let L denote the average length of forwarding paths between I and R. We define the path quality of π, denoted by $Q(\pi)$, as $\frac{L}{||\pi||}$ where $||\pi||$ represents the size of the forwarder set[1]. The system objective is to maximize $Q(\pi)$ by minimizing $||\pi||$.

[1] L is used to normalize the forwarder set size for a given average path length.

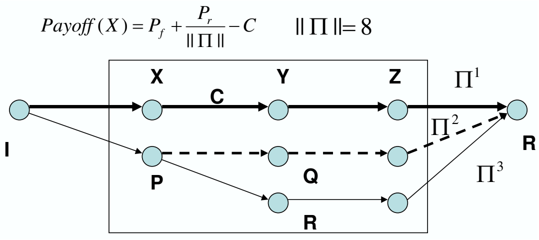

Figure 5.1. Random routing by P and unavailability of X leads to a large size of the forwarder set.

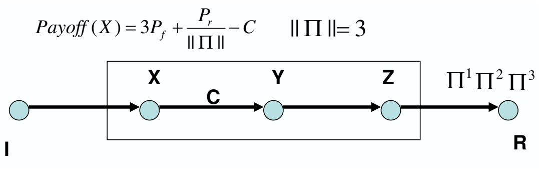

Figure 5.2. Stable set of forwarders.

5.3.2 Outline of Incentive Mechanism

We model the system as a network of N nodes which participate in anonymous forwarding of data packets. Each node s maintains information about a fixed number d of neighbors which can be used as potential forwarders. This neighbor set is denoted by $D(s)$ (for a detailed description of the system, we refer the reader to the corresponding technical report [66]). When an initiator I decides to set up a connection to a responder R, it uses the following mechanism. It makes a commitment to pay an amount P_f to any intermediate forwarder, per forwarding instance (**forwarding benefit**). In addition it also decides to pay a total shared benefit (**routing benefit**) equal to P_r to all the forwarders. Thus if a forwarder participates in m forwarding instances, its benefit is $mP_f + \frac{P_r}{||\pi||}$. The idea of separating the total benefit into routing and forwarding components serves the following purpose. The forwarding benefit induces availability of nodes because even if a forwarder makes a random routing decision, it still stands to gain by just participating in the forwarding process. The routing benefit induces the nodes to make routing decisions which are aligned with the system objective of minimizing $||\pi||$.[2] This can be achieved by making non-random routing decisions. Note that the routing benefit induces an implicit cooperation among forwarders. For example, in Figure 5.1, node X is not available for forwarding in π_2; consequently its forwarding benefit is smaller than the scenario in Figure 5.2. Moreover, the routing benefit for each forwarder is reduced from $\frac{P_r}{8}$ to $\frac{P_r}{3}$. Thus the utility function must be designed in such a way

[2] Note that we are basically concerned with the forwarder set for appropriate path lengths. The system objective is to ensure a minimum size forwarder set for path lengths which are appropriate for anonymity systems. For example in Crowds, tweaking the value of forwarding probability appropriately results in appropriate path lengths.

that in trying to maximize their utilities, nodes take forwarding and routing decisions which are aligned with the system objective.

We next define the utility function which is used by a forwarder X to select the next hop on the path from I to R. Let q_e be the quality[3] of the forwarding edge $e = (X, Y)$ from X to Y on some path $\in \pi$ and C be the sum of participation cost and forwarding cost (to Y) incurred by X. From a system perspective, we would want that a forwarders utility be aligned with the global objective, i.e. its utility should increase if it selects a high quality edge. We therefore define the utility for a forwarder X as

$$U_X(Y) \quad = \quad P_f + q_e P_r - C \tag{5.1}$$

Note that in trying to maximize its utility, X selects high quality forwarding edges which in turn increases its payoff due to a decrease in $||\pi||$. Thus this utility model captures the effect of local decision making on the final payoff to a forwarder and aligns its interest with the system objective. An intermediate forwarder X decides to participate or not participate in forwarding and routing of the payload on the basis of its utility. It calculates its utility corresponding to each neighbor $g \in D(X)$ and selects the neighbor which gives it the maximum utility as the next hop. Ties are broken by selecting a neighbor with a higher quality. Note that since the identity of the intermediate nodes (except first hop) is not known to the initiator, the establishment of the forwarding path is based on propagation of contract information (P_f and P_r) through the intermediate nodes (Note that both Crowds like probabilistic forwarding and hop-distance based forwarding are applicable to our model). Finally after R receives the payload, it sends back a confirmation through the reverse path. Each intermediate forwarder also includes path informtion which is then used by I to recreate the path and validate it. After evaluating the path quality, the initiator uses a central entity (bank) to make payments to the forwarders. Note that although the identity of R is known to the intermediate nodes, the identity of I is not leaked and therefore the system achieves initiator anonymity. The payment is made by I only after all the connections in π are completed. Details about path quality evaluation and payment mechanism can be found in the technical report [66]. For the initiator, a high benefit corresponds to a low value of $||\pi||$ (such that $A(||\pi||)$[4] increases with

[3] We introduce the notion of edge quality in the context of path reformations in section 5.3.3.

[4] We use A(.) as a function for quantifying the anonymity.

Connection id	Predecessor	Successor
cid	X	Y

Table 5.1. History profile at node s.

decrease in $\|\pi\|$) and the cost it incurs is equal to the payment it makes to the forwarders. Therefore

$$U_I = A(\|\pi\|) - \|\pi\|P_f - P_r \tag{5.2}$$

Relationship between forwarding and routing benefits: A high value of P_f increases the probability of peer participation in the forwarding process. A high value of P_r gives a higher weightage to the benefit and this results in a higher profit for a forwarder. Since a high benefit corresponds to a high quality path, a high value of P_f results in the formation of high quality paths between I and R. Thus depending on its anonymity requirements, the initiator can select appropriate values for P_f and P_r. Note that the ratio of P_f and P_r also affects the decisions taken by the forwarders. If $P_r = \tau P_f$, then a small value of τ would induce nodes to forward traffic; however, it may not align their routing decisions to the system objective. On the other hand a high value of τ will induce nodes to make effective forwarding and routing decisions.

5.3.3 Edge Quality

Connection history. Each node stores history information about connections passing through it. Thus if a node s lies on a path π^i with connection identifier $\mathtt{cid}$, it stores the corresponding predecessor and successor hops as shown in Figure 5.1. The history information at s for the k^{th} connection, represented as $H^{k-1}(s)$, consists of all outgoing edges from s which lie on $\pi^1, \pi^2, \cdots, \pi^i, \cdots, \pi^{k-1}$. Note that by using the predecessor information, a node can differentiate between outgoing edges for two different positions on the same path (e.g. if node s occupies two different positions on π^k).

Availability of neighbors. Each node also stores availability information about its neighbors. In the absence of a centralized entity for collecting availability information, each peer calculates availability of its neighbors using its own observations. A peer uses active probing [92] to monitor its neighbors. Mechanisms based on active probing have been used to estimate churn in peer-to-peer systems. We use a methodology similar to [16] to estimate availability. When a peer first joins the system, it initializes the session time of each of its neighbors to 0. At the start of each probing period a peer s checks the liveness of each neighbor. If the neighbor is alive, its session time t_s is updated as $t_s^{new} = t_s^{old} + T$, where T is the probing time period. If a new neighbor is found, its session time is updated as $t_s^{new} = rand(0, T)$ where $rand(0, T)$ is a uniformly distributed random value in the range $(0, T)$. Finally availability of a neighbor u, $u \in D(s)$ is calculated as $\alpha(u) = \frac{t_s(u)}{\sum\limits_{v \in D(s)} t_s(v)}$. Thus a neighbor with a higher observed session time has a higher availability. This is in accordance with observed session times of peers in peer-to-peer file sharing systems, which is modeled using a pareto distribution [53]. We represent the availability of a node v as observed by s as $\alpha_s(v)$ s.t. $0 \leq \alpha \leq 1$

Determining edge quality. We now outline a local mechanism for determining the quality of an edge at a node. Consider a node s which lies on π^k. Let $D(s)$ be the neighbor set of s. s calculates the quality of each outgoing edge, $q(s, v)$ using a procedure which takes as inputs v, $H^{k-1}(s)$ and $\alpha_s(v)$. Given an edge (s, v), s looks up its history information $H^{k-1}(s)$ (path information corresponding to $\pi^1, \pi^2 \cdots \pi^{k-1}$) for any entry corresponding to (s, v). The ratio of the number of entries corresponding to (s, v) and the maximum possible entries $(k-1)$ is called its selectivity and represented as $\sigma(s, v)$. Weights w_s and w_a are assigned to selectivity $\sigma(s, v)$ and availability $\alpha(v)$ respectively such that $w_s + w_a$=1. Finally, the edge quality is calculated as $q(s, v) = w_s\sigma(s, v) + w_a\alpha(v)$. The weights w_s and w_a signify the relative importance of selectivity and availability. A high value of w_a signifies a higher importance to the availability of the forwarders, with the objective that these forwarders would be available for forwarding for future connections. A high value of w_s on the other hand signifies higher importance for past history. Note that the edge quality of the last edge in the path π^k is always 1 because it ends in R. Note that w_s and w_a are system parameters which are set depending on the anonymity requirements of the system. The amount of history information stored at a node also influences the quality of the edge. The quality of a path π^k is then given by the sum of the qualities of the individual edges. We next show how incentive based non-random routing leads to reduction in path reformations.

Proposition 1. *Incentive based non-random routing by intermediate nodes leads to reduction in path reformations when compared with random routing.*

Proof. Consider an edge $e = (a, b)$ on path π^k. Let X be a random variable such that

$$X = \begin{cases} 0 & : \text{ if } e \in \bigcup_{i=1}^{i=k-1} \pi^i \\ 1 & : \text{ otherwise} \end{cases}$$

We need to show that $E[X]$ for random forwarding is greater than $E[X]$ for utility based non-random forwarding. If random forwarding is used, then $E[X] \geq \frac{N-(k-1+1)}{N} = 1 - \frac{k}{N}$. Since $k << N, E[X] \rightarrow 1$. In the case of utility based forwarding, a new edge is added only if there is no existing edge in $\bigcup_{i=1}^{i=k-1} \pi^i$. Let the probability that an edge $\in \pi^i$ is available in π^k be p_i. Then $E[X] = (1 - p_1)(1 - p_2) \cdots (1 - p_i) \cdots (1 - p_{k-1})\ \forall\ 0 < w_s, w_a < 1$. Note that since $w_a > 0$, $p_i \rightarrow 1$ as $i \rightarrow k$. Consequently $E[X] \approx 0$. From equation 5.1, $U_a(b)$ increases with an increase in $q(a, b)$ and therefore a rational forwarder would always try to select high quality edges which in turn leads to low path reformations. □

5.3.4 Forwarding and Routing Strategy

We model the forwarding and routing mechanism as a finite multi-stage game [38] where the peers are the players. Consider a system containing N peers represented by the set $V = \{1, 2 \cdots N\}$. A nodes forwarding and routing strategy space SS is the set of all nodes in the system (except itself) along with the NULL entity, which corresponds to the case when a node does not participate in the forwarding path. Therefore $SS_i = \{1, 2, \cdots, i-1, i+1, \cdots, N, NULL\}$. The strategy profile of node i is represented as $\{S_i, S_{-i}\}$ where S_i and S_{i-1} represent the strategies of i and other nodes respectively. For a path π, the strategy profile of the set of nodes is therefore given by $SP = \bigcup_{i=1}^{N}\{S_i\}$. Figure 5.3 shows an example of a game tree, where at each stage a node has three choices; a) not participate in forwarding, b) forward and route randomly, c) forward and route non-randomly.

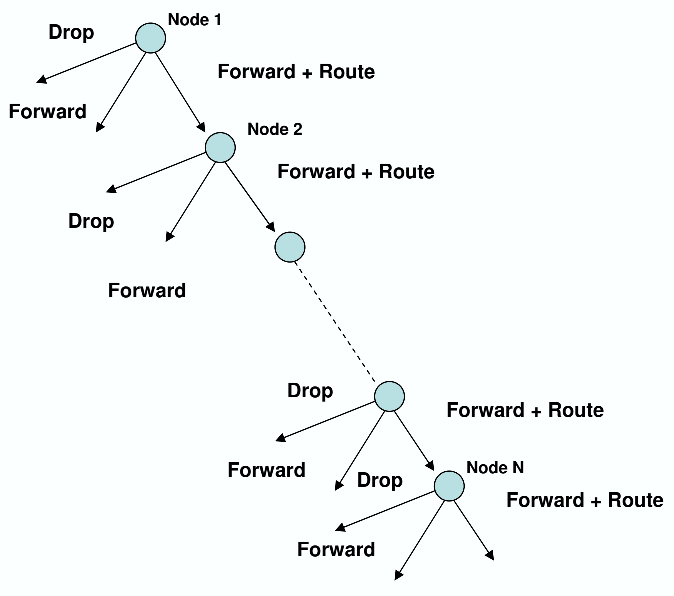

Figure 5.3. Game tree. A peer takes one of the three strategies: (1) does not participate in forwarding (Drop), (2) forwards with random routing (Forward), and (3) forwards with non-random routing (Forward+Route).

Note that the primary objective of an adversary in an anonymous forwarding system is to identify the end points of a communication and therefore its routing decision is not aligned with any economic incentive. We model an adversary's routing strategy as random routing. Note that the main objective of an adversary is to break initiator anonymity; therefore it is not concerned about the incentive. Our main objective here is to ensure that there is an equilibrium in the strategies for the selfish nodes (those who want to obtain maximum income) and in doing so the availability of nodes in the system increases. This in turn affects the anonymity of the system.

We first outline some game-theory basics. A *Dominant Strategy* [38] for a player i is a strategy which gives it an optimal utility irrespective of the strategies taken by other players. A *Nash Equilibrium* [38] represented as $\{S^*_i, S^*_{-i}\}$ is a strategy profile in which each players utility is optimal given that the other players have also played their optimal strategies. An equilibrium is a weaker property than a dominant strategy. Finally, a *Subgame Perfect Nash Equilibrium (SPNE)* [38] is a special case where irrespective of the past, playing the assigned strategies from the current stage is still an equilibrium.

5.3.4.1 Cost Model

The costs incurred by peers includes the following: a) Participation cost and b) Transmission cost. Any participating peer incurs a certain cost which depends on the nature of the application. A number of internet protocols including HTTP, FTP, NNTP and raw sockets etc. are characterized by a recurring traffic pattern [98] and therefore any client application which uses these protocols is vulnerable to path reformations. Consequently our cost model should be generic and must not be limited to any particular application. The cost of participation therefore includes the cost of running a software associated with a particular application for a peer session. This cost is represented as C^p. For the initiator, the participation cost is equal to the sum of payments that it makes to the intermediate forwarders. The transmission cost for a peer is associated with forwarding the payload to the next hop and is represented as C^t. If the payload size is b and per unit transmission cost to the next hop is l, then $C^t = bl$. We ignore the cost of transmitting control packets which is negligible. Note that the participation cost is incurred by a peer for participating in the anonymity system and is a one time cost, while the forwarding cost is incurred per forwarding instance. A peer tries to maximize its own access bandwidth for sending its own traffic and therefore during data forwarding for other peers, its rational (selfish) nature will make it forward traffic on low bandwidth links. This type of selfish behavior by peers has been modeled for peer-to-peer streaming [86].

5.3.4.2 Utility Model I

The forwarding and routing mechanism can be treated as a game where each forwarder can be modeled as a player. The path formation can be modeled as a sequential reasoning process at each forwarder such that the decision is influenced by the utility to the forwarders. Having outlined a possible mechanism for evaluating edge quality in section 5.3.3, we can now formalize the utility model for the i^{th} peer (from equation 5.1).

$$U_i(j) = P_f + q(i,j)P_r - (C_i^p + C^t(i,j))$$

In this model the benefit to a forwarder i is proportional to the quality of the edge from i to its successor j. The rationale behind using this model is that each forwarder can make a local decision based on the quality of the edges to each of its neighbors. Since a path is composed of edges, ensuring a high quality for each individual edge can also lead to formation of a path with high quality. In this case the determination of the optimal next hop requires sorting the utilities corresponding to each edge and has a complexity of $O(\log d)$. From a system perspective, we would like to derive the conditions under which there is a dominant forwarding and routing strategy for each forwarder. Ideally, we would like peers to participate in the forwarding process and route the payload to the best quality neighbor. We next show the conditions under which forwarding can be induced and also show the existence of a dominant routing strategy for good nodes.

Proposition 2. *If we assume a constant participation cost* C^p *and a constant fowarding cost* C^t *for all forwarders, then the condition* $P_f > \frac{C^p N}{Lk} + C^t$ *can induce peers to participate in forwarding.*

Proof. Assuming that any node is equally likely to be selected on the forwarding path, the probability that a forwarder X is selected for atleast one forwarding instance is $1 - (1 - \frac{1}{N})^{Lk} \approx \frac{Lk}{N}$. Therefore its expected payoff $E[Payoff] = \frac{Lk}{N}\{z(P_f - C^t) - C^p\} + (1 - \frac{Lk}{N})(-C^p)$. Here z is the total number of forwarding instances for the node. To induce the ”to participate in forwarding, we require that $E[Payoff] > 0$. Therefore $P_f > \frac{C^p N}{Lk} + C^t$. □

Proposition 3. *If* $P_f > (C_i^p + C_i^t)$*, forwarding is a dominant strategy for the forwarding stage.*

Proof. A non participating peer has a payoff of zero. However, if $P_f > (C_i^p + C^t(i,j))$, the payoff to a forwarder is > 0 irrespective of the forwarding decisions made by other peers. Thus forwarding is a dominant strategy for the forwarding stage. □

Proposition 3 gives us a possible condition which can induce the availability of nodes. Note that although in our model forwarding and routing decisions are decoupled, forwarding decisions influence future routing decisions. For example, if a peer i participates in the forwarding path

	$\tau = 0.5$	$\tau = 1$	$\tau = 2$	$\tau = 4$
f=0.1	409	390	391	456
f=0.5	299	298	332	306
f=0.9	85	91	72	122
Mean	296	303	301	360

Table 5.2. Routing efficiency for utility model I.

for connection π^k, it is highly likely that it will be selected for future connections. This in turn increases i's payoff and induces it to be available for future connections.

5.3.4.3 Utility Model II

We next consider an utility model whereby the utility is proportional to the quality of the path from i to the responder R. The intuition is that i can select a neighbor corresponding to a high quality path from i to R. Here $q(\pi(i, j, R))$ represents the quality of the path from i to R which goes through j.

$$U_i(j) = P_f + q(\pi(i, j, R))P_r - (C_i^p + C^t(i, j))$$

The path formation can be modeled as a L stage game for a path of length L such that at each stage only one player makes a move. We also define a history information at stage l. The history information corresponds to the position of the forwarder on the path and the identity of the predecessor. A subgame perfect nash equilibria is then a strategy profile SP such that each subgame $G_l \ \forall l = 1, 2 \cdots, L$ is a nash equilibrium. The equilibrium strategy profile (S_i^*, S_{-i}^*) can be derived using backward induction. We again refer the reader to the technical report [66] for a detailed discussion of this utility model.

5.4 Experiments

We analyze the effect of forwarding and routing benefits on the path quality and study the effect of malicious nodes on the path equilibrium and how it affects the payoff of good nodes. We use a discrete event simulator to perform the various experiments. For simulation simplicity, we use a small network size of $N = 40$ to study the effect of the utility models. A poisson process is

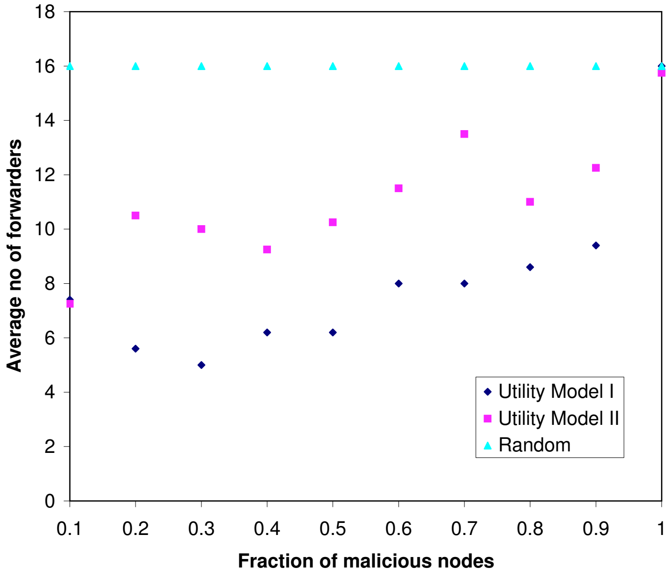

Figure 5.4. Comparison of the quality of a set of recurring connections between I and R using different routing strategies.

used to simulate the joining of nodes and each node randomly selects d nodes as its neigbors (unless otherwise specified, d is selected as 5 in our experiments). A set of nodes are randomly selected as Initiators and Responders. A (Initiator,Responder) pair is then randomly selected as the end points of an anonymous message transmission. The number of maximum transmissions for the same (I, R) pair is controlled using a parameter $max - connections$ in our simulations. A typical simulation setup involves 100 (I, R) pairs and a total of 2000 message transmissions, for an average of 20 communications rounds for a single (I, R) pair. The forwarding benefit, P_f for a (I, R) pair is randomly selected from the range $[50, 100]$ (since we did not have any particular application in mind, these values are arbitrary, however we believe they are reasonable to study the effect of benefits through simulations) and τ is selected from the set $(0.5, 1, 2, 4)$. Unless otherwise specified, the weights w_s and w_a are chosen as 0.5 and 0.5 respectively. We model the transmission cost between two peers as being proportional to the communication bandwidth between them. The session time of peers is modeled using a Pareto distribution and the median session time is set as 60 mins in accordance with the analysis done in [79]. A certain fraction f of nodes are selected as adversaries and an adversary's routing strategy is modeled as random routing.

Impact of malicious nodes on the payoff for good nodes Adversarial or malicious nodes randomly select the next hops for forwarding the packets which increases the size of the forwarder set for a set of connections between I and R. Consequently, the expected payoff for good nodes can decrease because the routing benefit gets shared by a large number of nodes and this can weaken their incentive to cooperate. We study this effect through simulations. We also analyze the effect of the size of forwarding and routing benefits and τ (ratio between routing and forwarding benefits) on the payoff to a good node. We use *routing efficiency* (ratio of average payoff and average number of forwarders) as a metric to quantify the effectiveness of the routing strategy of forwarders for a given value of τ. Note that a high value of *routing efficiency* is aligned with the system objective of inducing forwarders to make routing decisions so as to minimize the size of the forwarder set. Figure 5.2 shows that a high value of τ tends to increase the routing efficiency.

5.5 System Implementation Issues

In this section, we discuss various issues we have considered in design the whole system based on the proposed incentive mechanism. The first part is related to maintaining the anonymity in the mechanism itself; and the second part is about general implementation issues.

5.5.1 Maintaining Anonymity in the System Implementation

Incentives for peer availability. The proposed incentive mechanism induces nodes to select highly available next hops. Thus it is possible that malicious nodes become highly available and wait for paths to be reformed through them. This is especially effective at the initial stages when the first connections between (I, R) pairs is set up. Thus a malicious node can be selected as a next hop because of its high availability and due to history information stored at forwarders, it will occur in future connections. We use the following modification to thwart these types of **availability attacks**. If there is no history information at a node about previous connections (this can happen when it initially joins the system or after it has flushed its history table), then it should randomly select a neighbor as the next hop and therefore no preference is given to availability. With the passage of time, more and more connections would be set up which in turn would induce the availability of peers and therefore malicious peers would not have an added advantage of getting selected as forwarders over good nodes.

Collecting availability and cost information. Each peer collects availability and cost of transmission information about neighbors and therefore this results in additional traffic being generated. However this is not vulnerable to traffic analysis attacks [70] because of the following observations. The collection of availability information is periodic and each node (including the initiator and responder) executes it. Moreover since there is no correlation between the timing of collection of availability information and actual forwarding of anonymous traffic, the incentive based forwarding mechanism does not make the system any more vulnerable to timing attacks [73] than any non-incentive based forwarding scheme.

Connection identifier and history information about previous connections. Each peer stores history information for anonymous connections which pass through it. As described before, history information for a connection is tagged with its connection identifier. This raises the question that this connection identifier makes it simple for adversaries to correlate between the same (I, R) pair. However, the premise of an intersection attack is based on the assumption that there is a session identifying information available to the malicious entities in the transmitted packets [98] . Therefore the use of a connection identifier in our mechanism does not make the system any more weaker. However if a peer is compromised, the history table can give valuable information about connections passing through the peer. This can be avoided if each peer encrypts the contents of the history table using a secret key and physically protects this key using tamper proof hardware.

5.5.2 Other System Implementation Issues

Membership information. We assume a system containing N nodes which participate in requesting anonymous communications and forwarding traffic for other nodes. The membership information about nodes participating in the system can be maintained either in a centralized or in a distributed manner. We first compare the two approaches and then outline a distributed hash table based mechanism for maintaining membership information. In a centralized membership management system, a central server manages and updates information about existing members. It is simpler and easier to manage than a distributed approach. Moreover in such systems the membership information of the nodes participating in the system does not become disjoint. The disadvantage is that the central server can indulge in malicious reporting of membership information and can therefore make a “good” node forward its traffic to a “bad” node. Therefore for proper functionality of the system, the central server must be a trusted entity and therefore it becomes a bottleneck. On the other hand a distributed approach is better suited for systems in which information about only a few other nodes is required. In our model, d is a small fraction of N and therefore each node participating in the system only needs membership information about a small fraction of other nodes participating in the system. We can use a distributed hash table based approach for maintaining membership information. Note that in this case the DHT (e.g. Chord) is used only for maintaining membership information and not for object lookup. Therefore the routing decisions is influenced by the incentive mechanism and not by the chord routing structure. For example in Chord [91], each node maintains information about only $\log(N)$ other nodes in its finger table. These fingers can then be used as the forwarding neighbors. A node can then select $d(\leq \log(N))$ nodes as its forwarding neighbors. Therefore this ensures that a network overlay can be created with an uniform and bounded nodal degree of $d(\leq \log(N)$. Since the number of fingers increases with network size N, the maximum possible value of d also increases. Since the point at which a node joins the hash table is uniformly random (due to consistent hashing), the neighbor set selection is also uniformly random [5]. Moreover, the probability of network partitioning in Chord is very low and therefore this ensures that a forwarding path from I to R does not get restricted to a partitioned subset of nodes. Figure 5.5 shows the components of the system.

Maintaining history information and other issues. We assume that a communication channel set up from I to R has a unique identifier. For example, a one-way hash function $h(.)$ can be used and therefore connection identifier = $h(I, R)$. Note that from the connection identifier, it is not possible for the intermediate nodes to deduce the identity of I. As described earlier, each node maintains a history list of connections for which it has acted as a forwarder. Thus if a node

[5] We assume that the chord id can be used as the membership id in this case

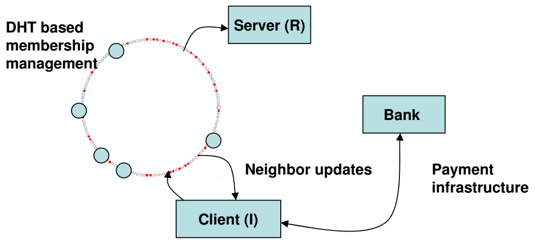

Figure 5.5. System model.

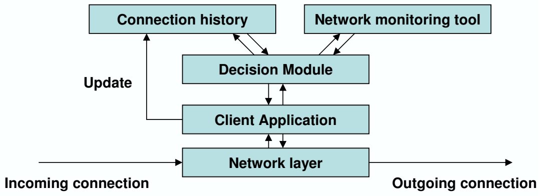

Figure 5.6. System architecture overview.

participates in a forwarding path for a given path identifier, it stores the previous and next hop information in the history list. Since theoretically a forwarder can lie on $O(N^2)$ connections, the number of entries in the history list can explode. Each node therefore maintains a fixed number of entries and uses a LRU replacement policy. Thus history information about the most recent connections is stored in the history list. Finally, the cost of forwarding the payload to a neighbor also influences the forwarding decision. The communication cost to a neighbor depends on the type of application for which the system is used. For example, in peer-to-peer file sharing, the bandwidth available to the neighbor dominates the communication cost (we refer the reader to the cost model presented in section 5.3.4.1. Existing network probing tools can be used for bandwidth estimation to arbitrary nodes in the overlay. Figure 5.6 shows the routing and forwarding module used at each intermediate node. For any incoming connection that has to be forwarded, the decision module uses information from history list and network monitoring tool(s) to identify the next hop (from its neighbor set) on the forwarding path.

Payment infrastructure. Here we outline a possible implementation of a payment mechanism. Our payment infrastructure consists of a central bank M for monetary transactions. The bank issues certificates for different amounts as requested by the peers for setting up anonymous communication channels to other peers. We assume that all communications between initiators and M take place through a trusted proxy which hides the identity of the initiator and use a pseudonym based approach [60] for authentication between I and the bank. When the initiator wants to set up a forwarding path to the responder, it embeds a certificate for an amount equal to P_f along with τ with the payload and sends it to the first forwarder. Note that the identities of the intermediate nodes is not known to the initiator at the time of setting up the path. Consequently a certificate for the routing benefit cannot be included with the payload without verifying the path formed from the initiator to responder. We therefore use a path verification mechanism in our scheme. The forwarders redeem the forwarding benefit by presenting the certificate corresponding to the forwarding benefit (along with a proof-of-forwarding) to the bank. The payment corresponding to the routing benefit is made to the intermediate forwarders after verification of forwarding and routing. We assume the existence of a Public key infrastructure for issuing public-private keys to the peers. Figure 5.7 shows the lifecycle of the forwarding and the payment mechanism. Due to space limitations, we refer the reader to the technical report [66] for a detailed description of the payment infrastructure.

Cryptographic operations in route formation and route verification. To initiate an anonymous connection with R, I selects the first forwarder from its set of neighbors and forwards the payload along with the certificate for amount P_f. Note that unlike payment mechanisms for onion-routing [35], the identity of the forwarders (except the first) is not known to the initiator and therefore the certificate is not bound to any particular identity. The message intended for R is encrypted with its public key. I also creates a public session key $E_{\{session\}}$ and includes it with the payload. The corresponding private key is used by I to regenerate the path information before it makes payments to the forwarders. Each forwarder uses the utility function to decide on the next hop and repeats the forwarding process until the payload reaches R. For example, in figure 5.7, I will send the following to A.

$$I \rightarrow A : \{RREQ, \{message\}_{E_R}, n, P_f, \tau, C_f\{P_f\}_{sign(I)}, E_{session}, R\}$$

RREQ identifies it as a request message and n is a random nonce. $C_f\{P_f\}_{sign(I)}$ is a certificate for the amount P_f which is signed by I. Once the payload reaches R, it decrypts the message using its private key. It then sends back a confirmation (receipt) through the reverse path as follows:

$$R \; \underrightarrow{\text{reverse-path}} : \{\text{RREP}, n, \{hash(message)\}_{D_R}, \{\text{previous hop}\}_{E_{session}}\}$$

$RREP$ signifies a reply message and n is the corresponding connection identifier. The hash of the message is used for integrity checking and is encrypted by R using its private key. This assures I that R has received the correct message because only R can encrypt it using its private key. Finally, to send the path information back to I, R also includes the previous-hop identity, which is encrypted with the public (session) key sent by I. Each intermediate node repeats the same forwarding process through the reverse path. When the next forwarder (in reverse path) receives this, it adds its own-id [6], its previous hop and next-hop information, all encrypted with the public session key. For example, in figure 5.7, the path information sent by A to I would be $\{\{A, I, B\}_{E_{session}}, \{B, A, C\}_{E_{session}}, \{C, B, R\}_{E_{session}}, \{R, C\}_{E_{session}}\}$. After receiving the receipt and path information, I can recreate the forwarding path and also know the identities of the intermediate forwarders. Note that the use of a private-public key ensures that only I can recreate the path information. We again refer the reader to the technical report [66] for details.

5.6 Payment Infrastructure

In this section we first outline a possible implementation of a payment mechanism and then present the design details along with the different cryptographic operations that are required. The payment infrastructure incorporates three main components; a central bank for issuing payment certificates, a path verification mechanism and finally the payment mechanism. Since our overall objective is to enhance the anonymity of communications, we must ensure that the payment mechanism does not compromise the identity of the initiator. We therefore incorporate the following design principles into our payment mechanism; a) The number of communications involving I should be minimal (so that traffic analysis attacks cannot be used to identify I) and b) Any centralized entity that is used should not be able to deduce either the identity of I or path information between I and R. We assume the existence of a PKI which is required for the cryptographic operations.

We assume the existence of a central bank M for monetary transactions. The bank issues certificates for different amounts as requested by the peers for setting up anonymous communication channels to other peers. We assume that all communications between initiators and M take place through a trusted proxy which hides the identity of the initiator. When the initiator wants to set

[6] Without revealing its actual identity, an intermediate forwarder can include its pseudonym

up a forwarding path to the responder, it embeds a certificate for an amount equal to P_{fo} along with τ with the payload and sends it to the first forwarder. Note that the identities of the intermediate nodes is not known to the initiator at the time of setting up the path. Consequently a certificate for the routing benefit cannot be included with the payload without verifying the path formed from the initiator to responder. We therefore use a path verification mechanism in our scheme. Finally, payment is made to the intermediate forwarders after verification of forwarding and routing. Figure 5.7 shows the lifecycle of the forwarding and the payment mechanism. The following sections describe in details the different steps of the lifecycle.

1. **Buying certificates from bank**

Before I can set up an anonymous connection with R, it has to buy a certificate for an amount P_{fo} from the bank. This amount corresponds to the benefit of forwarding for each intermediate forwarder. I can create a certificate and then send it to the bank for its signature or it can request the bank for the certificate (Steps 1 and 2). In either case, the bank needs some kind of identity information about I so that it can withdraw the desired amount from I's account. We use a pseudonym based approach [60] to ensure privacy of I during the transaction. A pseudonym is an identifier which cannot be directly linked to the actual identity of the initiator but can still be used for accountability. Thus I includes its pseudonym and the account number in the request. Finally after verifying the account information and ensuring the availability of the requested amount P_{fo}, M signs the certificate using some kind of blind signature scheme [22] (similar to the scheme presented in [35]) and sends it back to I.

2. **Cryptographic operations in route formation**

To initiate an anonymous connection with R, I selects the first forwarder from its set of neighbors and forwards the payload along with the certificate for amount P_{fo}. Note that unlike payment mechanisms for onion-routing [35], the identity of the forwarders (except the first) is not known to the initiator and therefore the certificate is not bound to any particular identity.

$$I \rightarrow F_1 : \{RREQ, \{message\}_{E_R}, n, P_{fo}, \tau, C_f\{P_{fo}\}_{sign(I)}, E_{session}, R\}$$

$RREQ$ identifies that it is a request message. The message intended for R is encrypted with its public key. n is a random identifier for the connection. $C_f\{P_{fo}\}_{sign(I)}$ is a certificate for the amount

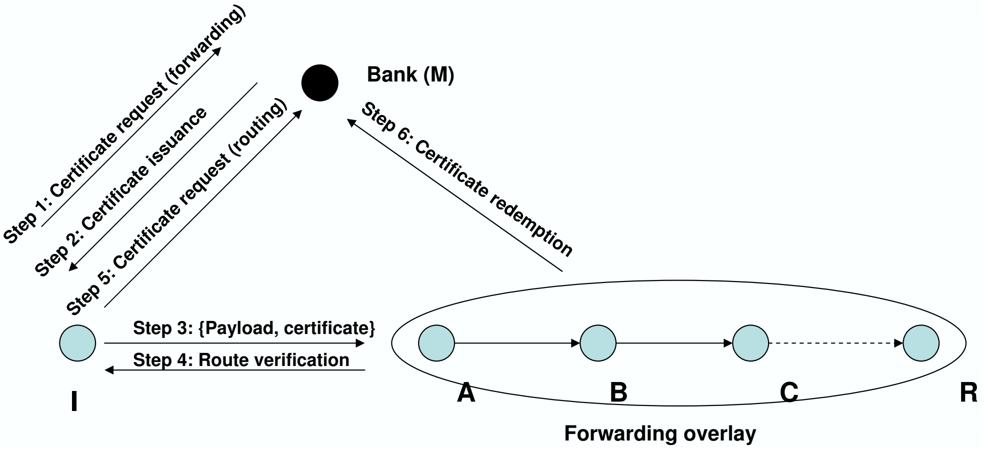
Bank (M)
Step 1: Certificate request (forwarding)
Step 2: Certificate issuance
Step 5: Certificate request (routing)
Step 6: Certificate redemption
Step 3: {Payload, certificate}
Step 4: Route verification
I
A
B
C
R
Forwarding overlay

Figure 5.7. Payment mechanism

P_{fo} which is signed by I. This is required to ensure non-repudiation by I before it makes payments to the forwarders. Note that the signature must not compromise the identity of I. Therefore a signature generated by I using its public key cannot be used. This is because if the number of public keys in the system is very less, it becomes very easy for the adversary to identify I from its signature (e.g. using a brute force attack). So an anonymous digital signature scheme [102] can be used. $E_{\{session\}}$ is a public key created by I for the session. The corresponding private key is used by I to regenerate the path information before it makes payments to the forwarders (details provided in Route regeneration section). Each forwarder uses the utility function to decide on the next hop and repeats the same process. Thus

$$F_i \rightarrow F_{i+1} : \{RREQ, \{message\}_{E_R}, n, P_{fo}, \tau, C_f\{P_{fo}\}_{sign(I)}, E_{session}, R\}$$

Each forwarder stores the previous hop and next hop information and also a copy of the certificate. These are required during the route regeneration and payment phases respectively as described later.

3. **Route regeneration by** I

Payment can be made by I only after it gets verification of forwarding by the intermediate nodes. As has been described in the section on path formation, I generates a private-public key for the session and includes the public key in the payload. Thus each intermediate node has access to the public key sent by I. Note that the session public key does not release any information about the identity of I. Once the payload reaches R, it decrypts the message using its private key. It then sends back a confirmation (receipt) through the reverse path as follows:

$$R\ \underrightarrow{reverse - path} : \{RREP, p, \{hash(message)\}_{D_R}, \{\text{previous hop}\}_{E_{session}}\}$$

$RREP$ signifies a reply message and n is the corresponding connection identifier. The hash of the message is used for integrity checking and is encrypted by R using its private key. This assures I that R has received the correct message because only R can encrypt it using its private key. Finally, to send the path information back to I, R also includes the previous-hop identity, which is encrypted with the public (session) key sent by I. Each intermediate node repeats the same forwarding process through the reverse path similar to the following:

$$F_i \to F_{i-1} : \{n, \{hash(message)\}_{D_R}, \{path\}_{E_{session}}\}$$

When the next forwarder (in reverse path) receives this, it adds its own-id [7], its previous hop and next-hop information, all encrypted with the public session key. Thus the entity $path$ behaves like an onion. For example, in figure 5.7, the path information sent by A to I would be $\{\{A, I, B\}_{E_{session}}, \{B, A, C\}_{E_{session}}, \{C, B, R\}_{E_{session}}, \{R, C\}_{E_{session}}\}$. After receiving the receipt and path information, I can recreate the forwarding path and also know the identities of the intermediate forwarders. Note that the use of a private-public key ensures that only I can recreate the path information.

4. **Payments to forwarders**

Using the path information, I calculates the appropriate routing benefits for each forwarder based on the selectivity of the path. For each forwarder, it requests a certificate C_r for the appropriate amount from the bank and includes its own pseudonym and the forwarders pseudonym. Finally each forwarder redeems the certificate C_r by submitting its credential to the bank. Since it also possesses C_f (from the RREQ phase), it redeems the certificate by presenting it to the bank.

[7] Without revealing its actual identity, an intermediate forwarder can include its pseudonym

5.7 Summary

We have presented an incentive mechanism to increase peer availability and reduce path reformations for forwarding based anonymity systems. We propose two utility models and use game theory to evaluate forwarding and routing strategies of intermediate peers. We also show the appropriateness of these utility models for forwarding based anonymity systems and evaluate their effectiveness in aligning the forwarding and routing strategies of peers. We compare the effectiveness of routing under these models with random routing. Our simulation results show that the incentive mechanism is quite effective both under churn and presence of malicious adversaries. We also outline a payment mechanism and show that in trying to increase the system anonymity, the payment mechanism does not actually decrease it. In designing the incentive mechanism, we have tried to address the two issues of compliance and availability in an anonymity system [35] within the same framework.

CHAPTER 6. Anonymity in Structured Peer to Peer Networks

6.1 Structured Peer to Peer Networks

In a structured overlay network, connections are made between nodes according to some algorithm. Most of the structured networks share common design features; a large identifier space of the order of $\mathbb{Z}_{2^{128}}$ or $\mathbb{Z}_{2^{160}}$, the nodes are partitioned on the identifier space and routing structure is based on butterfly, hybercube or ring like structures and in most cases the number of hops to the destination is of the order of $O(\log N)$. Data is stored on the nodes by hashing on content or data identifiers and then assigning the data to a node which is responsible for that identifier space. Therefore, these networks are also called Distributed Hash Table (DHT) based networks.

An important security issue in a DHT-based P2P storage system for file sharing is to provide recipient anonymity to the storage nodes in the system. In a P2P storage system using DHT, the DHT is used to return the network address of the storage node for a given document key. For example, in DHTs like Chord [91] and CAN [64] the query for a key returns the IP address of the node storing the documents that match that key. AChord [44] added anonymity feature to Chord by modifying “lookup-by-address” to “lookup-by-value”. However, AChord is still vulnerable to attacks to the index nodes[1]: If a hacker breaks into an index node, information stored in the node’s routing table will be leaked.

6.2 Effect of Design on Recipient Anonymity

The risk of leaking information from DHT routing tables has not been well analyzed in current research. DHTs were primarily designed for routing efficiency and scalability. For example, Chord uses a structured routing geometry such that the search takes up to $O(\log N)$ steps to com-

[1] It is possible that a node plays the role of an index node and a storage node simultaneously.

plete a query. The side effect is that the routing tables contain a substantial amount of information about other nodes in the system. This information can be used by adversaries, if the system is breached, to compromise the anonymity of storage nodes, i.e. the recipient anonymity. A recently proposed design called Neblo [25] uses imprecise routing to enhance recipient anonymity. Agyaat [87] proposes the use of unstructured clouds on top of a structured overlay to hide recipient anonymity (Table 6.1 shows the previous attempts at enhancing anonymity for structured networks). However, no research has been done to quantify and compare the information leak from routing tables in different DHT designs.

In this chapter, we outline an analytical model to analyze, quantify and compare the leak of privacy from the routing tables in existing DHT designs. There are two primary factors that influence the amount of information contained in a routing table: the type of routing geometry and the size of routing table. Our model gives valuable insight into how different routing geometries will affect the recipient anonymity. It uses entropy to calculate the amount of information leak. Based on this model, we compare information leak in different DHTs when index nodes are breached. We have compared the use of Chord [91], CAN [64], Kademlia [55], Pastry [78] in building a P2P storage system with the same routing complexity, i.e. the number of hops in the routing. Our analytical results show that for the same routing complexity, ring-based DHT (Chord) has the minimum information leak. The general trend is that the state information stored in routing tables increases with routing optimizations, thereby resulting in significant information leak. We observe that Kademlia (XOR routing) has a significant amount of information leak for small overlay sizes. Pastrys (hybrid routing) performance is quite close to that of the ring structure. The hypercube-based routing in CAN has an important side-effect, i.e. of localizing the information loss. We also analyse the effect of routing table size on leak of information. We believe that our preliminary findings can help us better understand the effect of routing geometry on the state information stored in routing tables which can lead to the development of DHT designs with an optimal balance between routing efficiency and information leak.

Privacy-enhanced DHT system	Type of information leak addressed	Method used
AChord	Query reply	Lookup by value
Neblo	Routing table	Imprecision in routing tables
Agyaat	Query reply	Unstructured cloud over a structured overlay

Table 6.1. Proposed privacy-preserving DHT designs.

6.3 Quantifying Information Leak from Routing Tables

Structured overlays like Chord [91], CAN [64], Pastry [78], Kademlia [55], and Viceroy [54] use distributed hashing to store keys at nodes. A node in such a DHT is identified by a tuple <IP address, Identifier>. The information contained in the tuple can be used by an adversary to compromise recipient (storage) anonymity. In the context of a distributed hash table, recipient anonymity is broken when the adversary can generate the mapping between a node's IP address and its identifier range. The objective of the adversary is to generate a map of the system. Such a mapping between a node's IP address and identifier range can be generated by compromising a sufficient number of routing tables [2]. The amount of information stored in routing tables is influenced by the type of routing geometry used in the DHT and also the size of the routing tables. Thus, our objective is to compare different DHT designs with respect to recipient anonymity through a common analytical framework and suggest improved design considerations. The important assumption here is that the lookup for a key is done through "lookup of data" and not "lookup of address" (See AChord [44]).

6.3.1 Information Stored in Routing Tables

A routing table of a node in a DHT either stores IP addresses of neighbors or mapping between IP addresses and identifier range. The information content of routing tables is directly related to routing efficiency. Consider the case of flooding (Gnutella which is an unstructured overlay). In Gnutella, each node only maintains information about its overlay neighbors and there is no

[2] In this context, an adversary may not necessarily compromise a node to get access to its routing table. An adversary can simply occupy a certain position on the identifier space and use information contained in its own routing tables to compromise recipient anonymity

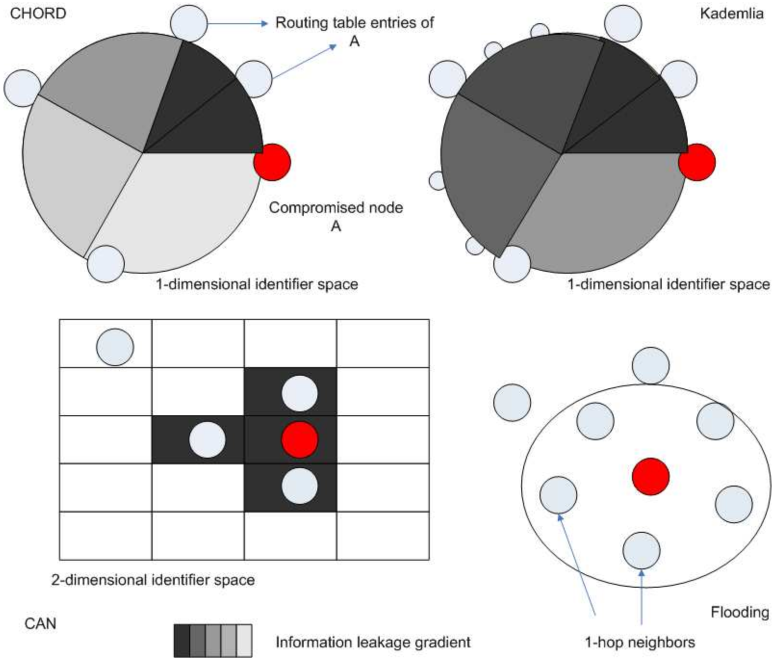
CHORD
Routing table entries of A
Compromised node A
1-dimensional identifier space
Kademlia
1-dimensional identifier space
2-dimensional identifier space
CAN
Information leakage gradient
Flooding
1-hop neighbors

Figure 6.1. Information leak from compromised routing tables. The gradient shows the degree of information loss, darker shades representing high information loss. (a) Chord: The amount of information loss decreases with the distance of fingers; (b) CAN: Complete information loss about the identifier ranges of neighbors; (c) Kademlia: It exhibits similar properties as Chord; however the replication of data through the use of a replication factor k requires maintenance of larger routing state information and thereby more leak of information; (d) Flooding (Gnutella): since no mapping information is maintained, negligible information about the keys stored at the neighbors is released; Pastry, although not shown here exhibits similar properties as Chord and Kademlia.

mapping between a node and the keys that it stores. Thus, compromising a node only reveals information about keys stored at that node and nothing about the neighbors. However, in a DHT mapping information about neighbors is also stored for increased routing efficiency. While this leads to improved routing efficiency ($O(\log N)$ as compared to $O(N)$ in gnutella), it also makes the DHTs vulnerable to leak of recipient anonymity (Figure 6.1 compares the information leak from routing tables for different DHT designs). We consider the effect of the following factors on the amount of information stored in routing tables: (a) routing geometry and (b) size of routing tables.

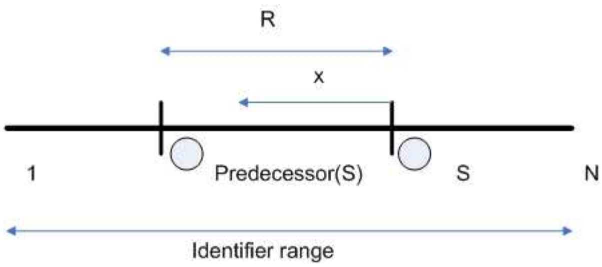

Figure 6.2. Identifier range of a node.

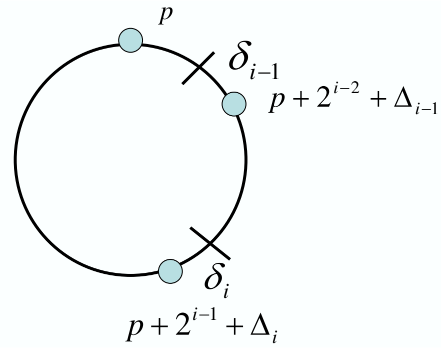

Figure 6.3. Identifier range of ith finger on Chordal Ring.

6.3.2 Information-Theoretic Framework for Analyzing Leak of Recipient Anonymity

We use the information-theoretic metric of *entropy* [83] to evaluate different DHT designs by calculating the leak of information in each design. Entropy is a measure of "randomness" in available information. Let X be the random variable which represents the identifer range of a node as observed by an adversary when a routing table is compromised. Let this observation correspond to the event ω. Figure 6.2 shows the identifier space of a DHT and the identifier range of node S corresponding to observation ω. We assume that x can take any value in R with equal probability. We next highlight the different elements of the information-theoretic framework.

The entropy of random variable X is given as,

$$H(X) = -\sum_{x \in R} \Pr(X = x) \log \Pr(X = x)$$

- Apriori Entropy: It is the entropy before any routing table has been compromised or in other words, before the adversary has made any observation about the mapping between a node and its identifier range. From the adversary's perspective, the first node can take any of N positions, the second node can take any of the remaining $N-1$ positions and so on and so forth (where N is the size of the overlay). For large N, the apriori entropy can be approximated as

$$H(X)^{\text{system}}_{\text{apriori}} = N \log N$$

- Aposteriori Entropy: When a routing table is compromised (corresponding to observation

ω), the information stored in the routing table can be used by the adversary (coalition of adversaries) to generate a mapping between node addresses and their identifier ranges. Thus, the aposteriori entropy corresponds to the entropy of the system after a routing table has been compromised. Higher the number of compromised routing tables, lower will be the system entropy (reduction in the randomness of the system).

- Information Loss

  Loss = Apriori Entropy (before any routing table has been compromised) - Aposteriori Entropy (after one or more routing tables have been compromised)

- Degree of Privacy [3]: To calculate the degree of privacy, we use the definition proposed in [90].

$$d(A) = \frac{H(X)^{\text{system}}_{\text{aposteriori}}}{H(X)^{\text{system}}_{\text{apriori}}}$$

6.4 Comparison of Existing DHTs

In this section, we compare how DHTs with different routing geometry perform in the face of anonymity attacks. We analyze how the routing geometry influences leak of information from compromised routing tables thereby affecting the privacy of the storage nodes. The routing geometry influences the amount of state information that is maintained in routing tables. Consider Chord which achieves logarithmic routing efficiency by maintaining $\log N$ entities in its routing table. However, two compromised routing tables might have some intersection in their routing tables. In comparison, 1 and 2-dimensional CAN maintain constant state information and the information loss is also localized (we discuss this in subsequent sections). However, to maintain the same routing efficiency as Chord, the state information in each routing table (d) is $\log N/2$. In DHTs like Kademlia the flexibility of routing improves routing efficiency (e.g. latency) but leads to an increased leak of information. In all these cases the state information maintained in routing tables increases with size. However, in systems like Viceroy, a constant number of entries are maintained in the routing tables and therefore information loss is constant. In contrast, flooding-based approaches are more secure because the mapping between neighbors and their overlay addresses

[3] We use the terms anonymity and privacy interchangeably

	CAN	Chord	Kademlia	Pastry	Gnutella
Size of adversarial coalition	$O(\frac{N}{\log^2 N})$	$O(\frac{N}{\log N})$	$O(\frac{N}{k\log N})$	$O(\frac{N}{\log_{2^b} N})$	$O(N)$

Table 6.2. Size of adversary set which can map the overlay of size N. The replication factor k affects the leak of information in Kademlia.

is not stored. We now evaluate different DHT designs and quantify the information loss from compromised routing tables. Subsequently, we derive expressions for the degree of privacy. We consider the following routing geometries: a) Ring, b) Hypercube, c) XOR, d) Hybrid.

6.4.1 Ring-based DHT

In a ring-based DHT design, nodes lie on a one-dimensional identifier space on which the distance between two identifiers is the clockwise distance between them. The Chord DHT represents a ring-based design. In Chord, each node stores information about $\log(N)$ fingers, such that if a node has identifier p, its i^{th} finger is the node closest to $p + 2^i$ on the identifier space. Moreover, the range information stored about a finger decreases as its distance from p increases.

Consider Chord with $O(\log N)$ finger table entries. When a finger table is compromised, information about $O(\log N)$ IP addresses are leaked to the adversary. Let the compromised node have an identifier p. Consider its i^{th} finger. We need to calculate the lower and upper bound of its address interval. We know that the i^{th} finger succeeds p by at least 2^{i-1} on the identifier circle. Therefore, the address interval of the i^{th} finger (R_i) is (p, id of i^{th} finger]. This address interval can be reduced using the $(i-1)^{th}$ finger to (id of $(i-1)^{th}$ finger, id of i^{th} finger]. In terms of p,

$$\begin{aligned} R_i &= |address - interval| \\ &= (p + 2^{i-1} + \Delta_i) - (p + 2^{i-2} + \Delta_{i-1}) \\ &= 2^{i-2} + (\Delta_i - \Delta_{i-1}) \end{aligned}$$

The aposteriori entropy corresponding to a single compromised routing table is then given as:

$$\begin{aligned} H(X)_{\text{aposteriori}} &= \sum_i \log R_i \\ &\leq \sum_i \log(2^{i-2} + \Delta_i - \Delta_{i-1}) \end{aligned}$$

Since the i^{th} finger is the first finger which exceeds p by atleast 2^{i-1}, there is no other node in Δ_i. Therefore, $R_i = (2^{i-2} + \Delta_i - \Delta_{i-1}) - \Delta_i = (id_i - id_{i-1})$

$$\begin{aligned} H(X)_{aposteriori} &\leq \sum_i \log(2^{i-2} - \Delta_{i-1}) \\ &= \sum_i \log 2^{i-2} \\ &= \sum_{3 \leq i \leq \log N} \log 2^{i-2} \\ &= \frac{1}{2}(\log N - 1)(\log N - 2) \end{aligned}$$

If c routing tables are compromised,

$$H(X)^{system}_{aposteriori} = (N - c\log N)\log N + \frac{c}{2}(\log N - 1)(\log N - 2)$$

Degree of anonymity can then be calculated as:

$$d(A) = \frac{(N - c\log N)\log N + \frac{c}{2}(\log N - 1)(\log N - 2)}{N \log N} \tag{6.1}$$

We observe that the information leak from a single compromised routing table is of $O(\log^2 N)$ and is influenced by the number of entries in the routing table.

Observation 1. *A coalition of* $O(\frac{N}{\log N})$ *adversaries can map the entire overlay. This can be derived by setting* $d(A) = 0$*.*

6.4.2 Hypercube-based DHT

The routing used in CAN resembles a hypercube geometry. A d-torus is partitioned among the nodes, such that each node owns a zone. In a d-dimensional coordinate space, two nodes are neighbors if their coordinate spans overlap along $d-1$ dimensions and abut along one dimension. Each node maintains a maximum of 2d neighbors. This information includes the IP address of the neighbor and its virtual coordinates. The virtual coordinates reveal the exact keyspace for which the neighbor is responsible. Thus, if a node is compromised, the exact identifier range of $(2d+1)$

nodes is revealed. Note that in CAN since each node maintains information about its neighbors which are close to it in the identifier space, the information loss from a compromised routing table is localized. Contrast this with Chord, where a compromised routing table can give information about distant nodes. We discuss this issue and its implications later.

Apriori Entropy: Using a similar analysis as Chord, a node is responsible for $\frac{1}{N}$ of the unit identifier volume. Therefore,

$$H(X)_{\text{apriori}} = \log N, \quad H(X)^{\text{system}}_{\text{apriori}} = N \log N$$

Aposteriori Entropy: We next derive the aposteriori entropy after c routing tables have been compromised. Note that in CAN, when a node is compromised, the exact information about the identifier space of the node and all its neighbors can be deciphered. This corresponds to zero entropy. Therefore,

$$H(X)^{\text{system}}_{\text{aposteriori}} = (N - c(2d+1)) \log N$$

Degree of privacy is then given by

$$\begin{aligned} d(A) &\geq \frac{H(X)^{\text{system}}_{\text{aposteriori}}}{H(X)^{\text{system}}_{\text{apriori}}} \\ &= \frac{(N - c(2d+1)) \log N}{N \log N} \end{aligned} \tag{6.2}$$

Lemma 1. *For the same scaling properties, CAN is less robust to privacy attacks than Chord.*

Proof. To achieve the same scaling properties, $d = \frac{\log N}{2}$ in CAN. Therefore a coalition of $\frac{N}{\log^2 N}$ adversaries is sufficient to map the overlay. □

Table 6.2 shows the size of the adversarial coalition required for mapping the overlay for different DHT designs. The values can be easily obtained by setting $d(A) = 0$ in the respective equations and calculating for c.

6.4.3 DHT with XOR Routing

The routing in Kademlia is based on the concept of XOR distance: the distance between two nodes is the numeric value of the exclusive OR (XOR) of their identifiers. If the identifier space is represented by m bits, for each $0 \leq i < m$, every node stores a list of $< IPaddress, UDPport, NodeID >$ triples for nodes of distance between 2^i and 2^{i+1} from itself. These lists are called k-buckets. While on one hand this gives routing flexibility (for example in comparison to Chord), a compromised routing table gives more information about other nodes in the system. The leak of information increases with k.

Consider the i^{th} k-bucket. We assume that the identifier range of a node in any bucket is equally distributed among the nodes in that bucket. Using a Chord-like analysis,

$$\begin{aligned} H(X)_{\text{aposteriori}} &= \sum_i \log R_i \\ &\leq k \sum_i \log(2^{i-2}/k) \\ &= \frac{k}{2}(\log N - 1)(\log N - 2) \\ &\quad -k \log k(\log N - 3) \end{aligned}$$

Degree of privacy is then given by

$$\begin{aligned} d(A) &= \frac{1}{N \log N}((N - ck \log N) \log N \\ &\quad +\frac{ck}{2}(\log N - 1)(\log N - 2) \\ &\quad -ck \log k(\log N - 3)) \end{aligned} \tag{6.3}$$

6.4.4 DHT with Hybrid Routing

We use Pastry as an example of hybrid routing. Pastry uses both tree and ring based routing to search for keys. Node identifiers are regarded as both the leaves of a binary tree and as points on a 1-dimensional circle. Each node maintains a leaf-set, neighbor-set and a routing table. We

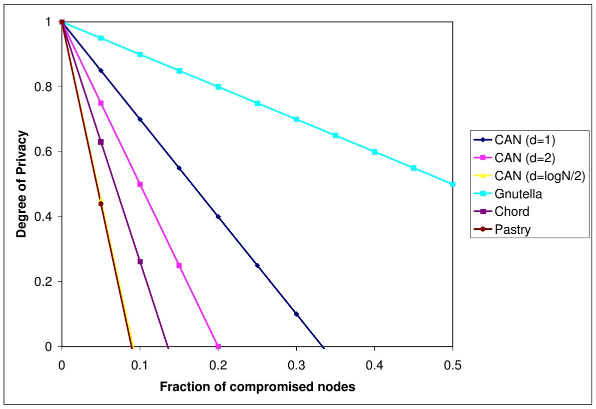

Figure 6.4. Leak of privacy for N=1000.

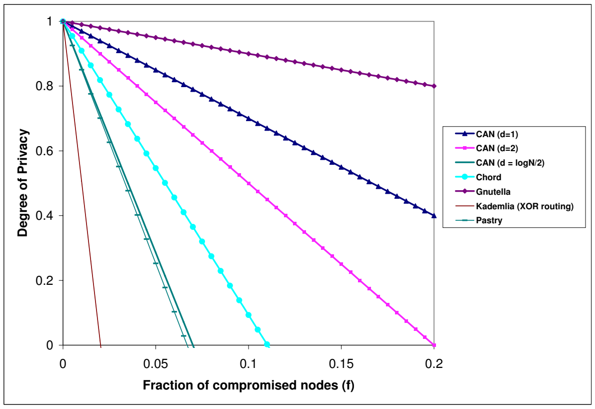

Figure 6.5. Leak of privacy for N=50000.

evaluate the range of an entry in the routing table as perceived by an adversary. Each row in a routing table has a maximum of $2^b - 1$ entries and there are a total of $\log_{2^b} N$ rows. Consider the i^{th} row. The range covered by the i^{th} row is $2^{b^{m-i}} - 1$. The range covered by the $(i-1)^{th}$ row is $2^{b^{m-(i-1)}}$. Therefore the effective identifier range covered by the i^{th} row is $2^{b^{m-(i-1)}}(2^b - 1)$. Since each row contains $2^b - 1$ entries, the effective identifier range corresponding to a single node in the routing table is $2^{b^{m-(i-1)}}$. We plug this range into entropy equation and evaluate information loss as outlined below.

$$\begin{aligned} H(X)_{aposteriori} &= \sum_i \log R_i \\ &\leq (2^b - 1) \sum_i \log(2^{b^{m-1}}) \\ &= (2^b - 1) b \log^2_{2^b} N/2 \end{aligned}$$

Degree of privacy is then given by

$$\begin{aligned} d(A) &= \frac{1}{N \log N}((N - c \log_{2^b} N(2^b - 1)) \log N \\ &+ \frac{c}{4}(2^b - 1) b \log^2_{2^b} N) \end{aligned} \tag{6.4}$$

Overlay Size	CAN (1-dim)	CAN (2-dim)	CAN ($d = \log N/2$)	Chord	Kademlia	Pastry	Gnutella
500	1	0.6	2	0.55	34.9	1.34	0.2
1000	0.5	0.3	1.1	0.28	18.8	0.75	0.1
10000	0.05	0.03	0.14	0.03	2.3	0.1	0.01
50000	0.01	0.006	0.03	0.005	0.5	0.02	0.002

Table 6.3. Percentage of information loss (measured in bits) when a routing table is compromised. A 100 % information loss corresponds to the case when the entire overlay can be mapped by the adversary (for kademlia, k=20 and for Pastry, b=2).

	CAN	Chord	Kademlia	Pastry	Gnutella	Viceroy
Size of routing table	$2d$	$\log N$	$k \log N$	$\log_{2^b} N$	Constant	Constant

Table 6.4. Size of routing table.

6.5 Analysis

Here we analyze the effect of routing geometry on the amount of information leak from compromised routing tables. We also compare DHTs based on the size of the routing table and how that affects leak of privacy. Finally we compare and contrast structured DHT designs with unstructured overlays which use flooding.

6.5.1 Routing Geometry

Figures 6.4 and 6.5 show the variation of degree of privacy with fraction of compromised nodes for different DHT designs. For the same scaling properties (in case of Distributed Hash Tables), Chord is the most robust against leak of information. In the case of 1- and 2-dimensional CAN, the decrease in privacy with fraction of compromised nodes is less than Chord. For 1- and 2-dimensional CAN, the information leak is a function of c only and is not dependent on N, since the size of the routing tables is constant at $2d$. However, 1- and 2-dimensional CAN take a larger number of routing steps to converge. On the other hand, to achieve the same scaling properties as Chord, $d = (\log N)/2$ and information leak increases with N. Observe that the plot is much steeper in case of CAN with $d = \log N/2$ as compared to Chord. Routing optimization in Kademlia is done through the maintenance of k buckets at each node. While this improves lookup latency, it requires the maintenance of a large amount of state information at each node. Therefore a compromised routing table leaks more information than Chord. For the same scaling properties, Pastry is the closest to Chord. However, for lookup optimization, Pastry maintains a leaf-set (besides the routing table) at each node. This leaf-set can leak information about an additional set of nodes in the system.

Table 6.3 shows the percent of information loss (in bits) when a routing table is compromised. We observe that Chord shows the maximum resilience to privacy leak for different overlay sizes. Observe that in the case of 1 and 2-dimensional CAN, the fraction of information loss $\propto \frac{1}{N}$ since the number of entries in the routing table is fixed (=2d). However, when $d = \log N/2$, the leak of privacy increases and is appreciably higher than Chord. In Pastry a routing table stores information about $(2^b - 1)\log_{2^b} N$ nodes. If $b = 1$, the state information maintained is same as that of Chord. Our analytical model shows that the privacy leak in that case is similar to that of Chord. However, the leak increases when the base is 4. In Kademlia the replication parameter k is typically set as 20. We observe that for small overlay sizes, the number of replicas has an adverse effect on the leak of information from routing tables. However, the information leak decreases with an increase in overlay size. The general trend is that DHT designs with routing optimizations tend to exhibit higher leak of information from compromised routing tables.

We also observe that the routing geometry of CAN leads to “localized” information loss when a routing table is compromised. By “localized” we mean that when a node is compromised, the routing table gives information only about neighbors which are close on the identifier space. Contrast this with Chord, in which the finger table stores information about distant nodes. The implication is that in CAN (as compared to other designs), compromised nodes in a certain region of the identifier space localize the information leak without affecting substantial portions of the overlay.

6.5.2 Routing Table Size

The size of the routing table affects the leak of information about the overlay. The routing geometry and routing optimizations influence the number of entries in the routing table (Table 6.4 shows the routing table size for different DHT designgs). We have observed that in all the aforementioned DHT designs the size of the routing table varies with the size of the overlay N. In contrast flooding-based approaches in unstructured overlays maintain constant state information (typically $3-8$ in gnutella). However, the important question to ask is can we have a DHT design which achieves logarithmic routing efficiency by maintaining constant state information. Viceroy [54], which emulates the butterfly network, achieves such efficiency. We did not include Viceroy in our analysis since we wanted to analyze DHTs which are based on a similar design principle. Each node maintains information about 7 other nodes in the overlay. As part of our future work, we plan to analyze the information leak from the Viceroy network.

6.5.3 Comparison with Unstructured Networks

In this section, we compare the information leak property of DHTs with that of flooding based search systems. We use the Gnutella [40] protocol as the basis for an unstructured routing geometry. In Gnutella and other flooding-based search systems, no state information corresponding to mapping between nodes and keys is maintained. Thus, each node only maintains information about its neighbors but is not aware of the keys that are stored at the neighbors. Any search query propagates through the unstructured network and eventually reaches the node responsible for the key. Therefore, if c nodes are compromised, the adversary can know only about the keys mapped to those compromised nodes.

The degree of privacy is then given by

$$d(A) = \frac{H(X)^{system}_{aposteriori}}{H(X)^{system}_{apriori}} \tag{6.5}$$

$$= \frac{(N-c)\log N}{N \log N} \tag{6.6}$$

Observe the bits of information leaked in the case of Gnutella (Table 6.3). Since each node maintains information only about its 1-hop neighbors and is blind with respect to the information stored in the neighbors, an adversarial coalition of size $O(N)$ is required to map the overlay. Thus, an unstructured overlay has very good privacy properties; however, it lies at the end of the routing efficiency spectrum.

6.6 Summary

We have presented an information-theoretic framework for evaluating the resilience of different DHT designs against leak of privacy. Our entropy-based analytical model helps us to quantify the leak of information from compromised routing tables. We analyze the effect of routing geometry, optimizations and route table size on the amount of information leak. Our analytical results show that for the same routing complexity, ring-based DHT (Chord) has the minimum information leak. The general trend is that the state information stored in routing tables increases with routing optimizations, thereby resulting in significant information leak. We observe that Kademlia (XOR routing) has a significant amount of information leak for small overlay sizes. Pastry's (hybrid routing) performance is quite close to that of the ring structure. The hypercube-based routing in CAN has an important side-effect, i.e. of localizing the information loss. We also analyse the effect of routing table size on leak of information. We believe that our preliminary findings can help better understand the effect of routing geometry on the state information stored in routing tables which can lead to development of DHT designs with an optimal balance between routing efficiency and information leak.

CHAPTER 7. SCUBE: Enforcing Location Privacy through Access Control

7.1 Vulnerability of Search Indexes in P2P File Sharing

Over the past few years, peer-to-peer (P2P) applications have become very popular with the most widespread application being file sharing. P2P based distributed storage systems use indexes to improve search performance [58, 47, 95]. While indexes improve search performance, they may make the distributed storage system more vulnerable to DoS attacks [89, 52]. This is because in a highly dynamic and open P2P system, search indexes may be stored at untrusted nodes which can be compromised by the adversary. Moreover, the location of these index nodes cannot be hidden from the adversary. Index-based search systems are therefore vulnerable to the following types of attacks:

1. File-server attacks: An index server node can be compromised by a malicious adversary and the adversary can then target specific file-servers because it knows the mapping between a file and it's location. This is also called Targeted File Attacks [89].

2. Search-infrastructure attacks: In such an attack, the malicious adversary tampers or destroys the indexes stored at the compromised node thereby decreasing the reliability of the search scheme.

File-server attacks may be partially prevented using an access-control mechanism where the indexes are encrypted using a cryptographic key and the key is shared with legal users. However, encryption alone cannot prevent Search-infrastructure attacks. The reason is that in distributed index-based search systems (for example in DHT based systems [91, 64]), there is a direct mapping between an index and its location (a distributed hash table allows a group of distributed hosts to collectively manage a mapping between keys and values through the use of distributed hashing). Therefore, the location of index nodes cannot be hidden from the adversary. Again cryptography can be used to enforce access control such that only legal users can generate the location of the indexes. This however does not ensure the availability of indexes in the face of random attacks by the adversary. Traditional approaches like replication can enhance availability. However the problem with replication in a security context is that even if a single replica is broken, confidentiality can be compromised.

In this chapter, we outline a **S**ecret **S**haring based **S**earch scheme which we call **SCUBE**. **SCUBE** uses Shamir's secret-sharing to split the location of a file (secret) into shares and generates file-identifier shares using obfuscation techniques. A file-identifier share and a location-share are then combined to create an index-share. A (t, s) secret-sharing scheme then requires atleast t of s shares to regenerate the index (location of a file). In comparison to traditional approaches like replication, secret sharing has better security guarantees and offers more flexibility. First, it makes it difficult for an adversary to correlate shares of the same index. Second, it offers the flexibility of selecting appropriate values of t and s, achieving a tradeoff between security strength and search performance.

7.2 Access Control using Secret-sharing and symmetric key cryptography

7.2.1 Shamir's Secret-sharing Scheme

The secret-sharing proposed by Shamir [82] is used to share a secret among a set of participants. A (t,s)-threshold scheme is a method of sharing a message M among a set of s participants such that any subset containing atleast t participants can construct the message. In the context of our fault-tolerant search scheme, the file location is a secret which is split into s shares such that any subset of atleast t shares is required to regenerate the location. Shamir's scheme uses Lagrange's polynomial interpolation on a field Z_p, where p is a prime. The dealer (content-provider in our case) generates a random polynomial of degree $t-1$:

$$f(x) \quad = \quad a_0 + a_1x + a_2x^2 + \cdots + a_{t-1}x^{t-1} \qquad (7.1)$$

This polynomial is constructed over a finite field Z_p and the coefficient is the secret (file-location). The value of p is public. The other coefficients are randomly selected by the content-provider and the location-shares are calculated as follows: $share_i = (x_i, f(x_i)), i = 1..s$. The secret can then be reconstructed as follows:

$$a_0 = f(0) = \sum_{i=1}^{t} y_i \prod_{j=1, j \neq i}^{t} \frac{-x_j}{x_i - x_j} (\text{mod } p) \tag{7.2}$$

The complexity of Lagrange interpolation is $O(t \log_2 t)$.

7.2.2 Adversary Model

We assume a threat model in which the adversary controls the actions of several adversary agents. An adversary agent controls the actions of a compromised node. Thus the adversary set consists of colluding adversary agents. We use terminology from [71] to describe the threat model. The adversary agents at the compromised nodes can tamper or destroy the file index shares hosted at these nodes (**Active/Internal adversary**). We also assume that the adversary can only compromise the index hosting nodes but cannot compromise communication mediums. On the other hand, the adversary can use packet sniffing on traffic originating at a good node and can therefore observe query packets (**Passive/External adversary**). We assume that the underlying network layer is secure and the adversary can only observe the query request packets that are addressed to it during a query lookup. Moreover, the adversary follows the Chord protocols correctly (Routing, Stabilization, Finger table updates and so on). The adversary can initiate two types of Denial of Service attacks. The adversary agents at different compromised nodes can tamper the file-index shares hosted at those nodes thereby breaking the search infrastructure. Moreover, by observing search queries and issuing bogus queries, an adversary can possibly reduce the size of the set of possible content providers for a particular file. This can result in targeted file attacks. We assume that a fraction f of a total of N nodes are compromised and therefore act as malicious nodes.

7.3 Protocol Overview

We first present a general overview of SCUBE. Each file in the storage overlay is associated with an unique identifier. Search protocols use keywords to search for documents. We assume that each file's identifier/keyword is unique which may be semantically attached to the content of the file. The index of each file is a tuple - {file-identifier, file-location}. The content-provider for a particular file first generates a **virtual address** using it's group key. The concept of virtual address has been used in MUTE [77] and is used for obfuscating the actual IP address. This virtual address is then split into s shares using Shamir's (t, s) secret-sharing scheme. Thus, even if an adversary manages to acquire atleast t shares, all it can generate is the virtual address. The virtual address concept therefore provides a second level of security. To generate the corresponding file-identifier shares, a keyed-hash function based **obfuscation technique** is used. This obfuscation of the file-identifier ensures that it is very difficult for an adversary or group of adversaries to correlate shares for the same file during a normal search operation. This is applicable to search schemes in which file identifiers are semantically attached to the file content. Each file-index share is then generated as {file-identifier share, file-location share} and inserted into the DHT (See figure 7.1 for the protocol steps). A searcher then generates the locations of atleast t untampered shares and generates the virtual address of the content provider. Finally the searcher generates the correct IP address of the content-provider by decrypting the virtual address using the group key. One of the basic requirements of SCUBE is that a searcher must possess the correct group key and must know the correct file-identifier. This aspect of SCUBE is similar to the concept of location-keys in LocationGuard [89]. We also assume that it is possible for a searcher to generate the locations of file replicas once the actual physical address of the file server can be generated.

Thus SCUBE has two levels of defense against malicious entities. First, by obfuscating the file identifiers and then distributing the shares, it makes it difficult for an adversary to correlate the shares. Thus it is very difficult for an adversary to regenerate the virtual location of a file. Moreover, by distributing the shares, the availability of indexes is improved. Second, even if the virtual address is generated by an adversary, it is very difficult to generate the physical address of the file server without knowing the correct group key. Table 7.3 shows the cryptographic operations used in SCUBE. The obfuscated file identifiers are 20 bytes in length.

7.4 System Architecture and Implementation

We assume an unstructured P2P based storage overlay with a total of N nodes. To enable search for files hosted at a node, a DHT based search structure is used. Each file is associated with a file-identifier which can be a keyword and a file-index is represented by the tuple {file-identifier,location}. To ensure location privacy, shares of a file-index are generated using the

K	Symmetric group key
N	Total number of nodes
f	Fraction of malicious nodes
(t,s)	Share scheme used
Z_p	Field used in secret-sharing
$E_K(x)$	Pseudo-random function with input x

Table 7.1. Protocol Notations

Protocol step	Overhead
Share generation	$O(st^2)$ {Computational} $O(s)$ {Cryptographic}
Share distribution	$O(s \log N)$ {Traffic}
Share gathering	$O(t \log N)$ {Traffic} $O(t)$ {Cryptographic}
File location generation	$O(1)$ { Cryptographic } $O(t \log t)$ {Computational}

Table 7.2. Protocol overhead

secret-sharing scheme and different shares are hashed to different nodes in the DHT. Each node has a symmetric group key and nodes in the same group have search-access to content hosted in that group. We leave the discussion on group key management to section 6.5. SCUBE uses the notations shown in Table 7.1.

7.4.1 Generation of File-index Shares

To ensure location-privacy, each file index is broken into s index-shares of which a minimum of t shares are required to regenerate the index. Each index share consists of two components: {fileid-share, location-share}. Figures 7.2 and 7.3 describe the steps of the search scheme.

Generation of Location Shares: The location shares are generated using shamir's secret-sharing scheme (see Section 7.2). The location of a file (32-bit ip address of the content-provider) is encrypted using K to generate a virtual address. Different shares of this virtual address (**location-shares**) are then generated using shamir's scheme.

Generation of File-identifier and Index Shares: The following procedure is used to generate file-identifier shares. Let us consider a file with identifier I. Then the j^{th} share of I is represented as

$$I_j = E_K(I||j), j = 1 \cdots s$$

where || can be a concatenation or any other operation. Thus each share of I is obfuscated using the pseudo-random function (keyed hash function in SCUBE) with group-key as input. The index-share is finally generated by a random mix-and-match of the file id and location shares.

Note that each location share generated in the previous step could have been combined with the file-identifier to generate the index share. This approach is vulnerable to leak of privacy. A group of colluding adversaries which host different index-shares of the same file can correlate the shares and gather enough shares to regenerate the virtual address of the file. Therefore the protocol requires the generation of file-identifier shares also.

7.4.2 Distribution of File-Index Shares

Each share generated in the previous step is then inserted into the DHT using chord routing protocol. We assume that the file servers are part of the chord ring and they insert the indexes of their files through trusted proxies or forwarders. To ensure byzantine fault tolerance, there should be a very low probability of two or more shares getting hashed to the same node. For a uniformly distributed chord ring, the **Birthday Paradox** [96] gives the relationship between N and s for which the aforementioned probability is low. Birthday Paradox states that if s keys are randomly hashed to N nodes and $s = \Omega(\sqrt{N})$, then atleast one of the nodes is likely to store more than one key. Thus, for a 1000 node network (N=1000), a value of $(s < 31)$ should be selected.

7.4.3 Searching File-Index Shares

Secret-sharing allows any searcher or a group of index-hosting nodes to generate the location of a file by collecting t or more shares. Thus the IP address of each file is encrypted into a virtual address. This ensures that only a legal searcher can generate the physical location of a file by collecting enough shares.

Cryptographic op.	Key used
Virtual address generation	192 bit AES key
Generation of obfuscated file identifiers	HMAC-SHA1

Table 7.3. Cryptographic operations

A legal searcher can generate the file-identifier shares using the method described in section 7.4.1. After generating s shares, it can initiate search for t or more shares. The search for each share uses the Chord routing protocol. Thus the average number of hops required for searching a single file is $\frac{t}{2}\log N$.

7.4.4 Generating File-Location

A legal searcher can determine the file location from the virtual address and atleast t untampered shares are needed to generate the virtual address. Thus a searcher collects a set of t shares and generates the virtual address. If the address generated is bogus (either the IP address does not exist or the node with the IP address does not possess the file), the searcher repeats the process until the correct virtual address is generated.

7.4.5 Illustrative example

Consider a file FILEA and its location ip address as 129.186.141.12. Using the group key, a 32 bit virtual address can be generated as 077EFACD which is used as the secret in the share scheme. If a (2,4,3) share scheme is used, then the 4 secret shares can be $(x_1,f(x_1))$, $(x_2,f(x_2))$, $(x_3,f(x_3))$ and $(x_4,f(x_4))$. The corresponding obfuscated file identifiers can be AELIF, ILEAF, LEAFI and EAFIL. Thus the content provider inserts a tuple of the form $\{$AELIF,$(x_1,f(x_1))\}$ into the chord network for each share. A legal searcher then generates any 3 obfuscated file identifiers say AELIF, ILEAF and EAFIL and then initiates a search for the corresponding shares. Finally, after downloading the shares from the index providers, the searcher generates the physical address. Note that the above values of the shares are only used for illustration purpose and may not reflect actual values.

7.5 Analyses of Attack Scenarios

In this section, we describe the different possible attack scenarios and how the protocol can address these attacks. We classify the different attacks into two categories: **Infrastructure Attacks** and **File-server Attacks**. Other DoS attacks like Routing Attacks, Partition Attacks and Storage and Retrieval Attacks have been addressed in literature [88]. We focus on attacks through which a set of colluding malicious nodes can make a file unavailable, either by destroying the file itself or destroying the pointer to the file location.

7.5.1 Search Infrastructure Attacks

A group of colluding adversaries can tamper the file-index shares stored at the compromised nodes which can break the search infrastructure. The adversary strategy would be to destroy atleast $(s - t + 1)$ shares of a single file index to make the file index unavailable to a legal searcher. This is because atleast t shares are needed to regenerate the virtual address of the file server.

The group of colluding adversaries can randomly compromise a set of nodes and tamper the file-index shares stored at those nodes. A second approach that can be used by adversaries is to observe queries routed through them and log information about the index providers. Due to Chord routing properties, an adversary can easily know the identity of the index provider if it is the predecessor node on the chord ring. Assuming that a group of adversaries gather information about index providers for queries routed through them, they first need to correlate index shares of the same file and then compromise the index provider nodes. Note that it is very difficult for an adversary or group of adversaries to correlate two obfuscated file identifiers as belonging to the same file. The keyed hash function (HMAC-SHA1) is used to generate the obfuscated file identifiers. Thus the obfuscated file identifier is a random 160 bit string and it is very difficult to correlate two random 160 bit strings as transformations of the same file identifier.

Effect of Routing Different queries of the same file identifier can only be correlated by an adversary if they all have the same initiator. The type of routing used has an effect on the Initiator anonymity in Chord. In Recursive routing, information about the index provider is returned to the initiator through the reverse path of the original query and therefore unlike Iterative routing, initiator anonymity is not compromised. SCUBE uses recursive routing.

Sybil Attacks In Sybil Attacks [32], an adversary introduces a large number of corrupt nodes to control certain regions of the chord ring. Targeted node attack [88] is a sybil attack in which the adversary corrupts certain areas of the identifier circle and this degrades the search performance. The effect of such localized attacks can be mitigated by placing different shares of the same file identifier in disparate regions of the chord identifier circle as has been observed in [88].

7.5.2 File-server Attacks

In File-server attacks, the adversaries employ different strategies to detect the location of a file and attack the file-server nodes. The adversaries can observe and log the queries routed through them. By observing the query traffic, the adversaries can then try to narrow down the set of possible initiators of the query and correlate the shares which originate from the same initiator. Let us assume that a group of adversaries can correlate a set of shares that they have observed and let p be the probability that an adversary lies on the search path. Then for a (t, s) share scheme, the probability that the virtual address can be generated (p_a) is $\frac{1}{p^t}$. When a large fraction of nodes are compromised, a higher number of shares is favorable.

Traffic Analysis Attacks In a Traffic Analysis attack, an adversary observes search traffic generated in the overlay and a group of colluding adversaries then use this information to correlate shares of the same query. Since for each file that is searched, atleast t queries are generated at the initiator, an adversary can use the traffic pattern to correlate shares. Note that SCUBE generates additional traffic due to the additional number of shares that have to be collected for integrity checking. One solution to thwart a Traffic Analysis Attack is to flatten the traffic pattern. Thus instead of generating a burst of traffic, the initiator spreads out the request for the different shares. A second solution is obfuscating the traffic pattern by **Share Mixing**. In **Share Mixing**, shares from different queries are mixed together in a single burst of traffic from the query initiator. We refer the reader to [46] for details.

Dictionary Attacks Dictionary attacks [57, 49] use the fact that a file identifier would usually be a common english word from a restricted vocabulary as found in a dictionary and therefore an adversary generates different possible identifiers to search for a file associated with it. SCUBE uses obfuscated identifiers which can only be generated by a legal searcher. Even if an adversary manages to figure out the original file identifier, the probability of generating valid obfuscated identifiers without knowing the group key is very low.

Phrase Attacks A Phrase attack [10] takes advantage of the fact that a file identifier is usually semantically attached to its content and therefore a query composed of a similar identifier can release information about the content provider. Again the use of obfuscated identifiers in SCUBE prevents such attacks. **Correlation Attacks** An adversary can passively observe search queries routed through it and store these queries. A group of adversaries can then try to correlate these queries to have originated from the same initiator. Assuming the correlation is successful, the adversaries can then issue these queries as bogus queries to generate the virtual address. Finally the adversaries need to break the encryption to generate the actual location of the file. Note that a successful correlation attack depends on the ability of the colluding adversaries to correlate a group of queries from the set of observed queries.

File frequency Attacks File frequency attacks was studied in [89] in the context of read access on file replicas and an optimum value for the number of replicas was suggested. These attacks use the property that if file popularity is known to the adversaries, then by observing query traffic the file location can be deciphered. If file popularity follows a zipf distribution [14], then the adversaries can try to relate the set of observed queries to a set of files. Note that in the context of SCUBE, this can help in deciphering the relationship between obfuscated file identifiers for a particular file and the file itself. We observe that for a (t, s) scheme, as s/t increases, the probability of a file frequency attack decreases (We refer the reader to [46] for a detailed proof). Other types of attacks and how SCUBE handles those attacks is also discussed in [46].

7.5.3 Other Passive Attacks

P2P networks are characterized by high churn [76, 51]. Thus it is possible that the malicious nodes get access to a large set of shares stored at their predecessor nodes. If we assume that the overlay contains file identifiers for Q files and if f is the fraction of malicious nodes in the network, then the probability that a sequence of R searches is successful is atleast $\{\frac{\binom{(1-2f)s}{t}}{\binom{s}{t}}\}^{R}$ (We refer the reader to [46] for a detailed proof). We observe that churn has an adverse effect on the search performance. For example, even for a low value of f (0.05), the success probability for a single search can be as low as 0.8 for a (2,10) share scheme. Over a period of time, a cascading effect can result in the disruption of the search infrastructure.

One solution to mitigate this adverse effect is **Share Regeneration and Distribution**. Proactive Secret-sharing [45] can be used by the file provider to periodically generate new shares and insert them into the chord overlay. Note that the location of the shares (identifiers on the chord identifier circle) is still determined by the obfuscation algorithm and is a requirement for a successful search. Proactive Secret-sharing only helps in regeneration of new shares.

7.6 Experimental Evaluation

We carried out some simulation based experiments to evaluate SCUBE using our discrete event simulator. A maximum of 1000 nodes were used in our simulations and the BRITE topology generator [15] was used to generate the physical topology. The topology was generated using the AS model. We use a chord identifier length of 16 bits. Besides the finger table and other maintenance structures, each node also maintains a share cache. We used a maximum of 5000 file-index insertions in our experiments. A search query was randomly generated every second from a node which does not possess the corresponding file. A random fraction of the nodes in the overlay were selected as malicious nodes with the maximum value being 0.2. To consider the effect of different sets of adversary nodes in the overlay, we generated different initial overlay configurations for each experiment.

We first consider the resilience of SCUBE to infrastructure attacks. The primary metric used in evaluating the DoS resilience of the protocol is **Reliability**.

$$\text{Reliability} = \frac{\text{No of successful searches}}{\text{Total number of searches}}$$

We vary the fraction of malicious nodes and consider different threshold schemes. In our experiments, we vary the values of t and s (Our choice of values for s and t satisfies the condition derived in section 7.4.2). An increase in the value of t increases the computational overhead at a legal searcher. Moreover, an increase in s and t generates additional traffic during share distribution and share gathering respectively. Since an adversary needs to tamper atleast $(s - t + 1)$ shares of an index to make it unavailable, an increase in t for given s decreases the search reliability. On the other hand, a small value of t increases the probability that the adversary can correlate shares and therefore launch file server attacks. Figure 7.6 shows the variation of reliability with the fraction of malicious nodes for different share schemes. We observe that the reliability is quite high (about 0.8) even for a large fraction of malicious nodes (f=0.1). With an increase in the value of t, reliability drops marginally. This is because the probability of the shares getting tampered increases. We also study the effect of overlay size on the DoS resistance of the protocol. Each experiment consisted of several runs to include the effect of different random distributions of malicious nodes. We assume a linear increase in the number of nodes that are compromised. Therefore, the fraction of malicious nodes is kept constant for each overlay size. If we consider the average case, the fraction of the identifier circle covered by the malicious nodes remains the same and hence there is not much of a variation in the reliability. Figure **??** shows reliability vs. overlay size plot for $f = 0.1$ when (2,4) share scheme is used. As the overlay size increases, the density of nodes on the identifier circle increases and each node then hosts a smaller fraction of indexes (assuming a constant number of searches). This decrease in the fraction of indexes hosted by malicious nodes is compensated by an increase in f and hence the number of tampered shares remains more or less constant.

P2P networks are characterized by dynamic membership because nodes join and leave rapidly. This results in **Churn**. Previous studies [51] have shown that a high churn has a significant negative impact on the performance of protocols. We use simulations to observe how realistic node join and leave scenarios affect search performance. Our churn model is based on the model proposed in [53]. We vary the join and leave rates of the nodes in our simulations to achieve varying degrees of churn. Figure 7.4 shows the effect of churn on the search reliability for different sets of parameters. A churn value of 0 means a static network and a value of 1 means a node join and leave rate of 1 per second. A 95% confidence interval is used in our plots. We observe that a there is no significant impact on the reliability at high churn rates. A (2,6) scheme shows a reliability of almost 1 for f = 0.05 and the reliability marginally drops for a higher fraction of malicious nodes

# Shares/search	1	2	3	4	5	6
Search time (ms)	710	1500	1922	2800	2942	4196

Table 7.4. Average search time

but does not show significant changes with an increase in churn rate (frequency of node joins and leaves). We see a similar pattern for a (2,4) scheme.

7.6.1 Search Performance

In Chord, the average number of hops required for searching a key is $\frac{1}{2}\log N$. Thus a (t, s) scheme would require $\frac{t}{2}\log N$ hops on an average. Besides Data-caching, SCUBE also uses Share-caching to improve the search performance. Each node maintains a share and data cache. A share cache consists of the most recently downloaded shares and each share is tagged with the file identifier. We compare the search performance of SCUBE (with share-caching) with Random scoped flooding [50] which is similar to Gnutella [40]. The number of neighbors for each node in the overlay is in the range [3,8], according to the original Gnutella protocol. For a given hop-limit, we plot the number of successful queries that are answered. Figure 7.5 shows the query success rate for an overlay size of 600. In the simulations the MAX-NEIGHBOR parameter is set to 8 and the initial SCOPE for each query is set to 5. For small networks, the search performance of SCUBE is slightly better than flooding and performance improves with a smaller number of shares. On the other hand, for a large network, SCUBE performs appreciably better than flooding. For a network with N=600, even a (2,6) share scheme shows an improvement of about 90% over random flooding. This is because in flooding, search time is $O(N)$ and therefore an increase in the network size has a linear increase in search time on an average. On the other hand, even with an increase in the number of shares, search time in SCUBE has a logarithmic increase. We also plot the search success rate for Chord as a reference. Observe that the use of a single index (as opposed to shares) would improve the search time but such an approach would not be fault tolerant.

7.6.2 Implementation of Prototype

We have implemented a prototype of SCUBE and tested it on the PlanetLab [63] testbed. It runs as an application and consists of about 6000 lines of java code. We use the Apache XML-RPC [1] library for remote procedure calls. Each node runs a Chord Server and searches are generated from Chord clients running on two Intel P4 3.2 GHz machines running Linux 2.6.5 and an Intel Pentium M 1.3 GHz machine running Windows XP. We run a Chord bootstrap node on one of the aforementioned Intel P4 machines. Each node in the overlay contacts this bootstrap server to join the network and thereafter stabilizes into the network within a short period of time. The chord client consists of a graphical user interface to insert file indexes and search files. Due to the transient nature of nodes on the PlanetLab testbed, we were able to use a small set of relatively stable nodes. In all our experiments, we used a set of 50 nodes. The PlanetLab Toolkit [2] was used to generate the scripts for remote execution. To maintain a more or less constant network size, we monitor the overlay network nodes from time to time and start a chord server on a new node as soon as any node leaves the network. In each experiment, we randomly select a set of malicious nodes and vary this number for different sets of experiments. The two primary metrics that we used in our experiments were **Reliability** and **Search time**.

We performed experiments to measure reliability for 2 different share schemes, (2,6) and (2,4). In each experiment, the chord client generates indexes for about 200 files and inserts them into the network and then a search is generated for a randomly selected file. Figure 7.7 shows the variation in reliability with fraction of malicious nodes in the network for two different share schemes. Similar to our simulation results, we observed that (2,6) share scheme performed better for high values of f. We also measured the search time for a share based scheme as a parameter to judge the QoS of the application. We generated a series of random search queries and measured the average response time of a query. Table 7.4 shows the search time for different number of shares. For a non-share scheme, the average search time was about 710 ms, while it increased to about 4196 ms when 6 shares are requested for each search. Thus an user has to wait for about 4 additional seconds when a share scheme is used. We believe that this overhead is acceptable when reliability of the application is of primary concern. Our current implementation does not include share-mixing or traffic obfuscation which we plan to do as part of future work.

7.7 Summary

We have presented a secret sharing based search protocol that uses a DHT-based index structure to speed up search. SCUBE employs two lines of defense against DoS attacks: (1) It obfuscates file identifiers and distributes file indexes at different locations in the network to ensure availability and integrity such that only an authorized searcher can generate the location of a file; and (2) it uses virtual address to obfuscate the actual location of a file to ensure confidentiality. We have analysed the resistance of SCUBE to two broad categories of DoS attacks, search infrastructure attacks and file server attacks, under different adversarial situations. We have also considered the effect of churn on the robustness of the protocol and suggested ways in which the performance of SCUBE can be improved. Our prototype implementation and performance measurements have shown that SCUBE delivers an acceptable level of search performance. As part of our future work, we plan to study the robustness of SCUBE under more complicated attack models and to incorporate a mechanism to detect malicious nodes within the network. We would also like to study the effect of share re-generation and relocation on search performance.

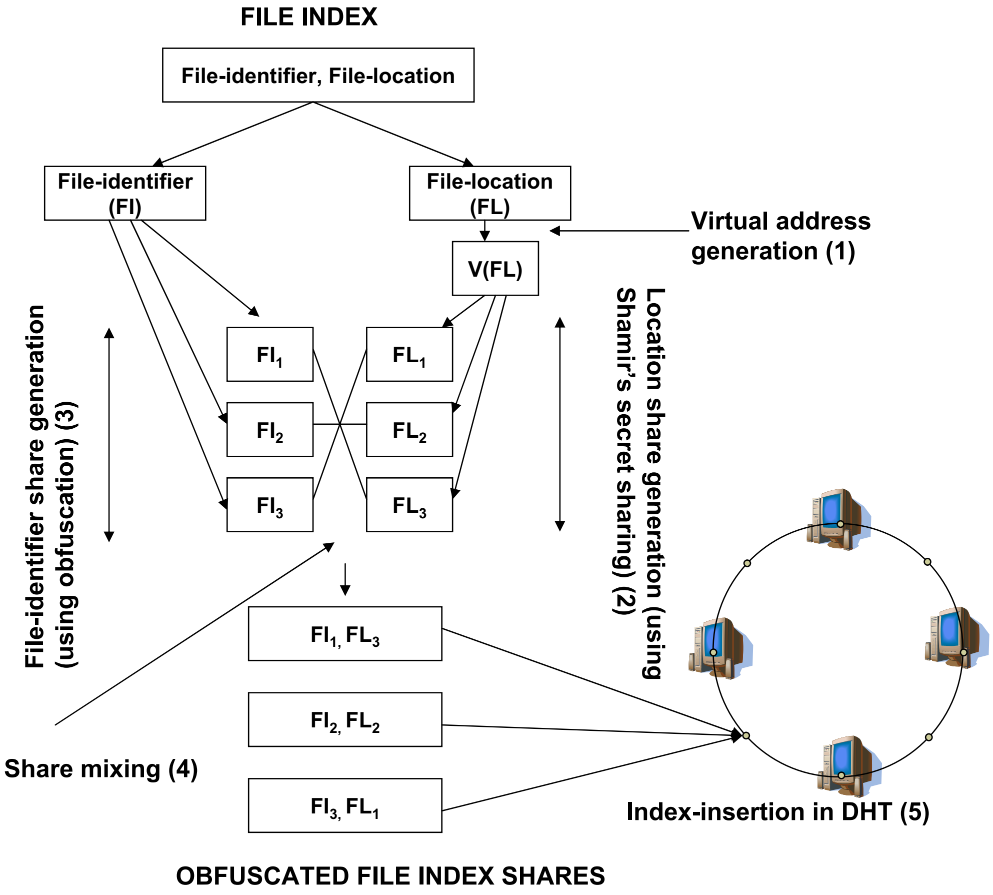
FILE INDEX
File-identifier, File-location
File-identifier (FI)
File-location (FL)
Virtual address generation (1)
V(FL)
File-identifier share generation (using obfuscation) (3)
Location share generation (using Shamir's secret sharing) (2)
FI1
FI2
FI3
FL1
FL2
FL3
FI1, FL3
FI2, FL2
FI3, FL1
Share mixing (4)
Index-insertion in DHT (5)
OBFUSCATED FILE INDEX SHARES

Figure 7.1. Protocol Steps: Step1 – The content provider generates the virtual address, Step2 – The virtual file-location is split into shares using secret-sharing, Step3 – File-identifier shares are generated using an obfuscation technique, Step4 – Random mixing of the file-identifier and file-location shares and generation of obfuscated file-index shares and Step5 – Insertion of the index-shares into the Chord DHT

```
/* Generate virtual address */
Virtual-address = $E_K$(ip-address)
/* Generate location shares using secret-sharing */
$Z_p$: public, (t,s): public
1. Select coefficients $a_1, a_2 ... a_{t-1}$
2. Generate $(x_i, f(x_i))$    $\forall i = 1..s$
/* Generate file identifier shares */
$I_j = E_K(I||j)$    $\forall j = 1...s$
/* Generate index shares */
Randomly mix and match locations shares and file-identifier shares to generate file-index shares $F_1, F_2 \cdots F_s$.
/* Distribute index shares */
Distribute the index shares on the chord ring using the following API:
for each $F_j$, j=1...s
    PUT($I_j$, $(x_j, f(x_j))$)
```

Figure 7.2. Share generation and distribution

```
/* Collect atleast t index shares */
Searcher generates the file identifier shares (same procedure as during generation) and initiates search using the following API:
for each $I_j$, j=1...t
    GET($I_j$)
After collecting atleast t untampered shares, searcher generates the virtual address as follows:
Virtual address = $a_0$ = f(0) {From eqn 7.2}
/* Generate file location */
Ip-address = $D_K(a_0)$
```

Figure 7.3. Share gathering and location generation

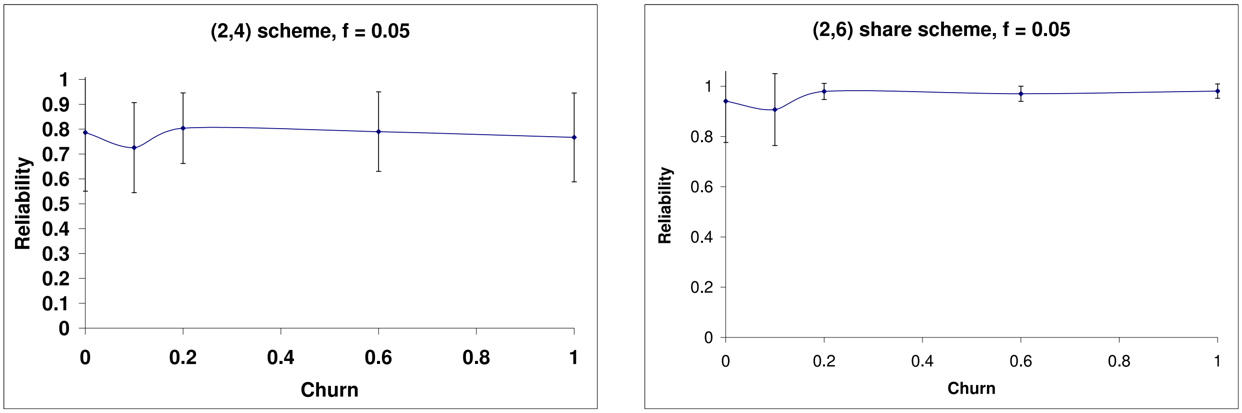

Figure 7.4. Variation of reliability with churn rates

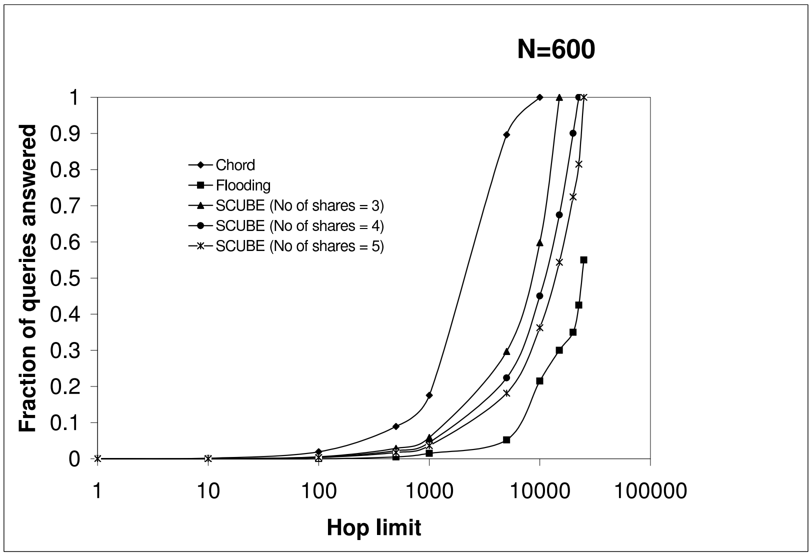

Figure 7.5. Search success rate (hop-limit in logarithmic scale)

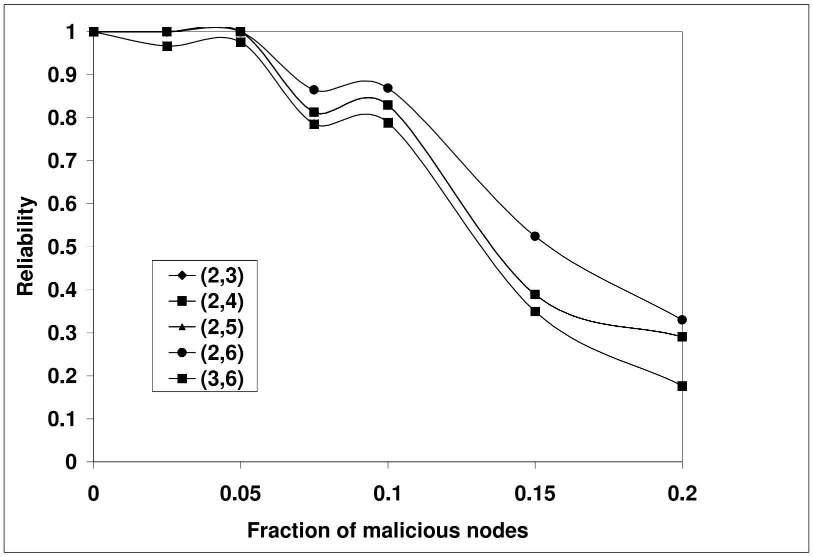

Figure 7.6. Reliability vs. share scheme

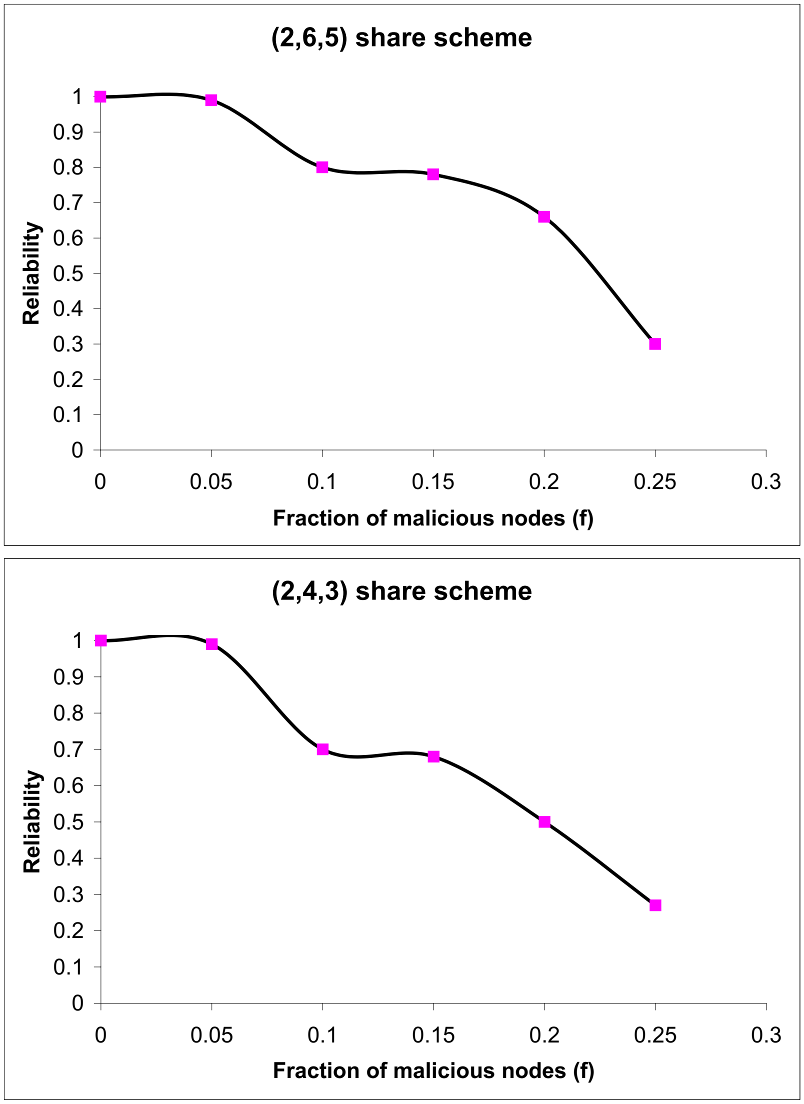

Figure 7.7. Variation of reliability with f (Implementation)

CHAPTER 8. Conclusion

- Anonymous data-transfer for large-scale data-intensive applications: We have tried to address the issue of anonymity for emerging distributed platforms like grid computing by proposing a lightweight forwarding protocol. The major challenge for such large scale data intensive applications is to achieve acceptable degree of anonymity while maintaining a low overhead. Our proposed protocol uses the inherent trust prevalent in such systems to our advantage. To the best of our knowledge, this is the first anonymity protocol for grid computing.

- Enhancing anonymity through the use of incentive mechanisms: Peer-to-Peer based anonymity systems are highly dependent on the participation of peers for achieving a high degree of anonymity. However, the very open nature of such systems makes it difficult to induce peers to forward traffic for others. We have tried to address this issue through the use of a game theory based model to incentivize peer's participation in the network. Our primary contribution is a model to generate optimal forwarding strategies of peers which is aligned with increased anonymity.

- Analyzing leak of privacy in structured Peer-to-Peer networks: Structured Peer-to-Peer networks use a mapping between a file's identifier and a node's location. Thus, lots of information is stored in routing tables of the peers participating in such networks. We have developed an entropy-based model to quantify the leak of information from the routing tables in such networks. We believe that our model can give us insights into the design factors which affect privacy leaks.

- Enforcing anonymity through access control: In this work, we have tried to address the vulnerability of DHT based search indexes by proposing a threshold cryptography-based access control mechanism. We have also implemented a prototype to demonstrate real world feasibility of our protocol.

BIBLIOGRAPHY

[1] Apache xml-rpc, http://ws.apache.org/xmlrpc.

[2] Planetlab toolkit, http://www.cs.virginia.edu/ mngroup/software/.

[3] *Comparison between two practical mix designs*, LNCS, France, September 2004.

[4] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In *Proceedings of the 7th International Financial Cryptography Conference*, 2003.

[5] M. Afergan and J. Wroclawski. On the benefits and feasibility of incentive based routing infrastructure. In *Proceedings of the ACM SIGCOMM Workshop on Practice and theory of incentives in networked systems*, 2004.

[6] L. Anderegg and S. EidenBenz. Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of the Ninth International Conference on Mobile Computing and Networking*, 2003.

[7] Anonymizer. http://anonymizer.com.

[8] F. Azzedin and M. Maheswaran. Evolving and managing trust in grid computing systems. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1424–1429, 2002.

[9] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo. Muon: Epidemic based mutual anonymity. In *Proceedings of the IEEE International Conference on Network Protocols*, 2005.

[10] M. Bawa, R. B. Jr., and R. Agrawal. Privacy-preserving indexing of documents on the network. In *Proceedings of the 29th Very Large Databases(VLDB) Conference*, Berlin, Germany, 2003.

[11] K. Bennett and C. Grothoff. GAP – practical anonymous networking. In R. Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. 2003.

[12] O. Berthold, H. Federrath, and S. Kpsell. Web mixes: A system for anonymous and unobservable internet access. pages 115–129. 2000.

[13] N. Borisov and J. Waddle. Anonymity in structured peer-to-peer networks. Technical Report UCB/CSD-05-1390, EECS Department, University of California, Berkeley, 2005.

[14] L. Breslau, P. Cao, and L. Fan. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the 18th IEEE Infocom*, 1999.

[15] BRITE. http://www.cs.bu.edu/brite/.

[16] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in p2p protocols. In *Proceedings of the International Workshop on Web Content Caching and Distribution*, 2003.

[17] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc networks. In *Proceedings of the First ACM Internation Symposium on Mobile Ad Hoc Networking and Computing*, 2000.

[18] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. In *Mobile Networks and Applications*, volume 8, 2003.

[19] M. Cagalj, S. Ganeriwal, I. Aad, and J. P. Hubaux. On selfish behavior in csma/ca networks. In *Proceedings of the IEEE Infocom*, 2005.

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

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

[22] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of the Advances in Cryptology*, 1988.

[23] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: Architecture and performance of an enterprise desktop grid system. 63(5):597–610, 2003.

[24] G. Ciaccio. Improving sender anonymity in a structured overlay with imprecise routing. In *Proceedings of the Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2006.

[25] G. Ciaccio. Improving sender anonymity in a structured overlay with imprecise routing. In *Proceedings of the Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2006.

[26] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[27] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[28] R. Dingledine, M. J. Freedman, and D. Molnar. The free haven project: Distributed anonymous storage service. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 67–95, 2000.

[29] R. Dingledine, N. Mathewson, and P. Syverson. Reliable mix cascade networks through reputation. In *Proceedings of the Sixth International Financial Cryptography Conference*, 2002.

[30] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[31] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[32] J. R. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems(IPTPS)*, 2002.

[33] M. Feldman and J. Chuang. Overcoming free-riding behavior in peer-to-peer systems. In *ACM Sigecom Exchanges*, volume 6.1, 2005.

[34] M. Feldman, J. Chuang, I. Stoica, and S. Shenker. Hidden-action in multi-hop routing. In *Proceedings of the ACM E-Commerce Conference*, 2005.

[35] D. Figueiredo, J. Shapiro, and D. Towsley. Incentives to promote availability in peer-to-peer anonymity systems. In *Proceedings of the IEEE International Conference on Network Protocols*, 2005.

[36] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15, 2001.

[37] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002.

[38] D. Fudenberg and J. Tirole. Game theory. 1991.

[39] C. Germain, V. Neri, G. Fedak, and F. Cappello. Xtremweb: Building an experimental platform for global computing. In *Proceedings of the 1st*, pages 91–101, Bangalore, India, 2000.

[40] Gnutella. http://gnutella.wego.com.

[41] I. Goldberg. Zeroknowledge to discontinue anonymity service. 2001.

[42] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[43] GriPhyN. http://www.griphyn.org.

[44] S. Hazel and B. Wiley. Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems(IPTPS)*, 2002.

[45] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *Lecture Notes in Computer Science*, 963:339–??, 1995.

[46] http://archives.ece.iastate.edu/secure/00000247/01/tr01.pdf. Scube: A dos resistant distributed search protocol. Technical Report TR-2006-04-23, Computer Engineering, Iowa State University, Ames, 2006.

[47] KaZaA. http://www.kazaa.com.

[48] D. Kesdogan and J. E. Rol. Stop-and-go mixes: Providing probabilistic anonymity in an open system. In *In Proceedings of Information Hiding Workshop (IH)*, pages 83–98. 1998.

[49] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, 1990.

[50] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. Random walks in peer-to-peer networks. In *Proceedings of the 23rd IEEE Infocom*, 2004.

[51] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating dht design tradeoffs under churn. In *Proceedings of the IEEE Infocom*, 2005.

[52] J. Liang, N. Naoumov, and K. W. Ross. The index poisoning attack in p2p file sharing systems. In *Proceedings of the 25th IEEE Infocom*, 2006.

[53] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing*, 2002.

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

[55] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems(IPTPS)*, 2002.

[56] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: An approach to universal topology generation. In *Proceedings of MASCOTS*, 2001.

[57] R. Morris and K. Thompson. Password security: A case history. *CACM*, 22(11):594–597, 1979.

[58] Napster. http://www.napster.com.

[59] C. W. O'Donnell and V. Vaikuntanathan. Information leak in the c hord lookup protocol. In *4th International Conference on Peer-to-Peer Computing*, 2004.

[60] A. Pfitzmann and M. Hansen. Anonymity, unobservability and pseudonymity: A consolidated proposal for terminology, (draft), 2000.

[61] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Isdn-mixes: Untraceable communication with very small bandwidth overhead. pages 451–463. 1991.

[62] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers & Security*, 6(2):158–166, Apr. 1987.

[63] PlanetLab. http://www.planet-lab.org.

[64] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM*, San Diego, CA, 2001.

[65] S. Ray, G. Slutzki, and Z. Zhang. Incentive-driven p2p anonymity system: A game-theoretic approach. In *ICPP*, page 63, 2007.

[66] S. Ray, G. Slutzki, and Z. Zhang. Incentive-driven p2p anonymity system: A game-theoretic approach. Technical report, Department of Computer Engineering, Iowa State University, Ames, 2007.

[67] S. Ray and Z. Zhang. An efficient anonymity protocol for grid computing. In *GRID*, pages 200–207, 2004.

[68] S. Ray and Z. Zhang. An information-theoretic framework for analyzing leak of privacy in distributed hash tables. In *Peer-to-Peer Computing*, pages 27–36, 2007.

[69] S. Ray and Z. Zhang. Scube: A dos-resistant distributed search protocol. In *ICCCN*, pages 128–134, 2007.

[70] J. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Proceedings of the Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[71] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 10–29. 2001.

[72] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. In *IEEE Journal on Selected Areas in Communications Special Issue on Copyright and Privacy Protection*, 1998.

[73] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. In *ACM Transactions on Information and System Security*, 1998.

[74] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, June 1998.

[75] M. Rennhard and B. Plattner. Practical anonymity for the masses with morphmix. In *Proceedings of Financial Cryptography (FC'04)*, pages 233–250, 2004.

[76] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *Proceedings of the USENIX Annual Technical Conference*, 2004.

[77] J. Rohrer. Mute: Simple, anonymous file sharing,http://mutenet.sourceforge.net,2004.

[78] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the ACM/IFIP/USENIX Middleware*, 2001.

[79] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. 2002.

[80] V. Scarlata, B. N. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *Proceedings of the IEEE International Conference on Network Protocols*, 2001.

[81] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, 2002.

[82] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613??, 1979.

[83] C. Shannon. The mathematical theory of communication. *Bell Systems Technical Journal*, 30:50–64, 1948.

[84] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[85] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet.

[86] V. Shrivastava and S. Banerjee. Natural selection in peer-to-peer streaming: from the cathedral to the bazaar. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2005.

[87] A. Singh, B. Gedik, and L. Liu. Agyaat: Mutual anonymity over structured p2p networks. volume 16, 2006.

[88] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems(IPTPS)*, 2002.

[89] M. Srivatsa and L. Liu. Countering targeted file attacks using locationguard. In *Proceedings of the 14th USENIX Security Symposium*, 2005.

[90] S. Steinbrecher and S. Köpsell. Modelling unlinkability. In R. Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. 2003.

[91] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM*, San Diego, CA, 2001.

[92] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2006.

[93] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.

[94] D. Talia and P. Trunfio. Toward a synergy between p2p and grids. *IEEE Internet Computing*, 7(4):94–96, 2003.

[95] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *Proceedings of the HotNets -I*, 2002.

[96] W. Trappe and L. C. Washington. Introduction to cryptography with coding theory. Technical report, 2001.

[97] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, 2000.

[98] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Symposium on Network and Distributed System Security*, 2002.

[99] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending anonymous communications against passive logging attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[100] M. Wright, M. Adler, B. N. Levine, and C. Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security(TISSEC)*, 4(7):489–522, November 2004.

[101] L. Xiao, Z. Xu, and X. Zhang. Low-cost and reliable mutual anonymity protocols in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):829–840, Sept. 2003.

[102] G. Yang, D. S. Wong, X. Deng, and H. Wang. Anonymous signature schemes. In *Proceedings of the 9th International Conference on Theory and Practice of Public Key Cryptography*, 2006.

[103] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of the IEEE Infocom*, 2003.

[104] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings of NSDI*, 2005.