TOWARDS A STRONGER PEER-TO-PEER ANONYMOUS SYSTEM

by

ARJUN R. NAMBIAR

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

To my Mom Prasanna, Dad K.M.R Nambiar and Anu- the most important people in my life.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Matthew Wright for giving me a chance to pursue my interest in Network Security, teaching me all I know, and letting me collaborate with him in this research project. It has been a valuable, eye-opening experience for me. I would also like to extend my appreciation to the NSF and Dr.Wright for providing financial support for my studies. He will always be a friend as well as a mentor, and the person behind any good work I do from now on.

I also wish to thank Dr. Mohan Kumar and Dr. Donggang Liu for their interest in my research and for taking time to serve in my Thesis committee.

I wish to thank members of the iSEC Lab at UTA, especially to Dr. Donggang Liu, Madhu Venkateshiah and Hatim Daginawala for all I learnt from them.

I would like to thank Sushant Prasad, who's been more of a brother than a friend, and for encouraging and inspiring me to pursue a Thesis.

Most importantly I would like to thank my parents for dedicating their entire life so as to give me the freedom to discover my path. I would like to thank my kid sister Anu for her positive and uplifting presence. These three people, more than anyone else, are responsible for anything good to come out of me.

April 17, 2006

ABSTRACT

TOWARDS A STRONGER PEER-TO-PEER ANONYMOUS SYSTEM

Publication No. ______

Arjun R. Nambiar, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Matthew K. Wright

Anonymous communications systems on the Internet provides protection against eavesdroppers and others that seek to link users with their communications. These systems have many important applications in areas such as law enforcement, intelligence gathering, business privacy, anonymous publishing, and personal privacy. Currently deployed systems rely on a relatively small set of advertised servers to forward messages for the user. These systems can suffer from scalability problems, with potentially large bandwidth and system overhead costs, and the servers themselves can be targets of direct attacks.

Peer-to-peer anonymous communications systems, such as Tarzan[1] and MorphMix [2], have been proposed as a way to alleviate these problems with a large and dynamic set of peers acting as servers. This makes direct attacks less effective and increases scalability. Tarzan, however, requires that each peer know the identity of all other peers, which makes it highly vulnerable to intersection attacks 2.3. MorphMix does not have this requirement, but it requires that users allow other peers to choose the peers that will help forward the users messages; attacker controlled peers will always select other colluding peers to be on the path. Although the authors of MorphMix propose a collusion detection scheme, attacker-controlled peers will be able to choose their colluding partners as forwarding nodes far too often while the detection algorithm is still learning. The fundamental problem is one of selecting peers independently at random, to ensure unbiased path selection, while not distributing lists of all the peers in the system to all other peers.

We propose a new peer-to-peer anonymous communications system using distributed hash tables (DHTs). Similar to peer-to-peer file-sharing systems that use DHTs 2.6, our system maps each IP address to a point on the id space using consistent hashing. We further divide the id space into groups, conceptually organized as a binary tree for purposes of node lookup. Each node has knowledge of all the nodes in its own group, as well as knowing a limited number of nodes in other groups. This knowledge is enough to effectively route lookups throughout the system. Nodes use redundancy and probabilistic checking when performing lookups to prevent malicious nodes from returning false information without detection.

We show that our scheme prevents attackers from biasing path selection, while incurring moderate overheads, as long as the fraction of malicious nodes is less than 20%. Additionally, the system prevents attackers from obtaining a snapshot of the entire system until the number of attackers grows too large (e.g. 77% for 10,000 peers and 1024 groups). The number of groups can be used as a tunable parameter in the system, depending on the number of peers, that can be used to balance performance and security.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 What is Anonymity

Anonymity is a technological solution to protecting one's online privacy. With its potential for monitoring user activity, the Internet is turning into the most pervasive surveillance system ever. Information that can be obtained by observing you includes to whom you send emails, what websites you browse, what chatrooms you visit, where you work, where you shop online and what you buy, what kind of physicians you visit, etc. Most of this information is trusted to several separate organisations, but there is a possibility of pooling in their resources to form a dossier on potentially anybody. There is thus a need for systems that ensure anonymity for these groups. Examples of people or organisations requiring anonymity include:

- *Socially sensitive communicants* visiting disease and crime victim chatrooms.
- *Corporations* trying to hide collaborations of sensitive business units and partners.
- *Law enforcement* authorities to encourage anonymous tips or for crime watching.
- *Government* to hide negotiations or procurement patterns.

Slightly more formally, anonymity is the state of being indistinguishable from other members of a set i.e., each member of the set is as likely the source of the message. This set is what is called the *anonymity set* [3]. Anonymity is not absolute. Larger this anonymity set is, greater is the degree of anonymity offered.

1.2 Anonymity over the Internet

The reason anonymity over the Internet is a problem has a lot to do with the medium and how the Internet operates. All machines on the Internet use the IP [4] protocol to communicate. The IP protocol, among its various header fields, has the source and destination IP address fields for effective two-way routing. These IP addresses can serve to identify and track communicating parties. Our goal is to hide the sender's IP address. As way of comparison, in conventional mail, in order to ensure anonymity one would omit the return address. Thus, the aim is to create an overlay network that masks the IP address while allowing real-time bi-directional communication. Information about the source and destination can be obtained by observing particular data move through a network, by matching amounts of data or by examining coincidences such as simultaneous opening and closing of connections. As a goal, we want to protect users against *traffic analysis* [5]. That is, we do not want an adversary capable of monitoring or compromising certain parts of the system, to be able to match sender and recipient without a certain degree of uncertainty.

1.3 Contribution

Traditional solutions for anonymity are mix-based systems [2.2], which consist of a small set of advertised servers used specifically to relay messages. These systems can suffer from scalability problems, with potentially large bandwidth and system overhead costs, and the servers themselves can be targets of direct attacks. Attempts have been made to extend these systems to a peer-to-peer environment for scalability among other reasons. Systems like Crowds[2.4.1], Tarzan[2.4.2] and MorphMix[2.4.3] which are examples of such systems suffer from certain attacks[2.5]. Our contribution is a proposal for a new peer-to-peer anonymous communications system using distributed hash tables (DHTs)[2.6]. The proposed system gives the user the ability to choose relays from among all the nodes of the system in spite of having just limited knowledge of the system. Nodes use redundancy and distance checking when performing lookups to prevent malicious nodes from returning false information without detection. We finally show that the system is resilient to certain attacks that plague the other p2p anonymous systems.

1.4 Thesis Organization

In Chapter 2, we discuss the background concepts and related work to justify the design decisions explained in our description of the system. In Chapter 3, we present our proposed system in detail and analyze the security of the system. In Chapter 4, we explain the simulation setup and aspects of the system that we tested. Section 4.5 presents the results of the simulations and their interpretation. Chapter 5 concludes with ideas for future work.

CHAPTER 2

BACKGROUND

Anonymity on the Internet is not a new security goal. In this chapter we present some background on the techniques in use to thwart traffic analysis. We also introduce key concepts crucial to understanding the reasoning behind some of the design decisions of our proposed system explained in Chapter 3. We also use this chapter to establish a nomenclature along the way to be used through this chapter and the rest of the document for purposes of homogeneity. The node attempting to send a message anonymously is called the *initiator* and the intended destination of that message is called the *responder*.

2.1 Single Proxy

The most intuitive solution to thwart traffic analysis is to add a level of indirection between the initiator and responder. However with only one level of indirection this is less suitable for real-time traffic like the Internet, but more for asynchronous traffic like electronic mail. Examples include Anonymizer [6] and remailers such as the now defunct Penet remailer [7]. As an example, Anonymizer is essentially a server with a web proxy that filters out identifying headers and source addresses from web browsers' requests. So instead of seeing the initiator's true identity, a web server only sees the identity of the Anonymizer server. This provides almost no safeguarding against traffic analysis. The system has other disadvantages too. It is a single point of failure. Also you have to trust the proxy, since they are in a position to observe everything. This is a less obvious decision than it may seem. The proxy may be the target of a legal attack by people in power via a subpoena. A trivial DoS attack seems feasible too.

2.2 Mix-Based System

Single-proxy systems like Anonymizer do very little to add indirection in the way of a determined attacker. Chaum presented the idea of mixes [8] between the initiator and the responder as a way to thwart the attacker. The purpose of a mix is to hide the correspondences between the messages in its input and those in its output. The order of arrival is hidden by outputting uniformly sized messages in lexicographically ordered batches. The messages are encrypted/decrypted at the mix so there is no danger of correspondence between messages at the input and output simply by appearance. A single mix does nothing but hide the correlation between incoming and outgoing messages. Using just one mix would make the sender dependant solely on the honesty of that one mix, and so several mixes are used in a chain. The first mix in the chain receives data from the intitiator, makes some cryptographic transformations and forwards it to the next mix. The last mix sends it to the responder. To make it resistant to traffic analysis, the mixes employ fixed-length messages as well as layered encryption of the messages. A vast number of solutions since have been modifications to this basic approach. Systems in use that provide anonymous real-time communication like Tor [9], Freedom [10], Onion-routing [11] are based on mixes. Our system is based on the same approach. Since our system, like most other solutions, does not provide a lot of functionality of the original mixes, we use the term *proxies* when refering to the mix-like intermediate nodes in our system. While a correct host runs only one *honest node*, which forwards packets properly, does not log addressing or timing information, and so on, an adversary can run potentially many *malicious nodes* or spoof many fake addresses. A node is malicious if it modifies, drops, or records packets, analyzes traffic patterns, returns incorrect network information, or otherwise does not properly follow the protocols.

In a generic mix-based system, to begin a session between an initiator and a responder, the initiator's proxy identifes a series of mixes forming a route through the network, and creates a virtual circuit(VC) through them. Based on this route, the initiator's proxy encrypts first for the responder, then for the preceding mix on the route, and so on back to the first mix to whom he will send the message. Hence mixes are said to employ *nested encryption*. Each proxy knows the previous and the next mix on the path, but knows nothing about the other mixes or where it lies in the chain. Since the system uses source routing, the mixes can be chosen over multiple domains so as to make the job of traffic analysis harder. mixes can be advertised so that they are easily accessible and individual mixes can be chosen based on trust, administrative domain or geographical location. However they do suffer from scalability issues. These systems can suffer from scalability problems as the number of participants increase, with potentially large bandwidth and system overhead costs. The mixes themselves can also be targets of direct attacks.

There are different possibilities to organize the co-operation of several mixes. One is where the sender can choose the mixes that make up her message's path. This is what is called the *free-route* topology. The contrasting setup is where nodes choose mixes from a set of fixed paths through the network. This is called the *mix-cascade* topology. [12] and [13] provide interesting analysis about whether the use of either affects the anonymity offered by the system. There are valid arguments for and against both of the topologies and the choice is based on attacker model, requirement of services and the incentives for participants [14]. Free-route topologies do however provide a larger number of exit and entry points, and require much less trust which seem more suitable for a dynamic system like ours, where nodes can easily join and leave the system.

Mix-based systems do not ensure anonymity. Different systems are vulnerable to different attacker models. Broadly the various types of attacks against low-latency mix-based systems are *timing analysis*, *predecessor attack*, and *intersection attack*. Mix-based systems used for real-time bidirectional traffic actually does very limited mixing , and hence are vulnerable to powerful adversaries. A substantial amount of work has been published about the various attacks on these systems. [5] categorizes the various types of traffic analysis attacks. [15] provides a good analysis of the security features of Onion Routing [16]. Timing analysis is where the attacker studies the timings of messages moving through the system to find correlations. It may make it possible for two malicious nodes to colloaborate to determine that they are on the same communication path. Systems like Tor use no specific techniques to prevent timing analysis. Hence in such systems when the first and the last mix on the VC are malicious, effective timing analysis may allow the attacker to link sender and receiver entities. A detailed analysis of timing analysis in low-latency mix systems is presented in [17].[18] talks about the threat of a single malicious entity presenting multiple identities, especially . Also as shown in [19]when a particular initiator continues communication with a particular responder over path reformations, existing protocols are subject to attack. The predecessor attack runs simply on the assumption that for an identifiable stream of communications through an attacker, the initiator is more likely to send the message to the attacker than any other participant. We talk about the intersection attack in more detail in the next section. [20] shows the threat from intersection attacks [2.3] and how they can be mitigated in systems with static routes.

2.3 Intersection Attack

The basic premise of the *intersection attack* is that users typically communicate with a relatively small number of parties. A classic example would be typical browsing behaviour- most users query the same websites in different session, the pattern is not random. It is carried out over a period of time. The two things that the attacker needs to be able to relate messages as belonging to same session as well as knowing which of the nodes are offline and online at the instant it observes them. The attaker then creates a set of possible senders for each message. For all messages belonging to the session, if the attacker intersecting the sets of possible senders will remove users who were not active during the transmission of all messages. This shrinks the set of suspects possibly till there is only one user left. The operation used here is an intersection of sets, hence the name.

2.4 P2P Anonymous System

Mix-based systems suffer from scalability problems since there is a disproportionate number of mixes to participants. Also, advertised mixes become targets of direct attacks. Although this is not a traditional client-server relationship, there is a definite hierarchy among the nodes in the system. Peer-To-Peer(p2p) anonymous systems extend known mix-based systems to the p2p environment. Nodes communicate through mix proxies chosen from an open ended pool of nodes [1]. Each node is an equal peer, therefore each node is a possible originator and each node is a possible relay/mix. This increases the size of the anonymity set. With no inherent hierarchy, there is no unfair load on certain nodes in the system. Hence, they scale well. There are a few p2p anonymous systems in existence. We distinguish among them based on the amount of knowledge of the system

[1] Synonymous with mixes in such a p2p environment.

that a node has, and show why we think less knowledge makes for a more secure system.

Current P2P anonymous systems include Crowds [21], Tarzan [1] and MorphMix [2]. We briefly introduce each of them below.

2.4.1 Crowds

Crowds consists of a lot of nodes that are run by the users of the system. Web requests are randomly chained through a number of them before being forwarded to the web server hosting the requested data. It has a node called a "blender" which maintains a list of nodes in the system. Nodes choose the next node(proxy) in the circuit randomly from the entire set of Crowds nodes, since they are all equal peers. The server will see a connection coming from one of the Crowds users, but will not be able to tell which one was the original sender. In addition Crowds uses encryption so that some protection is provided against attackers who intercept a user's network connection. However, this encryption does not protect against an attacker who cooperates with one of the nodes that the user has selected since the encryption key is shared by all nodes part of a connection. Traffic analysis is also trivial for an attacker who can observe all network connection since the messages are forwarded without modification and appear the same on different links. Crowds is named for the notion of *blending into a crowd* operates by grouping users into a large and geographically diverse group(crowd) that collectively issues requests on behalf of its members.

2.4.2 Tarzan

This is another p2p anonymous system based on the IP layer. It is a variant of onion routing in that it uses onion routing style layered encryption. Tarzan provides a higher level of practical security in some cases by having nodes select proxies based on domains. This means that the attacker cannot overload the network with malicious nodes from within the same domain since initiating nodes will not select proxies from that domain with any greater frequency. Attackers could gain an advantage however when the number of honest domains represented is small. An attacker may be able to operate nodes with no honest participants. With attackers in a few such domains, attackers could make it likely to appear on an initiators path despite only operating a few corrupt nodes.

2.4.3 MorphMix

MorphMix is another p2p anonymous system which uses a layered encryption scheme. It differs with the previous two in that it does not require nodes to have knowledge of the entire system. It consists of an open ended pool of volunteer nodes, with each node knowing only a small subset. In MorphMix the initiator only chooses the first proxy in the VC, and then each proxy in the path chooses the next. Nodes use witness nodes from among the nodes they know in order to act as the third party in the process of establishing the next hop of the anonymous tunnel. Finally, it also incorporates a *collusion detection* scheme to avoid collaborating malicious nodes from choosing the next proxy from the subset of malicious nodes.

2.5 Attack against p2p anonymous systems

In this section we talk about the attacks against the existing p2p anonymous system viz. Crowds, Tarzan, and MorphMix. The aim is to introduce the attacks before we explain how our system mitigates these threats.

2.5.1 Attacks against Tarzan, Crowds

Some peer to peer systems are particularly vulnerable to the intersection attack explained in Section 2.3. Both Crowds and Tarzan require complete knowledge of the system to function. So just placing one node in the system would ensure that the attacker gets a *snapshot* of the entire network. A snapshot here refers to the set of all the active nodes in the system at that instant. This makes them vulnerable to the intersection attack, since both are dynamic systems. For this reason, we feel the need for a p2p anonymous system where individual nodes know only a small subset of the entire network, thus making the intersection attack harder to carry out.

2.5.2 Attacks against MorphMix

MorphMix, as a system, is closest to ours w.r.t certain important design decisions. It is also a p2p anonymous system where the nodes have only limited knowledge of the entire system. However in MorphMix the initiator only chooses the first proxy in the VC, with each proxy in the path chooses the next. Thus one proxy in the path being an attacker allows it to choose all or some of the rest of the proxies as attackers. MorphMix does have a collusion detection scheme to detect such a case, but it has an initial learning phase. The system is vulnerable during this phase. Also, more the attackers in the network, higher is the possibility of choosing one, but slower is the learning curve. Our target is thus a p2p anonymous system, with nodes being able to choose proxies on the path from the complete set of nodes in the system, in spite of limited knowledge of the system.

2.6 Distributed Hash Tables

A *hash table* is used to map *keys*[2] onto values and is mainly useful for efficient lookups. Distributed Hash Table(DHTs) is a distributed infrastructure that provides hash table-like functionality on an Internet-like scale. A hash table is a table mapping keys to values, so intuitively a distributed hash table partitions ownership of those keys

[2]A key is any identifying information, like the name of a resource

among the nodes in the system. The ownership is partitioned in such a way so as to be able to efficiently route lookups to the node responsible for the key. Main features of DHTs are:

- *Decentralization:* Everyone maintains limited knowledge of the network viz. a small portion of the entire hash table.
- *Scalability:* Indexing is not a problem since there is no centralized authority responsible for the entire hash table. Large increase in the system results in only minute increments in the portion of the hash table of individual nodes.
- *Fault Tolerance* The system deals with node departure/failure gracefully. Also since there's no centralized authority, there is no single point of failure.

DHTs give us the tool to choose node verifiably at random from the complete set of nodes in the system inspite of having limited knowledge of the system. Current DHT-based p2p systems include Chord [22], CAN [23], Pastry [24], Tapestry [].

In order to give the flavor of such systems we provide an overview of the Chord system. Chord provides fast distributed computation of a hash function mapping keys to nodes responsible for them. What Chord does is provide a scalable protocol for lookup in a dynamic p2p system with frequent arrivals and departures. Chord assigns keys to nodes with consistent hashing. Consistent hashing has the property that with high probability the hash function balances load. Chord improves the scalability of consistent hashing by avoiding the requirement that every node know about every other node. A Chord node needs only a small amount of "routing" information about other nodes, thus making it more scalable. Because this information is distributed, a node resolves the hash function by communicating with other nodes. In an N-node network, each node maintains information about only $logN$ other nodes, and the number of messages required to resolve a lookup is $logN$.

CHAPTER 3

SYSTEM DESCRIPTION

To move closer to a practical and secure p2p anonymous communication system, we propose a new p2p anonymous system using DHTs, briefly introduce the key design decisions and discuss how the pieces fit together. Following the discussion in section 2.6, we use DHTs to create a dynamic self-organizing overlay in order to allow nodes to join, leave and effectively route between all the nodes of the system, while having only limited knowledge of the complete system. As explained in section 2.5.2, allowing every node to have only limited knowledge of the system helps protects it from the intersection attack. As with the p2p paradigm, every node in the proposed system is an equal peer, so each node is a possible originator as well as a relay. P2P systems means that the attacker has to cover more surface of attack since the number of entry and exit points are much larger than in mix-cascades. It also makes the system inherently more scalable, since every participant is involved in relaying messages and the system does not rely on few servers to do the relaying as in traditional mix-based systems like Tor [9].

The way we use DHT's here is similar to some of the p2p file-sharing systems like Chord [22], CAN [25], etc. Our system maps *keys*, or identifying information, to *ids* that are used within the system for routing. Since the aim of the lookup is to locate nodes, the keys here are the nodes' IP addresses. However, we present a novel way to perform lookups and node joins in a way that provides greater resilience to attacks. Each node has a list of known nodes, called the *contact table*, similar to a routing table, which it uses to resolve lookups throughout the id-space. Since a node keeps routing information of just these few " contacts", it has to contact other nodes in order to resolve a lookup. The way we organize this id-space and the way we use it to add nodes to the contact table is crucial to the correctness and security offered by our system. As explained earlier, the proposed system is similar to a mix-based system, so the anonymity is based on VCs created with a chain of nodes. We refer to each of the nodes in the chain as a proxy. Also, since this is based on DHTs, each lookup is resolved by contacting other nodes in a recursive manner. Each of the nodes contacted during a lookup is called a *lookup hop*.

3.1 Overview

Since this system is based on the free-route topology [2.2], the initiator chooses the proxies in the VC. These proxies should be chosen at random from the entire system so as to make the work of the attacker more difficult. However, since each node does not have a complete view of the system, we utilize the DHT to do lookups for random nodes. Each lookup is resolved by recursively contacting a number of other nodes till it reaches the *target* node. The lookup algorithm is based on the organisation of the id-space discussed in section 3.2. section 3.3 describes bootstrap methods in brief, section 3.4 explains how nodes join the system. With that background, section 3.5 explains the crux of the system-the lookup, but before explaining the details we take a look at the overall system.

With the system in place, when an initiator needs to communicate with a responder, it first needs to set up a VC. It does so by choosing three ids at random from the id-space, such that the owner of each of the ids will be a proxy on the VC. It then asks either all or a subset of its local contacts(depending on the selected redundancy) to do a lookup for the first id. Since this system is based on an open ended pool of volunteer nodes, the possibility of running into malicious nodes during a lookup exists. The redundancy helps the mitigate that risk. If the node receives different results from the separate redundant lookups, it indicates that either one or more of its local contacts or their lookup-hops are manipulating the results. The node can either choose the result whose id is closest to the target id[1], or it can choose to void all the lookups and choose another id for lookup. The system also incorporates a *distance checking* to determine if a malicious node was returned by a lookup. When a lookup is resolved, the node checks the offset of the node's id from the target id specified. The value is then normalized by the group size. If this offset is within the amount it pre-decides as safe[2], it accepts it or else it discards the id and decides on a new target id and repeats the process. The group size can be easily determined by dividing the number of points in the id space with the number of groups in the system. We use loose checking with more redundancy for the first proxy and tight checking with less redundancy for the later proxies. Either way, at the end of the process it has the node that will be the first proxy in the VC.

It establishes a connection with that node, and through that connection asks this first proxy to perform a lookup for the second id. The reason the initiator goes through the first proxy to resolve the lookup is to not expose the relationship of the initiator and later proxies. By going through the first proxy, the first proxy would not be able to deduce if the lookup was from the initiator or if the initiator itself was a proxy on some other node's VC. Now the first proxy goes through the same process to resolve the lookup and sends the result to the initiator. In this way the initiator sets up a VC consisting of three intermediate proxies. This VC is kept alive for a small time period, say three minutes, during which the initiator uses it for all communication. Note that this only sets up an anonymous connection. The intiator can choose to identify itself

[1]Since the owner of an id is the first node whose id is greater than the target id.

[2]The number of nodes in the system helps it decide the approximate separation of ids from their owners.

to the responder within the message. After the time period, the VC can be destroyed and another one can be used. The process of creating another VC can be done before the destruction of the first VC, so as to seamlessly move over fresh communication over the new pipe. Long standing connections could still be used for applications like remote login. The setting up of the VC, apart from the choice of proxies on the path, is similar to that of Tor [9].

3.2 Organisation of the id-space

The strength of DHTs is that a node can choose any other node in the system with only limited knowledge of the system. As explained earlier, our system maps IP addresses to id points. We use consistent hashing to map the IP address to a point on the id-space. Each node is said to "own" the id-space between its id and the id of the previous node in the id-space, i.e., it is responsible for the preceeding segment of the id-space. For purposes of lookup and adding nodes to the contact table, we divide the entire id space into groups. Each group is a contiguous portion of the id-space and is cyclic, i.e., the ends of the id-space of groups wrap around. All the nodes whose id's fall within a certain group know all the rest of the nodes within that group. Further let us say that a node 'n' belongs to a group if the id obtained by hashing n's IP lies in the id-space of that group. The notion of group also helps us define the concept of *neighbors* and *distant nodes* on the id-space. Nodes within the same group are called neighbors, while those in other groups are said to be distant nodes.

Groups are conceptually organized in the form of a binary tree. Let the number of groups be G. Then the first $log_2(G)$ bits of all id's in each group are the same. Thus, as an example, for an id-space divided into 8 groups, the first 3 bits($log_2(8) = 3$) of each id in group 0 would be 000, group 1 would be 001 and so on. So let us define the *group number* as the first $log_2(G)$ bits of each id within a group. What this binary structure also does is help to add nodes to the contact tables in a logical, verifiable, yet random manner. The height of the binary tree gives us the number of distant nodes present in a contact table. The first few entries in the contact table are all the rest of the nodes in the same group. So if a group has a population of p, the first $p - 1$ contacts are neighbours. Lets call them the *local contacts* and the distant nodes in the contact table as the *global contacts*. To choose the global contacts, for each contact, you go one level up in the binary tree and choose one node randomly from the other subtree that doesn't include itself. So for group 0(we'll call it G0), we go one level up and then choose from the subtree, thus choosing a node randomly from the right sub-tree i.e G1. Similarly for the next global contact, we go another level higher and choose a node randomly from the right sub-tree viz.G2 and G3, and so on. So to summarize, each node's contacts would include all the nodes from the same group, one node from the next group, another from the next two, another from the next four and so on[3]. This way each node has $log_2(G)$ global contacts. A feature of this scheme is that it gives a node more knowledge of nearby nodes and progressively less knowledge as one moves further in the id-space.

3.3 Bootstrap nodes

We haven't focused on the problem of bootstrapping during the modeling of the system. Most of the analysis is done with the assumption that the system already contains a few hundred nodes. However it is not hard to imagine a query for the name of the system to a central server yielding a set of nodes, among which one is chosen. Once you obtain even a single node in the system, the process of joining becomes a process of following the steps explained below. The name resolution for the system could change

[3]In increasing powers of 2.

periodically by an established method in order to introduce randomness in the choice and to avoid malicious nodes compromising the process. Alternatively, a node could join through a *trusted* friend. The friend could supply the joining node with a list of nodes in the system, which the node could use to join the system. This eliminates the risk of the bootstrap node being malicious.

3.4 Node Joins

The process of joining is similar to it's counterparts in other p2p file sharing systems like CAN, Chord. The process is best explained in steps as follows:

- Obtain a set of bootstrap nodes.
- Hash its own IP so as to obtain its corresponding id on the id-space.
- Perform a lookup for its id through a subset of the bootstrap nodes.
- The node returned by the lookup is it's successor in the id-space, and also one of it's neighbours.
- Populate the local portion of the contact table.
- Send an update message signed by the successor to its successor as well as its predecessor, which then let the message propogate throughout the group.
- Populate the global portion of the contact table.
- As nodes contact their global contacts within this group the information propogates across the network, thus updating the network view of other nodes in the system.

3.4.1 Populating Finger Tables

The task of populating the contact table is divided into the task of filling local as well as the global portions of the contact table. The local portion is all the neighbours of the node. For a joining node this is obtained by copying it from its successor. Updates from other joining nodes keep this portion up-to-date. The steps in order to populate the global portion are:

- Ascertain the values for its own group number(P) and number of groups(G).
- The number of global contacts is $g = log_2 G$.
- The function uses a binary search type algorithm. Initialize values of high and low as the extremities of the search range(in terms of groups). Here $low = 0$ and $high = G - 1$.
- Repeat the following steps for each of the global contacts starting from the last down to the first i.e $g - 1$ down to 0.
  - $mid = \lceil (low + high)/2 \rceil$
  - If $P < mid \Rightarrow$ the node is in the left half of the subtree under consideration;
    - Choose a node from the right half. Therefore choose an id from the range of groups(mid,high).
    - Change the search range for next iteration: $high = mid - 1$. Continue to next iteration.
  - Else, if $P \geq mid \Rightarrow$ the node is in the right half of the subtree under consideration.
    - Choose a node from the left half. Therefore choose an id from the range of groups(low,$mid - 1$).
    - Change the search range for next iteration: $low = mid$. Continue to next iteration.

3.5 Lookups

Lookups are performed in various situations: a)When a node needs to join the network, it performs a lookup for it's id point to find it's successor in the network, b)When a node is populating the global portion of its contact table, c)When a node chooses a proxy in the VC. In all the above cases the node performs a lookup for a certain id in the id-space. During lookup, a node may choose a subset of the local portion of its contact table, asking each of them to perform lookups independantly. This type of redundant lookup is done to reduce the possibility that a malicious node, appearing on the path in a subset of the lookups, has an influence on the result of the lookup. This *redundancy* is a tunable parameter with a tradeoff of security vs. performance. Since the system is based on DHTs, a series of lookup hops are needed up to resolve a lookup. A node chooses the next lookup hop in the series as the contact that is in the same portion of the id-space as the target id. Each consecutive lookup hop brings the lookup closer to the eventual target id. The process of choosing the next lookup hop continues till we get a node in the same group as the target id. Once that happens, the node looks at the local portion of its contact table for the node that owns the target id. The choice of each lookup hop is based on the following algorithm.

- Ascertain the values for its own group number(P), number of groups(G), the group number of the target id(T).
- The number of global contacts is $g = \log_2 G$.
- The function uses a binary search type algorithm. Initialize values of high and low as the extremities of the search range(in groups). At the start, $low = 0$ and $high = G - 1$.
- Repeat the following steps for contact values $g - 1$ down to 0, until you find the contact that is the next lookup hop.
  - $mid = \lceil (low + high)/2 \rceil$.
  - If $P < mid \Rightarrow$ the node is in the left half of the subtree under consideration,
    - Check which half the target id is in. i.e $T < mid$ or $T \geq mid$
    - If T is in the same half as P,i.e $T < mid$ this contact can't be the next lookup-hop since it is in the other half, so continue to the next iteration.

Change the value of the search range for the next iteration, so $high = mid - 1$.

    * Else if $T \geq mid \Rightarrow$, T is in the other half as P, this contact is the next lookup-hop.

- If $P \geq mid \Rightarrow$ the node is in the right half of the subtree under consideration;
    * Check which half the target id is in. i.e $T < mid$ or $T \geq mid$
    * If T is in the same half as P, i.e., $T \geq mid$ this contact cannot be the next lookup hop since it is in the other half, so continue to the next iteration. Change the value of the search range for the next iteration, so $low = mid$.
    * If $T < mid \Rightarrow$, T is in the other half as P, this contact is the next lookup hop.

3.6 Security Analysis

In this section we aim to analyze the security of the system. Generally mix-based systems are analyzed for the types of attacks they are vulnerable to. Our system is a mix-based system with anonymous tunnel creation along the lines of Tor, Onion Routing, etc. The system is thus inherently vulnerable to some of the attacks that Tor-like systems are vulnerable to[section 2.2]. However p2p anonymous systems like Crowds, Tarzan, and MorphMix have introduced other avenues for the attackers[2.5]. In this section we focus on showing that our system mitigates some of the threats that exisitng p2p anonymous systems face.

3.6.1 Intersection Attack

The intersection attack[2.3] requires that the attacker has full knowledge of the system. In our proposed system, the nodes need not have full knowledge in order to function effectively. It is possible however, for collaborating malicious nodes to pool their individual knowledge of the system in order to accumulate complete system knowledge. We now aim to quantify the amount of resources they need in order to get that information. Since each node knows of all the rest of the nodes in the group, the goal of the attacker would be to place at least one node in each group. That way when the nodes collaborate and pool information they have information of all the nodes in the system. The problem then, is to estimate how many nodes it takes to ensure at least one node in each group. The problem equates to the classic *balls-in-bins* problem referred to in randomized algorithms [26]. On an average it takes $Glog_2(G)$[4] attackers to have one in each node, and to do the same with high probability it takes twice the number, i.e., $2Glog_2(G)$.

The hash function means that the attacker cannot modify or change the id space and where his nodes go. If he owns an IP address, he could place another node in the system, but that is true for any p2p system. We do not address the problem of individual nodes presenting multiple identities in the system- the Sybil attack [18], but do note that the attacker would not be able to choose a part of the id-space to concentrate the malicious nodes at. Also, the last proxy is only dependent on the second to last proxy (second proxy in Tor [9]). So only if the second to last proxy is an attacker can the attacker influence the choice of the last proxy. This happens either by the X%[5] that the attacker owns the node normally, or by the additional chance that an attacker who owns the previous proxy can influence the choice of this proxy to be an attacker too.

3.6.2 Choice of Proxies

We have seen that MorphMix doesn't require nodes to know of all the other nodes in the system. However in not knowing all other nodes, it leaves the participant nodes

[4]The log is base 2 since the groups are organized as a binary tree.

[5]X refers to the percentage of malicious nodes in the system.

vulnerable to choosing a malicious node for a proxy and that node compromising the rest of the circuit. The strength of the system is the ability to choose nodes randomly from the entire set of nodes without knowing all of them, but the drawback is that some of the nodes chosen could be malicious. That seems an expected fallout when one chooses nodes at random from among unknown nodes. The best one can do is hope that the selection of nodes is uniformly at random i.e., the chance of choosing a malicious node as a proxy is just about the same as the percentage of malicious nodes in the system. That is our goal. For a system of with n nodes and c malicious nodes, our goal is that the chance of choosing a malicious node is not any more than c/n. As we show in section 4.6, our system provides results close to this. To minimize the risk of malicous nodes influencing the choice of the next proxy, we use redundancy and distance checking to attempt to identify malicious nodes returned from lookup requests.

Again, since our system is based on a Tor-like VCs, the attacks against them are still valid here. Hence timing analysis and the global passive adversary are also valid threats to our system. Avoiding those threats is beyond the scope of this thesis.

CHAPTER 4

SIMULATIONS

We performed a number of experiments aimed at testing the effectiveness as well as security of the system. We used a 30-bit hash space and considered systems of 1000 and 10,000 nodes. We used 128 and 256 groups for 1000 nodes and 256/512/1024 groups for 10,000 nodes. For each test, we simulated 1000 separate systems and made 1000 lookups per system. The system was simulated in Java. Most of the tests were to check the system under varying degrees of attack. Hence we tested with the percentage of malicious nodes changing from 0 to 20. We capped the tests at 20% malicious nodes to simulate a realistic scenario. Beyond 20% malicious nodes, we believe it would just be a matter of time before the entire system was compromised. 20% malicious nodes indicates a high fraction of total nodes in the system, realizable especially with the employment of bot-nets.

4.1 Routing

We now test our earlier claim that the id-space is organized such that it provides greater resilience to attackers. There are two places where we could encounter a malicious node.

- When a node chooses an id at random for one of the proxies on the VC, that id could be owned by a malicious node.
- While performing a lookup for an id, nodes need to communicate with other nodes in order to resolve the lookup. The lookup is performed in a recursive manner and any of the lookup hops could be malicious, thus potentially affecting the result of

the lookup.

In order to perform a lookup we chose one node at random from the set of all nodes. Lets call this the *source node*. We then chose an id at random from the entire id-space. We call this the *target id*. The source node was then asked to perform a lookup for the target id. The chance that the target id we chose is owned by a malicious node should approximately be the chance that we pick a malicious node from the set of nodes. We cannot avoid the fact that we will intermittently choose malicious target ids when we choose ids at random in an open system like ours. What we need to ensure is that over a number of lookups, this fraction should be equal to the percentage of malicious nodes in the system. What we also want to test is the case where we choose "honest" target ids, but during the lookup we encounter malicious nodes as lookup hops. These malicious lookup hops hold the potential to affect the result of the lookup. In our tests of the strength of routing in the system, if we chose an honest target id and the lookup hops turned out to be malicious nodes, we considered that the lookup *failed*. It must be noted that we used redundancy here to minimize the number of cases where that happened. So *successful* lookups were those where both the target id was honest and not all of the redundant lookups failed. We checked the percentage of the total lookups that were successful, and compared the results with the number of lookups with honest target ids. The difference- the percentage of lookups that failed gives us an idea of how resilient the system was to attackers randomly distributed[1] over the id-space. In a practical scenario, if a few, but not all of the redundant lookups were to fail, the node could distinguish the right result from the wrong one as the closest node whose id is greater than the id being lookup up.

[1]The hash function used ensures that the attackers are randomly distributed.

4.2 Redundancy

We use redundancy to minimize the effect of running into malicious node while looking up an id. The amount of redundancy we choose is a tunable parameter to balance security vs. performance. More the number of redundant lookups, higher is the chance of getting a successful lookup in spite of a subset of the lookups having encountering one or more malicious lookup hops. However, as can be imagined, more the number of lookups, worse is the performance of the system and more is the overhead of lookup messages in the network. Here we try different values of redundancy to pick the least value which does not degrade the security significantly.

4.3 Average Maximum Path Length

When a lookup is performed for an id, the node performing the lookup contacts a number of lookup hops in order to resolve the lookup. The number of lookup hops contacted to resolve the lookup is what we call the *path length* here. Since the system provides security through redundancy, we need to check how much security costs the system. Thus, when a node needs to resolve a lookup, it asks all or a subset of it's neighbours to perform the lookup for it. Since all the redundant lookup requests are shot simultaneously, the delay is equal to the delay introduced by the redundant lookup with the maximum path length. We calculate this average over all the lookups, so as to get an estimate of the amount of delay introduced by the system.

4.4 Detecting Wrong Results

When a node performs a lookup, the result could be tainted due to one or more malicious lookup hops. We chose a number of different id points at random, and observed their “owners” with the aim to devise a safe way of estimating whether the result returned is correct. The aim is to ensure security of the system in the face of malicious nodes, but at the same time, not increase the false positives so much as to create unnecessary burden on the system[2].

In the following sections, we present and analyze the results of the simulations aimed at testing the system w.r.t the aspects listed in this chapter.

4.5 Results

In this chapter we present the results of the simulations aimed at testing the system w.r.t aspects explained in the previous part of this chapter.

4.6 Lookup Success

For the first set of experiments we kept the redundancy at 5 i.e., for groups where the number of local contacts exceeded 5, only 5 of the local contacts were asked to do the lookup. As can be seen from Figure 4.1, about 20% of the target ids were owned by malicious nodes when there were 20% malicious nodes in the system. The value was seen to be consistently equal to the percentage of malicious nodes in the system as we incremented it from 0 to 20%. This shows the resilience of the system. The hash function means that malicious nodes are randomly distributed through the id space since the attacker cannot influence where it goes on the id-space. We see that even for the case of 20% malicious nodes in the system, the number of failed lookups amounted to less than 6%. This proves our assertion that the system is inherently resilient to attackers.

[2]When nodes detect an incorrect resolution to a lookup, they void the lookup and chooses another id and go through the process of lookup all over again

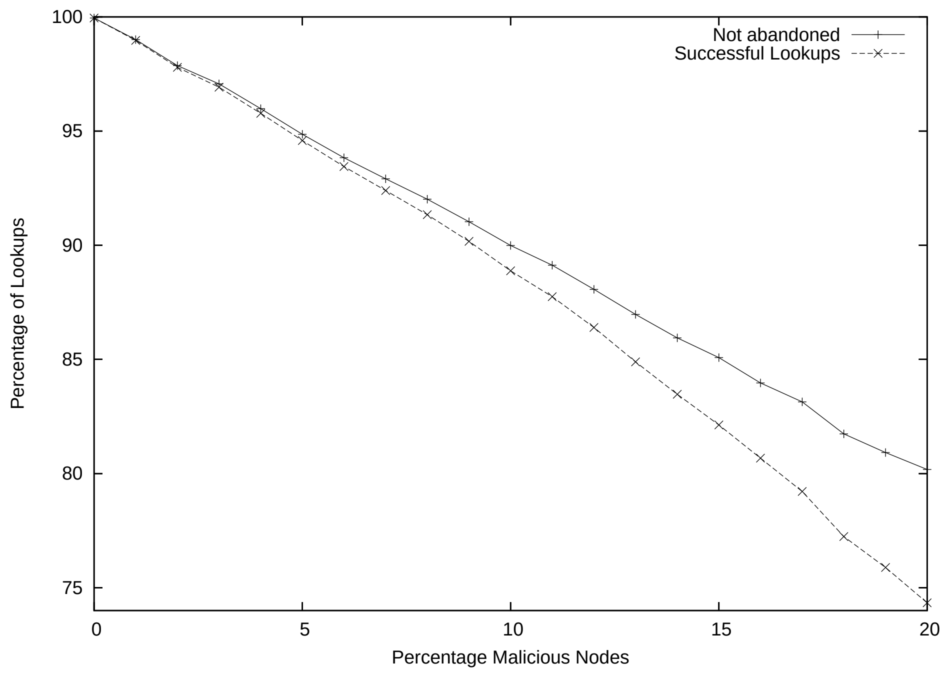

Figure 4.1 Lookup success
System details: 1,000 nodes, 128 groups, Redundancy=5.
X-axis[74:100] Y-axis[0:20]

4.7 Redundancy

The next set of experiments were to check the amount of redundancy required by the system to give reasonable results and to show the effect of redundancy on the success of a lookup. Figure 4.2 shows that as we increased the redundancy from 4 to 5 the number of failed lookups decreased by 2.44% and a further 1.57% as we increased the redundancy from 5 to 6. As the redundancy in the system is increased, nodes ask more of their local contacts to do lookups. This gives them a better chance of finding a path with no malicious lookup hops. However they do bear the burden of an extra lookup. Table 4.1 shows the increase in the number of messages exchanged as the value of redundancy is increased. As soon as the increase in security is not significant as compared to the overhead of the extra lookup, the extra redundancy can be done away with.

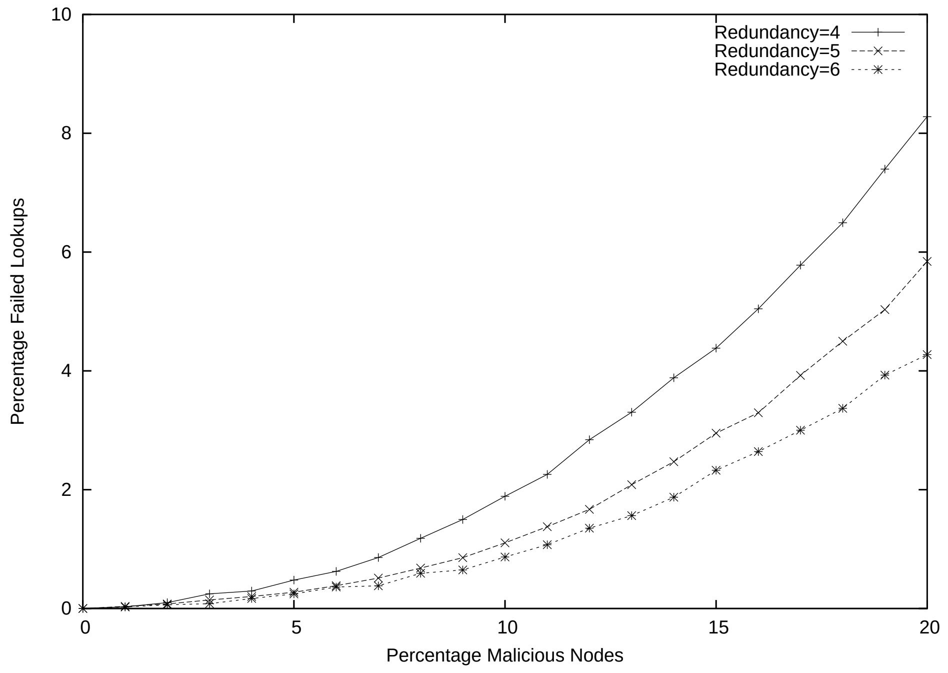

Figure 4.2 Percentage of Failed Lookups vs Redundancy
System details: 1,000 nodes, 128 groups, 20% malicious nodes

Next we did experiments to check general statistics of lookups and the system. Table 4.1 shows information like the average minimum and maximum group population, the number of messages exchanged to perform lookups as well as the average maximum pathlength. The number of messages indicates the overhead the redundancy places on the system. The average maximum pathlength indicates the delay induced in the system. Since the lookups are similar to a binary search on the groups, we expect the number of lookup hops to be about $log_2(G)$. The results show that to be true. Here each of the group members asked to do a redundant lookup is considered the first lookup hop.Of course, the other values in the table remain the same since the redundancy doesn't affect any of them.

Table 4.1 Group, Path, Message Statistics. *System Details: 1000 nodes, 128 groups, 10% malicious nodes.*

Redundancy	Avg. min. group size	Avg max. group size	Avg. messages sent	Avg. max pathlength
4	1.68	15.85	39.63	8.92
5	1.74	15.94	48.53	8.94
6	1.68	15.98	56.42	8.94

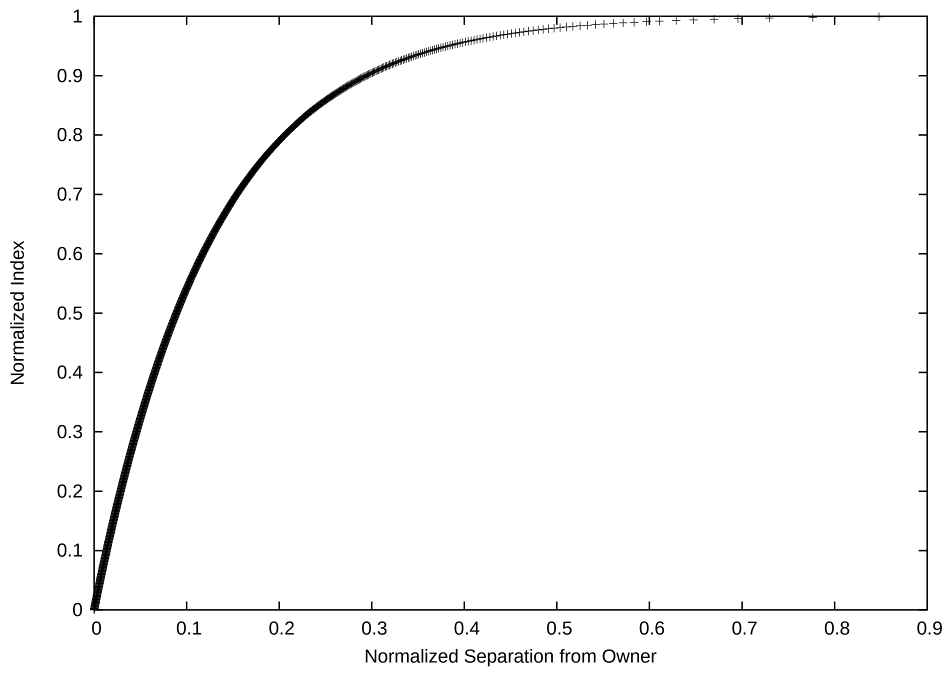

Figure 4.3 Distance Checking
System details: 1000 nodes, 128 groups.

Table 4.2 Percentage Malicious Nodes For Intersection Attack

Nodes/Groups	% Malicious Nodes
1000/128	70
10,000/1024	77

Table 4.3 Distance checking

Max. offset	False Positives	False Negatives 5%	10%	15%	20%
0.5	2.0%	9.2%	17.0%	24.2%	29.9%
0.1	45.8%	0.95%	2.0%	2.9%	3.8%

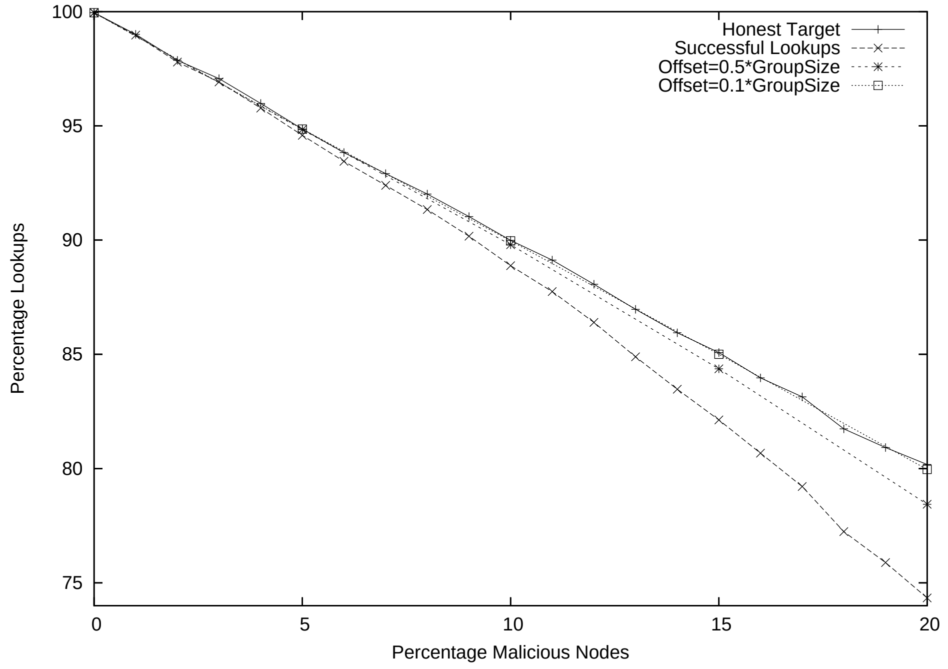

Figure 4.4 Lookup Success With Distance Checking
System details: 1,000 nodes, 128 groups.

We also checked how well distance checking worked. Figure 4.3 shows how the separation between ids and their corresponding owners are distributed. The values are normalized by the groupsizes. Table 4.3 shows the results in numerical format. False positives mean that the honest pick was out-of-bounds. False negatives mean that the picked attacker was in-bounds. The false positive rate is high, but that just means checking several random ids until you get a good one. At 50%, you would check 2 on average and no more than 10 with 99.9% probability. Also for false negatives, the real threat is much lower than the figures show. As shown in Figure 4.1, at 20% malicious

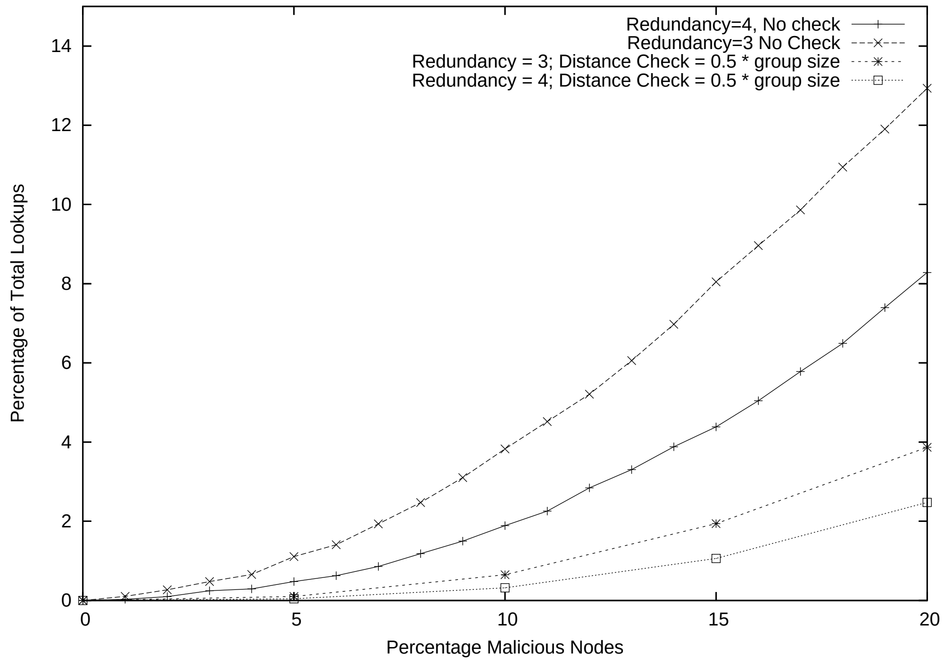

Figure 4.5 Lookup Success With/Without Distance Checking
System details: 1,000 nodes, 128 groups.

nodes in the system, only about 6% lookups turn false, so even if we pick the highest value of false negatives in the table i.e.,29.9%, it implies that 29.9% of those 6%, i.e. 1.8% of the total lookups returned malicious nodes and went undetected. This can be seen in Figure 4.4. With the offset set at 0.5 of group size, there are a few false negatives, but we still avoid a lot of malicious nodes as compared to when there was no distance checking. If we set the offset to a lot stricter 0.1 of group size, we eliminate almost all the undetected malicious nodes.

In order to better illustrate the effectiveness of the distance checking we plotted the percentage of failed lookups for redundancy values of 3 and 4, with and without distance checking. The figure 4.5 shows the significant decrease in the number of failed lookups once distance checking is employed. Another point to be noted is that the offset in this case is taken to be 0.5 of a group size, yet the drop in failed lookups is significant.

We spoke earlier about how many nodes an attacker needed to have in the system in order to be able to perform the intersection attack. We mentioned the value in terms of a formula. What table 4.2 does is show the percentage of malicious nodes that the attacker needs to inject in order to have one node in each group. The values are calculated for more common values of number of groups and nodes. For a system containing 10,000 nodes with 1024 groups the attacker needs to own 77% of the nodes in the system. The values we see are too high and hence we can deduce that the system is resilient to the intersection attack.

CHAPTER 5

CONCLUSION

In this thesis we propose a p2p anonymous communication overlay system based on DHTs. It is based on free-route topology, so the attacker needs to cover more surface in order to observe since there are a larger number of entry and exit points in the system. Since it is based on DHTs it makes for a scalable dynamic system. Due to the use of consistent hash function, attackers cannot influence where the nodes fall on the id space. We have shown that the way the system is organised makes it inherently resilient to attackers. The system uses redundant lookups to mitigate the risk of finding attackers on the path. This redundancy is a tunable parameter to balance performance and security. The system also uses distance checking to determine malicious nodes returned by lookups. We show that with a slightly strict offset, we can reduce the false negatives to 3.8% in a system containing 20% malicious nodes. Nodes have limited knowledge of the system, while still being able to choose proxies at random from the entire system. This helps protect it from the intersection attack. We also show that for an attacker to try to perform the intersection attack, he'd have to control a ridiculously large fraction of the system.

5.1 Future Work

An attacker proxy may attempt to get a fellow attacker as the next proxy by sending back nodes that are too far, when closer nodes exist, so that you choose new points that might be owned by or close to an attacker. We would have liked to have a method to be able to verify lookup resolutions without giving up the property of nodes having partial knowledge. Also, as we mentioned earlier, since the system is based on Tor- the attacks against Tor are still valid here. Hence, timing analysis and the global passive adversary are still threats here.

REFERENCES

[1] 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 (CCS 2002)*, Washington, DC, November 2002.

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

[3] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity: A proposal for terminology," Draft, version 0.14, July 2000.

[4] U. o. S. C. Information Sciences Institute, "Internet protocol-darpa internet program protocol specification," 1981.

[5] J. F. Raymond, "Traffic analysis: Protocols, attacks, design issues and open problems," in *Designing Privacy Enhancing Technologies: Proc. Intl. Workshop on Design Issues in Anonymity and Unobservability*, ser. LNCS, H. Federrath, Ed., vol. 2009. Springer-Verlag, 2001, pp. 10–29.

[6] "Anonymizer web site." Available at `http://www.anonymizer.com`.

[7] A. Engelfriet, "Anon.penet.fi has closed," Available at `http://www.stack.nl/~galactus/remailers/index-penet.html`, Jan 1998.

[8] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, February 1981.

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

[10] A. Back, I. Goldberg, and A. Shostack, "Freedom 2.0 security issues and analysis," Zero-Knowledge Systems, Inc. white paper, Nov 2000.

[11] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *Proceedings of Information Hiding: First International Workshop*, R. Anderson, Ed. Springer-Verlag, LNCS 1174, May 1996, pp. 137–150.

[12] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free mix-routes and how to overcome them," in *Proc. Intl. Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[13] R. Dingledine, V. Shmatikov, and P. Syverson, "Synchronous batching: From cascades to free routes," in *Proc. Privacy Enhancing Technologies workshop (PET 2004)*, 2004.

[14] R. Böhme, G. Danezis, C. Díaz, S. Köpsell, and A. Pfitzmann, "Mix cascades vs. peer-to-peer: Is one concept superior?" [Online]. Available: citeseer.ist.psu.edu/694708.html

[15] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[16] D. Goldschlag, M. Reed, and P. Syverson, "Hiding routing information," *Proc. Information Hiding: First International Workshop*, May 1996.

[17] B. N. Levine, M. Reiter, C. Wang, and M. Wright, "Timing analysis in low-latency mix systems," in *Proc. Financial Cryptography*, February 2004.

[18] J. Douceur, "The Sybil attack," in *Proc. IPTPS*, Mar 2002.

[19] M. Wright, M. Adler, B. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols," in *Proc. ISOC Sym. on Network and Distributed System Security*, Feb 2002.

[20] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communication against passive logging attacks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[21] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, June 1998.

[22] R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," 2001. [Online]. Available: citeseer.ifi.unizh.ch/ratnasamy02scalable.html

[24] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," Berkeley, CA, Tech. Rep. TR-00-010, 2000. [Online]. Available: citeseer.ist.psu.edu/ratnasamy02scalable.html

[26] R. Motawani and P. Raghavan, *Randomized Algorithms.* Cambridge University Press, 1995, ch. 4.

BIOGRAPHICAL STATEMENT

Arjun R. Nambiar was born in Mumbai, India, in 1979. He received his B.S. degree from Mumbai University, India, in Electronics Engineering in 2001, his M.S. degree with a Thesis from The University of Texas at Arlington in 2006 in Computer Science and Engineering. His interests lie in Cryptography, Network Security. He has been part of the Information Security Lab at UTA since its inception. His research interests include Network Security, Anonymity and Privacy on the Internet.