An Agent-Based Model of Anonymous Communication Protocols

Shigeki Kitazawa
Mitsubishi Electric Corporation
Information Technology R&D Center
Information Security Technology Dept.
5-1-1 Ofuna, Kamakura, Kanagawa,
247-8501, Japan
shigeki@iss.isl.melco.co.jp

Masakazu Soshi, Atsuko Miyaji
School of Information Science,
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Nomi, Ishikawa,
923-1292, Japan
{soshi,miyaji}@jaist.ac.jp

Abstract

So far various anonymous communication protocols have been proposed independently and aimed at different situations. As a result, it is hard to understand the essential structures of those protocols and each protocol must be evaluated and implemented independently.

To solve these problems, we propose an anonymous communication model that can represent a wide variety of anonymous protocols. In our model, we introduce agents *that work in cooperation to realize anonymous communication. This modeling is based on the observation that most of practical anonymous protocols have much commonality in that they have several relay nodes en route from the sender to the receiver to provide anonymity.*

On our model, such agents' behaviors are expressed in a small set of primitive functions. Using these primitive functions, the essential structures of anonymous communication protocols can be described clearly.

1 Introduction

Many researchers have vigorously proposed anonymous communication protocols so far [2, 3, 6, 8, 9, 10]. The most primitive way of providing anonymous communication is for a sender to deliver a message to a receiver via a *proxy* [8] indirectly (hereinafter we call this method "Proxy"). In this way the receiver can not learn the identity of the sender but only the identity of the proxy. Reiter and Rubin developed an anonymous communication system, called Crowds [9], in 1998. Crowds provides anonymity by making a relay node (called "jondo") probabilistically forward a message to another relay node before sending it to the ultimate destination. In 1999, Inoue and Matsumoto proposed a new method for anonymous communication [3] (in this paper we call this "IM"). In IM, a sender divides a message into fragments, then randomly separates them into two groups, and finally forwards each of them to two other relay nodes.

Such anonymous protocols have been developed independently and aimed at different situations. Therefore we are confronted with the following problems. First, it is hard to clarify the essential structures of those protocols. This immediately leads to the difficulty in understanding such protocols. Secondly, each anonymous communication protocols must be evaluated and implemented independently from scratch. That is, experiences of evaluation and implementation of an anonymous protocol would not help in the case of another protocol. Thirdly, it is difficult to evaluate the differences among those protocols. Thus a suitable choice of an anonymous communication protocol in a given situation is not so obvious.

To alleviate these problems, it might appear that what we have to do is only to represent anonymous communication protocols with FDT (Formal Description Techniques), some of which have already been standardized (e.g., SDL [1], LOTOS [5], and Estelle [4]). However, if we limit our discussion to anonymous communication protocols as is the case in this paper, the protocol description with FDT becomes too detailed and complicated unnecessarily. Consequently it is difficult to solve the problems mentioned above.

Therefore in this paper we propose an anonymous communication model that can represent a wide variety of anonymous communication protocols. In our model, we introduce *agents* that work in cooperation on behalf of users to realize anonymous communication. This way of modeling is based on the following observation; most of practical anonymous communication protocols [2, 3, 8, 9, 10] have much commonality in that they have several message relay nodes en route from the sender to the receiver to provide anonymity.

When we describe anonymous communication protocols on our model, behaviors of agents in the model can be ex-

pressed using a set of *primitive functions*. Actually the size of the set can be rather small, so that primitive functions can clarify the essential structure of anonymous communication protocols. Moreover they are useful to develop a new anonymous communication protocol.

Now we can summarize the advantages of our model as follows. First, our model has appropriate formalism and sufficient expressive power to describe anonymous communication protocols. Therefore, if we can develop a single mechanism to execute descriptions on our model, it is possible to actualize a wide variety of anonymous communication on such a single mechanism. Similar discussion can also be applied to evaluation on anonymous protocols. In other words, implementation and evaluation of a wide variety of anonymous communication protocols can be done in one framework of our model. Another advantage of our model is that it gives us a clue to evaluate and classify anonymous communication protocols. Therefore it becomes possible to choose an appropriate anonymous protocols depending on situations.

This paper is organized as follows. Section 2 defines our model. In Section 3 Proxy and Crowds are described on our model as examples. Moreover Crowds and IM are integrated in Section 4 to demonstrate the expressive power of our model. Finally, we conclude this paper in Section 5.

2 The anonymous communication protocol model

In this section, we define our anonymous communication model.

2.1 Basic Concepts

2.1.1 Entities

First of all, we define the entities involved in our anonymous communication model as follows: (1) **Initiator** I: a user who generates a message content V and initiates anonymous communication. (2) **Responder** r: the recipient of the message content V, with whom the initiator I wishes to communicate. (3) **Message relay agent** A: an agent that manages anonymous communication on behalf of a user. Each user has one such agent. The agents of initiator I and responder r are denoted by A_I, A_r, respectively.

2.1.2 Message form and type

On our model, a communication message m exchanged between entities has the following form: $m = (S, R, PI, Pr, PV)$. S is the sender that issues the message into a communication link and R is the receiver that receives the message directly in the link. PI is information about the initiator I, with which the responder r can return its reply to I, e.g., the initiator address or a message (or communication) identifier. Pr is information about r, with which the message can reach r, e.g., the responder address or the encrypted responder address. PV is a message content V, which has possibly been transformed, e.g., encrypted or fragmented.

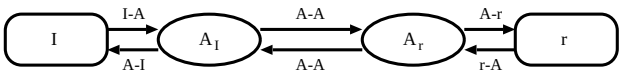

Figure 1. Message communication.

We call PI, Pr, PV *pseudo parameters*. Pseudo parameters are used to exchange a message between I and r while their identities are kept secret. Examples of messages in a few anonymous communication methods are given in Section 3.

Furthermore on our model, according to the type of the sender and the receiver, messages can be classified into the following types: (1) **type I-A**: messages from I to A_I, (2) **type A-A**: messages from an agent to other agent(s), (3) **type A-r**: messages from A_r to r, (4) **type r-A**: messages from r to A_r, and (5) **type A-I**: messages from A_I to I.

Figure 1 depicts the case where the initiator and the receiver communicates with each other directly.

2.1.3 Databases

In our model, an agent's state and knowledge are expressed in its own databases. An agent decides where to send a message, depending on its state and the received message.

The databases are classified into the following types, based on the data stored in them: (1) `Key`: encryption keys and decryption keys, (2) `SDB`: all messages the agent sent, and (3) `RDB`: all messages the agent received. In our model, we do not distinguish keys in public key cryptosystems from keys in symmetric key cryptosystems.

2.2 Anonymous communication function

Message relay agents actualize an anonymous communication protocol in cooperation. This section defines some functions to represent such agent's behavior.

2.2.1 Message routing function and message generating function

In this section we introduce two auxiliary functions $MsgRoute^{type1}_{type2}(m, \mathsf{Key}, \mathsf{SDB}, \mathsf{RDB})$ and $MsgGen^{type1}_{type2}(m, \mathsf{Key}, \mathsf{SDB}, \mathsf{RDB})$ in order to define anonymous communication function $AnonComm(m)$ later.

$MsgRoute^{type1}_{type2}(m, \mathsf{Key}, \mathsf{SDB}, \mathsf{RDB})$ determines the path a message generated by the agent follows. This

function returns true or false. $MsgGen^{type1}_{type2}(m, \mathsf{Key}, \mathsf{SDB}, \mathsf{RDB})$ generates the messages which are sent to the next addresses. In both cases $type1$ and $type2$ show the types of a received message and a newly generated message, respectively.

The input parameters of both functions are message m the agent received and the agent's state (Key, SDB, and RDB). They are used to determine the message route and to generate new messages.

2.2.2 Anonymous communication function *AnonComm*(*m*)

Using the definitions above, now we are ready to define $AnonComm(m)$. In our model $AnonComm(m)$ expresses the behavior of an agent when it receives a message.

SDB: $\mathtt{SDB}_{\mathrm{A-A}} \cup \mathtt{SDB}_{\mathrm{A-r}} \cup \mathtt{SDB}_{\mathrm{A-I}} \cup \mathtt{SDB}_{\{\mathrm{A-A,\,A-r}\}} \cup \mathtt{SDB}_{\{\mathrm{A-A,\,A-I}\}}$
RDB: $\mathtt{RDB}_{\mathrm{I-A}} \cup \mathtt{RDB}_{\mathrm{A-A}} \cup \mathtt{RDB}_{\mathrm{r-A}}$
m: a message the agent receives.
type(m): returns the type of the received message.
$\mathcal{M}$: the set of the output messages.

```
1  function AnonComm(m)
2    begin
3      x ← type(m);  M ← ∅;  RDB_x ← RDB_x + {m};
4      if (x = "I − A") then
5        begin
6          M ← MsgGen^{I−A}_{A−A}(m, Key, SDB, RDB); SDB_{A−A} ←
   SDB_{A−A} + {M};
7        end
8      else if (x = "r − A") then
9        begin
10         M ← MsgGen^{r−A}_{A−A}(m, Key, SDB, RDB); SDB_{A−A} ←
   SDB_{A−A} + {M};
11       end
12     else if (x = "A − A") then
13       begin
14         if MsgRoute^{A−A}_{{A−A, A−r}}(m, Key, SDB, RDB) then
15           begin
16             M ← MsgGen^{A−A}_{{A−A, A−r}}(m, Key, SDB, RDB);
   SDB_{{A−A, A−r}} ← SDB_{{A−A, A−r}} + {M};
17           end
18         else if MsgRoute^{A−A}_{{A−A, A−I}}(m, Key, SDB, RDB) then
19           begin
20             M ← MsgGen^{A−A}_{{A−A, A−I}}(m, Key, SDB, RDB);
   SDB_{{A−A, A−I}} ← SDB_{{A−A, A−I}} + {M};
21           end
22         else if MsgRoute^{A−A}_{A−A}(m, Key, SDB, RDB) then
23           begin
24             M ← MsgGen^{A−A}_{A−A}(m, Key, SDB, RDB); SDB_{A−A} ←
   SDB_{A−A} + {M};
25           end
26         else if MsgRoute^{A−A}_{A−r}(m, Key, SDB, RDB) then
27           begin
28             M ← MsgGen^{A−A}_{A−r}(m, Key, SDB, RDB); SDB_{A−r} ←
   SDB_{A−r} + {M};
29           end
30         else if MsgRoute^{A−A}_{A−I}(m, Key, SDB, RDB) then
31           begin
32             M ← MsgGen^{A−A}_{A−I}(m, Key, SDB, RDB); SDB_{A−I} ←
   SDB_{A−I} + {M};
33           end
34         endif /* line 14, 18, 22, 26, 30 */
35       end /* line 13 */
36     endif /* line 4, 8, 12 */
37     return (M)
38   end
```

Note that since $AnonComm$ is an agent's function, the types of messages it receives must be one of A-A, I-A, or r-A. Therefore $type(m)$ returns only such three types. When the agent receives a message m, first m is stored in $\mathtt{RDB}_x$ (Note that x is the message type of m). Next the execution of $AnonComm$ branches to the block corresponding to x. For example, if the agent receives m from another agent, then the message type x is A-A and the control flow jumps to line 12. Now *MsgRoute* function determines the destination(s) of a newly generated message(s). Note that when A_I and A_r receive the message typed I-A and r-A, respectively, they are assumed to send messages to other agents on our model and in those cases *MsgRoute* are unnecessary (see lines 5–7 and lines 9–11). Next, messages $\mathcal{M}$ are generated by *MsgGen* and according to the type(s) of $\mathcal{M}$, say y, they are stored in $\mathtt{SDB}_y$. To see how *MsgGen* works, let us further assume that the newly generated messages should be sent to the other agent(s) and the initiator, for the example above. Then *MsgRoute* on line 18 returns true and *MsgGen* (line 20) generates new messages, which are stored in $\mathtt{SDB}_{\{\mathrm{A-A,\,A-I}\}}$. Finally, on line 37, *AnonComm* returns the generated messages $\mathcal{M}$, which are supposed to be sent some time later.

2.3 Primitive functions

In this section we define *primitive functions*, which are basic building blocks that compose anonymous communication protocols on our model. From our experience of modeling anonymous communication, we believe that the primitive functions in this section are expressive and comprehensive enough to describe anonymous protocols proposed so far [6].

The primitive functions are classified into the following four types:

1. Cryptographic function[1]

$Encrypt(Key, M)$: encrypts M with *Key*, and outputs the ciphertext.

$Decrypt(Key, C)$: If C was encrypted with a key which corresponds to *Key*, outputs the plaintext. If not, outputs invalid data.

2. Identifier function

$CIDGen()$: outputs a communication identifier. (e.g., a random number, a port number, etc.)

[1] Although cryptographic functions are not used in the protocol description in Section 3, they are included here for completeness. However, they are actually necessary for the description of some sort of anonymous communication, say MIX. See [6] for the description of MIX on our model.

$LocalAddr()$: outputs the address of the machine on which the agent is running.

3. List function

$Shuffle(t, V)$: divides V into t parts of the same length, shuffles and outputs them.

$Separate(t, \mathtt{S})$: divides S into t lists and outputs them.

$Pickup(t, \mathtt{S})$: picks t elements from list S at random and outputs them.

$Size(\mathtt{S})$: outputs size s of S.

4. Probabilistic function

$Trial(P)$ outputs 1 (with probability P) or 0 (with probability $1 - P$).

In the definitions above, the data type of variables in boldface is a list, otherwise scalar. It should be noted that clear characterization of anonymous communication protocols by primitive functions is realized for the first time on our model.

Using this set of primitive functions, the message routing functions and the message generating function can be represented simply. Moreover they are useful to develop a new anonymous communication protocol. This will be exemplified in Section 4.

In addition the primitive functions are not necessarily limited to the functions above. Arbitrary primitive functions can be added if necessary. Thus we can say that our model is highly extensible.

3 Protocol descriptions

In this section, we describe Proxy and Crowds as the examples of the protocol description on our model. However, since *MsgRoute* functions in those methods are rather simple and easy to describe, we omit the descriptions of them due to lack of space. Furthermore, reply procedures from the responder to the initiator are also not described, but it is almost straightforward to do so.

In the rest of this section, we pay much attention to the descriptions of $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}$ and $MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}$. The reasons for this are as follows: (1) A_I receives a message of initiator I and initiates anonymous communication by using $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}$. Therefore generally speaking, $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}$ plays the most important role in various anonymous communication protocols. (2) $MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}$ describes agent-to-agent communication, which in general occupies the largest part of anonymous communications.

3.1 Proxy

There is only one message relay agent (named Proxy) in Proxy method. Proxy method is a simple modification of ordinary message communication.

The *MsgGen* is following:

function $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}((I,\ A_I,\ I,\ r,\ V),\ (\mathtt{Key},\ \mathtt{SDB},\ \mathtt{RDB}))$
  begin
    $S \leftarrow A_I$; $R \leftarrow \mathrm{Proxy}$; $PI \leftarrow I$; $Pr \leftarrow r$; $PV \leftarrow V$;
    return $\{(S,\ R,\ PI,\ Pr,\ PV)\}$
  end

function $MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}((A_I,\ \mathrm{Proxy},\ I,\ r,\ V),\ (\mathtt{Key},\ \mathtt{SDB},\ \mathtt{RDB}))$
  begin
    $S \leftarrow \mathrm{Proxy}$; $R \leftarrow A_r$; $cid \leftarrow CIDGen()$;
    $PI \leftarrow cid$; $Pr \leftarrow r$; $PV \leftarrow V$;
    return $\{(S,\ R,\ PI,\ Pr,\ PV)\}$
  end

3.2 Crowds

In the following description of Crowds, $jondo_x$ are the message relay agents. Note that the probability p_f of forwarding messages is fixed in advance in Crowds system. Moreover Group is a list of members in the anonymous communication group (called *crowd*).

The *MsgGen* is following:

function $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}((I,\ A_I,\ I,\ r,\ V),\ (\mathtt{Key},\ \mathtt{SDB},\ \mathtt{RDB}))$
  begin
    $S \leftarrow A_I$; $R \leftarrow Pickup(1,\ \mathtt{Group})$; $cid \leftarrow CIDGen()$;
    $PI \leftarrow cid$; $Pr \leftarrow r$; $PV \leftarrow V$;
    return $\{(S,\ R,\ PI,\ Pr,\ PV)\}$
  end

function $MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}((jondo_{x_{k-1}},\ jondo_{x_k},\ cid,\ r,\ V),$
  $(\mathtt{Key},\ \mathtt{SDB},\ \mathtt{RDB}))$
  begin
    $S \leftarrow LocalAddr()$;
    if $(Trial(p_f) = 1)$ **then** $R \leftarrow Pickup(1,\ \mathtt{Group})$;
    else if $R \leftarrow A_r$; **endif**
    $PI \leftarrow cid$; $Pr \leftarrow r$; $PV \leftarrow V$;
    return $\{(S,\ R,\ PI,\ Pr,\ PV)\}$
  end

3.3 Discussion on the descriptions

We summarize the descriptions of Proxy and Crowds together in Table 1. Moreover we summarize IM in Table 2 without describing it in this paper. The interested reader is referred to [6] for its description.

As we can readily see from Table 1, Proxy and Crowds share certain similarities as follows; (1) they provide anonymity by interposing one or more agents between the initiator and the responder, and (2) they make it possible for the responder to reply to the initiator with the help of communication identifier *cid* while keeping the identity of the initiator secret.

With respect to Crowds (Table 1) and IM (Table 2), it is not very hard to find similarities between them. The big difference between Crowds and IM is that the former determines the destination of a message according to the output of *Trial* function, on the other hand, the latter does so according to the number of fragments of V.

As discussed above, our model gives us a clue to evaluate and classify anonymous communication protocols. This is mainly due to the way of our modeling, especially to the introduction of primitive functions.

		Proxy	Crowds	
$MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}$	S	A_I	A_I	
	R	Proxy	$Pickup(1,\ \texttt{Group})$	
	PI	I	cid	
	Pr	r	r	
	PV	V	V	
$MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}$			$Trial(p_f) = 1$	$Trial(p_f) \neq 1\ (= 0)$
	S	Proxy	$LocalAddr()$	$LocalAddr()$
	R	A_r	$Pickup(1,\ \texttt{Group})$	A_r
	PI	cid	cid	cid
	Pr	r	r	r
	PV	V	V	V

Table 1. Message summary of Proxy and Crowds.

		IM		
$MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}$	S	$(S_1,\ S_2) \leftarrow (A_I,\ A_I)$		
	R	$(R_1,\ R_2) \leftarrow Pickup(2,\ \texttt{Group})$		
	PI	$(PI_1,\ PI_2) \leftarrow ((gid,\ cid),\ (gid,\ cid))$		
	Pr	$(Pr_1,\ Pr_2) \leftarrow (r,\ r)$		
	PV	$(PV_1,\ PV_2) \leftarrow Separate(2,\ Shuffle(n,\ V))$		
$MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}$		$Size(\texttt{V}) = 1$	$Size(\texttt{V}) = 2$	$Size(\texttt{V}) \geq 3$
	S	$LocalAddr()$	$(S_1,\ S_2) \leftarrow (LocalAddr(),\ LocalAddr())$	$(S_1,\ S_2,\ S_3) \leftarrow (LocalAddr(),\ LocalAddr(),\ LocalAddr())$
	R	A_r	$R_1 \leftarrow A_r,\ R_2 \leftarrow Pickup(1,\ \texttt{Group})$	$R_1 \leftarrow A_r,\ (R_2,\ R_3) \leftarrow Pickup(2,\ \texttt{Group})$
	PI	$(gid,\ cid)$	$(PI_1,\ PI_2) \leftarrow ((gid,\ cid),\ (gid,\ cid))$	$(PI_1,\ PI_2,\ PI_3) \leftarrow ((gid,\ cid),\ (gid,\ cid),\ (gid,\ cid))$
	Pr	r	$(Pr_1,\ Pr_2) \leftarrow (r,\ r)$	$(Pr_1,\ Pr_2,\ Pr_3) \leftarrow (r,\ r,\ r)$
	PV	$Pickup(1,\ \texttt{V})$	$PV_1 \leftarrow Pickup(1,\ \texttt{V}),\ PV_2 \leftarrow \texttt{V} - \{PV_1\}$	$PV_1 \leftarrow Pickup(1,\ \texttt{V}),\ (PV_2,\ PV_3) \leftarrow Separate(2,\ \texttt{V} - \{PV_1\})$

Table 2. Message summary of IM

4 Integration of Crowds and IM

In order to demonstrate that our model has enough power and flexibility to express anonymous communication protocols, we shall integrate Crowds and IM in this section.

Although Crowds is simple and efficient, it needs enough storage on relay nodes in order to keep information necessary for delivering reply message to the initiator. In IM, meanwhile, relay nodes are not necessarily to keep information for replying because the replying is multi-casted to all members in the group. However since the message content is fragmented in many pieces, each of which must be sent separately, it is possible that heavy load for processing such operation is imposed on intermediate relay nodes.

Hence we integrate Crowds and IM to take advantages of both methods for various network conditions. The integrated method (called "Crowds-IM") can be fitted to practical network environments by only changing parameters accordingly.

In Crowds-IM, we use two system parameters which are the probability of p_f (in Crowds) and the number of division of the message n (in IM). These parameters influence the distance from the initiator to the responder sensitively.

For the integration, we incorporate p_f and *Trial* function (one of the primitive functions) into IM. Note that `Group` means the list of members (expressed as "M") in the anonymous communication group (identified by gid).

function $MsgGen^{\mathrm{I-A}}_{\mathrm{A-A}}((I,\ A_I,\ I,\ r,\ V),\ (\texttt{Key},\ \texttt{SDB},\ \texttt{RDB}))$
begin
$S_1 \leftarrow A_I$; $S_2 \leftarrow A_I$;/* $(A_I = \mathrm{M}_{x_0})$ */
$(R_1,\ R_2) \leftarrow Pickup(2,\ \texttt{Group})$;
$cid \leftarrow CIDGen()$; $PI_1 \leftarrow (gid,\ cid)$; $PI_2 \leftarrow (gid,\ cid)$;
$Pr_1 \leftarrow r$; $Pr_2 \leftarrow r$;
$(PV_1,\ PV_2) \leftarrow Separate(2,\ Shuffle(n,\ V))$;
return $\{(S_1,\ R_1,\ PI_1,\ Pr_1,\ PV_1),$
$(S_2,\ R_2,\ PI_2,\ Pr_2,\ PV_2)\}$
end

function $MsgGen^{\mathrm{A-A}}_{\mathrm{A-A}}((\mathrm{M}_{x_{k-1}},\ \mathrm{M}_{x_k},\ (gid,\ cid),\ r,\ \texttt{V}),$
$(\texttt{Key},\ \texttt{SDB},\ \texttt{RDB}))$
begin
if $(Trial(p_f) = 1)$ **then**
if $(Size(\texttt{V}) \geq 2)$ **then**
begin
$S_1 \leftarrow LocalAddr()$; $S_2 \leftarrow LocalAddr()$;
$(R_1,\ R_2) \leftarrow (Pickup(2,\ \texttt{Group}))$;
$PI_1 \leftarrow (gid,\ cid)$; $PI_2 \leftarrow (gid,\ cid)$;
$Pr_1 \leftarrow r$; $Pr_2 \leftarrow r$;

```
        (PV1, PV2) ← Separate(2, V);
        return {(S1, R1, PI1, Pr1, PV1),
          (S2, R2, PI2, Pr2, PV2)}
      end
    else if (Size(V) = 1) then
      begin
        S ← LocalAddr(); R ← Pickup(1, Group);
        PI ← (gid, cid); Pr ← r; PV ← Pickup(1, V);
        return {(S, R, PI, Pr, PV)}
      end
    endif
  else
    if (Size(V) ≥ 3) then
      begin
        S1 ← LocalAddr(); S2 ← LocalAddr();
        S3 ← LocalAddr();
        (R1, (R2, R3)) ← (Ar, (Pickup(2, Group)));
        PI1 ← (gid, cid); PI2 ← (gid, cid);
        PI3 ← (gid, cid); Pr1 ← r; Pr2 ← r; Pr3 ← r;
        PV1 ← Pickup(1, V);
        (PV2, PV3) ← Separate(2, V − {PV1});
        return {(S1, R1, PI1, Pr1, PV1),
          (S2, R2, PI2, Pr2, PV2),
          (S3, R3, PI3, Pr3, PV3)}
      end
    else if (Size(V) = 2) then
      begin
        S1 ← LocalAddr(); S2 ← LocalAddr();
        (R1, R2) ← (Ar, Pickup(1, Group));
        PI1 ← (gid, cid); PI2 ← (gid, cid); Pr ← r;
        PV1 ← Pickup(1, V); PV2 ← V − {PV1};
        return {(S1, R1, PI1, Pr1, PV1),
          (S2, R2, PI2, Pr2, PV2)}
      end
    else if (Size(V) = 1) then
      begin
        S ← LocalAddr(); R ← Ar; PI ← (gid, cid);
        Pr ← r; PV ← Pickup(1, V);
        return {(S, R, PI, Pr, PV)}
      end
    endif
  endif
end
```

In Crowds-IM, anonymity of the initiator for the responder depends on the size of Group. In addition, if we assume $n = 1$, *Size* always returns 1 and the performance is equal to Crowds. On the other hand, if we assume $p_f = 0$, *Trial* always rerutns 0 and the performance is equal to IM. Moreover, since Crowds and IM can be considered as an enhanced method of Proxy, if we assume $n = 1$ and p_f, the performance is equal to Proxy. Thus Crowds-IM can cope with different network situations and environments.

5 Conclusion and Discussion

In this paper we have proposed an anonymous communication model. We have introduced agents that work in cooperation to provide anonymity. This modeling is based on the observation that most of practical anonymous protocols have much commonality in that they have several relay nodes en route from the sender to the receiver. More importantly such agents' behaviors can be expressed in a small set of primitive functions. Using primitive functions the essential structure of anonymous communication protocols can be made clear. Another advantage of our modeling is that implementation and evaluation of a wide variety of anonymous communication protocols can be done in one framework. Finally, Crowds and IM were integrated in this paper to demonstrate the expressive power of our model.

Now we are trying to examine the degree of anonymity achieved in previous anonymous protocols described on our model. Unfortunately, it is very difficult to precisely evaluate it. Therefore we are considering to estimate the anonymity by investigating how robust each protocol is against the collusion attack [3, 7, 9] because it is known that the attack can pose a serious threat to the anonymity provided by anonymous communication protocols.

References

[1] CCITT(ITU). Specification and Description Language (SDL). Recommendations Z.100, 1988.

[2] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, Feb. 1981.

[3] D. Inoue and T. Matsumoto. Anonymity Achieved by Group Communication. In *JW-ISC 2000*, pages 199–206, Jan 2000.

[4] ISO. Estelle: A Formal Description Technique Based on Extended Stated Transition Model. ISO9074, 1989.

[5] ISO. Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behavior. ISO8807, 1989.

[6] S. Kitazawa, M. Soshi, and A. Miyaji. Modeling anonymous communication protocols with message relaying. In *Symposium on Cryptography and Information Security (SCIS 2001)*, pages 321–326 (6C–1), Jan. 2001.

[7] S. H. Low and N. F. Maxemchuk. A collusion problem and its solution. *Information and Computation*, 140(2):158–182, 1998.

[8] A. Luotonen and K. Altis. World-Wide Web Proxies. *Computer Networks and ISDN Systems*, 27(2):147–154, 1994.

[9] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Trans. Info. Syst. Security*, 1(1):66–92, Apr. 1998.

[10] P. F. Syberson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, 1997.