

在疊蓋式網路中的匿名且容錯的路由協定

AFATOR: Anonymous and FAult-TOLerant Routing in
overlay networks

研究生：吳佳貞

Student : Chia-Chen Wu

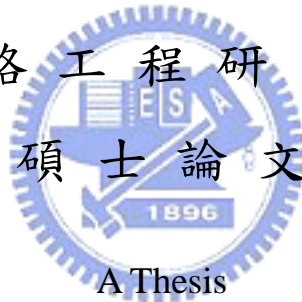
指導教授：謝續平 博士

Advisor : Dr. Shiuhpyng Shieh

國立交通大學

網路工程研究所

碩士論文



Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

在疊蓋式網路中的匿名且容錯的路由協定

研究生：吳佳貞

指導教授：謝續平 博士

國立交通大學

網路工程研究所

摘要

在疊蓋式網路中，匿名對於資料要求者與資料提供者是非常重要的。如何達到匿名通訊最主要取決於路由協定是如何傳遞訊息。在疊蓋式網路中提供匿名路由的挑戰在於找尋資料的時候會洩漏識別碼(identity)。我們提供使用者可抵擋攻擊的匿名通訊並且能容忍網路中節點錯誤。我們隨機選取中繼節點來轉送訊息，並利用層層加密(layered encryption)方式隱藏資料要求者的身分，不被其他中繼節點發現。我們利用Fuzzy Identity-Based Encryption (Fuzzy-IBE)的方法來達到容忍在路由路徑中的節點發生錯誤。利用Fuzzy-IBE，在兩個使用者的識別碼在一定的距離內，使用者可自己的私鑰去解密被另外一個使用者的公開金鑰所加密的密文。因此，當網路中的有一節點發生錯誤或離開，則其識別碼相近的鄰居可幫忙做訊息傳遞。在此篇論文的最後，我們會分析與評估所提之路由協定的匿名性和容錯能力。

AFATOR: Anonymous and fault-tolerant routing in overlay networks

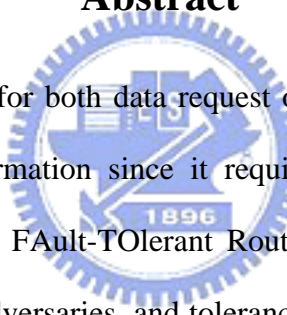
Student: Chia-Chen Wu

Advisor: Dr. Shihpyng Shieh

Department of Network Engineering

National Chiao-Tung University

Abstract

The logo of National Chiao-Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized blue and white design that includes a book and a torch, with the year '1896' at the bottom.

Anonymity is important for both data request or response in overlay networks. Overlay routing reveals information since it requires identity to locate data. We proposed an Anonymous and FAult-Tolerant Routing protocol (AFATOR), which provides anonymity against adversaries, and tolerance of node failures. We randomly select intermediate nodes to forward the messages and use layered encryptions to hide the originator from the intermediate nodes. We also apply Fuzzy Identity-Based Encryption (Fuzzy-IBE) scheme for tolerance of node failures in the routing path. Leveraging Fuzzy-IBE, a user can decrypt a ciphertext encrypted with other's public key if and only if the two users are within a certain distance. Thus, a node can easily take over message forwarding if its neighbor node fails. Analysis of anonymity and failure tolerance of the proposed protocol is also given.

誌 謝

首先感謝指導教授謝續平教授兩年來的諄諄教誨，尤其推薦我去美國加州大學柏克萊分校(University of California, Berkeley)進行為期四個月的資訊安全學術研究，同時接受 Prof. John Kubiawicz 和謝老師的共同指導。感謝老師肯定我的能力並給予難得的機會使我能體驗國外求學的生活，以及學習國外學者做研究的方式。不只是學業方面，對於未來，謝老師也提供許多寶貴的意見和人生經歷，我會記得老師的教誨，做自己有興趣的事情並且享受它，努力、不要輕易放棄。

再來我要感謝黃育綸教授在美國那四個月對我的照顧，教我做研究的方法，在我思考未來時給我引導，在我身體微恙時幫忙我找醫生，在我心情低落、氣餒時幫我想出路。另外，感謝實驗室的碩二同學們(繼偉、鼎鈞、政仲和經偉)，在我研究的過程中給予許多寶貴的意見，在我論文遇到瓶頸的時候，幫我指引方向，讓我能順利完成；也很感謝學長們分享許多經驗、給我信心；感謝碩一的學弟妹們(雨芊、家維、家銘、秉翰和蘇偉)和可愛的助理們(明華和順瑩)，多虧了你們撐住實驗室，讓我們能專心準備論文口試。

最後要感謝我的家人，在這段時間照顧我、包容我，讓我無後顧之憂的專注於研究。我也要感謝明暉同學，是我心靈的支柱，沒有你在背後默默的支持，我將無法順利完成論文，由衷地謝謝你陪伴我度過所有歡樂與痛苦。祝福所有人，事事順心如意！

Table of Content

摘要	I
Abstract	II
誌 謝	III
Table of Content	IV
List of Figures	VI
List of Tables.....	VII
Chapter 1 Introduction	1
1.1 Motivation.....	1
1.2 Contribution	3
1.3 Synopsis	3
Chapter 2 Background.....	4
2.1 Overlay Networks	4
2.2 Identity-Based Cryptosystem.....	5
2.3 Summary	8
Chapter 3 Related Work.....	9
3.1 Corwds	9
3.2 Onion/ Tor	10
3.3 Tarzan.....	11
3.4 Agyaat	12
3.5 Cashmere.....	13
3.6 SurePath	15

3.7 Summary	17
Chapter 4 Proposed Scheme (AFATOR).....	18
4.1 Notation.....	19
4.2 Primitives	20
4.3 Node Registration	21
4.4 Topology Formation.....	23
4.5 Content Request	26
4.6 Example	30
4.7 Summary	35
Chapter 5 Comparison.....	36
5.1 Performance	36
5.2 Security	40
5.3 Summary	42
Chapter 6 Analysis.....	43
6.1 Threat Model.....	43
6.2 Anonymity Analysis.....	44
6.3 Resilient to Node Failure	45
6.4 Against traffic Analysis.....	47
6.5 Summery	48
Chapter 7 Conclusion	49
Reference	50



List of Figures

Figure 1.1 Communications through overlays.....	1
Figure 1.2 Node failures in the routing path.....	2
Figure 2.1 Overlay networks.....	4
Figure 3.1 Crowds network	9
Figure 3.2 Onion network.....	11
Figure 3.3 Tarzan network	12
Figure 3.4 Agyaat network.....	13
Figure 3.5 Cashmere network.....	15
Figure 3.6 SurePath network.....	16
Figure 4.1 Procedure of key extraction.....	21
Figure 4.2 Private key components.....	22
Figure 4.3 Network topology	24
Figure 4.4 Failure detection in Chord-like protocol	25
Figure 4.5 Return path formation.....	27
Figure 4.6 Forward path formation.....	27
Figure 4.7 Routing path stripping process	28
Figure 4.8 Transitions of request and response.....	32
Figure 5.1 Number of the public and private key pairs' comparison.....	40
Figure 5.2 Key storage comparison	40
Figure 6.1 Probability to be Initiator.....	45
Figure 6.2 Forward probability with various error tolerant factors	47

List of Tables

Table 4.1 Notation.....	19
Table 4.2 Messages flow.....	35
Table 5.1 Performance comparison	36
Table 5.2 Security comparison.....	40



Chapter 1

Introduction

1.1 Motivation

Overlay networks [1] have become tremendously popular over the last few years. Users distribute lots of information which can be easily observed by others through overlay networks. They may want to request or provide some secret file, such as gossip, movie, or music, without revealing identities or eavesdropping by other adversaries. It is important to develop a secure protocol for anonymity protection.

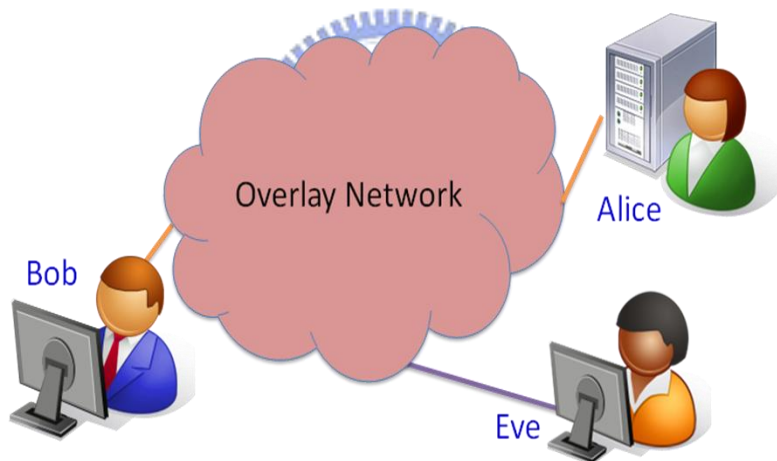


Figure 1.1 Communications through overlays

There are several kinds of anonymity [2] including initiator anonymity, responder anonymity, and relationship anonymity (i.e. unlinkability). Initiator anonymity means the identity of the initiator is hidden to all other peers during communications. Responder anonymity hides the responder from all other peers including the initiator. But it is against the basic DHT lookup scheme: identifying a node mapping to an object with the closest identifier. Mutual anonymity provides both initiator anonymity and responder anonymity. Relationship anonymity, also called initiator-responder

unlinkability, means that even if the initiator and the responder are identified as participating in some communication, they cannot be identified as communicating with each other.

Most anonymity related systems, such as Onion routing [3], make all the anonymous connections go through a fixed set of trusted nodes, like onion routers, which are not preferred in decentralized overlays. By monitoring the traffic of either a colluding entry or exit, adversaries may easily identify initiator or responder. Failure of any onion router in the routing path fails to deliver the message. That results data loss and jitter before the forwarding path is recovered or a new one is constructed.

Now many systems have been designed to exploit peer-to-peer overlays in anonymous communication, including Agyaat [12], Tarzan [13], EDR [16], and NEBLO [18] [39] [40]. They use a sequence of random nodes as anonymous paths or tunnels. If any node along the route fails, or misbehaves, then the message is never delivered to the destination (See Figure 1.2). It is difficult for the initiator to figure out which node has failed.

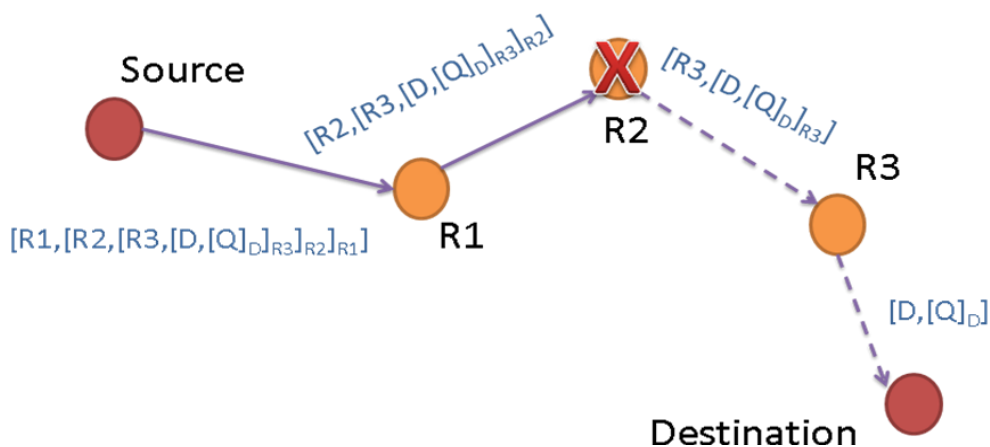


Figure 1.2 Node failures in the routing path

Therefore, we design a protocol to provide an anonymous and fault-tolerant routing protocol named AFATOR.

1.2 Contribution

At first, we provide anonymous routing without proxies. We use layered encryption and random intermediaries to achieve anonymity. Secondly, we achieve unlinkability between initiator and responder without being identified from adversaries. Every node in the path knows only the previous hop and the next hop. An intermediate node cannot tell the initiator or responder. Thirdly, it is easy to recover the routing path without request re-transmission. By using Fuzzy Identity-Based Encryption (Fuzzy IBE) [4], a user can decrypt a cipher-text encrypted with other's public key if and only if the two users are within a certain distance. Thus, any node can easily take over message forwarding if its neighbor node fails. At last, AFATOR uses smallest key storage and leaks less information about the responder.

1.3 Synopsis

The rest of this thesis is organized as follows. In Chapter 2, background is surveyed. Chapter 3 reviews other relevant research in anonymous systems. In Chapter 4, we proposed a scheme to provide anonymity against adversaries and tolerance of node failures. In Chapter 5, we compare AFATOR with other existing researches in performance level and security level. We give a detailed analysis of the security of AFATOR in Chapter 6. Finally, the conclusions and further work are given in Chapter 7 and Chapter 8.

Chapter 2

Background

We give a brief overview of overlay networks and ID-based cryptosystems.

2.1 Overlay Networks

Overlay networks [1] are on top of other networks (See Figure 2.1). The important feature for privacy and mobility is that they decouple network address from physical placements of peers. There are two kinds of existing overlay systems: unstructured overlays and structured overlays.

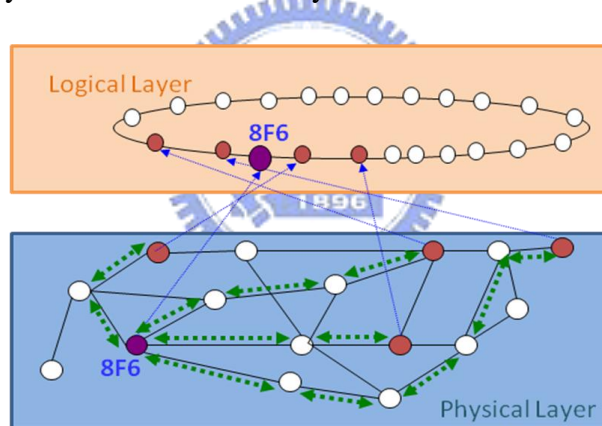


Figure 2.1 Overlay networks

An unstructured overlay network system [30] [34] [36] [37] [38] is composed of peers joining the network without any prior knowledge of the topology. The peers send queries by flooding across the overlay within a limited scope. When a peer receives the query, it replies a list of content matching the query to the originating peer. It is effective to locate the highly replicated data with flooding, and resilient to join or leave the network. But they are unsuited to locating rare data in the large size network.

The disadvantages of unstructured overlay system are there's no guaranteed lookup of a data and a large resource is needed due to the broadcast of queries. The advantage is easy to achieve mutual anonymity.

Structured overlay networks like Chord [5], Tapestry [6], Kademia [35], and Pastry [7] provide scalable and guaranteed lookup of data, a feature especially missing in unstructured overlay networks. They are receiving more attention due to the performance of routing.

Structured overlay is composed of a set of nodes, where nodes are assigned identifiers uniformly at random from a large identifier space. Keys are assigned to data items and nodes are organized into a graph that maps each key to a node.

Most structured overlays support Key-Based Routing (KBR) [8], enabling applications to route a message to any specified key selected from the identifier space. Applications are allowed to locate data in a probabilistically bounded and small number of hop counts while every node contains only a small number of information in its routing table.

The advantage is more efficient than unstructured overlays. But the basic DHT lookup scheme (i.e. to identify a node mapping to an object with closest identifier) against the responder anonymity.

2.2 Identity-Based Cryptosystem

ID-Based Cryptosystem [9] was introduced by Shamir in 1984. The main idea is that the public key of a user can be derived from public information that uniquely identifies the user, such as an email address or IP address. The traditional approach is to use the public key infrastructures, in which a certification authority (CA) issues a certificate which binds a user's identity with his/her public key. The need to make

available authentic copies of entities' public keys is a major drawback to the use of public-key cryptography. By using ID-Based Cryptosystem, all the participants need not to access public key directory. The major advantage is that it simplifies the key management process which is a heavy burden in the traditional certificate based cryptosystems.

Identity-Based Encryption

In 2001, Boneh and Franklin presented an efficient Identity-Based Encryption (IBE) scheme [10]. They performed encryption and decryption operation by using a bilinear map (the Weil pairing) over elliptic curves.

The bilinear map transforms a pair of elements in group G_1 and sends it to an element in group G_2 in a way that satisfies some properties. The most important property is the bi-linearity that it should be linear in each entry of the pair. Weil pairing on elliptic curves is selected as the bilinear map. That is, they use the elliptic curve group as G_1 and the multiplicative group of a finite field as G_2 .

Their ID-based encryption scheme works as follows. A trusted third party called the private key generator (PKG) initially chooses a secretive master key s and announces the public information including elliptic curve equation, the base point P , the public key sP of the system, and other needed hash functions.

Each user has the public key $KU = QID$ that is a point on elliptic curve corresponding to his ID and is known to all other users. The private key is generated by $KR = sQID$, which is obtained from the PKG.

To encrypt a message M , the sender randomly chooses an integer r and sends $(U, V) = (rP, M \oplus h(e(QID, sP)r))$ to the receiver, where h is a hash function announced by PKG in the public information and e is the Weil pairing function to be elaborated in Section II-D. To decrypt the received cipher text (U, V) , the receiver uses the private

key $sQID$ to compute $M = V \oplus h(e(sQID, U))$. This decryption procedure yields the correct message due to the bilinearity of the Weil pairing (i.e., $e(sQID, U) = e(sQID, rP) = e(QID, sP)r$).

Fuzzy Identity-Based Encryption

In 2005, Sahai and Waters proposed Fuzzy Identity-Based Encryption (Fuzzy IBE) [4], where a user can decrypt a cipher-text encrypted with other's public key if and only if the two users are within a certain distance judged by some metric.

Fuzzy-IBE gives rise to two interesting new applications. The first is an Identity-Based Encryption system that uses biometric identities. That is we can view a user's biometric, for example an iris scan, as that user's identity described by several attributes and then encrypt to the user using their biometric identity. Since biometric measurements are noisy, we cannot use existing IBE systems. However, the error-tolerance property of Fuzzy-IBE allows for a private key (derived from a measurement of a biometric) to decrypt a cipher-text encrypted with a slightly different measurement of the same biometric.

Secondly, Fuzzy IBE can be used for an application that we call "attribute-based encryption". In this application a party will wish to encrypt a document to all users that have a certain set of attributes. For example, in a computer science department, the chairperson might want to encrypt a document to all of its systems faculty on a hiring committee. In this case it would encrypt to the identity {"hiring-committee", "faculty", "systems"}. Any user who has an identity that contains all of these attributes could decrypt the document. The advantage to using Fuzzy IBE is that the document can be stored on a simple untrusted storage server instead of relying on trusted server to perform authentication checks before delivering a document.

Setup(d): Providing some security parameter as input, the Private Key Generator

(PKG) runs this algorithm to generate its master key mk and public parameters $params$ which contains an error tolerance parameter d . Note that $params$ is given to all interested parties while mk is kept secret.

$Extract(mk, ID)$: PKG runs this algorithm to generate a private key associated with ID , denoted by D_{ID} by providing the master key mk and an identity ID as input. User's identity, ID , as a set of strings representing a user's attributes.

$Encrypt(M, ID', params)$: Providing the public parameters $params$, an target identity ID' , and a plaintext M as input, a sender runs this algorithm to generate a cipher-text C' .

$Decrypt(C', ID, params)$: Providing the public parameters $params$, a private key D_{ID} associated with the identity ID and a cipher-text C' encrypted with an identity ID' such that $|ID' \cap ID| \geq d$ as input, a receiver runs this algorithm to get a decryption, which is either a plaintext or a "Reject" message. If the set overlap $|ID \cap ID'|$ is greater than or equal to d the algorithm will output the decrypted message M .

When PKG is creating a private key for a user, he will associate a random $d - 1$ degree polynomial, $q(x)$, with each user with the restriction that each polynomial have the same valuation at point 0, that is $q(0) = y$.

If the user is able to "match" at least d components of the cipher-text with their private key components, then they will be able to perform decryption. However, since the private key components are tied to random polynomials, multiple users' are unable to combine them in any way that allows for collusion attacks.

2.3 Summary

In Chapter 2, we give a brief introduction and backgrounds of overlay networks and ID-based cryptosystems.

Chapter 3

Related Work

There are many research on anonymous systems such as Crowds, Onion Routing, TOR, Tarzan, Cashmere, Agyaat, and Surepath. We give a brief introduction about their features, advantages and disadvantages.

3.1 Corwds

Crowds [14] simply uses no public-key encryption, so any node on a circuit can read users' traffic. It provides anonymity by having messages route through anonymous paths involving a randomly chosen sequence of nodes. The initiator sends the message to a randomly-chosen node called "jondo." Upon the message, each jondo randomly decides to either send the message to the responder or to forward it to another jondo. (see Figure 3.1)

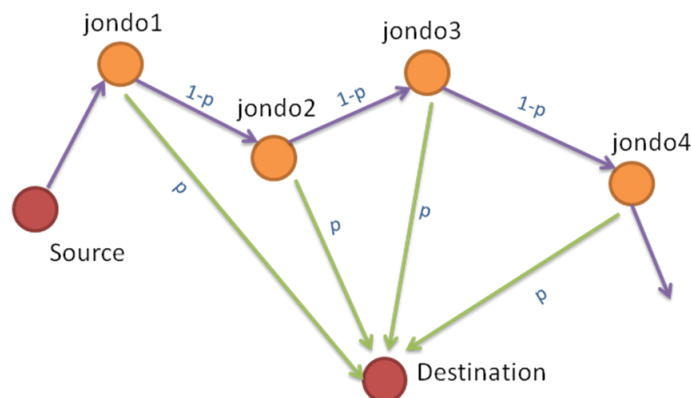


Figure 3.1 Crowds network

In such systems, each node is a mix and an anonymous path can follow any possible path through the system. However, the resulting anonymous paths are vulnerable to node failures: If a node on a tunnel is down, the request/reply message is

not able to route through the tunnel to the destination. Consequently, node failures pose a functionality problem for anonymous paths.

3.2 Onion/ Tor

Onion Routing [3] uses a static set of dedicated onion routers to redirect network traffic. Before sending a message, the sender selects a set of currently active routers to forward through. Session keys are distributed to the chosen routers during the setup phase. The sender creates an onion by encrypting the message with the public key for every router in the routing path.

To transfer a message, each onion router decrypts the outside layer by its private key. After that, it discovers the next hop and forwards the message. Every relay node knows only its previous and next hops.

Node churns, frequent node arrivals, departures, and failures, limit the scalability of Onion Routing.

Tor [17], the second generation of Onion Routing, is one of the most popular privacy enhancing systems. Its goal is to provide initiator anonymity and responder anonymity against non-global adversaries by using rendezvous points.

Tor proposes using a directory server to maintain router information but this approach is also limited in scalability. It has also been shown that if the first or last router is compromised in an Onion Routing network (see Figure 3.2), the source or destination is revealed [31].

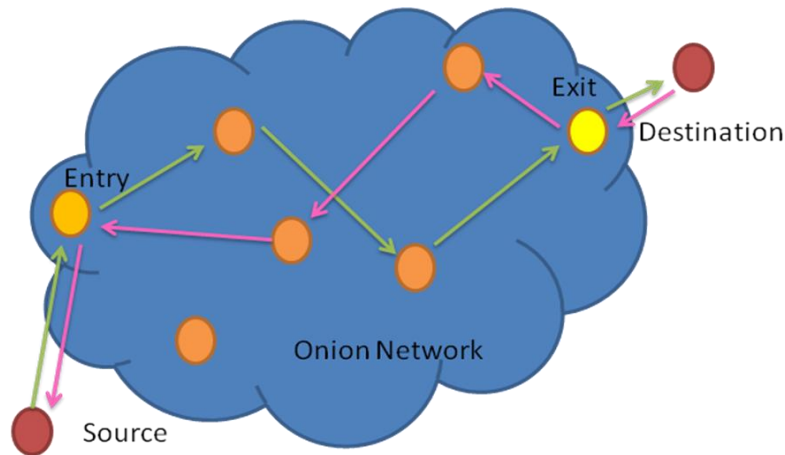
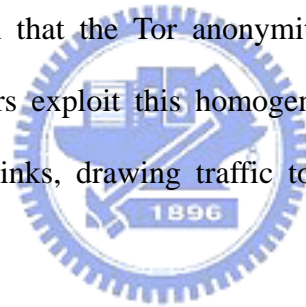


Figure 3.2 Onion network

In their current implementations, all of the approaches do not allow client applications to choose routes that adhere to application-specific criteria. Routes are chosen using pre-defined immutable heuristics.

Recent work has shown that the Tor anonymity network is vulnerable to an attack in which eavesdroppers exploit this homogeneous routing policy by falsely advertising high bandwidth links, drawing traffic towards mixes under its control [32].



3.3 Tarzan

Tarzan [13] provides anonymity with high resistance against traffic analysis by using layered encryption, multi-hop routing, cover traffic and a special mix selection protocol.

The source chooses a set of relays to act as a path and iteratively establishes a tunnel through these relays with symmetric keys between them. The creation of a tunnel incurs both significant computation overhead and delay. The source wraps the packets in several layers of encryption and sends it through relay nodes. The relay node strip off one layer and sends it to next relay node. The exit point of the tunnel,

pseudonymous network address translator (PNAT), decrypts the last layer to extract the original packet, and operates as a network address translator (NAT). After translating the private address to one of PNAT's real addresses, PNAT forwards the message to the Internet (See Figure 3.3). The response repeats the process in reverse.

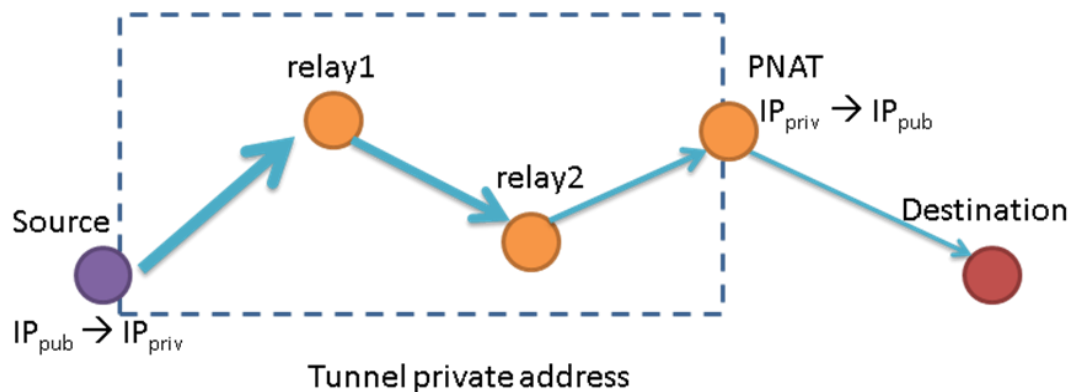


Figure 3.3 Tarzan network

Since none of the peers on a tunnel know the whole path, an adversary cannot figure out communicating peers. But this design is still vulnerable if an adversary can observe traffic throughout the Internet.

Another vulnerability of Tarzan is the resilience of node failures. The message cannot reach the destination if any node on a tunnel fails. Consequently, node failures pose a functionality problem for anonymous paths. The tunnels are static and any relay failure requires formation of a new tunnel.

Although Tarzan provides a high level of sender and recipient anonymity, the sender still has to know the address of the recipient in order to communicate.

3.4 Agyaat

Agyaat [12] provides a compromise between anonymity and efficiency by means of a two-level hybrid organization in which the Chord structured overlay works together with the Gnutella unstructured system, Gnutella-like “clouds” are connected

with one another by means of a Chord ring.

As shown in Figure 3.4, initiator S can flood its request to every peer in its cloud A. One of those peers takes the request out of the cloud, onto the main DHT ring. A normal DHT lookup takes over to locate the cloud to which the responding peer D belongs. At the responder's end, some peers in the cloud of the responder get the request and then broadcast it in its cloud. After receiving the request, the responder peer replies by following a similar path back to the initiator's cloud.

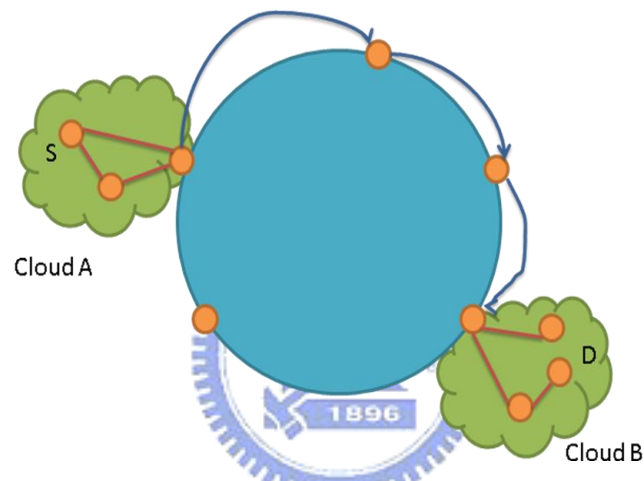


Figure 3.4 Agyaat network

Normal DHT maps a key onto a peer, but Agyaat makes a key mapping onto a cloud which links to the appropriate peer.

3.5 Cashmere

Cashmere [11], a resilient anonymous routing on structured overlay networks, provides both source anonymity and unlinkability of source and destination. Cashmere is designed to use a prefix-routing based on structured overlay network, such as Tapestry and Pastry. The routing path used in Cashmere is a set of distributed

relay groups rather than a single node. There are k unique prefixes and public/private key pairs for each k -bit nodeID. Each relay group has a m -bit GID, where $1 \leq m \leq k$.

Layered encryption is applied on the routing path encryption by the public/private key pair shared with all members of each relay group. Except all the members of the relay group in the routing path fail, the routing path is remained valid. The source node can randomly orders the relay groups to hide the destination relay group containing the destination node. All nodes in a relay group are capable of decrypting a message (only the forwarding path information) which was addressed to that relay group. While a node receives and decrypts the message, it sends the result to the next relay group and broadcasts the result to all the other members in its relay group.

The key benefit of Cashmere over traditional approaches is that it provides an increased resilience to node failures and node churns which generally degrades the performance of traditional anonymous routing protocols based on Chaum-Mixes. Traditionally Chaum-Mixes based routing protocols achieve anonymity by relaying the traffic through a sequence of nodes, such that any two nodes, which are not adjacent to each other along the path, are unable to identify each other. Thus, if the relayed path contains more than two nodes, then there is no way the destination can identify the source. More specifically, no downstream node can identify the upstream nodes.

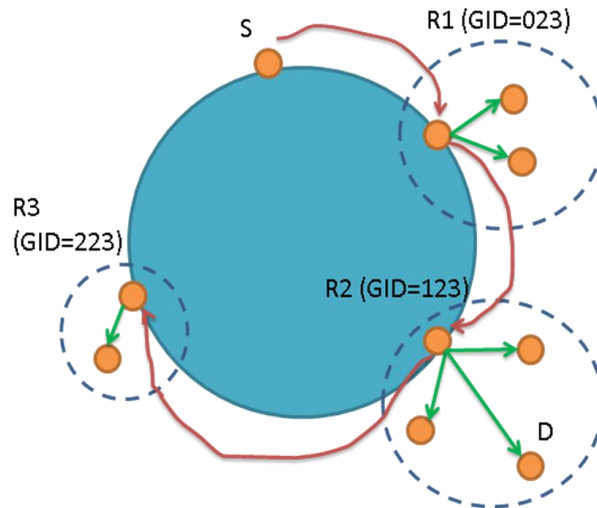


Figure 3.5 Cashmere network

3.6 SurePath

In SurePath [2], a node seeking initiator anonymity generates a small number of RSAs containing session keys, deploys the RSAs into the DHT overlay, forms an anonymous path using a subset of the deployed RSAs, and sends messages through the resulting anonymous path. Like a normal file, a RSA is stored on k nodes whose nodeIds are numerically closest to its associated rsetId. These k nodes are the replica set for the RSA and k is the replication factor. Leveraging the DHT routing infrastructure and data replication mechanism, SurePath is fault-tolerant to node failures. A malicious node can disclose the RSAs stored in its local storage to other colluding nodes such that the malicious nodes can pool their RSAs to break anonymity of other users.

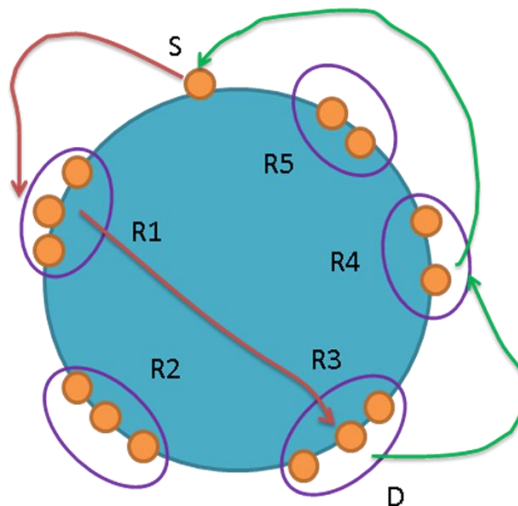


Figure 3.6 SurePath network

The following are the advantages:

- a) To improve resilience of anonymous routing in overlay networks
- b) Leveraging the DHT routing infrastructure and data replication mechanism, SurePath is fault-tolerant to node failures.
- c) By carefully choosing the replication factor and tunnel length, SurePath can strike a balance between functionality and anonymity.
- d) SurePath's performance optimized tunnelling mechanism can greatly improve routing performance.
- e) Users seeking anonymity must reform their tunnels periodically against colluding malicious nodes in dynamic P2P networks to reduce the risk of having their anonymity compromised.

And the disadvantages list as follows.

- a) SurePath lacks the ability to control future hops along a tunnel. It trades this ability for functionality.
- b) The admission control problem in SurePath has not been addressed. In securing routing, the certified nodeIds could control the admission of peers,

and we believe trust management could be used to control the admission and exclude malicious peers from the system. In addition, other incentive mechanisms could possibly be introduced to encourage nodes to protect others' anonymity.

- c) SurePath does not have a mechanism to detect compromised tunnels. It requires users to reform their tunnels periodically against colluding malicious nodes.

3.7 Summary

In Chapter 3, we describe several related works of anonymous network system. It includes Crowds, Onion/TOR, Agyaat, Tarzan, Cashmere and SurePath.



Chapter 4

Proposed Scheme: AFATOR

There are three main phases in AFATOR:

- Phase1: Node Registration

In the first phase, Private Key Generator (PKG) would do the Setup operation. Every node must do registration from PKG to get some system parameters and its private key while joining the network.

- Phase2: Topology Formation

Chord-like protocol is used to setup network topology including route discover and routing table maintenance. It can route bi-direction by using routing table with several predecessors and successors.

- Phase3: Content Request

For content request phase, we use layered encryption and random intermediaries to achieve anonymity. Every node in the path knows only the previous hop and the next hop.

We also apply Fuzzy Identity-Based Encryption (Fuzzy-IBE) scheme for tolerance of node failures in the routing path. By using Fuzzy-IBE, a user can decrypt a cipher-text encrypted with other's public key if and only if the two users are within a certain distance. Thus, any node can easily take over message forwarding if its neighbor node fails.

We have investigated the use of the different constructions [23] [41] for Fuzzy IBE scheme. The new construction we used [23] is more efficient in both extract and encryption operations.

4.1 Notation

At first, we shall introduce the notations used in this protocol. The notations and their interpretations are listed in Table 4.1.

Symbol	Description
S	Initiator
R _i	The intermediaries chosen from the initiator S
R _i '	The real existing node whose id is close to R _i
F	The file which initiator requests
D	The destination which stores the file F
ID _X	Identities of X
SK	Secret key produced by S
PuK _X , PrK _X	Public key and private key for X
T _B	Return path specified by S
Q	Query

Table 4.1 Notation

4.2 Primitives

We give a brief review of admissible bilinear pairing [10]. Let G_1 and G_2 be groups of the same prime order p . An admissible bilinear map, denoted by \hat{e} , has the following properties:

$\hat{e}: G_1 \times G_1 \rightarrow G_2$:

1. Non-degenerate:

g is a generator of $G_1 \Rightarrow \hat{e}(g, g)$ generates G_2

2. Bilinear:

$\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$, $\forall a, b \in \mathbb{Z}, g \in G_1$

3. Efficiently computable:

An efficient algorithm to compute $\hat{e}(g, g)$ exists for any $g \in G_1$

Recall that we apply Fuzzy IBE scheme for allowing a cipher text created by identity w can be decrypted with a private key of identity w' where $|w \cap w'| \geq d$. Let G_1 be a bilinear group of prime order p and make g a generator of G_1 . And let the bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$.

The definition of universe U is $\{0*2^0, 1*2^0, \dots, 0*2^{n-1}, 1*2^{n-1}\}$. For each identity is viewed as a set of attributes or bits, $w = \{w_n, w_{n-1}, \dots, w_2, w_1\}$. The identities of nodes will be element subsets in the universe U . Each element would be associated with a unique integer in U .

The Lagrange coefficient $\Delta_{i,S(x)}$ for $i \in \mathbb{Z}_p$ and a set, S , of elements in \mathbb{Z}_p :

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

Identities will be element subset of some universe. And we will associate each element with a unique integer in \mathbb{Z}_p^* .

4.3 Node Registration

Private Key Generator (PKG) first generates a group G_1 of prime order q , and constructs a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$, where G_2 is a group of the same order q . PKG picks a generator g of the group G_1 .

Second, PKG randomly picks $g_1 \in G_1$, $s \in \mathbb{Z}_q^*$ and compute $g_2 = g^s$. Then, PKG chooses a hash function $H: \mathbb{Z}_q^* \rightarrow G_1$ and selects an error tolerance factor d . After that, PKG generates its master key: $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, s \rangle$ and keeps it secret. PKG also generates system parameters which contain an error tolerant factor d and publishes them to other registered nodes: $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, d \rangle$.

Third, for any node provides its ID: (μ_1, \dots, μ_n) to PKG, PKG picks a random polynomial $p(\cdot)$ of degree $d-1$ over \mathbb{Z}_q such that $p(0)$ is equal to s . Then, PKG computes each private key component D_{μ_i} for $i=1, \dots, n$:

$$D_{\mu_i} = (\gamma_{\mu_i}, \delta_{\mu_i}) = ((g_1 H(\mu_i))^{p(\mu_i)}, g^{p(\mu_i)})$$

As result, the private key of identity ID is composed of n components as follows:

$$\text{PrK}_{\text{ID}} = \langle D_{\mu_1}, \dots, D_{\mu_n} \rangle = \langle ((g_1 H(\mu_1))^{p(\mu_1)}, g^{p(\mu_1)}), \dots, ((g_1 H(\mu_n))^{p(\mu_n)}, g^{p(\mu_n)}) \rangle$$

After finishing computation, PKG returns private key $\text{PrK}_{\text{ID}} = (D_{\mu_1}, \dots, D_{\mu_n})$.

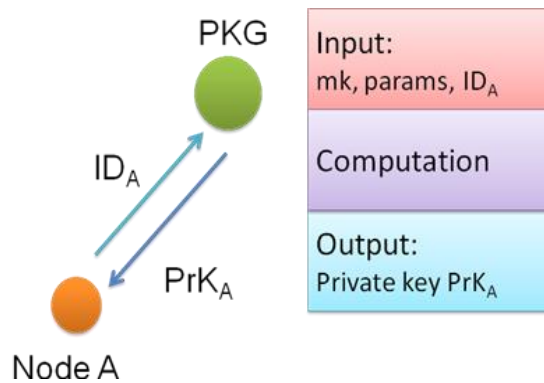


Figure 4.1 Procedure of key extraction

For example, node A can get its private key PrK_A from PKG by providing its ID_A .

The procedure of key extraction is shown in Figure 4.1.

As you can see in Figure 4.2, for each of the attributes or bits associated with a user's identity, PKG will issue a private key component that is tied to the user's random polynomial $p(x)$. Each identity with the same restriction that the value at point 0 for each polynomial are the same, that is $p(0) = s$.

The private keys of different users are generated from different random polynomials. No group of users should be able to combine their keys in such a way that they can decrypt a cipher that none of them could. That is the adversaries cannot combine their keys to form a new one for decrypting some other cipher text.

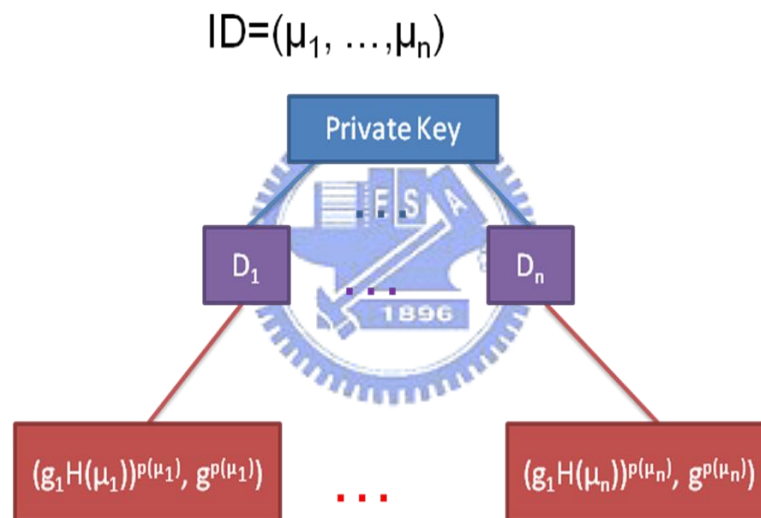


Figure 4.2 Private key components

4.4 Topology Formation

AFATOR can be built over a structured overlay network, which provides a scalable routing substrate for building resilient, large-scale decentralized systems. The routing protocol used in such overlay substrate is similar to Chord [5], where every node is assigned a unique identifier from a large key space, and the routing between any two nodes typically contains $O(\log N)$ hops, where N is the total number of nodes.

Chord-like protocol is used to setup network topology, including route discovery and routing table maintenance.

Each node, acting as a proxy and router, stores information about only a small amount of its neighbors. But the information generally is not enough to determine the node where data located. The information can be used by attackers to compromise the anonymity of storage nodes, i.e. recipient anonymity. The data maps onto a node by using identity. The data is assigned to the first node whose identifier is equal to or follows the identifier of the data. The distance between data and node is within d .

When a node joins the overlay network, it first finds a neighbor node and initializes its routing table. The new joined node exchanges some information with neighbors and updates the routing table. Each node maintains a neighbor set of m nodes ($m/2$ neighbors clockwise and $m/2$ neighbors counter clockwise). Whenever a node wants to lookup data, it can choose clockwise way or counter clockwise way to route the message.

We give an example to show how to route a message through the CHORD ring network. As you can see in Figure 4.3, N_8 wants to route the request to the node which stores data 50 by checking its neighbor list. After finding the closest neighbor to data 50, N_{32} , N_8 routes the request through N_{32} . Upon N_{32} receiving the request,

it checks its neighbor, finds the neighbor that close to data 50, and routes the message to the neighbor.

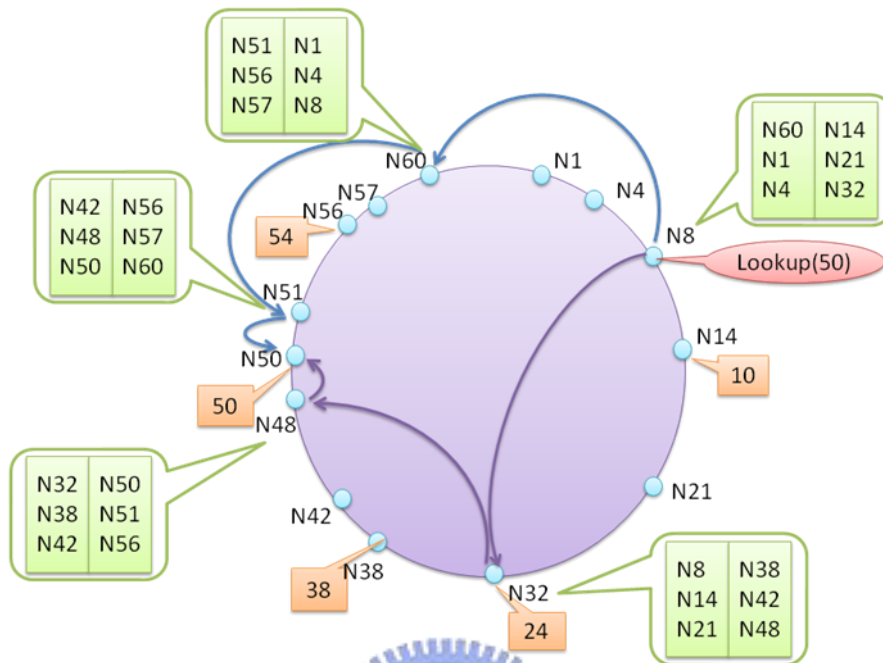


Figure 4.3 Network topology

If a node cannot find an active neighbor which acts as proxy to join the network, it creates a new ring after a timeout. The network can contain multiple rings.

The correctness of Chord-like routing protocol relies on every node knows the previous hop and the next hop. The nodes which is compromised or failed will lead to incorrect lookups. In order to increase the robustness, it is important to pick a suitable m. The problem is dependent with Chord and I'm not going to discuss it in this thesis.

It's easier to detect path failure by using soft state than hard state. Each node periodically broadcast hello message to see if its neighbors are active or not. Without any response from the neighbor node, it can mark the neighbor as failed node in the routing table. The node can remove the failed neighbor node from the routing table after a period without any messages. A hello message also indicates that sender is active.

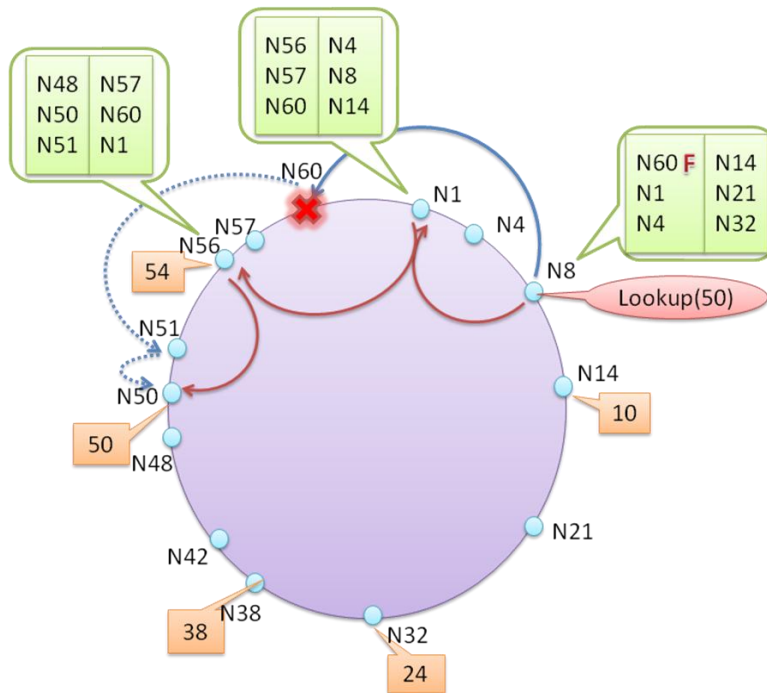


Figure 4.4 Failure detection in Chord-like protocol

As shown in Figure 4.4, N8 originally routes to N50 through N60. While N8 detects that N60 is not active, N8 chooses another neighbor N1 to send lookup message and marks N60 as faulty node in the neighbor set. If N60 is still not active after a period, N8 removes N60 from the neighbor set. Otherwise, N8 clears the faulty mark of N60.

4.5 Content Request

We use layered encryption and random intermediaries to achieve anonymity. Every node in the path knows only the previous hop and the next hop.

We separate it into two parts as the routing path formation and the routing path stripping process. Initiator can randomly choose the intermediate nodes and create a message onion by encrypting with the intermediate nodes' public keys. The paths between initiator and intermediate nodes may pass through some other nodes in network. When the intermediaries get the packets, they strip off the outside layer and then forward the message to the next hop. After receiving the request, Responder would reply it using the return path specified by initiator.

For any node S wants to request file F in the network, it can perform the following procedures, return path formation and return path formation.

Figure 4.5 illustrate the flow diagram of return path formation. Followed the flow diagram, S firstly generates fakeOnion and decides the length of the return path. After generating L random intermediate nodes (R_1, \dots, R_L), S does the encryption with fakeOnion, system parameters, and L random intermediate nodes: R_1, \dots, R_L . Finally, the result is T_B which indicates the return path that the responder can follow.

While S finishes the return path formation, S would use the result T_B , Query Q , and session key SK to do the forward path formation. Since the data is stored on the node which identity is the most closest to the data identity, S use the hash value of the data identity as public key to encrypt the message: $\langle Q, T_B, SK \rangle$. Therefore, the node who stores the data can decrypt the cipher text due to the Fuzzy IBE scheme. After that, S decides the path length, generates node identities and encrypts the routing message (See Figure 4.6).

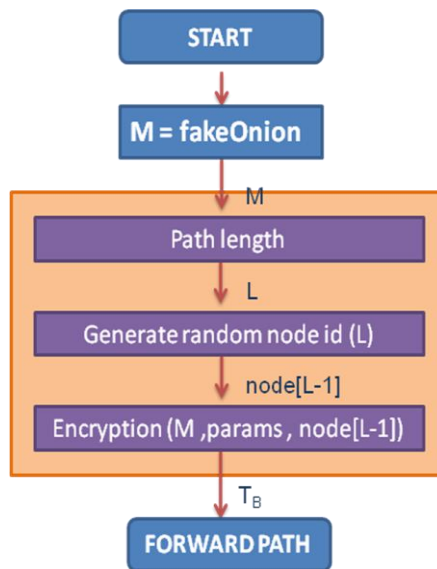


Figure 4.5 Return path formation

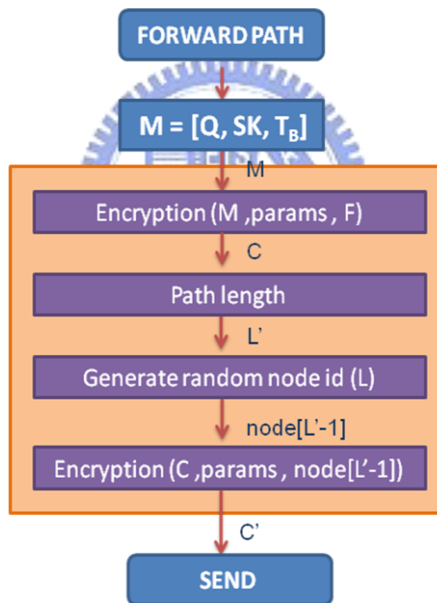


Figure 4.6 Forward path formation

The following equation presents the generation of the intermediate nodes' identifies in a formal way.

$$ID_{Rj} = \text{hash}(t, ID_F, ID_S)$$

A uniform collision-resistant hash function such as SHA-1 can be used. Time t and the identifier of file are added to avoid collision.

S encrypts the message M in a layered manner from the last hop to the first hop in the routing path by their public keys which are the hash value of their identities.

To do encryption operation with ID_{R_i} : (μ_1, \dots, μ_n) , S choose a random value $r \in \mathbb{Z}_p$. Recall that the publish parameters $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, d \rangle$ are given during the node registration. S generates C_{R_i} :

$$\langle ID_{R_i}, U, V_{\mu_1}, \dots, V_{\mu_n}, W \rangle = \langle ID_{R_i}, g^r, H(\mu_1)^r, \dots, H(\mu_n)^r, \hat{e}(g_1, g_2)^r M \rangle.$$

While any intermediate node (R_i') receives the encrypted message, it can perform the routing path stripping process in Figure 4.7.

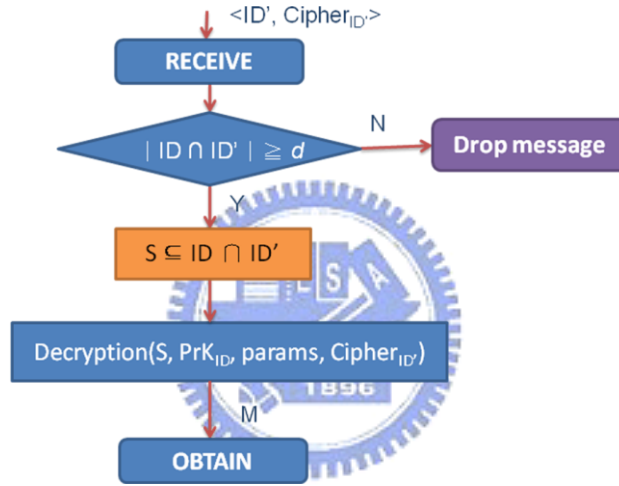


Figure 4.7 Routing path stripping process

Due to $| ID_{R_i} \cap ID_{R_i}' | \geq d$, R_i' does the following steps to decrypt C_{R_i} which is encrypted with ID_{R_i} (Recall that $ID_{R_i} = (\mu_1, \dots, \mu_n)$). R_i' chooses a d -element arbitrary set, S , of $ID_{R_i} \cap ID_{R_i}'$ and runs the decryption algorithm:

$$M = \frac{\prod_{\mu_j' \in S} \hat{e}(v_{\mu_j'}, \delta_{\mu_j'}^{\Delta_{\mu_j', S(0)}})}{\hat{e}\left(\prod_{\mu_j' \in S} \gamma_{\mu_j'}^{\Delta_{\mu_j', S(0)}}, U\right)} \cdot W \quad (\text{Notice that } \mu_j = \mu_j' \text{ if } \mu_j' \in S)$$

The above decryption algorithm is correct as

$$\begin{aligned}
\frac{\prod_{\mu_j \in S} e(V_{\mu_j}, \delta_{\mu_j}^{\Delta_{\mu_j, S(0)}})}{e(\prod_{\mu_j \in S} \gamma_{\mu_j}^{\Delta_{\mu_j, S(0)}}, U)} \cdot W &= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^r, g^{p(\mu_j)\Delta_{\mu_j, S(0)}})}{e(\prod_{\mu_j \in S} (g_1 H(\mu_j))^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot W \\
&= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)}{e(\prod_{\mu_j \in S} (g_1 H(\mu_j))^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot W \\
&= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)}{e(\prod_{\mu_j \in S} g_1^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot \frac{W}{e(\prod_{\mu_j \in S} H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \\
&= \frac{1}{e(\prod_{\mu_j \in S} g_1^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot e(g_1, g_2)^r M \\
&= \frac{1}{e(g_1^s, g^r)} \cdot e(g_1, g_2)^r M = \frac{1}{e(g_1, g_2)^r} \cdot e(g_1, g_2)^r M = M.
\end{aligned}$$

The fakeOnion added in the return path makes S like an intermediate node and it can also confuse the adversaries observing the network traffic. The return path T_B is specified by using the same manner. The routing information from S to R1 can be shown as follows:

$\langle ID_{R1}, [ID_{R2}, \dots, [ID_{Ri}, [ID_F, [Q, SK, T_B]_{PuKF}]_{PuKRi}]_{PuKRi-1} \dots]_{PuKR1} \rangle$, where

$T_B = \langle ID_{Ri+1}, [ID_{Ri+2}, \dots, [ID_{Rm}, [ID_S, fakeOnion]_{PuKRm}] \dots]_{PuKRi+1} \rangle$

Since the identifier of intermediate node ID_{Rj} is randomly generated from S, it probably does not exist in the network. But the routing protocol will routes ID_{Rj} to $ID_{Rj'}$ which is really close to ID_{Rj} . $R_{j'}$ and R_j are in a certain distance d , so that $R_{j'}$ can decrypt the cipher-text encrypted with R_j 's public key PuK_{Rj} .

If the user is able to “match” at least d components of the cipher text with their private key components, then they will be able to perform decryption. However, since the private key components are tied to random polynomials, multiple users' are unable to combine them in any way that allows for collusion attacks.

An intermediate node determines the next hop and forwards the message to it according to the identifier in the header after removing one layer of encryption using its private key. The message onion has been stripped off one layer:

$\langle ID_{R2}, [\dots, [ID_{Ri}, [ID_F, [Q, SK, T_B]_{PuKF}]_{PuKRi}]_{PuKRi-1} \dots]_{PuKR2} \rangle$

This process is continued until the encrypted message arrived at the destination node D, responder, whose identifier is closest to the file ID_F . The responder can use its private key PrK_D to decrypt the encrypted query due to the distance between ID_F and ID_D are larger than d .

Responder D retrieves the file f from its local storage and encrypts it with a symmetric key K extracted from the receiving message and the public key of the next hop. Then Responder sends the reply message to the next hop specified by the return path:

$$\langle ID_{R_{i+1}}, [F]_{SK}, [ID_{R_{i+2}}, \dots [ID_{R_m}, [ID_S, fakeOnion]_{PuKR_m}] \dots]_{PuKR_{i+1}} \rangle$$

The intermediate nodes in the return path would do the same procedure like the nodes in forward path, such as strip off a layer and send it to the next hop.

It is hard for adversaries to correlate a request with a response because the forward path is different from the return path. Messages passed along the anonymous connection appear different to each node, so they cannot be tracked en route and compromised nodes cannot cooperate.

4.6 Example

In this subsection, we would like to demonstrate the way how to apply AFATOR. For example, the network N contains a private key generator (PKG), and many nodes including initiator S , responder D , intermediate nodes R_i , R_i 's neighbor R_i' , and some other nodes where $i = 1, \dots, 4$. (See Figure 4.8)

The definition of universe U is $\{0*2^0, 1*2^0, \dots, 0*2^{n-1}, 1*2^{n-1}\}$. For each identity is viewed as a set of attributes or bits, $w = \{w_n, w_{n-1}, \dots, w_2, w_1\}$. The identities of nodes will be element subsets in the universe U . Each element would be associated with a unique integer in U .

PKG randomly picks $g_1 \in G_1$, $s \in Z_q^*$ and compute $g_2 = g_1^s$. Then, PKG chooses a hash function $H: Z_q^* \rightarrow G_1$ and selects an error tolerance factor d . After that, PKG generates its master key: $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, s \rangle$ and keeps it secret. PKG also generates system parameters which contain an error tolerant factor d and publishes them to other registered nodes: $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, d \rangle$.

In order to generate a private key for a node with identity ID: (μ_1, \dots, μ_n) , PKG needs to randomly choose a $d-1$ degree polynomial p such that $p(0)=s$. Then, PKG computes each private key component D_{μ_i} for $i=1, \dots, n$:

$$D_{\mu_i} = (\gamma_{\mu_i}, \delta_{\mu_i}) = ((g_1 H(\mu_i))^{p(\mu_i)}, g^{p(\mu_i)}).$$

As result, the private key of identity ID is composed of n components as follows:

$$\text{PrK}_{\text{ID}} = \langle D_{\mu_1}, \dots, D_{\mu_n} \rangle = \langle ((g_1 H(\mu_1))^{p(\mu_1)}, g^{p(\mu_1)}), \dots, ((g_1 H(\mu_n))^{p(\mu_n)}, g^{p(\mu_n)}) \rangle$$

To do encryption operation with $\text{ID}_{R_i}: (\mu_1, \dots, \mu_n)$, the initiator chooses a random value $r \in Z_p$. Recall that the publish parameters $\langle G_1, G_2, \hat{e}, g, q, H, g_1, g_2, d \rangle$ are given during the registration. The initiator generates C_{R_i} :

$$C_{R_i} = \langle \text{ID}_{R_i}, U, V_{\mu_1}, \dots, V_{\mu_n}, W \rangle = \langle \text{ID}_{R_i}, g^r, H(\mu_1)^r, \dots, H(\mu_n)^r, \hat{e}(g_1, g_2)^r M \rangle.$$

Due to $|\text{ID}_{R_i} \cap \text{ID}_{R_i'}| \geq d$, R_i' does the following steps to decrypt C_{R_i} which is encrypted with ID_{R_i} (Recall that $\text{ID}_{R_i} = (\mu_1, \dots, \mu_n)$ and $\text{ID}_{R_i'} = (\mu_1', \dots, \mu_n')$). R_i' chooses a d -element arbitrary set, S , of $\text{ID}_{R_i} \cap \text{ID}_{R_i'}$ and runs the decryption algorithm mentioned in previous subsection.

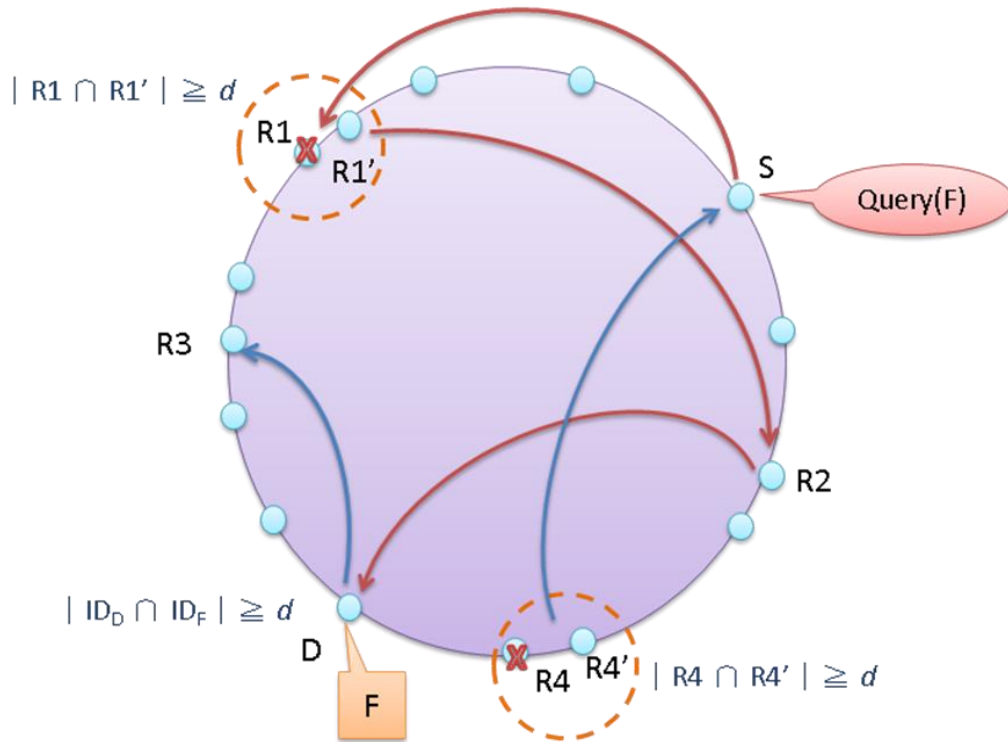


Figure 4.8 Transitions of request and response

Figure 4.8 shows an example of how initiator S looks up and gets back the file F. In order to lookup a file F, initiator S randomly chooses the intermediaries R1, R2, R3, R4 and then forms the forward path and the return path. The forward path consists of R1 and R2. R3 and R4 are in the return path. To confuse other nodes knowing the real destination of the message, a fakeOnion is generated. S uses the public keys of R3 and R4, the hash value of ID_{R3} and ID_{R4} , to encrypt the message M' including the identity of initiator ID_S , a fakeOnion and the return path T_B .

$$M' = \langle ID_S, \text{fakeOnion} \rangle$$

S picks a random r and encrypts the message M with $R4$'s public key PuK_{R4} and then obtains cipher text C_{R4} : $\langle ID_{R4}, U_{R4}, V_{R4\mu 1}, \dots, V_{R4\mu n}, W_{R4} \rangle$ where ID_{R4} consists of n attributes and $W_{R4} = \hat{e}(g_1, g_2)^r M'$.

By the similar way, S also uses $R3$'s public key PuK_{R3} to encrypt the result C_{R4}

and then obtains cipher text C_{R3} : $\langle ID_{R3}, U_{R3}, V_{R3\mu1}, \dots, V_{R3\mu n}, W_{R3} \rangle$, where $W_{R3} = \hat{e}(g_1, g_2)^r C_{R4}$. The result C_{R3} acts as the return path T_B .

The request message M containing the query Q , the return path T_B , and a session key SK is encrypted by the public keys of $R2$ and $R1$. The session key, SK , is used to protect the reply message.

$$M = \langle Q, T_B, SK \rangle$$

S uses F 's public key PuK_F to encrypt the message M and then obtains cipher text C_F : $\langle ID_F, U_F, V_{F\mu1}, \dots, V_{F\mu n}, W_F \rangle$, where $W_F = \hat{e}(g_1, g_2)^r M$.

By the similar way, S uses $R2$'s public key PuK_{R2} and $R1$'s public key PuK_{R1} to encrypt the message E_F . Finally, S obtains the layered encrypted routing message C_{R1} : $\langle ID_{R1}, U_{R1}, V_{R1\mu1}, \dots, V_{R1\mu n}, W_{R1} \rangle$, where $W_{R1} = \hat{e}(g_1, g_2)^r C_{R2}$.

S routes the request message E_{R1} to $R1$. If $R1$ fails or does not exist, then its neighbor node $R1'$ can take over the message forwarding since the overlap between $R1$ and $R1'$ is larger than or equal to error tolerant factor d . Upon receiving message, $R1'$ uses its private key $PrK_{R1'}$ to decrypt the message. At first, $R1'$ chooses an arbitrary d -element subset S of $\{ ID_{Ri} \cap ID_{Ri'} \}$ and then does the decryption operation.

$$U = \{ u_0, u_1, \dots, u_{d-1} \}$$

$$M = \frac{\prod_{\mu_j \in S} e(V_{\mu_j}, \delta_{\mu_j}^{\Delta_{\mu_j, S}^{(0)}})}{e(\prod_{\mu_j \in S} \gamma_{\mu_j}^{\Delta_{\mu_j, S}^{(0)}}, U)} \cdot W$$

(Here, notice that $\mu'_j = \mu_j$ if $\mu_j \in S$). Return M .

$$\begin{aligned}
\frac{\prod_{\mu_j \in S} e(V_{\mu_j}, \delta_{\mu_j}^{\Delta_{\mu_j, S(0)}})}{e(\prod_{\mu_j \in S} \gamma_{\mu_j}^{\Delta_{\mu_j, S(0)}}, U)} \cdot W &= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^r, g^{p(\mu_j)\Delta_{\mu_j, S(0)}})}{e(\prod_{\mu_j \in S} (g_1 H(\mu_j))^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot W \\
&= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)}{e(\prod_{\mu_j \in S} (g_1 H(\mu_j))^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot W \\
&= \frac{\prod_{\mu_j \in S} e(H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)}{e(\prod_{\mu_j \in S} g_1^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot \frac{W}{e(\prod_{\mu_j \in S} H(\mu_j)^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \\
&= \frac{1}{e(\prod_{\mu_j \in S} g_1^{p(\mu_j)\Delta_{\mu_j, S(0)}}, g^r)} \cdot e(g_1, g_2)^r M \\
&= \frac{1}{e(g_1^s, g^r)} \cdot e(g_1, g_2)^r M = \frac{1}{e(g_1, g_2)^r} \cdot e(g_1, g_2)^r M = M.
\end{aligned}$$

After computation, the result C_{R2} : $\langle ID_{R2}, U_{R2}, V_{R2\mu_1}, \dots, V_{R2\mu_n}, W_{R2} \rangle$, can be extracted. At last, $R1'$ discovers the next hop $R2$ and then routes the message to $R2$. If $R2$ exists, $R2$ can strip off one layer of the message and then forwards the result to the destination D which stores the file F .

The file F maps onto the destination D whose identifier is closest to ID_F . Due to the intersection between ID_D and ID_F exceeds the error tolerant value d , D decrypts the request with its private key PrK_D . Upon obtaining the query Q , D retrieves the file F from its storage, uses the session key K to protect the responding file F and replies the message through the return path.

The reply message can be transferred through the return path which is specified by initiator by similar way. We can also see the detailed message flow in Table 4.2

Source → Dest.	Message
$S \rightarrow R1'$:	$\langle ID_{R1}, [ID_{R2}, [ID_F, [Q, SK, T_B]_{PuKF}]_{PuKR2}]_{PuKR1} \rangle$, where $T_B = \langle ID_{R3}, [ID_{R4}, [ID_s, fakeOnion]_{PuKR4}]_{PuKR3} \rangle$
$R1' \rightarrow R2$:	$\langle ID_{R2}, [ID_F, [Q, Key, T_B]_{PuKF}]_{PuKR2} \rangle$
$R2 \rightarrow D$:	$\langle ID_F, [Q, Key, T_B]_{PuKF} \rangle$
$D \rightarrow R3$:	$\langle ID_{R3}, [F]_{SK}, [ID_{R4}, [ID_s, fakeOnion]_{PuKR4}]_{PuKR3} \rangle$
$R3 \rightarrow R4'$:	$\langle ID_{R4}, [F]_{SK}, [ID_s, fakeOnion]_{PuKR4} \rangle$

$R4' \rightarrow S:$	$\langle ID_s, [F]_{SK}, \text{fakeOnion} \rangle$
----------------------	--

Table 4.2 Messages flow

4.7 Summary

In Chapter 4, we describe the whole scheme of AFATOR including three phases. AFATOR protects anonymity for initiator and responder and provides tolerance for node failures by using Fuzzy Identity-Based Encryption (Fuzzy IBE).



Chapter 5

Evaluation

This chapter focuses on evaluation of AFATOR in performance and security level. By comparing with other anonymous systems, we discuss benefits and drawbacks of AFATOR.

5.1 Performance

Table 5.1 presents a performance comparison of AFATOR and other research, SurePath, Agyaat, Tarzan, and Onion/TOR.

	SurePath	Cashmere	Tarzan	Onion/ TOR	AFATOR
Send requests without setting tunnels & sym. Key first	N	Y	N	N	Y
Get public key without CA	N	N	N	N	Y
Tolerate node failures	Y	Y	N	N	Y
Tolerate attackers in routing path	Y	Y	Y	N	Y
Message transmission	Tunnel +relay group	relay group	Tunnel	Onion routers	Ring DHT
Storage costs	1pub&pri key N-1 pairwise key	logN pub&pri key			1pub&pri key sys. params

Table 5.1 Performance comparison

Public key from CA

Tarzan and Onion need to establish tunnels or circuits and distribute session keys before sending message. Cashmere and SurePath use a set of relay groups to instead a single relay node. Every node with the same prefix of its identifier in a Cashmere relay group shares the common public/private key pair. SurePath randomly generates

the relay groups and distributes and the symmetric keys associated with the relay groups. All of them need a third party Certificate Authority (CA) to get public key and private key pairs. But AFATOR need not to do so. With only system parameters and the target node identity, every node can easily generate the public key of the target node. The private keys of all nodes in AFATOR are obtained from Private Key Generator (PKG).

Resilient to node failures

In Tarzan, Onion and Tor, the message may not reach the destination if any node in the routing path fails. Cashmere, SurePath and AFATOR can achieve resiliency for node churns. The routing path of Cashmere is formed with a set of relay groups with the same prefix of identifiers. Each relay group shares the common public/private key pair due to the same prefix. SurePath uses relay sets with k candidates to help message forwarding. We apply Fuzzy IBE scheme to ensure that even if a node fails, its neighbors can take over message forwarding.

Compare AFATOR with SurePath and Cashmere, neither key distribution nor key discovery is need for AFATOR. Lots of churns in AFATOR and Cashmere do not affected the other existing nodes which need not do extra operation because the identities of nodes are used as public keys. But in SurePath, adding or deleting a node results extra key distribution or index information change. The performance of SurePath decrease with increasing churns in overlay networks.

Computation cost

In AFATOR, the initiator must do most of the encryption operations to form the routing message including the forward path and return path. For each intermediate node, the operation is decrypting one layer of the message. The responder does two operations: decrypts the message to get the query and encrypts the reply by symmetric

key from the receiving message.

In SurePath, before sending a query, the initiator needs to setup tunnels, discover the public key of destination and distribute symmetric keys encrypted by the public key of relay nodes. And then the initiator route the messages encrypted by symmetric keys from every relay sets. Every relay node decrypts the message using the pre-share key. After decrypting the message by the private key of the responder, the responder obtains the symmetric key to encrypt the reply message.

The initiator in Cashmere first generates symmetric keys for each relay groups to encrypt the payload. Secondly, the initiator encrypts the routing information: the next forward path, the identity of the next relay group, and the symmetric key by using the public keys of each relay groups from the last relay group to the first one. Not only forward path but also return path will the initiator create. Any node in the relay group receiving the message becomes the root of its group. The root decrypts the path with public key and the message with symmetric key, and then forwards the result to the next relay groups and broadcast the result to all members in the same relay group. The responder decrypts the message with its public key and then encrypts the reply by symmetric key extracted from the receiving message.

An ECC benchmark shows that an instruction set extensions of a typical embedded processor for ECC that can efficiently replace a coprocessor that is typically used for improving performance of ECC [42].

Key storage cost

Nowadays, Elliptic Curve Cryptography (ECC) is becoming more and more used to alternate traditional public key methods. The reason is that ECC can use shorter key, faster computation time and less memory to achieve the same security level with traditional public key methods. For example, RSA encryption with 1024-bit key is at

the same security level as the use of 163-bit key in the case of ECC over GF (2m). As a result, ECC offers higher throughput on the server side and smaller implementations on the client side. [44]

The cost depends on the keys stored at each node. The key size of AFATOR based on Elliptic Curve Cryptography is much smaller than the key size of Cashmere and SurePath based on traditional public key methods.

In AFATOR, every node stores its private key and public parameters. Whoever wants to query the data, it needs to generate a symmetric key attached in routing message so that the responder can encrypt the reply message by the symmetric key without knowing the initiator.

All members having m-bit identifier in Cashmere own m public/private key pairs corresponding with each prefix of identifiers. The initiator would also create each symmetric key for each relay group in order to encrypt the payload. In SurePath, each node stores lots of keys, including its private key, the public keys of other nodes and the corresponding symmetric keys for each relay set. The initiator also needs to generate a symmetric key for the responder to reply the message.

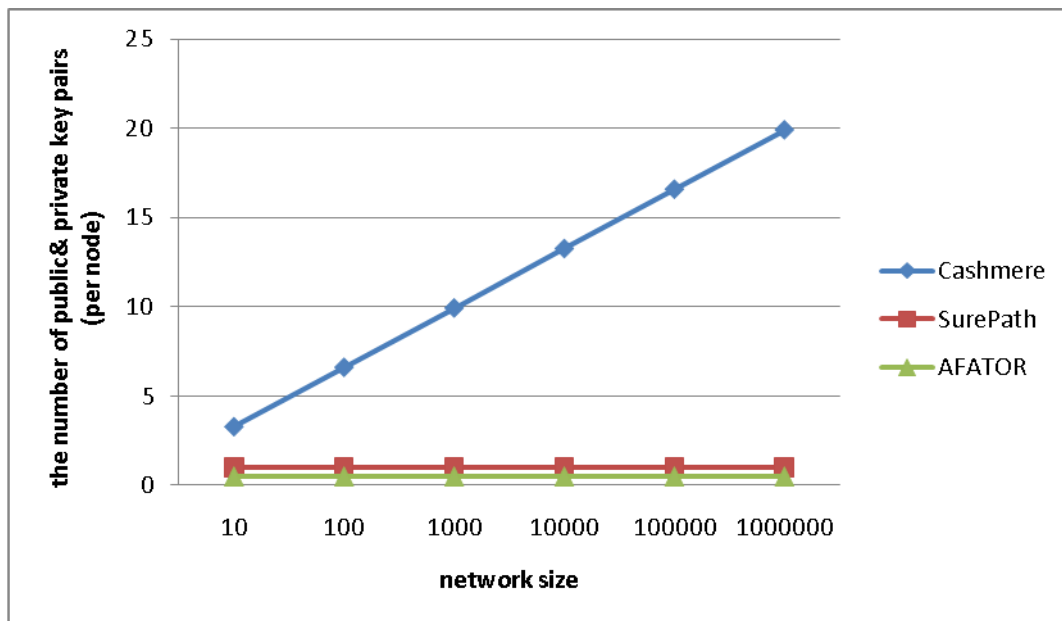


Figure 5.1 Number of the public and private key pairs' comparison

In Figure 5.1, the number of keys in AFATOR is the least than others. And AFATOR only needs the least key storage shown in Figure 5.2.

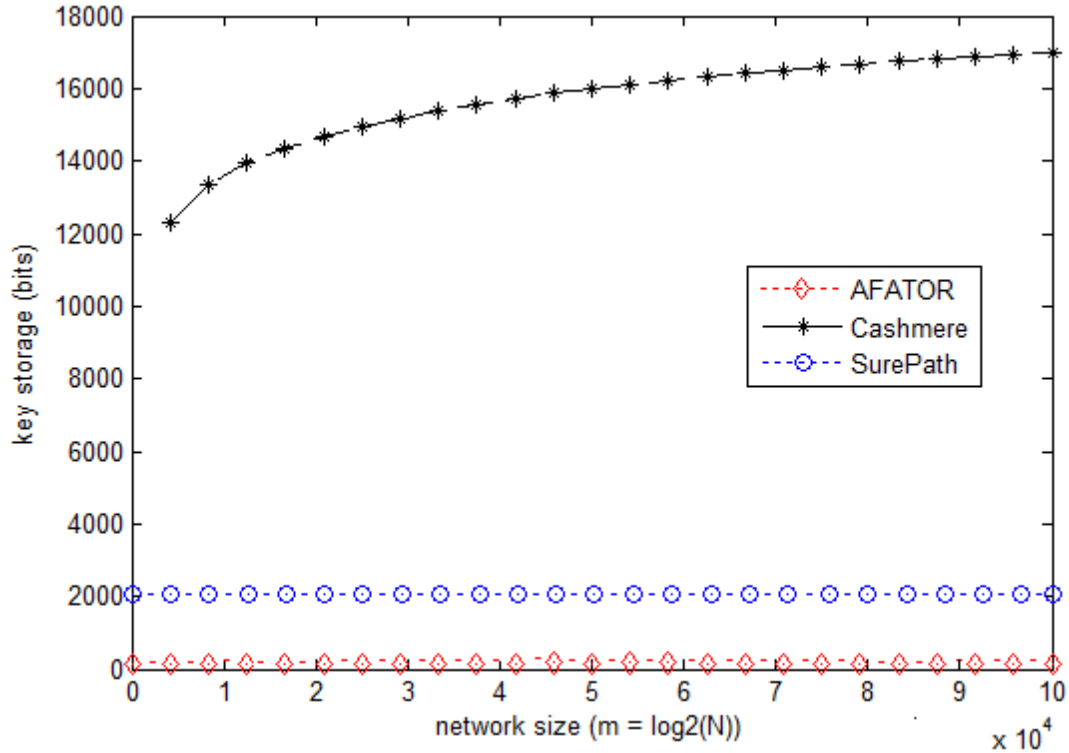


Figure 5.2 Key storage comparison

5.2 Security

The comparison about security issue is shown in Table 5.2.

	SurePath	Cashmere	Tarzan	Onion/TOR	AFATOR
Initiator anonymity	Y	Y	Y	Y	Y
Responder anonymity	Y	N	N	N	Y
Use different paths to do request & response	Y	Y	N	N	Y
Information leaking	Receiver id & public key	Receiver id & public key	Receiver id & public key	Receiver id & public key	Data id & data public key

Table 5.2 Security comparison

Tarzan, Onion, SurePath, Cashmere and AFATOR use layered encryption and multi-hop routing to achieve anonymity. Except AFATOR, all the others use Public

Key Infrastructure (PKI) and symmetric key to protect data secrecy. Moreover, the third party they trusted is Certificate Authority (CA). Onion routing also trusts onion routers to route through the message.

AFATOR use Fuzzy Identity-Based Encryption (Fuzzy IBE) to do public & private key generation and data encryption & decryption. Instead of CA, AFATOR use PKG.

All the systems we mentioned can achieve initiator anonymity by random routing path. Some of them, like SurePath and AFATOR, do the message forwarding and returning by different routing path. The advantage of different routing path is to avoid traffic analysis.

Information leaking

The initiator in AFATOR knows only the desired data name but nothing about where the data located. So the initiator uses the public key, the hash value of data identifier, to encrypt the request. SurePath and Cashmere know not only the data name but also the public key of the node which stores the data. For responder anonymity, the information leaking of AFATOR is less than SurePath and Cashmere.

In Cashmere, if one node with m -bit identifier is compromised, then the attacker would obtain m public and private key pairs associated with each prefix. When the same case occurs in SurePath, the attacker can control $N-1$ share keys where network size is N . However, only one private key is revealed to the attacker during the same situation in AFATOR. Compared AFATOR with other systems, the attacker knows less information about secret keys.

Souvik *et al.* [26] proposed an information theoretic framework for analyzing leak of privacy in DHT. With the same routing complexity, the analytical result shows that ring-based DHT (CHORD) has the minimum information leak than the other

DHTs, such as tree-based, hypercube-based, and hybrid-based DHT.

CHORD-like routing protocol with ring-based DHT is used in AFATOR; however, both Cashmere and SurePath apply Pastry, a routing protocol with hybrid-based DHT. As a result, AFATOR leaks less information than Cashmere and SurePath.

To provide sender and receiver anonymity, these systems like Tarzan [13] and Cashmere [11] require the overlay nodes to have public-private keys obtained through a trusted authority; i.e., they require a public key infrastructure (PKI). A few systems (e.g., Crowds [14]) do not require PKI, but they expose the receiver and message content.

5.3 Summary

It is commonly held that there is a tradeoff between performance and anonymity. The routing protocol that provided best anonymity usually came with associated performance costs.



Chapter 6

Analysis

6.1 Threat Model

The adversaries can do the following things:

- a. compromise the existing node
- b. observe packets destined to itself or its local network
- c. collude and share information with others
- d. follow the protocol and forwards all the messages pass through it

For each compromised node, the attacker will obtain the private key of itself and the error tolerant distance d . Therefore, the attacker may read the cipher text encrypted with its neighbor's public key if they are within a certain distance d . We apply Byzantine failure model to allow compromised node behave arbitrarily.

Eavesdropper

There are two kinds of eavesdropper. The first one is global eavesdropper who can observe all the traffic of the network. Even the message is encrypted, he still can use timing attacks or statistic attacks to break anonymity: identifying the route from the initiator to the responder. But it is not realistic in overlay network with thousands of nodes. It is impossible to know the information of the whole network at any time due to lots of churn in overlays. The other one is local eavesdropper who can only observe packets destined to itself or its local network. He cannot get enough information to identify real destination.

6.2 Anonymity Analysis

We analyze anonymity using three parameters: N (number of nodes in the network), f (fraction of malicious nodes in the network), and L (number of intermediate nodes in the routing path).

The routing path is generated by the initiator, a non-malicious node. When the initiator decides the path, it passes the message to the first intermediate node. Due to layered encryption, every intermediate node knows only the previous hop and the next hop. The attackers in the routing path act as intermediate nodes and try to guess which node is the initiator or the responder. Since the messages are encrypted, the attackers would suspect the previous node which passes the message to itself is the initiator. We distinguish the two cases to analyze the probability that the intermediate previous node is in fact the initiator. Notation f means the probability of choosing an attacker to be an intermediate node. In contrast, the probability of choosing a non-attacker as intermediate node is $1-f$.

Case I: 1st intermediate node is attacker

The attacker can guess its previous node is the initiator with probability of 1. The probability of case I is $p = \frac{1}{L} \sum_{i=1}^L i f^i (1-f)^{L-i}$.

Case II: 1st intermediate node is not attacker

The attacker suspects its previous node with probability of $\frac{1}{N(1-f)}$, where $N(1-f)$ represents the number of non-malicious nodes in the network. The probability of case II is $1-p = 1 - \frac{1}{L} \sum_{i=1}^L i f^i (1-f)^{L-i}$.

Thus, the probability that node x is the initiator shows as follows.

$$P = \frac{1}{L} \sum_{i=1}^L i f^i (1-f)^{L-i} + \frac{1}{N(1-f)} \left(1 - \frac{1}{L} \sum_{i=1}^L i f^i (1-f)^{L-i} \right)$$

Figure 6.1 displays the results for the probability of guessing a node to be the initiator from the attacker with different path length and fraction of attackers. As f (fraction of attackers) increases, the probability a node to be the initiator increases. The length of routing path L influences the variation of the initiator probability of a node. The more intermediate nodes pass by, the less probability to guess right.

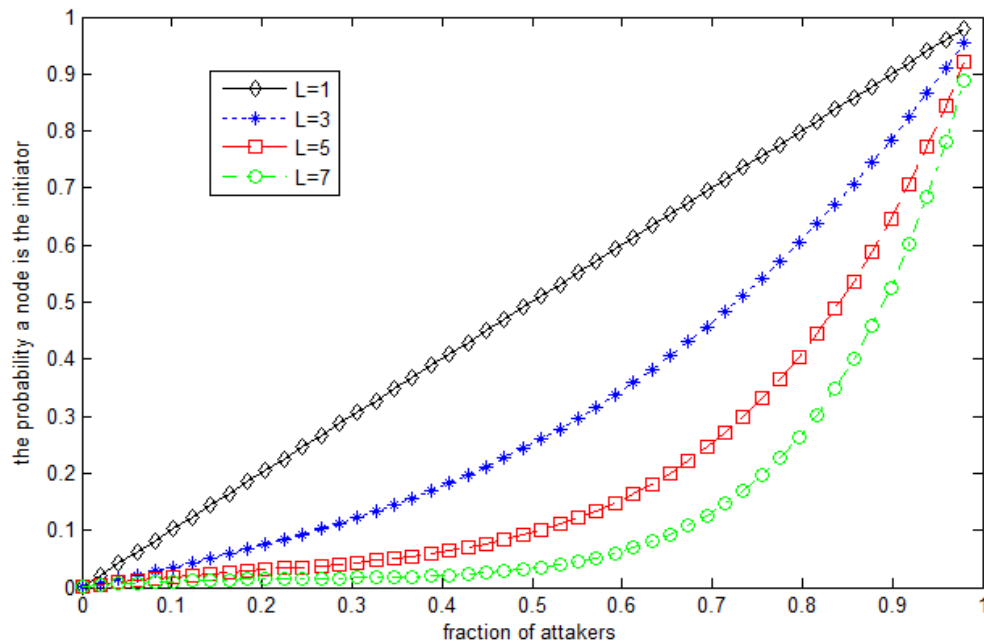


Figure 6.1 Probability to be Initiator

By the similar way, we can also analyze the probability that the intermediate next node is in fact the responder.

6.3 Resilient to Node Failure

We use probability to analyze the resilience and fault tolerance. The parameters list as follows.

- ◆ 2^m : id space;
- ◆ N : number of nodes in the network, network size;
- ◆ L : number of intermediate nodes in the path, the length of routing path;
- ◆ d : error tolerant factor

Since the probability that each ID maps onto a node is $\frac{N}{2^m}$, a node maps onto a non-existing node with the probability of $1 - \frac{N}{2^m}$. The number of nodes Z can help forwarding the message if any of them exists. The restriction is that the overlap of identities is larger than or equal to the error tolerant factor d . The estimation of Z lists as follows.

$$Z = C_d^m 2^{m-d} - C_{d+1}^m 2^{m-(d+1)} + \dots + (-1)^{m-d} C_m^m 2^0$$

It is clear that Z is influenced by the value of error tolerant factor d . If all the Z nodes that can help forwarding do not exist, then the message would be failed to transfer. Hence, when at least one node to help forwarding exists, the probability to forward the message successfully is $1 - \left(1 - \frac{N}{2^m}\right)^Z$. The routing path has L intermediate nodes to route through. If every intermediate node has at least one node that can help forwarding, then the message can be transferred successfully through the routing path.

Therefore, the probability of path success can be shown as: $\left[1 - \left(1 - \frac{N}{2^m}\right)^Z\right]^L$, where L

means the length of routing path.

The probability that each ID maps onto non-existing node decreases with increasing N . For every node, the successful forward probability increases while Z is getting larger. However, the path length L grows inversely proportional to the forwarding probability. Therefore, the level of anonymity provided by AFATOR is inversely proportional to the successful forwarding probability of the routing path. This is a tradeoff between efficiency and the level of anonymity.

Figure 6.2 presents the result of the forward probability with different error tolerant factors and network size. According to the system requirements, fault-tolerance can be tuned by error tolerant factor and the length of routing path.

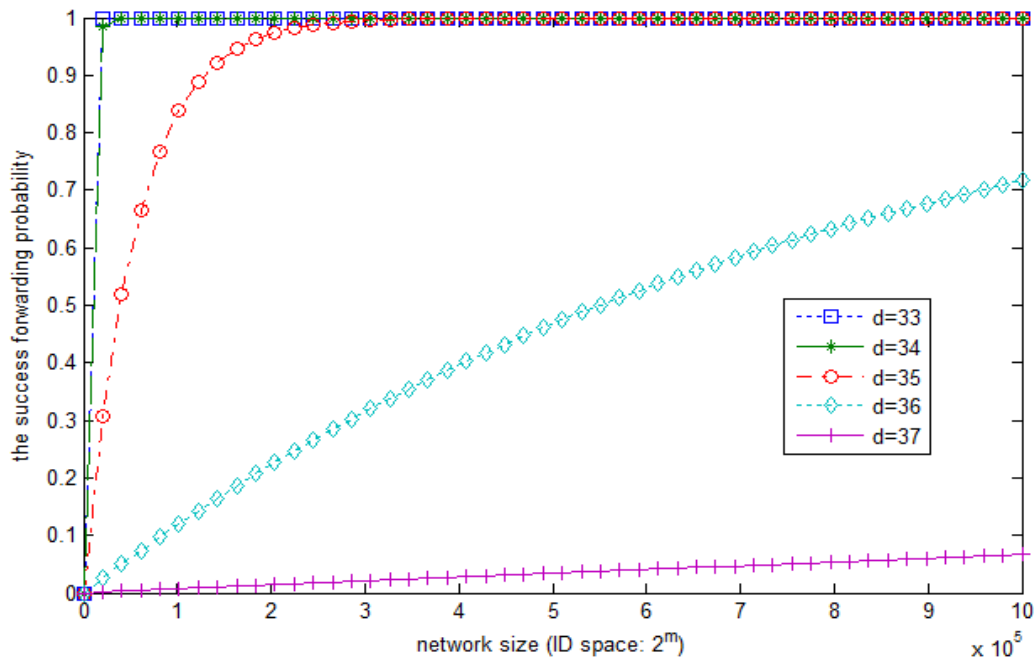


Figure 6.2 Forward probability with various error tolerant factors

If the routing path fails with high probability, then the routing path should reconstruct frequently. After a large number of reconstructions, to identify the initiator participating in the path is much easier.

In AFATOR, the probability of routing path failure is very low because of the tunable error tolerant factor d . Therefore, we not only improve performance by reducing the path reconstruction time but also strengthen our robustness to the degradation attacks [43].

6.4 Against traffic Analysis

Since all the packets are encrypted in a layered manner from the last hop to the first hop by their public keys, the incoming packets and the outgoing packets for every intermediate node are different in packet headers, size, and patterns. The encryption makes the packets indistinguishable from data flows. Cover traffic, which means fake messages would be send from every node per random time period,

prevents a global observation from using traffic analysis to identify the initiator.

But the adversary can find some relationships between those incoming and outgoing packets for the node by using timing analysis.

6.5 Summery

The initiator can select the number of the intermediate nodes in the path and the value of the error tolerant factor to control tradeoffs between churn resilience, anonymity and overhead.



Chapter 7

Conclusion

Our routing protocol, AFATOR, provides anonymity against adversaries without proxies. We use layered encryption and random intermediaries to achieve anonymity. We also achieve unlinkability between initiator and responder without being identified from adversaries. Every node in the path knows only the previous hop and the next hop. It is easy to recover the routing path without request re-transmission. By using Fuzzy Identity-Based Encryption (Fuzzy IBE) [4], a user can decrypt a cipher-text encrypted with other's public key if and only if the two users are within a certain distance. Thus, any node can easily take over message forwarding if its neighbor node fails. At last, AFATOR uses smallest key storage and leaks less information about the responder.

Reference

- [1] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma and Steven Lim, “**A Survey and Comparison of Peer-to-Peer Overlay Network Schemes,**” IEEE Communications survey and tutorial, Mar. 2004.
- [2] Yingwu Zhu and Yiming Hu, “**SurePath: An Approach to Resilient Anonymous Routing,**” International Journal of Network Security (IJNS) Mar. 2008.
- [3] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed, “**Anonymous Connections and Onion Routing,**” IEEE Journal on Selected Areas in Communication Special Issue, 1998.
- [4] Sahai and B. Waters, “**Fuzzy Identity-Based Encryption,**” In Eurocrypt 2005, LNCS 3494, pp. 457-473, Springer-Verlag, 2005.
- [5] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan, “**Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications,**” IEEE/ACM Transactions on Networking.
- [6] Ben Y. Zhao, John Kubiawicz, and Anthony D. Joseph, “**Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing,**” UC Berkeley, 2001.
- [7] A. Rowstron and P. Druschel, “**Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,**” In Middleware, November 2001.
- [8] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Anthony D. Joseph, and John D. Kubiawicz, “**Exploiting Routing Redundancy via Structured Peer-to-Peer**

- Overlays,”** ICNP, 2003.
- [9] Adi Shamir, “**Identity-Based Cryptosystems and Signature Schemes,**” Advances in Cryptology, Lecture Notes in Computer Science, 1984
- [10] Dan Boneh, and Matthew Franklin, “**Identity-based Encryption from the Weil pairing,**” In J. Kilian, editor, Advances in Cryptology, Springer-Verlag, Lecture Notes in Computer Science, pp. 213-229, 2001.
- [11] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron, “**Cashmere: Resilient Anonymous Routing,**” NSDI, 2005.
- [12] Aameek Singh, Bugra Gedik, Ling Liu, “**Agyaat: Mutual Anonymity over Structured P2P Networks,**” In Emerald Internet Research Journal (Special Issue on Privacy and Anonymity in the Digital Era), Volume-16, Issue-2, 2006.
- [13] Michael J. Freedman and Robert Morris, “**Tarzan: a peer-to-peer anonymizing network layer,**” ACM CCS, Nov. 2002.
- [14] Michael K. Reiter and Aviell D. Rubin, “**Crowds: anonymity for Web transactions,**” ACM Transactions on Information and System Security, 1998.
- [15] Nikita Borisov, and Jason Waddle, “**Anonymity in Structured Peer-to-Peer Overlay Networks,**” Technical report, UC Berkeley, May 2005.
- [16] Michael Kinateder, Ralf Terdic, and Kurt Rothermel, “**Strong pseudonymous communication for peer-to-peer reputation systems,**” ACM symposium on Applied computing, Mar. 2005.
- [17] Roger Dingledine, Nick Mathewson, and Paul Syverson, “**Tor: The Second-Generation Onion Router,**” USENIX Security Symposium, Aug. 2004.
- [18] Giuseppe Ciaccio, “**Recipient Anonymity in a Structured Overlay,**” AICT-ICIW, Feb. 2006.
- [19] M. Caesar, M. Castro, E. Nightingale, G. O’Shea and A. Rowstron, “**Virtual**

- Ring Routing: Network routing inspired by DHTs,”** Sigcomm, Sep. 2006.
- [20] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, “**Secure routing for structured peer-to-peer overlay networks,”** In OSDI, December 2002.
- [21] B. Ford, “**Unmanaged Internet Protocol: Taming the edge network management crisis,”** In HotNets II, November 2003.
- [22] M. Choudary Gorantla, Raju Gangishetti, and Ashutosh Saxena, “**A Survey on ID-Based Cryptographic Primitives,”** Cryptology ePrint Archive: Report 2005/094. <http://eprint.iacr.org/2005/094>.
- [23] J. Baek, W. Susilo and J. Zhou, “**New Constructions of Fuzzy Identity-Based Encryption,”** ASIACCS, Mar. 2007.
- [24] Khanh V. Nguyen, “**Simplifying Peer-to-Peer Device Authentication Using Identity-Based Cryptography,”** ICNS 2006.
- [25] Charles, “**Information Leak in the Chord Lookup Protocol,”** Peer-to-Peer Computing, 2004.
- [26] Souvik Ray and Zhao Zhang, “**An Information-Theoretic Framework for Analyzing Leak of Privacy in Distributed Hash Tables,”** Peer-to-Peer Computing, Sept. 2007.
- [27] Youn-Ho Lee, Heeyoul Kim, Byungchun Chung, Jaewon Lee and Hyunsoo Yoon, “**On-demand Secure Routing Protocol for Ad Hoc Network using ID based Cryptosystem,”** PDCAT, 2003.
- [28] Song Hong, Yang Luming, Wang Weiping, and Duan Guihua, “**A Delay Demand-based Anonymous Communication Mechanism,”** Communications and Networking in China, Oct. 2006.
- [29] Wei Ren, Yoohwan Kim, Ju-Yeon Jo, Mei Yang and Yingtao Jiang, “**IdSRF:**

ID-based Secure Routing Framework for Wireless Ad-Hoc Networks,” ITNG, 2007.

[30] Gnutella (2002), available at: <http://gnutella.wego.com/>

[31] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr, “**Towards an analysis of onion routing security,**” In Proc. of PET, July 2001.

[32] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “**Low-resource routing attacks against anonymous systems,**” Technical Report CU-CS-1025-07, University of Colorado at Boulder, Feb 2007.

[33] Geng Yang, Chunming Rong, Christian Veigner, Jiangtao Wang, Hongbing Cheng, “**Identity-Based Key Agreement and Encryption for Wireless Sensor Networks,**” IJCSNS, May 2006.

[34] Napster, available at <http://www.napster.com/>

[35] P. Maymounkov and D. Mazieres, “**Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,**” Proc. IPTPS, Cambridge, MA, USA, Feb. 2002, pp. 53–65.

[36] I. Clarke et al., “**Freenet: A Distributed Anonymous Information Storage and Retrieval System,**” available at <http://freenetproject.org/freenet.pdf>, 1999.

[37] FastTrack Peer-to-Peer Technology Company, available at <http://www.fasttrack.nu/>, 2001.

[38] The Overnet File-sharing Network, available at <http://www.overnet.com/>, 2002.

[39] G. Ciaccio. The NEBLO homepage, available at <http://www.disi.unige.it/project/neblo/>.

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

2006.

- [41] Matthew Pirretti, Patrick Traynor, Patrick McDaniel and Brent Waters, “**Secure attribute-based systems,**” In Proc. of the ACM conference on Computer and communications security, 2006.
- [42] Bartolini, S. and Branovic, I. and Giorgi, R. and Martinelli, E., “**A Performance Evaluation of ARM ISA Extension for Elliptic Curve Cryptography over Binary Finite Fields,**” Computer Architecture and High Performance Computing, 2004.
- [43] Wright, M., Adler, M., Levine, B. N., and Shields, C. “**An analysis of the degradation of anonymous protocols,**” In Proc. of NDSS (Feb 2002).
- [44] Eberle, H. and Gura, N. and Shantz, S.C. and Gupta, V. and Rarick, L. and Sundaram, S., “**A public-key cryptographic processor for RSA and ECC,**” Application-Specific Systems, Architectures and Processors, 2004.

