Node Management without Directory Servers in DHT-based Anonymous Communication Systems using ID-based Encryption

Hiroyuki Tanaka, Shoichi Saito, and Hiroshi Matsuo
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan
tanaka@matlab.nitech.ac.jp, shoichi@nitech.ac.jp, matsuo@nitech.ac.jp

Abstract

Since the problem of disclosing personal information on the Internet continues to increase, many anonymous communication systems have been studied. Such systems usually use directory servers to manage the public keys of participant nodes and such node information as IP address, port number, and NodeID. However, this reduces anonymity because the query messages for the directory servers can give adversaries route information of anonymous communication channels. To solve this problem, applying ID-Based Encryption has been proposed, but in the existing method, directory servers continue to exist.

Our novel method, which can grasp assigned NodeIDs without directory servers, can be applied to existing Distributed Hash Table (DHT)-based anonymous communication systems. Our proposal enhances scalability. This paper describes the structure of our proposed system and its application.

1. Introduction

As Internet proliferation continues, it provides more and more services, some of which need high confidentiality, including medical treatment and psychological counseling. Hence, research is intensely investigating anonymous communication systems to ensure anonymity on the Internet. Anonymous communication must satisfy the following three requirements [11]: a source node cannot be identified, a destination node cannot be identified, and the data flow cannot be traced. These properties are collectively called *anonymity*, and communication with anonymity is called *anonymous communication*. Furthermore, communication channels with anonymous communication are called *anonymous communication channels*.

A typical technique to realize anonymous communication is a multistage relaying method that uses such multiple encryptions as Onion Routing [4, 16]. This method sends messages to a destination by multiple relay nodes, and a source multiply encrypts messages using the public keys of relay nodes and the destination to conceal the final destination from all relay nodes. Relay nodes can only get the next-hop node after decrypting the messages.

The method requires a source to obtain the public keys and such node information of each relay node as IP address, port number, and NodeID. For that purpose, Cashmere [20], Bifrost [7], and Bluemoon [13] need directory servers to manage the public keys and node information. However, the method compromises anonymity because query messages for directory servers can reveal the routing information of anonymous communication channels to adversaries. This is obvious from research that infers client behavior from name resolution by DNS [5, 18].

To solve the first problem of obtaining public keys, anonymous communication with ID-Based Encryption (IBE) [2] has been proposed [6]. With IBE, communication for obtaining public keys can be omitted. However, sources must choose the NodeIDs, which are IDs for IBE, of all relay nodes before building new anonymous communication channels. But the sources do not know how many nodes have already joined and been assigned NodeIDs. Hence, they cannot choose any NodeIDs for relay nodes. Because Kate et al.'s proposal [6] requires directory servers to distribute assigned NodeIDs, it cannot remove directory servers from anonymous communication systems. Moreover, directory servers become bottlenecked and reduce scalability. We propose a novel method that can grasp the assigned NodeIDs without directory servers and improve scalability.

The second problem is how to acquire the node information of the destination. Tor [3] introduces Introduction Points to solve it without directory servers. Introduction Points are nodes that arrange to meet sources and destinations. However, the solution reduces anonymity, because an Introduction Point and a destination share a one-on-one relationship. Therefore, the Introduction Point can learn the related destination. We expand Introduction Point to improve anonymity.

In Section 2, we describe the outline and the problems of existing anonymous communication systems. Section 3 explains ID-based encryption and its problems. Section 4 introduces the details of our proposal. Section 5 describes how we apply it to existing Distributed Hash Table (DHT)-based anonymous communication systems. Section 6 describes its implementation and performance analysis. Section 7 provides a conclusion.

2. Related work and its problems

This section outlines and describes the problems of existing anonymous communication systems.

2.1. Multistage relay using multiple encryptions

In this section, we describe existing anonymous communication methods using multistage relay and multiple encryptions.

2.1.1. Outline. In IP communication, a source and a destination have to know each other’s IP addresses for communication. Hence, in anonymous communication over an IP to conceal the source and the destination, messages must be relayed by many nodes [4, 16].

No relay node can identify a source and a destination to protect anonymity with the multistage relay. However, messages must obviously be delivered. The realization of the above requires that each relay node only learn the node information of the nodes located before and after it on the anonymous communication channel. Public key cryptography solves this problem. Sources encrypt each piece of next-hop node information as routing information in the reverse order of the relaying. Relay nodes can only get their next-hop node. The relay nodes cannot identify other nodes, except themselves, the next-hop node, and the previous node on the channel. In the end, no relay nodes can learn the entire anonymous communication channel. The above outlines anonymous communication with multistage relay and multiple encryptions. This method is used by many anonymous communication systems [3, 4, 7, 8, 10, 19, 20].

2.1.2. Bifrost. Bifrost is an example of an anonymous communication system that combines multiple encryption and node management using DHT. Bifrost uses Chord [15] for DHT.

Bifrost has the following three features. First, a source can locate an arbitrary destination position in an anonymous communication channel. Second, routes to and from a destination are difference. Third, Bifrost introduces a *receiver area* to reduce the processing time of message encryption and searching for a next node. A receiver area is a subspace of an ID space and consists of consecutive Chord IDs. The beginning node of a receiver area receives a message, which is repeatedly relayed to successors by nodes in the receiver area until the message reaches the receiver area’s end node. Finally, the end node of the receiver area sends the message to the beginning node of the next receiver area. A source can arbitrarily set the number of receiver areas and a beginning and ending nodes for them.

The construction of anonymous communication channels in Bifrost is different from that in Tor, where a source constructs an anonymous communication channel by telescoping construction. By contrast, in Bifrost, a source can construct an anonymous communication channel by one round trip of one message. Hence, Bifrost has both construction and data messages. Construction message M_c consists of header H_c and body B_c. The header has shared keys between the source and relay nodes and the route information. The body has data to a destination. The header and body are separately encrypted. The structure of M_c is as follows:

$$M_c = H_c|B_c,$$

$$H_c = P_1(S_1|ID_2|P_2(S_2|ID_3|P_3(\ldots P_N(S_N|null)))),$$

$$B_c = S_1(S_2(S_3(\ldots P_D(S_D|H_R)))),$$

where “|” is union. P_i is a public key of node i and $P_i()$ represents encryption using P_i. S_i is a shared key between a source and node i and $S_i()$ represents encryption using S_i. ID_i is an ID of the node i. H_R, which is a reply header with reply route information differently from a route to a destination, is created by the source and used to reply by a destination. D represents a destination node, and N represents the Nth relay node, which is an end node of an anonymous communication channel. In general, D differs from N because a source can set an arbitrary destination position in an anonymous communication channel in Bifrost.

The end nodes of each receiver area decrypt messages using their own private keys and send the messages to the next relay node in the next receiver area. The innermost messages are encrypted using a destination’s public key. Only the destination can decrypt it and receive the plain data.

Reply construction message M_{cr} and relay header H_{cr} are created as follows:

$$M_{cr} = H_{cr}|B_{cr}$$

$$H_{cr} = P_{T_1}(S_{T_1}|ID_{T_2}|P_{T_2}(\ldots P_{T_n}(S_{T_n}|null))),$$

where the nodes from T_1 to T_n are relay nodes on the reply channel. The source, which is located the next of T_s ($s<n$) in the reply route, receives the reply message through the reply channel and gets the reply messages, which are enclosed in a reply body. Reply body B_{cr} is encrypted and relayed to the source by every reply relay nodes. The destination only encrypts a reply message, *rmsg*, once as follows:

$$B_{cr} = S_D(rmsg).$$

Eventually B_{cr}, which has reached the source, becomes:

$$B_{cr} = S_{T_s}(\ldots S_{T_1}(S_D(rmsg))).$$

Note that B_{cr} is encrypted on the relay nodes that successfully decrypted the corresponding header. In this way, the destination never learns about the shared keys used on the reply route.

After the construction of an anonymous communication channel, a source communicates using data messages, which are sent by a constructed anonymous communication channel built by a construction message. Therefore, data messages only have body B_d and do not have a header for routing. The following is the structure of data message M_d:

$$M_d = null|B_d,$$

$$B_d = S_1(S_2(S_3(\ldots S_D(msg)))).$$

Each relay node decrypts the message and relays it. Only a destination can decrypt the message using S_D and obtain *msg*. On the other hand, in a reply channel, the structure of relay data message M_{dr} is the same structure with M_d and each relay node encrypts reply data body B_{dr} on the same way as reply construction message B_{cr}.

2.2. Problems of node and public key management

To construct an anonymous communication channel using multiple encryptions, a source first arbitrarily chooses some relay nodes. Hence, the source must obtain their public keys. There are three methods for managing nodes and distributing public keys. First, a source beforehand obtains a list of all participant nodes, including node information and a public key, as in the case of Tor. Second, all nodes are controlled by DHT, whose public keys are managed by directory servers. A source obtains node information with a DHT and public keys with such directory servers as Bifrost, Cashmere, or Bluemoon. Cashmere's off-line central authority (CA) and Bluemoon's official CA are equivalent to directory servers. Third, public keys are built by IBEs such as Pairing Based Onion Routing [6].

In the first method, a source can arbitrarily choose relay nodes from the list of all participant nodes. It does not need extra communication to get relay node information. Consequently, relay nodes cannot be inferred by communication. A source can also obtain public keys with the list without extra communication. However, a source periodically gets a large list in this method. Therefore, it needs to pay a large cost to maintain the list.

In the second method, a source only chooses arbitrary DHT-IDs, and nodes managing them become relay nodes. A source does not need to look up relay nodes because each relay node looks up the IP address of the next-hop node using the next-hop DHT-ID in a relaying message. The method does not require communication to obtain relay node candidates. However, a source needs to look up public keys in directory servers. Adversaries can surreptitiously discover relay nodes by monitoring query messages from the source to the directory servers.

In the third method, a source can arbitrarily choose relay nodes, as in the first method, because both methods need directory servers to distribute a list of participant nodes. Therefore, this method also needs to pay a large cost to maintain the list. Moreover, adversaries can learn all nodes easily, and participant anonymity suffers.

In the first method, node management and public key management do not have scalability. In the second method, node management has scalability but public key management does not. In the third method, public key management has scalability but node management does not, just like in the first method. Hence, anonymous communication systems can ensure scalability by combining the second and third methods.

2.3. Problems of retrieval destination's NodeID

Generally, sources do not know the destination's NodeIDs as node information in DHT-based anonymous communication systems. They need to search for the NodeIDs of destinations before starting to connect. There are some searching methods. The first, a simple discovery way, floods the destination's NodeIDs. However, flooding is needed whenever a destination's NodeID is renewed. Furthermore, this method is not scalable. The second way is an existing secure service discovery method [17]. In this method, because search keywords are clear text, a third person can learn what service is being provided and being searched for by any nodes. The last, Introduction Point, was proposed by Tor to contact hidden servers. Introduction Points are nodes that arrange to meet sources and destinations and can respond to the changes of the destination NodeIDs. An Introduction Point and a destination share a one-on-one relationship. Therefore, an Introduction Point can learn the related destination. We believe this relationship reduces anonymity.

3. Introducing IBE and its Problems

This section outlines IBE and shows three new problems for applying IBE to DHT-based anonymous communication systems.

3.1. ID-Based Encryption

IBE is a kind of public-key cryptography that derives a public key from an ID. IBE encrypts with a hash function and a destination's ID called an IBE-ID, which can be an arbitrary string. The hash function is shared by all users. IBE has a Private Key Generator (PKG), which we assume is operated by a trusted third party. PKG generates the master private key and a hash function, which it disseminates to all nodes. PKG also generates a private key of each node using the master private key and each IBE-ID. For IBE, a destination's public key does not need verification because it is based on the reliability of an IBE-ID and the hash function. In addition, Boneh and Franklin [2] suggest using a distributed PKG to realize a robust system.

3.2. Introducing IBE

Introducing IBE can prevent anonymity from being reduced by monitoring query messages for directory servers to solve the problems defined in 2.2. Since regarding a NodeID as an IBE-ID can retrench the communication for obtaining a public key, we can achieve a robust anonymous communication method. Our proposal can be applied to various DHT-based anonymous communication systems.

3.3. Problems of introducing IBE

To introduce IBE into a DHT-based communication system and realize a system that does not depend on directory servers, we need to solve the following challenges. Our proposal, which solves them, is described in detail in the following sections.

Unique NodeID assignment With IBE, PKG generates and assigns a private key corresponding to a NodeID as an IBE-ID. If a NodeID overlaps, some nodes have identical private keys and can decrypt the same message. Therefore, a collision-free NodeID assignment method is needed.

Verification of assigned NodeIDs Sources can even adopt arbitrary NodeIDs as relay nodes that have not been assigned yet in a system and encrypt messages with those NodeIDs as IBE-IDs. IBEs can create public keys that are not corresponding nodes, and no node can decrypt encrypted messages with the public keys. Since the messages must be encrypted only using the previously assigned NodeIDs, a new problem occurs: *how to get the previously assigned NodeIDs*. If a source communicates to verify previously assigned NodeIDs, anonymity is reduced. Therefore, we need a verification method for previously assigned NodeIDs that does not rely on communication.

Retrieval of a destination's NodeID In the NodeID assignment of anonymous communication systems, a NodeID is mainly a hash value of an IP address or a random value. The reasons for a random value for a NodeID are to uniformly assign the NodeIDs in DHT's ID space [8] and to prevent adversaries from choosing the NodeIDs they want [7, 9]. In addition, some systems [13, 20] use a random value for ID assignment without describing why. If NodeIDs are assigned randomly, we need to retrieve a destination's NodeID before beginning communication. But a source cannot communicate to retrieve the destination's NodeID for the same reason as the retrieval of public keys.

4. Proposed methods

We propose novel methods that resolve the problems mentioned in 3.3 and ensure that a node grasps the assigned NodeIDs without directory servers. Our proposal also enhances scalability.

4.1. Unique NodeID assignment

An IBE-based system must not allow duplication of identical NodeIDs because a public key is derived from a NodeID as an IBE-ID. Therefore, we introduce a NodeID Allocator (NIA) that assigns one NodeID to just one node when the node joins an anonymous communication system. We assume that NIA is operated by a trusted third party.

The NIA assigns a NodeID with its digital signature to a participating node. A PKG verifies the NIA's digital signature, generates a private key corresponding to the NodeID, and hands it over to the participating node. This prevents faked NodeIDs. Connections between participating nodes and NIA/PKG are over SSL.

4.2. Verification of previously assigned NodeIDs

To encrypt messages using only the assigned NodeIDs and to protect anonymity, obtaining assigned NodeIDs needs no communication for looking up nodes. However, it is difficult to obtain the latest assigned NodeID. The goal for obtaining assigned NodeIDs is to discern nodes that can decrypt messages rather than acquire the latest assigned NodeID. Therefore, we propose a method that loosely grasps the assigned NodeIDs by dividing them into groups. The details are discussed below.

Such a simple ascending order NodeID assignment as $0, 1, \cdots$ does not allow nodes to be sought effectively

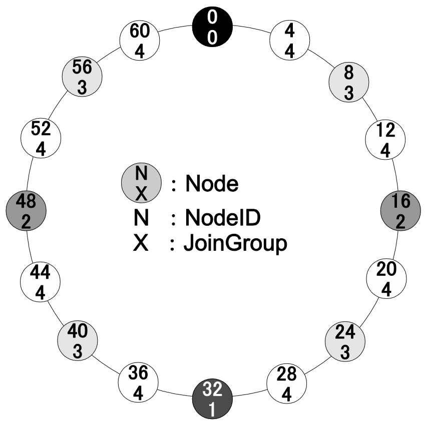


Figure 1. NodeID and JoinGroup when NodeID consists of 6 bits

because these NodeIDs are not equalized in a DHT's ID space. Moreover, nodes cannot know the latest assigned NodeID because the latest node is not necessarily on their routing table. Therefore, we introduce *JoinNumbers,* which are assigned in ascending order to the nodes. Reversing a JoinNumber in the order of the bits is a NodeID (e.g., in case of 4 bits, 0001→1000, 0010→0100). Searching efficiency with DHT is not reduced because these NodeIDs are evenly spread over the DHT's ID space. All new NodeIDs with this method are middles of the adjacent nodes at all times. Hence, all nodes can know the latest state of the NodeID assignment by reversing the next NodeIDs.

However since this method does not have a random nature, we add a partially random nature to it. We call a set of nodes whose JoinNumbers can be found from 2^{i-1} to $2^{i}-1(i\geq1)$ the "*ith JoinGroup,*" and randomly assign JoinNumbers to each JoinGroup. Here, the 0th JoinGroup has only one node of NodeID 0. JoinGroup is assigned in ascending order. More than one JoinGroup cannot be assigned simultaneously.

Figure 1 shows an example of a NodeID assignment when the size of the ID space is 6 bits. First, one node of NodeID 0 joins the 0th JoinGroup. If a new node joins, it is assigned to JoinNumber 000001, NodeID is 100000 (32). This belongs to the 1st JoinGroup. New participating nodes are randomly assigned JoinNumbers 000010 and 000011, their NodeIDs are 010000 (16) and 110000 (48) as the 2nd JoinGroup. The NodeIDs of the new JoinGroup are located between the NodeIDs of the already assigned JoinGroups. New participating nodes are assigned NodeIDs in the same way.

The proposed method has the following characteristic: a node itself or the next nodes of it also belong to the current JoinGroup, which is the latest or the immediately preceding JoinGroup. Therefore, all nodes can know the current JoinGroup only using their own routing tables because the routing tables of each node have the next node. Each node can judge that nodes belong to the current and previous JoinGroups have already been assigned.

Here, the proposed method can have a partial random nature on the NodeID assignment. Our verification method requires that any NodeIDs which belong to the current and previous JoinGroups have already been assigned before being assigned to a new JoinGroup. Our proposed method ensures that the assignment of JoinGroups the current and previous JoinGroups have already finished. The order of the NodeID assignment in a JoinGroup is not designated in the above requirement and guarantee. In other words, if NodeIDs are randomly assigned in a JoinGroup, it does not affect the verification of the previously assigned NodeIDs.

Consequently, our proposed method provides verification without communication of the previously assigned NodeIDs and the random NodeID assignment.

4.3. Retrieval of destination's NodeID

To send messages, sources need to know the destination's NodeID. But it is assigned randomly. We suppose that destinations offer their own *Service Names* (SN) to the public in place of their own NodeIDs. Hence, sources can search for destinations with their SNs.

Our proposed method of searching and connecting to a destination is based on IBE and the Introduction Points of Tor. We enhance the Introduction Point to increase anonymity and call it a *Multi-connected Introduction Point* (MIP). A node managing the hash value of an SN is an MIP of the SN. Since MIPs simultaneously maintain connections to many destinations and relay ID retrieval messages from sources to all the connected destinations, many destinations that share one MIP receive the same message. We propose using IBE so that the only true destination can receive it. A sender encrypts an ID retrieval message by an SN as a public key of IBE. The encrypted message can only be decrypted by the true destination that has the IBE's private key. Therefore only the destination can receive and respond to it.

To keep multiple connections to one MIP, a number of MIPs needs to be less than a number of destinations. Nodes belonging to a small number of JoinGroups can only be an MIP. The number of JoinGroups for MIPs depends on the number of destinations in a system.

5. Application of proposal to existing systems

In this section, we apply our proposal to existing DHT-based anonymous communication systems.

5.1. Applications to DHT-based anonymous communication systems

To apply our proposal to DHT-based anonymous communication systems, the system needs to add PKG and NIA. Nodes are needed to add the following change and mechanisms:

- Changing join-protocol to our proposal
- MIP mechanism
- Requesting a NodeID mechanism for sources
- Providing a NodeID mechanism for destinations

The procedures after retrieving the destination's NodeID are not changed.

5.2. Join protocol of proposal

To apply our proposal to DHT-based anonymous communication systems, we need to introduce an NIA and a PKG. All participant nodes assume that NIA and PKG can be trusted. Therefore, they can be combined into one. The join-protocol consists of the following four steps:

1. A joining node sends an assignment request of a NodeID to the NIA when the node enters the system.
2. The NIA assigns a new NodeID with NIA's digital signature to the node.
3. The node notifies the PKG about the assigned NodeID with the NIA's digital signature.
4. The PKG verifies the NIA's digital signature, generates a corresponding private key, and sends it and the IBE's hash function.

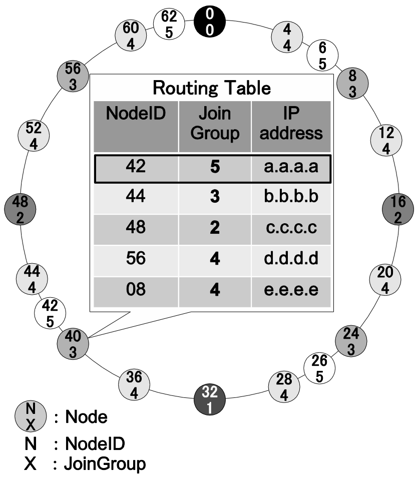
These communications are done over SSL.

5.3. NodeID assignment method

This section describes our proposed NodeID assignment method over Chord and Pastry [14], which are base DHTs. Chord is used in SALSA [10] and Bifrost. Pastry is used in Cashmere. Our proposal assigns NodeIDs to infer previously assigned NodeIDs from such known information as a DHT routing table. The NodeID assignment method is shown in Figure 2. In this method, the NodeIDs in one JoinGroup are randomly assigned. This process is repeatedly executed whenever new JoinGroups are assigned. For Chord and Pastry, the ID space sizes are 2^{160}. As an initial condition, we consider that just one node (NodeID:0) has participated in the system. NIA assigns a NodeID from the 1st JoinGroup. In this method, if a node

```
NodeIDAssgin(JoinGroup $i$){
   Generate $JoinNumber$[] from $2^{i-1}$ to $2^i - 1$
   RandomSort($JoinNumber$[])
   **for** j=0 to $2^{i-1} - 1$ **do**
      $NodeID$ = bit_order_reverse($JoinNumber$[j])
   **end for**
}
```

Figure 2. Pseudo code of NodeID assignment method



Routing Table

NodeID	Join Group	IP address
42	5	a.a.a.a
44	3	b.b.b.b
48	2	c.c.c.c
56	4	d.d.d.d
08	4	e.e.e.e

Figure 3. Verification of assigned NodeID

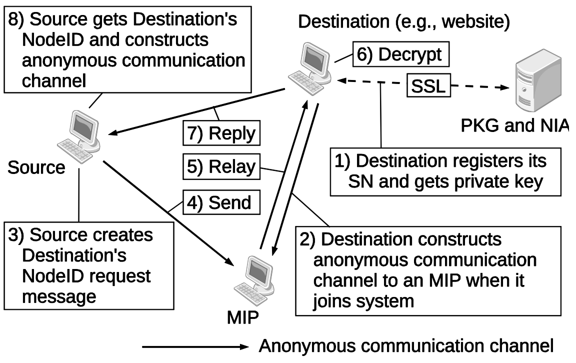
belonging to the *x*th JoinGroup is in its own routing table, which is the Finger Table of Chord and the Leaf Set, the Routing Table and the Neighborhood Set of Pastry, the previous assigned JoinGroups prove to be less than *x*th. Consequently, sources can choose relay nodes from nodes belonging to from 0th to the (*x*–1)th JoinGroups.

Examples of the proposed NodeID assignment method and the verification method of the assigned NodeID are illustrated in Figure 3. Here, the 5th JoinGroup is assigning. At this time, the node of NodeID 40 starts to verify the assigned NodeIDs. The node checks its own routing table and finds a node of the 5th JoinGroup. Therefore, it can judge that the nodes in the 0th to the 4th JoinGroups have already been assigned.

We need to separately consider an assignment method for other DHTs.

5.4. Message structure of Bifrost with proposed method

This section describes the structure of a Bifrost message with the proposed method. The following is the structure of construction message M'_c:



8) Source gets Destination's NodeID and constructs anonymous communication channel
Destination (e.g., website)
6) Decrypt
SSL
PKG and NIA
7) Reply
Source
5) Relay
1) Destination registers its SN and gets private key
4) Send
3) Source creates Destination's NodeID request message
2) Destination constructs anonymous communication channel to an MIP when it joins system
MIP
Anonymous communication channel

Figure 4. Retrieval of destination's NodeID using MIP

$$M'_c = H'_c | B'_c,$$
$$H'_c = P'_1(S_1|ID_2|P'_2(S_2|ID_3|P'_3(..P'_N(S_N|null)))),$$
$$B'_c = S_1(S_2(S_3(..P'_D(S_D|H_R))))),$$

where P'_i is a public key derived from NodeID i and $P'_i()$ is encryption using P'_i. This structure is the same as M_c of traditional Bifrost. The only difference is that the public keys are provided by IBE. The following is the structure of data message M'_d:

$$M'_d = null|B'_d,$$
$$B'_d = S_1(S_2(S_3(\dots S_D(msg)))).$$

M'_d is encrypted using shared keys. The structure of a data message is the same as M_d in traditional Bifrost.

5.5. Retrieval of destination's NodeID

Figure 4 shows a proposed NodeID retrieval protocol using MIP for Bifrost. The following are its details:

Step 1) A destination registers its SN with PKG and obtains a private key corresponding to its SN as a public key.

Step 2) The destination constructs an anonymous communication channel between itself and the MIP in charge of the hash value of the SN.

Step 3) When a source connects to the destination, it generates a NodeID retrieval request message for the destination. The message is multiple encrypted using some relay nodes' NodeIDs and the destination's SN as public keys. Furthermore, it contains a reply header to construct a reply anonymous communication channel. The following is the request message's REQ structure:

$$REQ = P'_{SN}(H_{DtoS}|IdR|S_D),$$

where H_{XtoY} is a header of a multistage encrypted message from node X to node Y, IdR is an ID retrieval command, and node S is the source. At this time, since the source does not know the destination's NodeID, it cannot send the message to the destination.

Step 4) The source sends a construction message to the MIP managing the hash value of the SN. The message structure is the same with M'_c. Message header H_{StoMIP}, which has route information of an anonymous communication channel between the source and the MIP, is encrypted as follows:

$$H_{StoMIP} = P'_{E_1}(S_{E_1}|ID_{E_2}|P'_{E_2}(..P'_{E_m}(S_{E_m}|null))),$$

where notes from E_1 to E_m are relay nodes. Message body B_{StoMIP}, which is sent from the source to the MIP, is encrypted as follows:

$$B_{StoMIP} = S_{E_1}(..S_{MIP}(REQ)).$$

Step 5) The relayed messages go through the reply channels built at Step 2 and reach the true destination and other destinations. Encryption of the reply channels is the same with B_{dr}. The relayed messages are encrypted by every relay nodes and become:

$$B_{MIPtoD} = S'_{F_d}(..S'_{F_1}(S'_{MIP}(REQ))),$$

where S'_i is a shared key between node i and node D, which is a source of this channel, and $S'_i()$ is encryption using S'_i. The nodes from F_1 to F_d are relay nodes, and F_d is each previous node of the destinations in each route from the MIP to the destinations.

Step 6) The true destination and other destinations sharing the same MIP receive the message and decrypt REQ using their private keys that correspond to their own SNs.

Step 7) The true destination, which can completely decrypt the message, sends a reply with its own NodeID, ID_D, to the source using an enclosed reply header H_{DtoS} as following:

$$H_{DtoS} = P'_{G_1}(S_{G_1}|ID_{G_2}|P'_{G_2}(..P'_{G_l}(S_{G_l}|null))).$$

Note that nodes from G_1 to G_l are nodes in the route from the destination to the source. Body B_{DtoS}, which has reached the source, becomes:

$$B_{DtoS} = S_{G_s}(..S_{G_1}(S_D(ID_D))),$$

where the node G_s ($s<l$) is the previous node of the source in the route from the destination to the source.

Step 8) Shared keys from S_{G1} to S_{Gs} and S_D are shared by the source and the relay nodes. The source completely decrypts the body B_{DtoS}, obtains the destination's NodeID and constructs an anonymous communication channel between the source and the destination using the destination's NodeID.

Because the source, the MIP, and the destination are connected by anonymous communication channels, they cannot identify each other. Moreover, because the MIP forwards a NodeID request message to two or more destinations, it cannot identify which destination is the true destination of the request message. Hence, the source can retrieve the destination's NodeID without reducing anonymity.

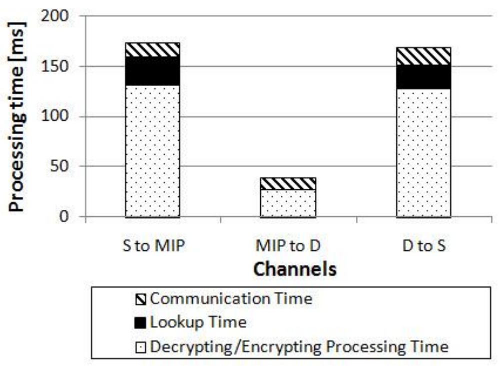


Figure 5. NodeID retrieval time: LAN

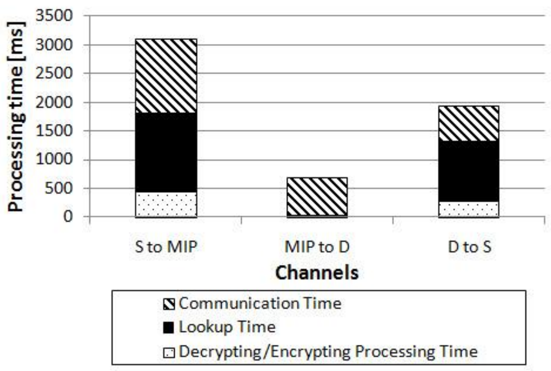


Figure 6. NodeID retrieval time: PlanetLab

Table 1. Performance evaluation of MIP

Destinations relayed by MIP	1	5	
Order of relaying to the true destination	1st	1st	5th
NodeID retrieval time	372ms	376ms	387ms

6. Evaluations

We implemented our proposal and measured its execution time for NodeID retrieval using MIP, which is assumed to be a challenge posed by the introduction of IBE. We combined Bifrost and the IBE library [1].

First, we successfully confirmed that anonymous communication can be realized without directory servers. Next, we performed the following three evaluations: time of destination's NodeID retrieval using MIP, MIP performance, and scalability.

6.1. NodeID retrieval using MIP

In the first measurement, we evaluated in two environments. The first was a local network system connected by 1G-bps networks, and the second was a PlanetLab [12] connected all over the world. In both cases, 32 computers were joined to the environment systems. NodeID request messages go through three channels: the first is a channel from a source to an MIP (Step 4 in Section 5.5), the second is a reply channel from the MIP to a destination (Step 5), and the third is a reply channel from the destination to the source (Step 7). Each channel has three receiver areas and six relay nodes: nine receiver areas and 18 relay nodes in total. An MIP only has one destination in this evaluation.

Figures 5 and 6 show the processing times of three channels for NodeID retrieval in the two environments. Each processing time is composed of communication, lookup, and decrypting/encrypting processing times. The communication time is for sending a message, the lookup time is for searching for the IP addresses of the next relay node, and decrypting/encrypting processing time is for decrypting/encrypting a message. The total times of the NodeID retrieval are 382 ms in the LAN and 5,726 ms in the PlanetLab. The first and third channels occupied about 40% of each channel in both environments because they are constructed when the request messages are conveyed. In PlanetLab, communication delay significantly influences NodeID retrieval time. Therefore, the communication and lookup times in PlanetLab are larger than LAN. NodeID retrieval in PlanetLab requires excessive time. For practical use, this retrieval time is considerable. Hence reducing the channel construction time is a critical future issue.

6.2. Performance evaluation of MIP

We compared NodeID retrieval times when an MIP is in charge of one destination and five destinations. In the case of five destinations, we measured two times in the first and the last orders of relaying to the true destination. This evaluation is performed in the LAN environment. The result is shown in Table 1. The difference between one and fifth of five is 15 ms and between first and fifth of fives is 11ms. Increments are 3ms or less a destination and 1% or less of the total NodeID retrieval time in the LAN. The MIP, a sender of the second channel, only encrypts a message once each destination, because the second channel is a reply channel. Therefore, when the number of destinations relayed by MIP grows, the increase of the NodeID retrieval time is negligible.

6.3. Evaluation of scalability

In this section, we focus on the costs of node management and compare the scalability of the proposed method with an existing method that distributes node information by directory servers. Here, we assume that the number of nodes is *N*.

The existing method needs directory servers to manage the participating nodes. Directory servers manage the node information of *N* nodes. Therefore, the data transfer cost of the directory servers is proportional to the product of the data size of the node information and the number of nodes to which the directory servers communicate. As a result, the cost of the directory servers is $O(N^2)$. In addition, directory servers periodically gather and provide the node information for *N* nodes. On the other hand, our proposed method does not need a directory server. In the proposal, *N* nodes access NIA and PKG only once when joining the system. There is no periodic communication. In addition, NIA only gives out a hash function to participant nodes. Thus, the cost of our proposed method is O(N). For these reasons, our proposed method has more scalability than the existing method that uses directory servers.

7. Conclusion

We proposed a novel method that can grasp previously assigned NodeIDs and obtain a destination's NodeID without directory servers. This method enhances scalability. In addition, we applied our proposed method to an existing DHT-based anonymous communication system and realized a system that does not need directory servers. Furthermore, we described the behavior of existing DHT-based anonymous communication systems by applying our proposal not only to Bifrost but also to many other DHT-based anonymous communication systems. However, robust anonymity is realized in exchange for the high cost of NodeID retrieval processing. In the future, we will address the excessive retrieval time of the destination NodeIDs in Internet environments. Remaining challenges include measures against dynamic joining and leaving of nodes.

Acknowledgements

This research was partly supported by the Grant-in-Aid for Scientific Research (#20500064) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

References

[1] BAO Yiyang. ibe-javapairing. http://en.sourceforge.jp/projects/sfnet_ibe-javapairing/ (Access Date: March 28, 2011).

[2] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[3] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of 13th USENIX Security Symposium*, pages 303–320, 2004.

[4] D. Goldschang, M. Reed, and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *ACM SIGCOMM Computer Communication Review*, 42(2):39–41, 1999.

[5] K. Ishibashi, T. Toyono, and K. Toyama. Detecting Mass-Mailing Worm Infected Hosts by Mining DNS Traffic Data. In *ACM SIGCOMM Workshop on Mining Network Data*, pages 159–164, 2005.

[6] A. Kate, G. M. Zaverucha, and I. Goldberg. Pairing-Based Onion Routing. *In Proceedings of 7th Privacy Enhancing Technologies*, pages 95–112, 2007.

[7] M. Kondo, S. Saito, K. Ishiguro, H. Tanaka, and H. Matsuo. Bifrost: A Novel Anonymous Communication System with DHT. In *Second International Workshop on Reliability, Availability, and Security*, pages 324–329, 2009.

[8] 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*, pages 590–599, 2009.

[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*, pages 161–172, 2009.

[10] 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*, pages 17–26, 2006.

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

[12] PlanetLab. http://www.planet-lab.org/ (Access Date: March 28, 2011).

[13] K. P. N. Puttaswamy, A. Sala, C. Wilson, and B. Y. Zhao. Protecting Anonymity in Dynamic Peer-to-Peer Networks. In *IEEE International Conference on Network Protocols*, pages 104–113, 2008.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the Middleware 2001: IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer–To–Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 SIGCOMM Conference*, pages 149–160, 2001.

[16] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion routing. *IEEE Journal on Specific Areas in Communications*, 16(4):482–494, 1998.

[17] S. Trabelsi and Y. Roudier. Secure Service Discovery with Distributed Registries. In *2nd IEEE Workshop on Service Discovery and Composition in Ubiquitous and Pervasive Environments*, pages 1–6, 2008.

[18] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based Detection of Scanning Worms in an Enterprise Network. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005.

[19] Y. Zhu and Y. Hu. TAP: A Novel Tunneling Approach for Anonymity in Structured P2P Systems. In *International Conference on Parallel Processing*, pages 21–28, 2004.

[20] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient Anonymous Routing. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation*, pages 301–314, 2005.