A NEW SOLUTION OF PEER-TO-PEER ANONYMOUS COMMUNICATION

By

YANGBAE PARK

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2012

I dedicate this thesis to my parents.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Shigang Chen. This thesis would not have been possible without his advice, guidance, and persistent help. I would also like to thank Dr. Sartaj Sahni and Dr. Jonathan Liu for serving on the supervisory committee. Their constructive assistance and comments have always inspired me to think creatively. It was my great pleasure and honor to have them on the committee.

In addition, I thank Min Chen, Tao Li, Wen Luo, Zhen Mo, Yan Qiao, and Yian Zhen who have studied and discussed a number of interesting topics with me. I am grateful to Dr. Lingguo Cui for giving me valuable comments. I am also indebted to my other colleagues, including, but not limited to, Inchul Choi, Hokyu Kang, Dunam Kim, and Coralie Richard, who helped me stay sane and happy. I wish them all well in their future endeavors.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A NEW SOLUTION OF PEER-TO-PEER ANONYMOUS COMMUNICATION

By

Yangbae Park

May 2012

Chair: Shigang Chen
Major: Computer Engineering

Anonymous communication prevents network sniffers or any third parties from identifying communication parties. Whenever we use the Internet, our IP addresses are exposed to anyone along the routes; however, these addresses often allow an adversary to trace identities of senders and recipients. Since privacy protection has become more important, the demands for anonymous communication have also increased a lot.

In particular, the Tor network is the most popular and widely used anonymous communication system, but it is not very scalable. Many researchers have suggested peer-to-peer (P2P) based solution to cope with this limitation of Tor. However, none of them have yet offered the anonymity that Tor provides.

Our goal is to improve anonymity of a Tor-like system based on P2P architecture. It should not depend on any central authority or trusted third party that limits scalability. In addition, it should be resistant to large-scale coordinated eavesdropping. In this thesis, we propose a new anonymous communication solution that satisfies these requirements.

CHAPTER 1
INTRODUCTION

Anonymous communication prevents network sniffers or any third parties from identifying communication parties. Cryptography has solved many issues for confidentiality and integrity. However network addresses in packets are still exposed, and anyone along the routes can monitor the addresses. Such addresses often become critical hints, enabling an adversary to trace identities of senders and recipients. Since privacy protection has become more important, the demands for anonymous communication have also increased.

The Tor network [1] is the most popular and widely used anonymous communication system. Tor allows hundreds of thousands of users [2] to surf the Internet without scarificing privacy. There are a variety of users and countries accessing Tor, from journalists in Egypt to Iranian, Indian, Japanese, and Russian embassies [3].

In the Tor network, the number of trusted directory servers is limited, and all users have to store a global view of the system. Although this design prevents attackers from poisoning the directory or circuits, it causes a scalability problem. McLachan et al. [4] show that traffic to manage a global view will soon become larger than actual anonymous traffic. With regard to this issue, some researchers consider adapting peer-to-peer (P2P) approaches on Tor or its variants [5] [6] [7] [4] [8] [9].

P2P architecture cannot be not easily applicable for Tor because it causes a new problem to anonymous communication. For example, AP3 [7] and Salsa [6] utilize Distributed Hash Table (DHT) to distribute centralized overhead, and users are required to maintain only a partial view of a system. However Mittal and Borisov [10] showed that attackers can reveal identities of communication parties during the lookup procedure.

More recently, NISAN [8] has proposed an anonymous lookup mechanism. NISAN provides better redundancy and bound checking against active attackers while it hides the relationship between users and relays from passive attackers. However Wang et al. [11] show that a group of compromised nodes may still break anonymity during lookup.

In this thesis, we propose a new P2P anonymous communication solution based on Chord [12], which is widely used by many P2P researchers and applications. Our solution is completely decentralized, and it is robust against a large scale adversary. It does not rely on any trusted third party on the system, nor redundant activities that can cause another type of vulnerability.

The key idea of our solution is called *predictive lookup*. We find a mechanism to predict another node's routing table. Based on this mechanism, we design a new lookup protocol to initiate anonymous communication. We further develop our idea into a general condition, regardless of the size of nodes, or the density of the network.

The rest of this thesis is organized as follows. In Chapter 2, we describe previous research and applications. In Chapter 3, we show details of routing table prediction which will be later a key mechanism for our new solution. In Chapter 4, we propose a new solution, and we expand our idea to generalize the solution. In Chapter 5, we conducted a simulation, and the experimental results are shown. Finally we conclude in Chapter 6.

CHAPTER 2
PREVIOUS RESEARCH AND APPLICATIONS

2.1 Low Latency vs. High Latency Anonymous Communication

Anonymous communication can be categorized based on latency. Low latency communication is typically for interactive applications which require short delay of transmission. For instance, web browsers will show HTTP Error 408 (Request timeout) unless they retrieve a web page within a few seconds. Likewise, most people will close voice chatting if their partners become mute.

Although low latency communication can benefit most applications, it is weak against a powerful global adversary who can monitor the entire network [13] [14] [15] [16]. The global adversary can measure end-to-end data transmission and reception time, and they can then discover senders and corresponding recipients.

High latency communication usually takes hours or even days to transmit a message, and it is robust against a global adversary. Mixminion [17] and Mixmaster [18] are well-known examples of high latency communication systems. However due to high latency, only limited applications are able to use high latency communication systems, such as emails.

In this thesis, we focus on low latency anonymous communication. We assume that there is no global attacker on the Internet, but we still consider the existence of semi-global attackers.

2.2 Attack Models for Anonymous Communication

Typically attack models for anonymous communication are classified into active and passive attacks. Active attackers join in and manipulate anonymous communication channels actively so that they can break anonymity, or they damage the system itself. Although active attacks could cause critical damage, they are usually visible due to abnormal activities.

On the other hand, passive attackers only observe some portion of anonymous traffic. Unlike active attackers, passive attackers do not expose themselves, so they are rarely detected. Furthermore, Mittal and Borisov [10] show that several defense techniques against active attacks create new vulnerabilities of anonymity from passive attacks.

We focus on passive attacks particularly in this thesis.

2.3 Anonymous Proxy Servers

A proxy server is a network node that forwards incoming packets to others. An anonymous proxy server has an anonymizer that conceals the original sender's identity. Figure 2-1 shows how an HTTP anonymous proxy system works.

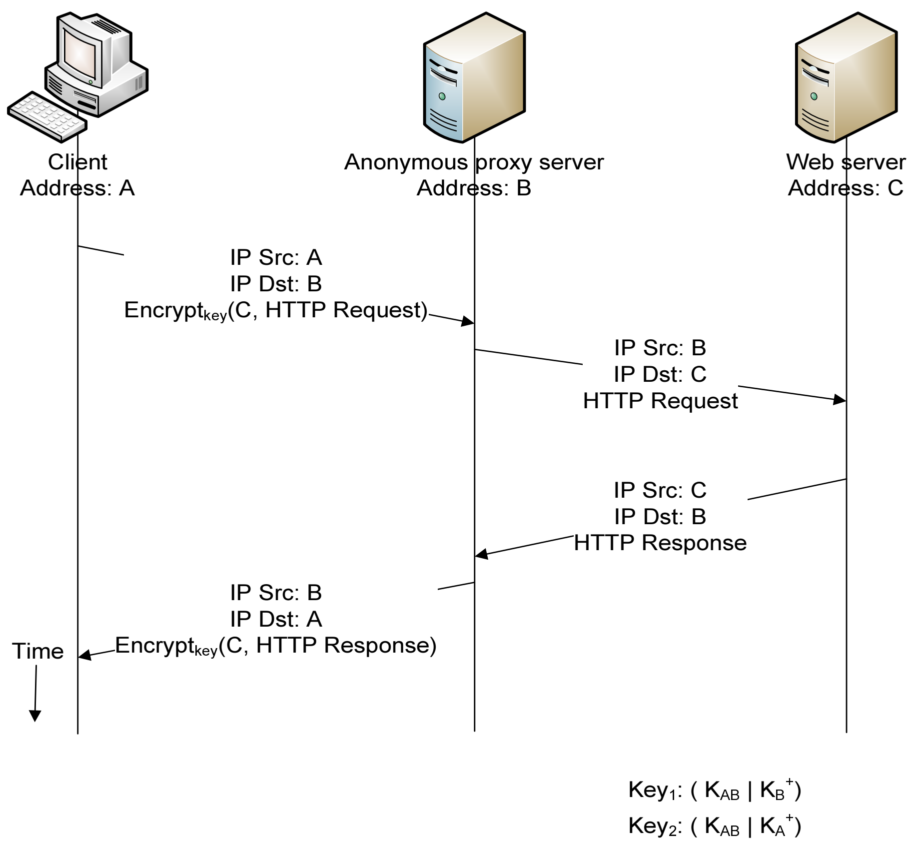


Figure 2-1. Anonymous HTTP proxy

First client A encrypts a HTTP request message with a shared secret key K_{AB} or B's public key ${K_B}^+$. Next A sends the encrypted request to the anonymous proxy server B. Authentication is optionally required at this moment. Once B accepts A and its message, the anonymizer removes any A's identity from the message. Proxy B then forwards the anonymous message toward the actual destination C. When B receives a response from C, it encrypts the response again with K_{AB} or A's public key ${K_A}^+$, and then forwards to A.

A key role in this model is the proxy B. If B is compromised, attackers can trace A's address. Moreover even without controls over B, attackers can still use traffic or timing analysis attacks to find a connection from A to B, and a corresponding connection from B to C.

2.4 Onion Routing and Tor

Reed et al. [19] proposed a freely available anonymous communication system called onion routing. Onion routing utilizes multiple network nodes to prevent eavesdropping and traffic analysis. Tor is a predominant implementation of onion routing, serving hundreds of thousands of users [2]. Tor typically calls the network nodes relays, and anyone can run a Tor relay voluntarily.

Onion routing is based on Public Key Encryption. Each relay or user has a public and private key pair. Public keys are available for all, while private keys should be kept in secret. The list of relays is called the directory. In the Tor network, all clients, as well as several trusted directory servers maintain the directory.

Before transmitting actual messages, an onion routing client has to choose several relays. Typically Tor selects three random relays. The more relays the client uses, the higher anonymity and performance are obtained [20], but latency will also increase more.

Next, the client creates a virtual circuit composed of the chosen relays, as Figure 2-2 shows. A virtual circuit is a network tunnel on top of the Internet. If the client selects three relays, messages will be transmitted through the three relays.

As a powerful adversary may attempt to trace the sequence of relays to discover communication parties, Tor expires each virtual circuit every 10 minutes.

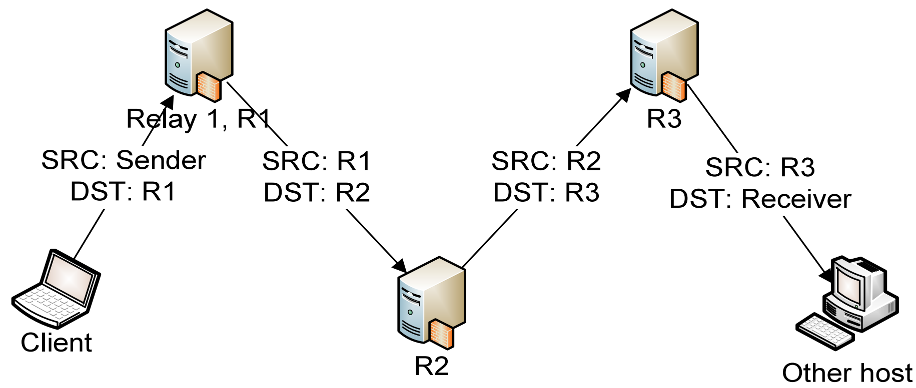


Figure 2-2. An example of a virtual circuit

In order to hide identities of senders and recipients, an actual message is wrapped in several layers of encryption. Figure 2-3 shows how a client transmits a message through a virtual circuit that consists of three relays $R1$, $R2$, and $R3$. It first encrypts the message with $R3$'s public key ${K_{R3}}^{+}$. Next, the client encrypts $R3$'s address and the previously encrypted message with $R2$'s public key ${K_{R2}}^{+}$. This procedure continues until the client encrypts with the first relay's public key, which is ${K_{R1}}^{+}$ in Figure 2-3.

When the first relay $R1$ receives the wrapped message from the client, $R1$ decrypts the message with $R1$'s private key ${K_{R2}}^{-}$ to extract the payload and the next destination. This is like peeling an onion, but $R1$ can only peel the first layer because the other layers are encrypted with other keys, ${K_{R2}}^{+}$ and ${K_{R3}}^{+}$.

From the perspective of $R1$, the destination is $R2$, and no other relays or destination are visible because their addresses are encrypted with different keys. Once $R1$ forwards the message to $R2$, $R2$ has no way to recognize the client's existence, though $R2$ can see $R3$. $R3$ knows the final destination, but it is unable to trace who sent this message. Thus, no one can deanonymize the communication.

Although Tor is a very successful onion routing application, it depends on a single directory authority. In addition, each client manages a global picture of the network, and

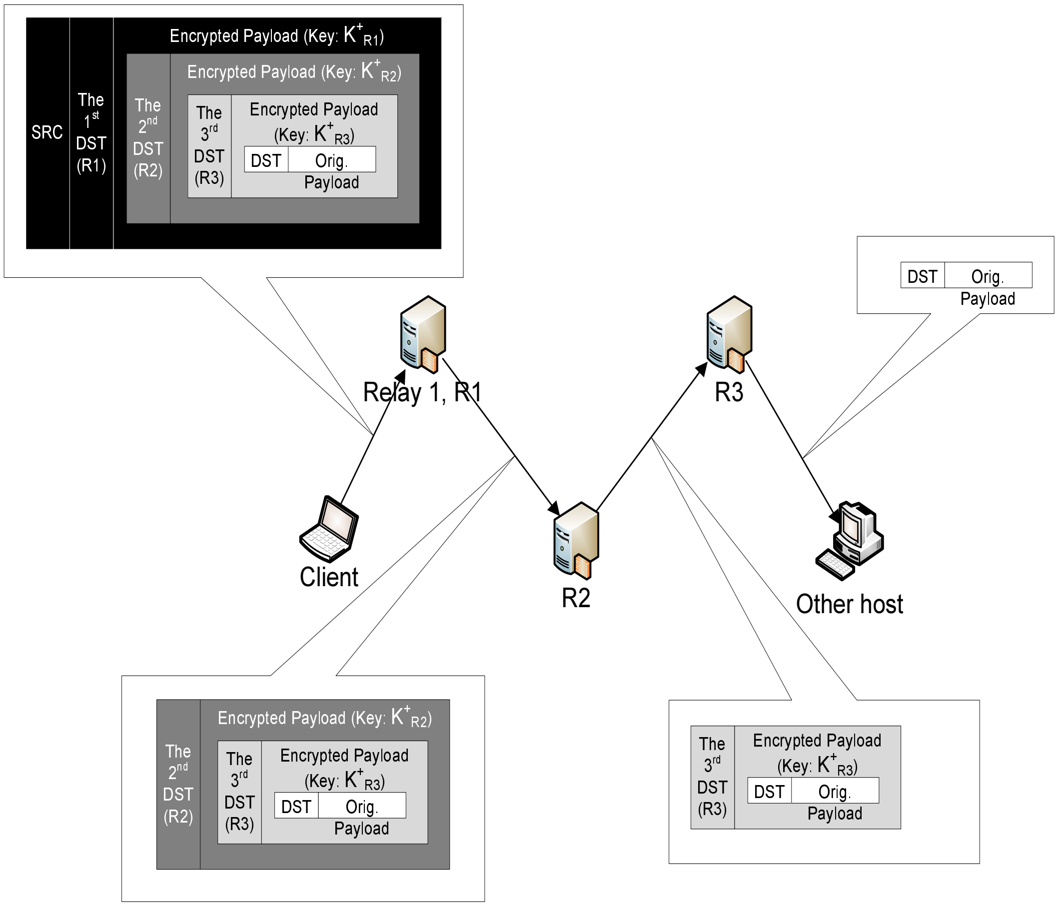


Figure 2-3. Message encapsulation and decapsulation ($K_R{}^+$: R's public key)

these cause scalability issues. McLachan et al. [4] show that traffic to manage the global view will become larger than actual anonymous traffic in the near future.

2.5 Distributed Hash Tables

Peer-to-peer (P2P) systems have been very successful in addressing resource sharing and content access over the Internet [12, 21–23]. Some researchers have considered P2P approaches to resolve scalability issues in a Tor network [7] [6] [4] [8] [9]. A variety of P2P systems are available now, but they are briefly categorized into centralized and decentralized systems. Decentralized systems are again classified into unstructured and structured systems.

In centralized P2P systems, a single directory authority maintains a centralized directory. This structure has several advantages. It prevents malicious nodes from

Table 2-1. Comparison of P2P

P2P types	Lookup time	Storage requirement	Application(s)
Centralized	$O(1)$	$O(N)$	Napster, eDonkey, BitTorrent
Decentralized & Unstructured	$O(N)$	$O(1)$	Gnutella
Decentralized & Structured	$O(lg(N))$	$O(lg(N))$	DHTs, BitTorrent

poisoning the directory, and it also responds to queries very quickly because the directory is stored in a local area. However the directory becomes bottlenecked when the number of nodes and queries soars, so this architecture is not very scalable.

Decentralized P2P systems are based on overlay networks. Each node stores and shares only a small portion of the directory. In particular, unstructured P2P systems do not impose any topology or structure on the network, so the network expands arbitrarily. In an N-node system, this causes querying time to expand to $O(N)$ in the worst case.

On the other hand, a structured P2P system has a consistent protocol that restricts nodes from forming an inefficient overlay network. Distributed Hash Tables (DHTs) are of this class. Because of the strict structure, important operations, such as joining, Lookup, and quitting, can be done within $O(lg(N))$ or $O(lg^2(N))$ [12]. Thus, DHTs are generally faster than unstructured P2P systems, and more reliable than centralized P2P. Chord [12], Pastry [24], CAN [25], and Tapestry [26] are famous examples of DHTs.

2.6 DHTs and Tor

DHTs have been adapted by many researchers to resolve Tor's scalability issues, and Salsa [6] is one of the pioneering works. Salsa relies on a DHT system to store directory information. It calculates a cryptographic hash value of the node's IP address to create an identity in the DHT. Unlike file sharing applications, anonymous communication does not need external data to share, such as files, so nodes only share directory information. When a Salsa client needs to locate a relay, it generates a random value, and finds the corresponding node which is unique in the DHT id space. However later Mittal and Borisov [10] show that this is not adequately secure. Moreover they also prove that Salsa's defense mechanism against active attacks ironically increases threats of passive attacks.

Torsk [4] proposes *secret buddy* scheme. Instead of anonymizing lookup itself, a Torsk client executes random walks to select *secret buddy* nodes, and these nodes will serve as proxies during lookup. The client then generates a random value in the DHT id space, and find the corresponding node to select a relay. However Wang et al. [11] present buddy exhaustion attacks, and this blocks honest nodes from choosing a *secret buddy*. They also show that Torsk is weak agains passive attacks, as the random value is leaked to others. This enables intermediate nodes to query the random value, which eventually exposes the relay.

Panchenko et al. [8] introduce an alternative approach, named NISAN. In NISAN, a client also needs a random value x. However, instead of announcing x to other nodes, the client asks other nodes to send their routing tables[1] . This prevents other nodes from knowing the value x, which should be kept secret.

Although NISAN protects x from being directly revealed, it is still far from perfect. Wang et al. [11] prove that attackers can still shrink range of x significantly. This is called *range estimation*, and it is based on the fact that a client will query only nodes preceding x. When querier Q in Figure 2-4 queries to a compromised node C, C can estimate x's boundary δ_x as follows:

m: the number of bits in a given id space

id_n: the identifier of node$n$

$$\delta_x = (id_{MAX}, id_{MIN})$$

[1] NISAN is based on Chord-like DHT, and Chord calls routing tables finger tables. In this thesis, we always use routing tables to prevent confusion.

$$id_{MIN} = id_C$$

$$id_{MAX} = (id_Q + 2^i) mod 2^m$$

where $id_Q + 2^{i-1} < id_C < id_Q + 2^i$ in the DHT id space ring.

This estimation is recursively applicable when queries leak to a group of malicious nodes belong to a single adversary.

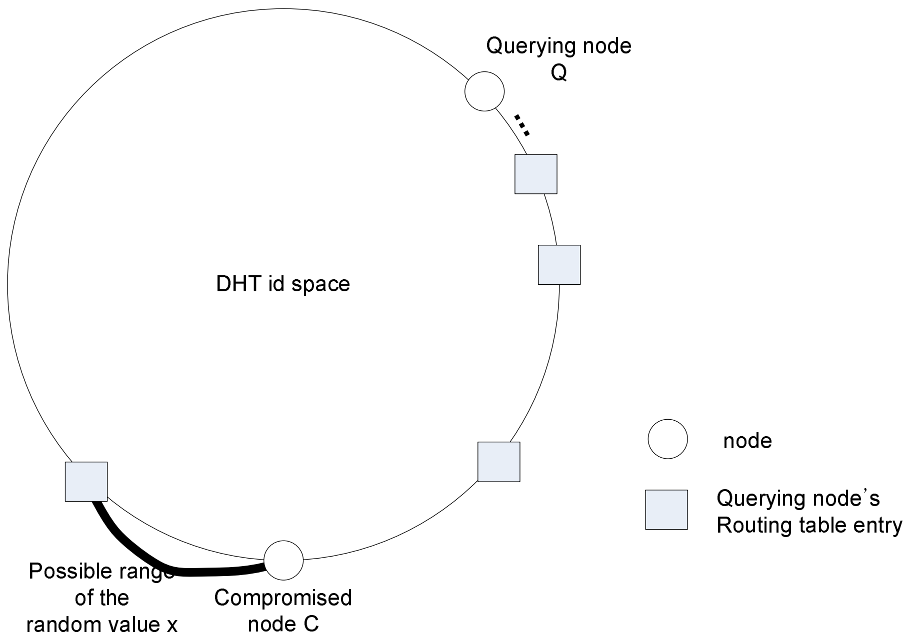


Figure 2-4. *Range estimation* during lookup in NISAN

CHAPTER 3
MOTIVATION

In this chapter, we describe motivating ideas related to our new solution. It consists of several steps. We first explain a lookup procedure of Chord-like DHT, and then discover how to predict other nodes' routing tables.

3.1 DHT Lookup

Every DHT node has a single m-bit identifier (id) positioned in a shared id space. Typically an id is generated by running a cryptographic hash function such as SHA-1. For simplicity, we use small id spaces to describe examples. Figure 3-1 shows 5 nodes in 8 bit DHT id space.

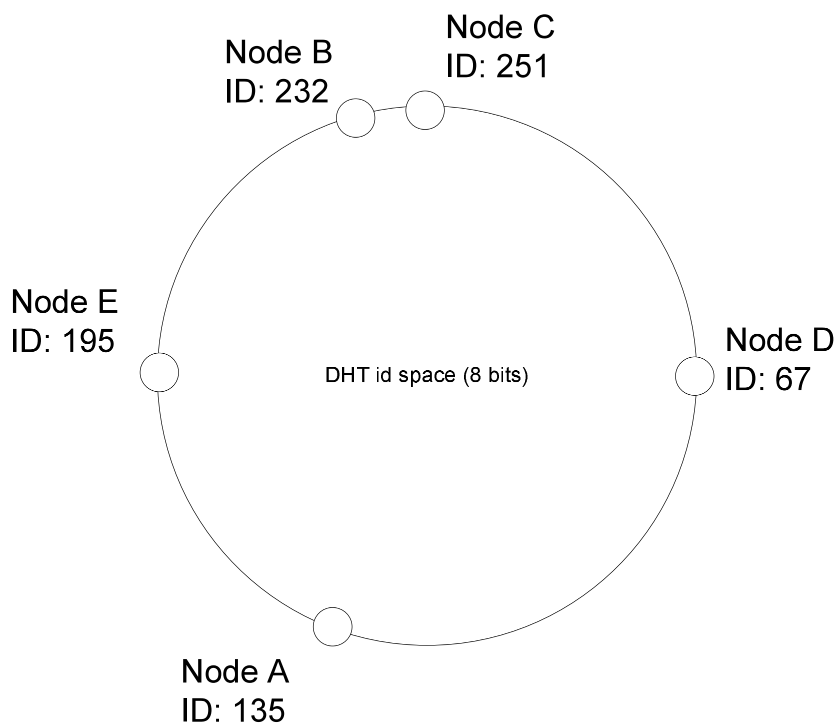


Figure 3-1. DHT nodes

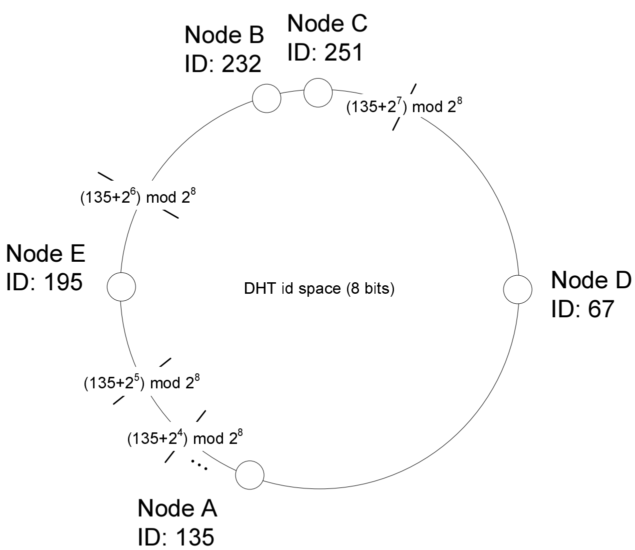
Chapter 2 shows that a DHT system does not rely on a central directory service. Instead, every DHT node n has a routing table T_n that consists of m routing entries. Each entry $E_{n,i}$ has a key $k_{n,i}$ and its corresponding node $o_{k_{n,i}}$.

$$T_n = \{E_{n,i} | 0 \leq i < m, i \in \mathbb{Z}\}$$

$$E_{n,i} = \{k_{n,i}, o_{k_{n,i}}\}$$

$$k_{n,i} = (id_n + 2^i) mod 2^m$$

$o_{k_{n,i}}$ is a node whose id is exactly $k_{n,i}$ if it exists. Otherwise it is the first successor of $k_{n,i}$. Figure 3-2 shows an example of node A's routing table when $id_A = 135$ and $m = 16$. Since there is no node between $id_A + 1$ and $(id_A + 2^5) mod 2^8$, Node E is the first successor of keys $k_{A,i}$, where $0 \leq i \leq 5$. Thus, node E becomes $o_{k_{A,i}}$. Node B is the first successor of the key $(id_A + 2^6) mod 2^8$, so B becomes $o_{k_{A,6}}$. Node D is $o_{k_{A,7}}$ likewise.



i	key	owner
0	$(135+2^0)$ mod 2^8 = 136	E
1	$(135+2^1)$ mod 2^8 = 137	E
2	$(135+2^2)$ mod 2^8 = 139	E
3	$(135+2^3)$ mod 2^8 = 143	E
4	$(135+2^4)$ mod 2^8 = 151	E
5	$(135+2^5)$ mod 2^8 = 167	E
6	$(135+2^6)$ mod 2^8 = 199	B
7	$(135+2^7)$ mod 2^8 = 7	D

Node A's routing table

Figure 3-2. A DHT routing table example

Since every node already knows m keys and corresponding nodes, the lookup function returns immediately for any of the m keys. However when a node n needs to find another key k' where $k' \notin k_{n,i}$, n has to ask the biggest preceding node from k' in its routing table T_n. This procedure continues iteratively or recursively until it reaches $o_{k'}$. Since each hop reduces at least half of the possible range of k' in m-bit id space, the lookup operation can be done within $O(m) = O(lg(N))$ hops. Figure 3-3 describes an example of multi-hop queries, and Figure 3-4 shows the pseudo-code of the DHT query function.

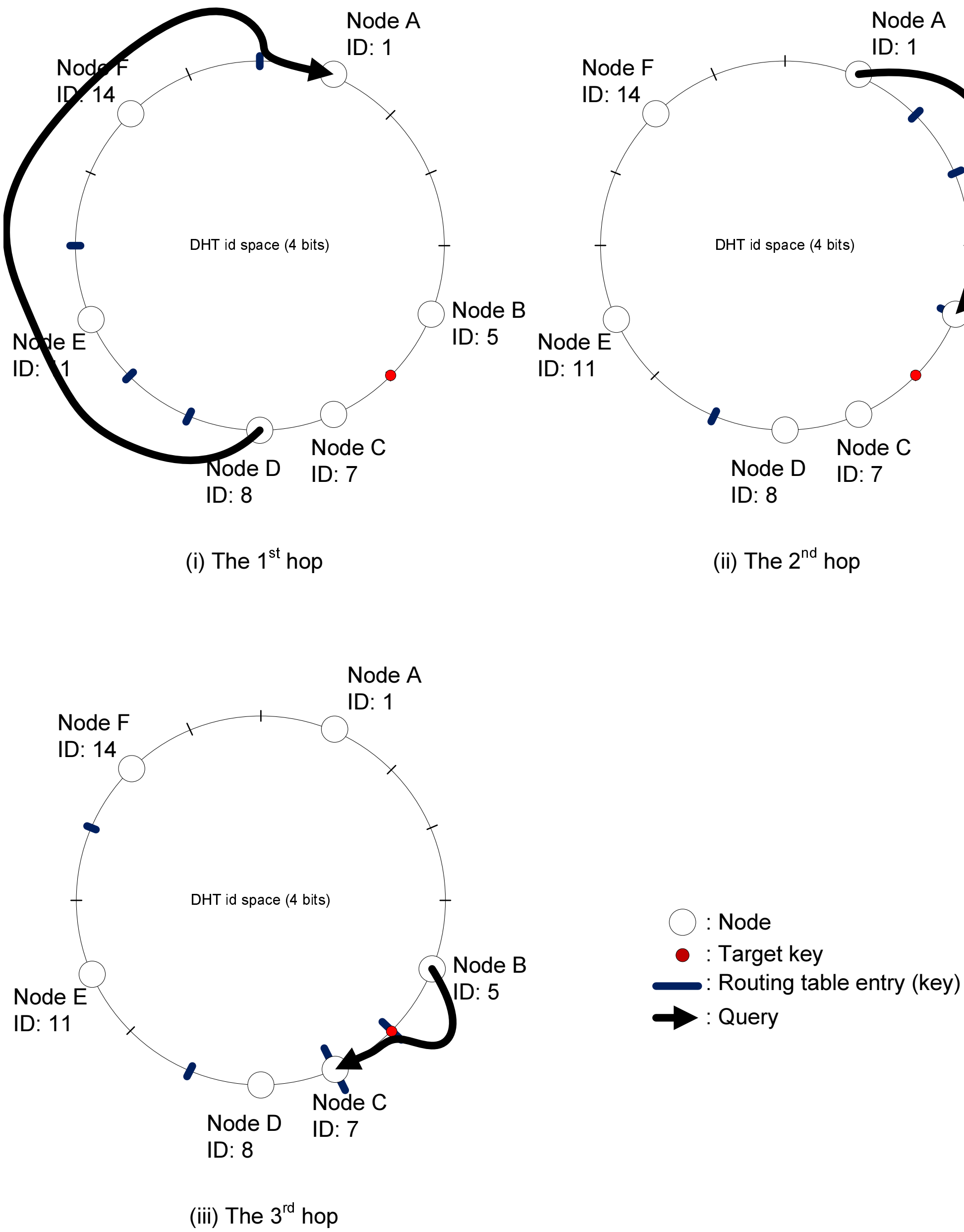


Figure 3-3. A scenario of multi-hop queries

3.2 Routing Table Prediction

Let's first assume that a DHT id space is fully filled, like Figure 3-5. In Figure 3-5, node A shall have node B, C, and E in its routing table because the distance from A

```
find_owner_by_traditional_lookup(n, id)
{
	n' = find_predecessor(n, id)
	return n'.successor
}

find_predecessor(n, id)
{
	n' = n
	while (NOT( id ∈ (n', n'.successor] ))
		n' = cloest_preceeding_node(n', id)
	return n'
}

closest_preceeding_node(n, id)
{
	for i=0 to m-1
		if (n.routingtable[i].node ∈ (n, id))
			return n.routingtable[i].node
	return n
}
```

Figure 3-4. Pseodu-code of lookup function

to B, C, or E is exactly the power of 2. Likewise, node C shall have node D, E, and G. Node D shall have node E, F, and H.

We note that nodes A, C, and D must contain node E in their routing tables. In other words, E is reachable from A, C, or D within a single hop. The following formula shows how to derive a collection C_{id_T} of keys id_n of nodes that must have a specific node T in their routing tables. C_k is the generalized version of C_{id_T} for any id k.

$$C_{id_T} = \{id_n | id_n = (id_T - 2^i) mod 2^m, 0 \le i < m, i \in \mathbb{Z}\}$$

$$C_k = \{id | id = (k - 2^i) mod 2^m, 0 \le i < m, i \in \mathbb{Z}\}$$

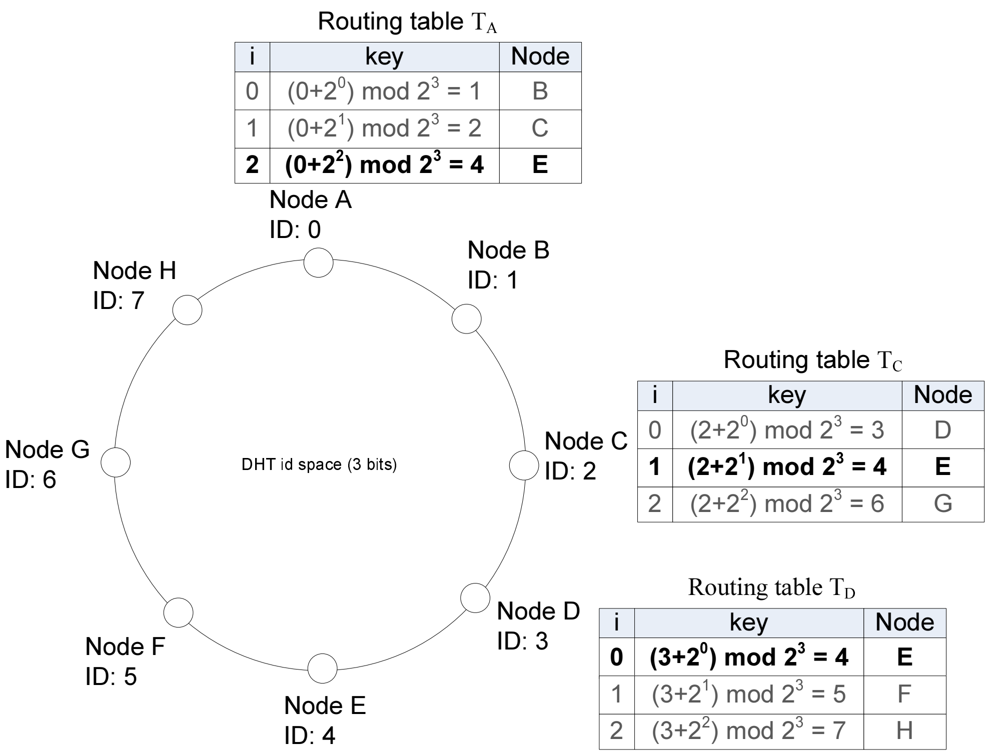


Figure 3-5. Fully-filled 3-bit id space

CHAPTER 4
PROPOSED SOLUTION

In this chapter, we explore a new lookup mechanism called *predictive lookup* to obfuscate *range estimation*. We first assume that the DHT id space is full, and then we generalize our method by removing the assumption. We also deal with how to build a virtual circuit using the new lookup mechanism to make it even more difficult for attackers to estimate the range.

4.1 Predictive Lookup: Locating a Relay in a Special Condition

The previous chapter shows that when DHT id space is full, we can predict C_k, a set of m nodes that have a specific key k in their routing tables. We call C_k *prediction table*. With this knowledge, we can design a new lookup process.

First we generate a random id x and corresponding *prediction table* C_x. Next we generate a random index i ($0 \leq i < m$) to choose the ($i + 1$)-th entry in C_x. We use x_{alt} to denote the key of the selected entry, and o_{alt} to denote the corresponding node whose id is x_{alt}. Once o_{alt} is discovered by a traditional DHT lookup, the original target x is then reachable within a single hop because the $(i+1)$-th routing entry of o_{alt} tells x. We name this method *predictive lookup version 1*.

Figure 4-1 shows an example of how node H locates a relay. Node H first generates a random id $x = 4$, and then it builds the prediction table $C_x = C_4$. As the number of bits of the DHT id space is three, C_4 shall consist of three prediction table entries. Next, node H randomly chooses one entry on C_4 to select x_{alt}. In this example, node H selects the second entry whose index is $i = 1$, and whose key is $(2 - 2^i) mod 2^8 = 2$. This key is called x_{alt}. Once $x_{alt} = 2$ is selected, node H sends queries to find the node o_{alt} whose key is x_{alt}. In this example, o_{alt} is node C. After discovering node C by traditional lookup, node H can jump to $x = 4$ directly from node C because node C's second routing entry tells the key $x = 4$ and corresponding node o_x, which will serve as a relay for anonymous communication.

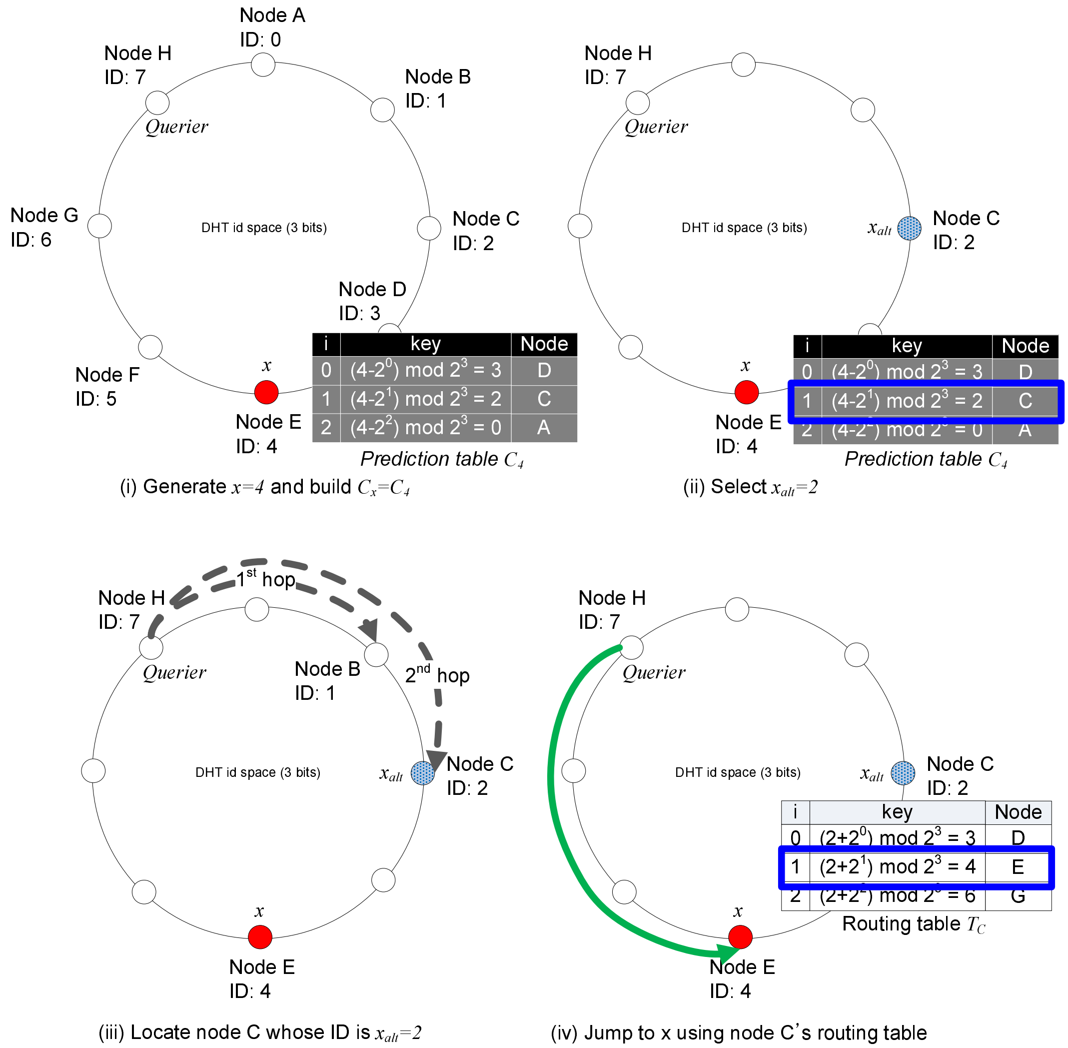


Figure 4-1. *Predictive lookup version 1*

This mechanism suppresses range estimation because it forces the majority of queries to head to x_{alt}. Therefore passive attackers are highly unable to estimate the correct range of x. Figure 4-2 shows the pseudo-code of *predictive lookup version 1*.

```
find_owner_by_predictive_lookup_v1 (n, x)
{
        i = random([0,m))
        n' = find_predecessor(n, (x-2^i) mod 2^m)
        return n'.routingtable[i].node
}
```

Figure 4-2. Pseudo-code of *predictive lookup version 1*

4.2 Predictive Lookup: Generalized Version

Although *predictive lookup version 1* limits *range estimation*, this is not always applicable. Practical DHT id spaces are so big; for instance, Kademlia [27] is a famous DHT protocol used by many applications, such as BitTorrent, and it has a 160-bit id space. Such an id space is so spacious that it is unrealistic to be filled out.

When *predictive lookup version 1* locates o_{alt}, it returns the first successor of x_{alt} unless o_{alt}'s id is exactly x_{alt}. Because o_{alt} succeeds x_{alt}, o_{alt}'s (i+1)-th routing entry also refers o_x's successor denoted by o_{xsucc}. Thus, the querier will fail to locate the correct o_x. In Figure 4-3, the querier generates $x = 8$ in the 4-bit id space, and it chooses the last entry on C_8, so $x_{alt} = 0$. Since there is no such node whose id is 0, node A becomes o_{alt}. If the querier follows node A's last routing entry, it will eventually arrive at node F, which is an incorrect destination.

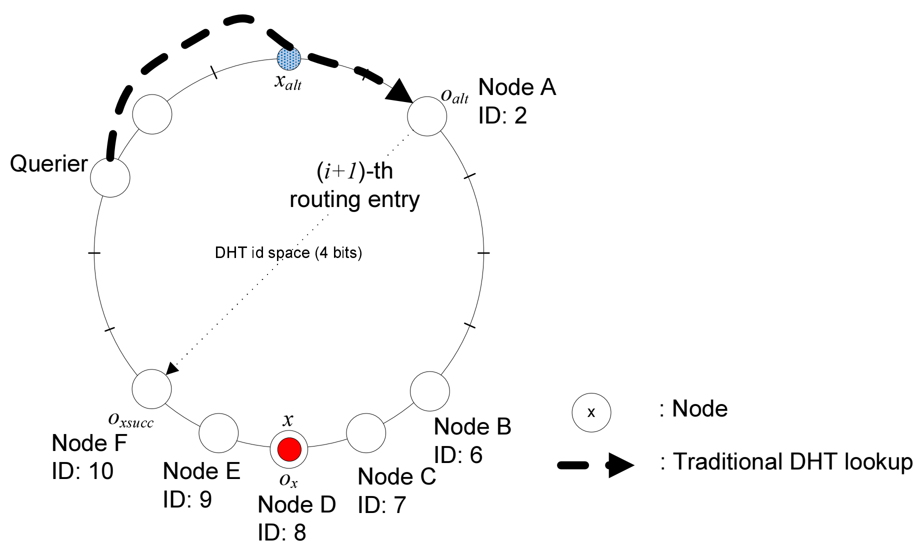


Figure 4-3. Failure of *predictive lookup version 1*

Such a failure causes additional lookup to relocate x. There are two ways to relocate x from the incorrect destination o_{xsucc}. The first option is searching backward from o_{xsucc}. This requires to send a query to every node between x and o_{xsucc} because each node only holds the closest predecessor pointer in the backward direction. The other option is searching forward, but this is also a bad idea because x is too far away from o_{xsucc}. Therefore, all these options are risky enough to enable *range estimation* again.

We focus on how to minimize the relocation process. Instead of looking for o_{alt} which is the first successor of x_{alt}, the querier can search o_{alt}'s first predecessor $o_{altpred}$. In Figure 4-4, node A is o_{alt}, whereas node G is $o_{altpred}$. $o_{altpred}$ can be obtained within constant time because each Chord node maintains a predecessor pointer. We note that $o_{altpred}$ is the closest predecessor from x_{alt}, and o_{alt} is the closest successor from x_{alt}. Thus, unlike o_{alt}, $o_{altpred}$'s routing entry points x's predecessor, which is denoted by o_{xpred}. Since Chord id space is directional, the new distance from o_{xpred} to x is shorter than either the forward distance or backward distance from o_{xsucc} to x. Figure 4-5 compares the new distance with the backward distance when nodes are near-uniformly distributed.

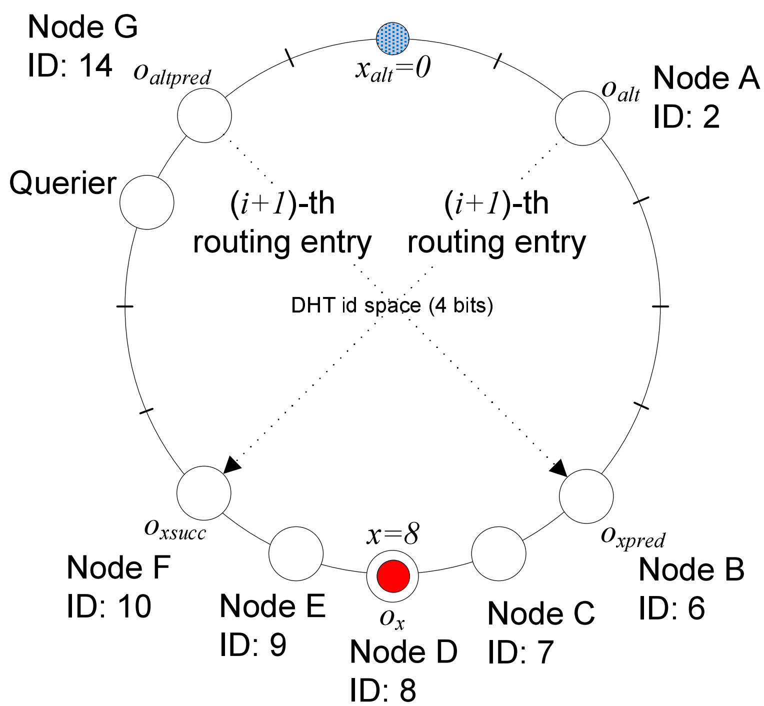


Figure 4-4. $o_{altpred}$ and o_{xpred} in the 8-bit DHT id space

Although this method does not guarantee a one hop increment during lookup, it adds reasonably small hops in general because the distance from o_{xpred} to x is typically shorter than the distance from the querier to x_{alt}. We name this method *predictive*

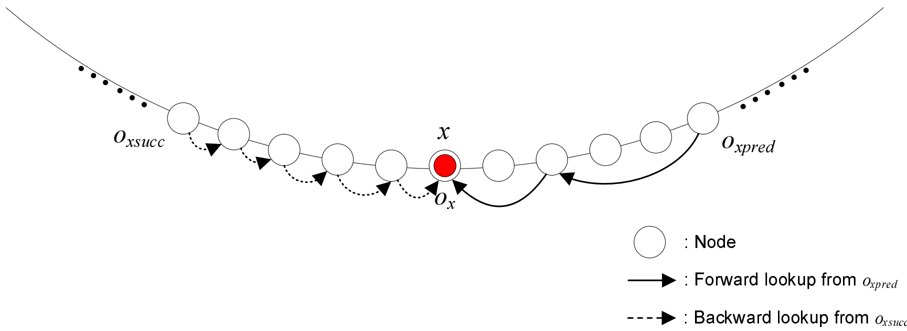


Figure 4-5. o_{xsucc} vs. o_{xpred} in a large DHT id space

Table 4-1. Classification of results of range estimation for *predictive lookup*

Class	Leakage while locating x_{alt}	Leakage while locating x	Safety
Safe	X	X	Safe
Misestimated	O	X	Safe
Leaked	X	O	Unsafe
Confusing	O	O	Almost safe

lookup version 2, and Figure 4-6 and Figure 4-7 show the revised way to locate x using *predictive lookup version 2*. For simplicity, Figure 4-6 is drawn in a recursive way although it is actually done iteratively.

With regard to *predictive lookup version 2*, an attempt for *range estimation* results in one of following:

1. The relay is completely safe: when no query is leaked during the whole process, the relay is obviously safe.
2. The relay is misestimated: when one or more queries only during the first traditional lookup step are leaked, the attackers misestimate the range of the relay.
3. The relay is leaked: when one or more queries only during the *predictive lookup* step are leaked, the attackers can correctly estimate the range of the relay.
4. The relay is confusing: when both traditional lookup step and *predictive lookup* step are leaked, the attackers have to decide which range includes a correct relay. We use shuffling and concurrent querying to confuse attackers even more in this scenario. More details about the techniques are described in Section 4.4.

4.3 Yet Another Issue of Predictive Lookup and Solution

There is yet another minor issue: which entry does a lookup initiator have to choose in C_x? There are total m entries available in C_x, but choosing a key nearby the target x

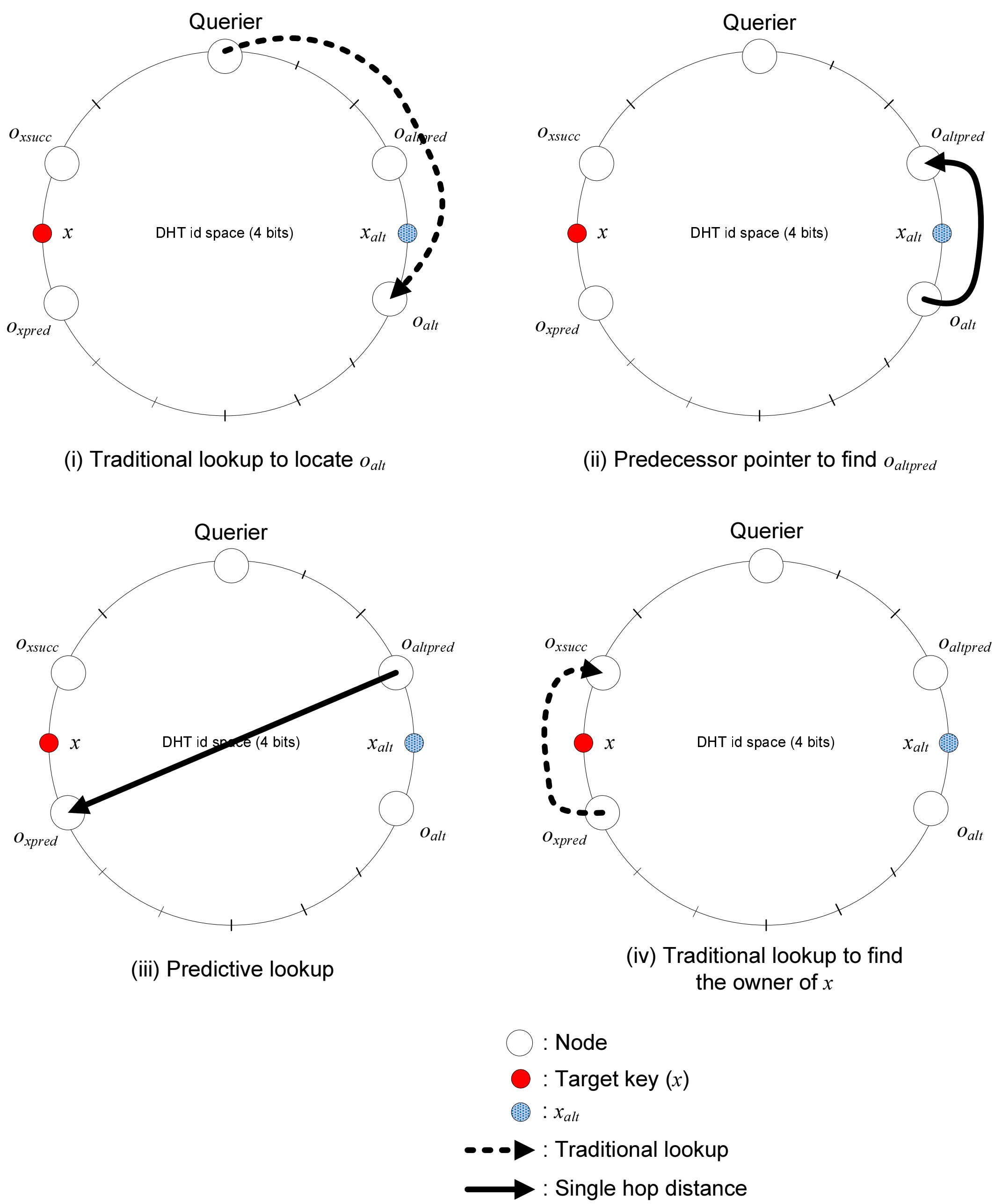


Figure 4-6. Locating x using *predictive lookup version 2*

```
find_owner_by_predictive_lookup_v2 (n, x)
{
        i = random([0,m))
        n' = find_predecessor(n, (x-2^i) mod 2^m)
        n' = find_predecessor(n', x)
        return n'.successor
}
```

Figure 4-7. Pseudo-code of the *predictive lookup version 2* is generally insecure. This is because when attackers estimate range of x, the result will accidentally intersect with both x and x_{alt} if x and x_{alt} are closed, as Figure 4-8 shows.

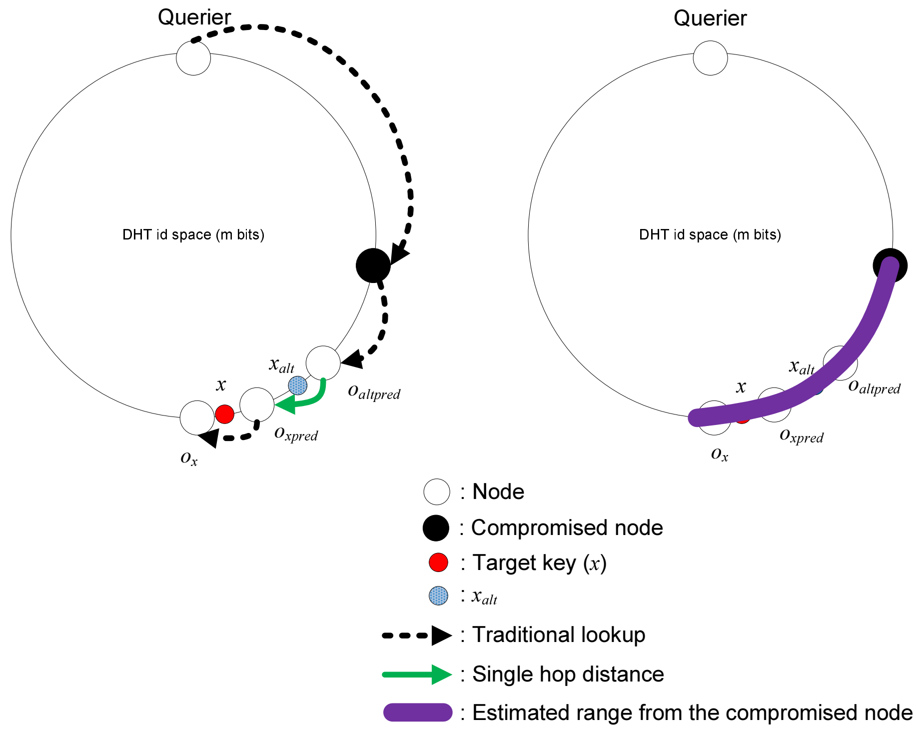


Figure 4-8. Successful range estimation when x_{alt} is nearby x

When nodes are near-uniformly distributed, the probability $Pr_{neighbor}$ that a client selects the target's neighbor is as follows:

$$\delta_{distance} = \frac{2^m}{N}$$

$$\delta_{x_{alt},x} = 2^i$$

$$Pr_{neighbor} = Pr(\delta_{distance} > \delta_{x_{alt},x})$$

$$= Pr(\frac{2^m}{N} > 2^i) = Pr(lg(\frac{2^m}{N}) > i)$$

$$= Pr(m - lg(N) > i), (0 \leq i < m)$$

where i is the selected index on the prediction table, N is the number of nodes, m is the number of bits in a given id space, $\delta_{distance}$ is the average distance between two neighbor nodes, and $\delta_{x_{alt},x}$ is the distance from x_{alt} to x. Since m is fixed, and $log(N)$ does not vary dramatically, $Pr_{neighbor}$ depends on i. When i is too small, *predictive lookup* is no longer beneficial.

With regard to this issue, we propose *prediction table cutoff* technique, which restricts clients from selecting small i. A basic prediction table has m element, but clients cut the front part (the first $m \times f_{cutoff}$ entries) of the table optionally so that they do not select small i. We will explore the impact of cutoff ratio f_{cutoff} more by observing the simulation results in Chapter 5.

```
find_owner_by_predictive_lookup_v3 (n, x)
{
        i = random([⌊m * f_cutoff⌋, m))
        n' = find_predecessor(n, (x-2^i) mod 2^m)
        n' = find_predecessor(n', x)
        return n'.successor
}
```

Figure 4-9. Pseudo-code with *predictive table cutoff*

4.4 Building a Virtual Circuit

We have discussed how to locate a single relay with *predictive lookup* mechanism, but in order to build a virtual circuit, we have to select multiple relays. Tor typically requires three relays, but more relays are recommended in a P2P based environment because of security and scalability issues.

First, any P2P anonymous communication is inevitably weaker than Tor in terms of anonymity because the lookup process relies on queries to other nodes. Whenever a querier sends a request message using a DHT protocol, the receiver at least recognizes that the querier is attempting to initiate anonymous communication. Thus using the same constant number of relays that Tor uses is not a good idea.

Moreover, P2P architecture is more scalable; the system can support more volunteer nodes, and this provides more options to utilize more relays. Although adding a relay increases delay during circuit initialization and communication, it more importantly guarantees improved anonymity.

A naïve approach to locate r relays is running r independent *predictive lookups* sequentially. A client initially locates the first relay, then second, and subsequent relays. However, a large set of passive attackers belonging to a single adversary can analyze querying time to discover the order of queries.

In order to prevent timing attacks during r relays selection, we adapt two strategies: shuffling and concurrent querying. Shuffling randomizes order of relays. Each *predictive lookup* requires generating a random value, so we need r random values, named x_i, where $0 \leq i < r, i \in \mathbb{Z}$. Instead of searching from x_0 to x_{r-1}, we shuffle the set of x_i so that an adversary cannot guess the order of relays.

In addition to shuffling, a client can search each x_i concurrently. A large set of malicious nodes in a single adversary will simultaneously listen to a set of independent queries heading to different x_i. These concurrent queries mislead the adversary into a wrong relay if the adversary uses *range estimation*.

CHAPTER 5
SIMULATION RESULTS

We write a Java application to simulate a DHT based anonymous communication system which has 32-bit id space. ($m = 32$) Then we setup 500,000 nodes, ($N = 500,000$) and assign a different identifier for each node randomly. Once all nodes join into the DHT, we select 10,000 random sources src_i and corresponding random target keys x_i. ($0 \leq i < 10000, i \in \mathbb{Z}$)

5.1 Simulation for Predictive Lookup

We simulate a traditional lookup method and our new *predictive lookup* method to see how much our solution improves anonymity. For both methods, we consider the following criteria.

1. The average number of queries (hops) that source nodes send. (= hop count)
2. Range Estimation Success Ratio f_{RES}

$$f_{RES} = \begin{cases} 0, & \text{if } N_{range} = 0 \text{ or } x \notin [range] \\ \frac{1}{N_{range}}, & \text{if } x \in [range] \end{cases}$$

where $range$ is the estimated range by the attackers and N_{range} is the number of nodes in the range. $f_{RES} = 1$ means that the attackers exactly find the relay. f_{RES} is zero when they fail to estimate range. $f_{RES} = 0.5$ means that they find two nodes, and one of them shall be the relay. Likewise, $f_{RES} = \frac{1}{3}$ means that they find three nodes, and one of them is certainly the relay. The lower f_{RES} is, the more anonymous it is.

We fix $f_{cutoff} = 0$, and set the ratio of compromised nodes f as a variable. We then simulate traditional lookup and *predictive lookup* 10,000 times for different f. We first measure how many hops are additionally required for *predictive lookup*. Figure 5-1 shows that the average hop count of *predictive lookup* is approximately 25% larger than a traditional lookup. Furthermore, we simulate 1 million nodes which greatly exceed

Table 5-1. Average hop count of traditional and *predictive lookup*

N	Average hop count of traditional lookup	Average hop count of *predictive lookup*
10000	6.464	7.996
20,000	7.048	8.769
30,000	7.366	8.952
40,000	7.539	9.422
50,000	7.734	9.501
60,000	7.742	9.500
70,000	7.948	9.883
80,000	7.992	10.045
90,000	8.112	10.278
100,000	8.130	10.291
...	...	...
500,000	9.244	11.960
...	...	...
1,000,000	9.908	12.952

current Tor users, and we find that *predictive lookup* requires only 3 more hops which is near-constant.

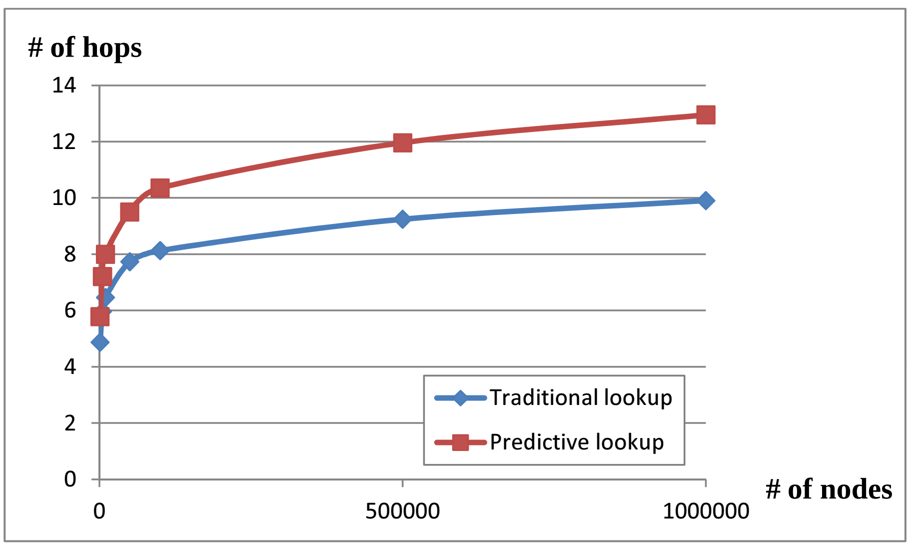


Figure 5-1. Average hop count of traditional and *predictive lookup*

When it comes to f_{RES}, Table 5-2 and Figure 5-2 show that *predictive lookup* is approximately 4 times more secure than the original DHT lookup when 20% of nodes are compromised. When it comes to large f, the difference becomes more

Table 5-2. Comparison between traditional and *predictive lookup*

f	f_{RES} of traditional lookup	f_{RES} of *predictive lookup*
0	0	0
0.05	0.0029	0.0007
0.10	0.0112	0.0033
0.15	0.0278	0.0078
0.20	0.0326	0.0082
0.25	0.0661	0.0136
0.30	0.0985	0.0200
0.35	0.1277	0.0232
0.40	0.1581	0.0267
0.45	0.1733	0.0254
0.50	0.2315	0.0326
0.55	0.2088	0.0328
0.60	0.2458	0.0358
0.65	0.2817	0.0386
0.70	0.3092	0.0405
0.75	0.3509	0.0419
0.80	0.3915	0.0468
0.85	0.4364	0.0564
0.90	0.4626	0.0535
0.95	0.5043	0.0548

significant–up to 10 times. However, using any P2P based anonymous communication solution is not recommended when the portion of compromised nodes is too high.

5.2 Simulation with f_{cutoff}

While we analyze the previous simulation results, we note that many queriers attempt to select x_{alt} which is very closed to x. We set fixed $f = 0.3$ and variable f_{cutoff} ratio at this time. We measure *range estimation success ratio* f_{RES} for the different f_{cutoff} values. Figure 5-3 shows that range estimation is more likely to fail when f_{cutoff} is large.

However, the attackers may find the f_{cutoff} value if they reverse-engineer onion routing applications. When attackers know the f_{cutoff} value, they can design a new range estimation algorithm to exclude the cutoffed range. Figure 5-4 shows that f_{cutoff} should not exceed 0.8 when $f = 0.3$ and an adversary knows f_{cutoff}.

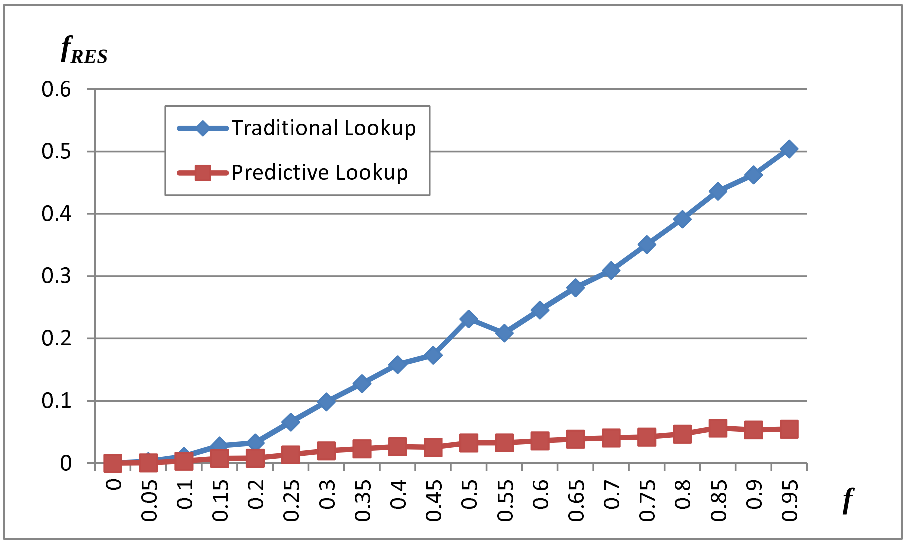


Figure 5-2. Comparison between traditional and *predictive lookup*

Table 5-3. Range estimation success ratio f_{RES} with f_{cutoff}

f_{cutoff}	f_{RES} when f_{cutoff} is secret	f_{RES} when f_{cutoff} is revealed
0	0.0200	0.0200
0.05	0.0140	0.0147
0.10	0.0152	0.0169
0.15	0.0172	0.0202
0.20	0.0152	0.0190
0.25	0.0128	0.0171
0.30	0.0118	0.0169
0.35	0.0148	0.0228
0.40	0.0109	0.0182
0.45	0.0092	0.0167
0.50	0.0089	0.0178
0.55	0.0079	0.0176
0.60	0.0066	0.0165
0.65	0.0073	0.0209
0.70	0.0040	0.0133
0.75	0.0049	0.0196
0.80	0.0040	0.0200
0.85	0.0035	0.0233
0.90	0.0033	0.0330
0.95	0.0017	0.0340

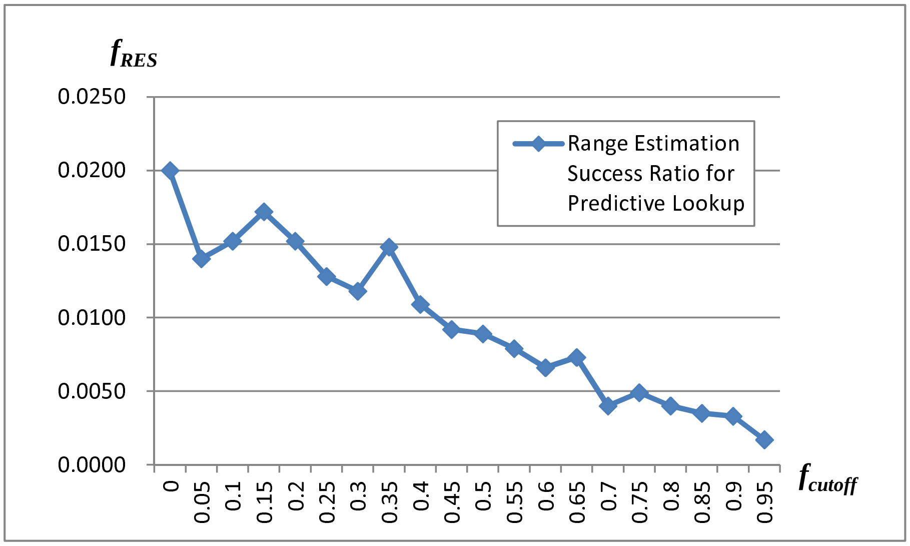


Figure 5-3. Range estimation success ratio f_{RES} when f_{cutoff} is secret

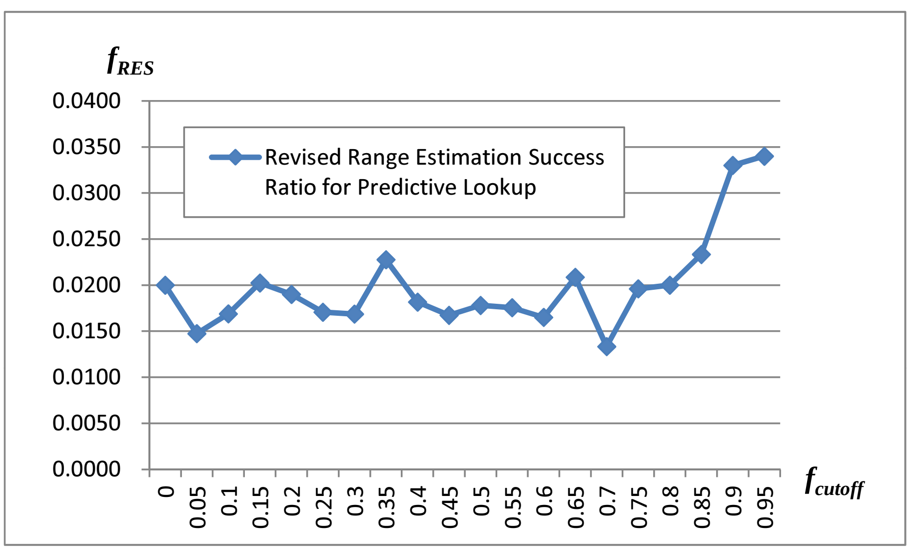


Figure 5-4. Range estimation success ratio f_{RES} when f_{cutoff} is revealed

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

In this paper, we proposed a new anonymous communication system based on P2P architecture. In particular, we focused on Chord-like DHTs, and we presented *predictive lookup*. The new lookup mechanism is designed to protect anonymity without relying on any trusted parties which usually become easy targets of active attacks. Our solution suppresses the possibility of range estimation from passive attackers.

We have also dealt with side effects of our solution, and suggested *prediction table cutoff* to reduce the risk of accidental deanonymization. Moreover, shuffling and concurrent querying obfuscate a large group of timing attackers while locating multiple relays.

We run simulations to assess our solution in a practical environment, and the simulations show that when 30% of nodes are occupied by an adversary, the anonymity increases up to 5 times by sacrificing only tiny additional latency during circuit initialization. This shows that our solution is not only secure, but it is also very practical.

6.2 Future Research

Although this thesis discovers a new anonymous communication solution, more research is still necessary in this field. None of the P2P based solutions guarantees perfect anonymity yet. Interesting research issues will rise when we jointly consider other aspects of networking such as QoS/resource management/distributed computing [28–36], DDoS attacks [37, 38], wireless clients [39], etc. We believe this area is still immature, so enthusiastic researchers may consider jumping into this field.

The compatibility with Tor is another important issue. Since Tor is a dominant anonymous communication application, many people are considering using Tor only. Without absorbing the Tor users, any other solution may not replace Tor even if it provides better anonymity or scalability.

We focused on passive attacks in this thesis, but we also need precise analysis against active attacks. As different solutions may introduce different types of attacks, a new vulnerability can be always discovered, and our solution needs to be analyzed and tested more.

Finally, the types of identity leakage shown in Section 4.2 could be more precisely classified. Each class may have hidden sub-classes which may have different vulnerabilities, and this is another interesting issue specified for our solution.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM '04. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21.

[2] S. Hahn and K. Loesing, "Privacy-preserving ways to estimate the number of tor users," Tor Project, Tech. Rep., 2010, tech. rep., Tor Project, https://metrics.torproject.org/papers/countingusers-2010-11-30.pdf.

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

[4] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable onion routing with torsk," in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 590–599.

[5] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 193–206.

[6] A. Nambiar and M. Wright, "Salsa: a structured approach to large-scale anonymity," in *Proceedings of the 13th ACM conference on Computer and communications security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 17–26.

[7] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "Ap3: cooperative, decentralized anonymous communication," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, ser. EW 11. New York, NY, USA: ACM, 2004.

[8] A. Panchenko, S. Richter, and A. Rache, "Nisan: network information service for anonymization networks," in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 141–150.

[9] P. Mittal and N. Borisov, "Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies," in *Proceedings of the 16th ACM conference on Computer and communications security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 161–172.

[10] ——, "Information leaks in structured peer-to-peer anonymous communication systems," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 267–278.

[11] Q. Wang, P. Mittal, and N. Borisov, "In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems,"

in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 308–318.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 149–160, August 2001.

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

[14] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: attacks and defenses," in *Proceedings OF ESORICS*, 2006, pp. 18–33.

[15] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *International Workshop On Designing Privacy Enhancing Technologies: Design Issues In Anonymity and Unobservability*. Springer-Verlag New York, Inc., 2001, pp. 96–114.

[16] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Privacy Enhancing Technologies workshop*, 2004, pp. 26–28.

[17] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: design of a type iii anonymous remailer protocol," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, may 2003, pp. 2 – 15.

[18] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster protocol version 2," *IETF Internet Draft*, 2005.

[19] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 4, pp. 482 –494, may 1998.

[20] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect," in *Proceedings of the Fifth Workshop on the Economics of Information Security*, ser. WEIS '06, 2006.

[21] Z. Zhang, S. Chen, and M. Yoon, "MARCH: A Distributed Incentive Scheme for Peer-to-peer Networks," in *Proc. of IEEE INFOCOM*. IEEE, 2007, pp. 1091–1099.

[22] Z. Zhang, S. Chen, Y. Ling, and R. Chow, "Capacity-aware Multicast Algorithms on Heterogeneous Overlay Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 2, pp. 135–147, 2006.

[23] S. Chen, B. Shi, S. Chen, and Y. Xia, "Acom: Any-source Capacity-constrained Overlay Multicast in non-DHT P2P Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 9, pp. 1188–1201, 2007.

[24] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware ’01. London, UK: Springer-Verlag, 2001, pp. 329–350.

[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 161–172, August 2001.

[26] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, “Tapestry: A resilient global-scale overlay for service deployment,” *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 41–53, 2004.

[27] P. Maymounkov and D. Mazières, “Kademlia: A peer-to-peer information system based on the xor metric,” in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS ’01. London, UK: Springer-Verlag, 2002, pp. 53–65.

[28] Y. Tang, S. Chen, and Y. Ling, “State Aggregation of Large Network Domains,” *Computer communications*, vol. 30, no. 4, pp. 873–885, 2007.

[29] R. A. Guérin and A. Orda, “QoS routing in networks with inaccurate information: theory and algorithms,” *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 3, pp. 350–364, 1999.

[30] S. Chen and K. Nahrstedt, “Maxmin Fair Routing in Connection-oriented Networks,” *Proc. Euro-Parallel and Distributed Systems Conf*, pp. 163–168, 1998.

[31] K. Lui, K. Nahrstedt, and S. Chen, “Hierarchical QoS Routing in Delay-bandwidth Sensitive Networks,” in *Proc. of 25th Annual IEEE Conference on Local Computer Networks*. IEEE, 2000, pp. 579–588.

[32] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[33] S. Chen, M. Song, and S. Sahni, “Two Techniques for Fast Computation of Constrained Shortest Paths,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 105–115, 2008.

[34] S. Bhatnagar and B. Nath, “Distributed Admission Control to Support Guaranteed Services in Core-stateless Networks,” *Proc. of IEEE INFOCOM*, 2003.

[35] S. Chen and Y. Shavitt, “SoMR: A Scalable Distributed QoS Multicast Routing Protocol,” *Journal of Parallel and Distributed Computing*, vol. 68, no. 2, pp. 137–149, 2008.

[36] S. Chen, Y. Deng, P. Attie, and W. Sun, “Optimal Deadlock Detection in Distributed Systems based on Locally Constructed Wait-for Graphs,” in *Proc. of the 16th*

International Conference on Distributed Computing Systems. IEEE, 1996, pp. 613–619.

[37] K. Park and H. Lee, “On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets,” *Proc. of ACM SIGCOMM'2001*, August 2001.

[38] S. Chen, Y. Tang, and W. Du, “Stateful DDoS Attacks and Targeted Filtering,” *Journal of network and computer applications*, vol. 30, no. 3, pp. 823–840, 2007.

[39] Y. Jian and S. Chen, “Can CSMA/CA Networks Be Made Fair?” in *Proc. of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 235–246.

BIOGRAPHICAL SKETCH

Yangbae Park was born in Busan, South Korea in 1982. He received his Bachelor of Engineering degree in computer engineering at Ajou University in 2004. Upon graduation, he worked at the Third Logistics Support Command in the Republic of Korea Army as a military officer, serving three years. In 2009, he participated in the development of hardware testing algorithm and platform at EAST Laboratory. Since 2010, he has been studying computer engineering at the University of Florida, and he has particularly worked on anonymous communication with his advisor, Dr. Shigang Chen.