A Conceptual Framework for Specification, Analysis, and Design of Anonymity Services

Marzieh Ispareh
Department of Computer Engineering
University of Isfahan, Iran
ispareh@eng.ui.ac.ir

Behrouz Tork Ladani
Department of Computer Engineering
University of Isfahan, Iran
ladani@eng.ui.ac.ir

ABSTRACT

Anonymity is an important issue in information security, which its main goal is to protect entities privacy in the systems. Different methods and protocols (with different types of anonymity services) have been developed so far to provide special anonymity requirements of applications. Each of these systems has been developed with different ad hoc approaches. In this paper we present a conceptual framework that makes specification, analysis and design of anonymity applications more systematic. To do this, first we go toward presenting a conceptual model of anonymity which can be used in clear description of different aspects of anonymity. Then we extract a list of anonymity primitives from the existing anonymity providing methods. These primitives are base functions which can be composed to form anonymity services to provide specified anonymity requirements of the system.

Keywords

Security, Anonymity, Anonymity Services, Anonymity Requirements, Conceptual Model.

1. INTRODUCTION

Anonymity is an important issue in information security context which ignoring it in some applications leads to entities privacy violation. Some applications that require anonymity can be categorized as follows [1, 2]:

- Anonymous search of information
- Protecting communication patterns of entities to prevent traffic analysis.
- E-voting
- Providing freedom of speech in fanatic environments
- Anonymous use of location based services
- E-payment
- Sending and receiving messages anonymously

It can be deduced from the above list that different applications require different types of anonymity. For example the anonymity requirements of an e-voting system are different from the requirements of an anonymous search application. Different methods and protocols have been developed so far to provide special anonymity requirements of applications [3, 4, 5, 6, 7, 8], but each of these systems has been developed with different and ad hoc approaches. The goal of this paper is to present a conceptual framework that makes specification, analysis and design of anonymity applications more systematic.

Like any software system, the first step in anonymity system development is analysis and specification of anonymity requirements. To do this step, we need a descriptive model of anonymity concepts but there is no complete framework for comprehensible description and classification of concepts and requirements of anonymity yet. So first we go toward presenting a conceptual model of anonymity which can be used in clear description of different aspects of anonymity.

After requirement analysis and specification, the next step is to define anonymity services to provide these requirements. Most of the existing anonymity methods and protocols use a combination of some reusable primitive anonymity techniques. We derivate and present these anonymity primitives by investigating the existing anonymity methods. These primitives are base functions for providing anonymity and can be composed to form anonymity services to provide specified anonymity requirements of the applications.

We first study and classify architectures used in different anonymity methods and present a layered architecture for providing anonymity in an application. Using this architecture, application layer implementation can be separated from anonymity provider layer.

As we studied there is no work yet to present a complete classification and comprehensive model of anonymity concepts.

In [9] anonymity is classified into two categories: Data anonymity and Connection anonymity. Data anonymity is about filtering any identifying information out of the data that is exchanged in a particular application. Connection anonymity, on the other hand, is about hiding the identities of source and destination during the actual data transfer.

Freedman in [10] presents an elementary taxonomy of anonymity services, but this categorization is just based on which entities become anonymous in communication process.

In [1], another taxonomy of anonymity properties is presented. Although this classification is more complete than [4], but it has not covered different roles of entities which are someway participant in anonymity process.

In [12] three scenarios are presented for anonymity in e-commerce:

- Only one communication party wants to be anonymous.
- Both of parties want to be anonymous.
- Two parties are known to each other but want to be anonymous to the others.

As can be seen, this taxonomy is only based on "What entity wants to be anonymous and toward which entities".

The paper is structured as follows: First a layered architecture for anonymity provider is presented in section 2. Then in section 3 we present a conceptual model of anonymity which can be used in clear description of different aspects of anonymity. We show the usage of model by analysis and specification of requirements of two sample anonymity applications and sketching the properties of two anonymity protocols. In section 4 we try to derive a list of anonymity primitives as complement to the presented model. The application of primitives as reusable modules of two anonymity method is also shown as case study in this section. Finally section 5 concludes the paper.

2. A LAYERED ARCHITECTURE FOR ANONYMITY SERVICES

Different protocols and mechanisms are developed so far to provide anonymity requirements. We can divide them into the following three categories based on layers they are applied:

1. Communication layer methods:

Some of the required anonymity properties are related to the communication and are application independent. So most of protocols in this category are general-purpose protocols and their goal is to provide anonymity in communication layer independent of application logic. Some examples of these methods are Mix Net [13], Crowd [14], Onion routing, [15], Dining Cryptography [16], and P5 [17].

2. Application layer methods:

These methods are normally special-purpose and provide special requirements of applications using top-level anonymity techniques. An important assumption in these methods is that we have anonymous communication in lower layers. So all attempt in this layer is to provide just special anonymity requirements of application. Applying these methods in a system without anonymous communication is useless. Examples of these methods are Joris Claessens's work [18] for E -Payment and Giuseppe Ateniese's work [19] for E-Prescriptions.

3. Composed methods

These methods need to modify anonymity protocols in communication layer to provide anonymity requirements of the application layer. In other words in these methods, general-purpose anonymous communication methods are customized to provide special anonymity requirements of an application in combination with anonymity requirements of communication layer. Theses methods are normally ad hoc methods which are inserted into the application code and are not reusable in similar cases. Examples of these methods are Net Cash [6] and Mix-based Electronic Payments [7].

We can consider the architecture of an anonymity provider as Fig. 1. Dashed area shows anonymity provider system and upper layer provides only application requirements without considering anonymity requirements.

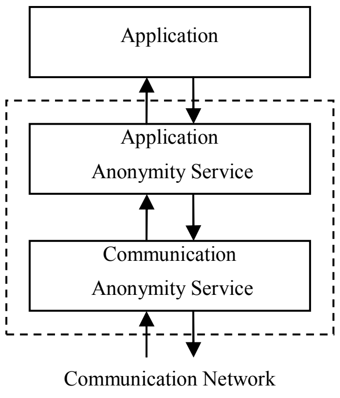
Application
Application Anonymity Service
Communication Anonymity Service
Communication Network

Figure 1. Anonymity provider architecture

3. A CONCEPTUAL MODEL OF ANONYMITY

In this section, we try to present a conceptual model of anonymity based on the existing anonymity requirements and services in different applications and protocols. To achieve this goal, first we introduce the key concept "identification information" which we use it to clearly define the anonymity concepts. Then we describe our model components and we summarized the model. Finally in last part of this section, we apply our model on four operational cases including two applications and two anonymity protocols.

3.1 Identification information

Anonymity is usually translated to "without name" or "nameless", however we think that it is more general than only "without name" conceptually. We introduce "*Identification Information*" or *Idinfo* in short that is more general than "name". After defining Idinfo and its instances, we can define anonymity concepts.

Definition 1 (Idinfo). Idinfo is data or information that can be used to indicate the real identity of an entity or her messages precisely. Idinfo may belong to one entity or a group of entities.

Note that in most cases the identity of messages of an entity is the same identity of the entity herself.

Based on the above definition, different instances of Idinfo in a system are Entity Idinfo and Message Idinfo.

3.1.1 Entity Idinfo

Entity Idinfo instances falls into the following categories:

- **Name Idinfo**: this type of Idinfo is a contractual pattern for identifying an entity in the system. An entity in an organization can have different name Idinfos. We classify name Idinfos into two types: *"Personal Name Idinfo"* and *"Organizational Name Idinfo."* The latter is in fact the role of entity in organization.
- **Operation Idinfo**: Sometimes, an entity can be truly identified based on her operations. Regarding this, operation Idinfos can be divided into two types: *"Operator Idinfo"* and *"Operation Coherency Idinfo"*. The former is such information that from it we can identify the operator uniquely, and the later is information that lets us identify a

source by finding out the relation between its separate operations.

- **Property Idinfo**: data or information about an entity which is not directly includes the entity's identification information, but joining them to other information or inferring from them, helps us to identify the entity itself. If this information is the entity's own property, we call it *"Inherent Property Idinfo"*, but if it is indirectly related to the entity, it is called *"Adventitious Property Idinfo"*. K-Anonymity concept is mainly about handling this kind of Idinfo [8].

Entity Idinfo categories are summarized in Fig. 2.

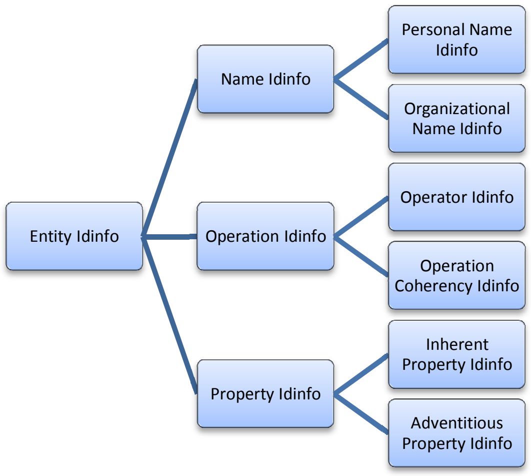

Figure 2. Entity Idinfo types

3.1.2 Message Idinfo

Message Idinfo instances falls into the following categories:

- **Channel Idinfo**: Sometimes messages of an entity are truly identified by studying the properties of the communication channel between entities. We call these properties channel Idinfos. Some of these Idinfos that can help us trace the communication are *message size*, *message content*, *delay between sending and receiving messages*, *order of messages*, *route of messages,* etc.
- **Connected Entities Idinfo**: This kind of Idinfo is in fact about the message sender and receiver entities which were studied in previous section. So they have entity Idinfo properties.

Message Idinfo categories are summarized in Fig. 3.

3.2 Conceptual model components

As mentioned earlier, our conceptual model of anonymity has three components including anonymity types, anonymity structure and anonymity constraints. In this section we explain them in details.

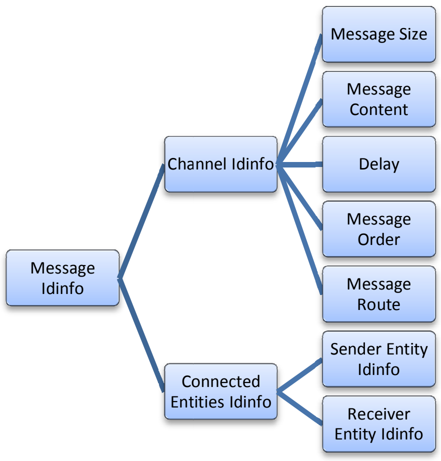

Figure 3. Message Idinfo types

3.2.1 Types of anonymity

We can make entities anonymous by preventing access to their identification information. So we define anonymity service based on the Idinfo concept previously defined as follows:

Definition 2 (Anonymity service). Each anonymity service is a set of activities which provide a valid combination of inaccessibility to some Idinfos instances from the viewpoint of other entities.

So we can have two classes of anonymity service: entity anonymity services, and message anonymity service.

- **Entity anonymity**: We divide this class of anonymity services into seven different anonymity types shown in table 1. Considering the table, when all of Idinfos of an entity are reachable, we have no anonymity. In contrast, when none of Idinfos of an entity is reachable, we have full anonymity.
- **Message anonymity**: We divide this class of anonymity services into seven types presented in table 2. Like entity anonymity, when all of the message Idinfos are reachable, we have no anonymity.

Table 1. Types of entity anonymity service

Anonymity service type	Name Idinfo	Property Idinfo	Operation Idinfo
Without anonymity	✓	✓	✓
EA1	✓	✓	×
EA2	✓	×	✓
EA3	✓	×	×
EA4	×	✓	✓
EA5	×	✓	×
EA6	×	×	✓
EA7 (Full anonymity)	×	×	×

✓: Access to Idinfo, ×: No access to Idinfo

Table 2. Types of message anonymity service

Anonymity service types	Connected entities Idinfos		Channel Idinfo
	Sender Entity Idinfo	Receiver Entity Idinfo	
Without anonymity	✓	✓	✓
MA1	✓	×	✓
MA2	×	✓	✓
MA3	×	×	✓
MA4	✓	✓	×
MA5	✓	×	×
MA6	×	✓	×
MA7 (Full anonymity)	×	×	×

✓: Access to Idinfo, ×: No access to Idinfo

Note that we can increase the granularity of Idinfos to make more valid and detailed combinations in both cases of entity and message anonymity.

3.2.2 Structure of anonymity

Each anonymity application needs a combination of one or more anonymity service types based on its requirements and most of applications need more than one anonymity service type.

As we told earlier in definition 1, an anonymity service is a set of activities that provide one kind of anonymity for an entity from the viewpoint of other entities.

Thus two other main factors should be considered in such a service:

- **Anonymous entity:** this entity has one of the following roles: *Sender*, *Receiver, or entity which its information is accessible.*
- **Anonymity observer entity:** the entity that an anonymous entity makes herself anonymous for him. This entity may have one of the following roles: *Sender, Receiver, Local observer, Global observer, Members of anonymity provider system, or Users who have access to entities information*.

Another noticeable issue is sender's abilities. When the sender of information needs to be anonymous, it can have the following abilities based on application requirements:

1- Authentication:

Though it is impossible to identify sender of messages, anonymous sender authentication is feasible.

2- Reply:

Though it is impossible to identify sender of messages, reply to anonymous sender is feasible.

3.2.3 Anonymity constraints

Each anonymity service can be applied absolutely or conditionally. In the case of Absolute anonymity, entity becomes anonymous without any condition. On the other hand, in Conditional anonymity, the entity becomes anonymous condition to satisfying some constraints. Three kinds of anonymity constraints could be defined:

- **Temporal anonymity constraints:** Anonymity is established or preserved based on some temporal conditions e.g. until when some special event is happen.
- **Spatial anonymity constraints:** Anonymity is established or preserved based on some spatial conditions e.g. an agent is anonymous else in hosts with a special (authenticated) ID or IP address.
- **Committed anonymity constraints:** Anonymity is established or preserved so long as entity is faithful to some special rules e.g. while she does not received a special secret token or is obeyed to a certain rule.

3.3 Summary of the model

Our conceptual model of anonymity has three components:

- **Types of anonymity**

  First step in specification of anonymity requirements in applications and protocols is to specify the required anonymity service types.

- **Structure of anonymity**

  Second step is to specify factors involved in each required service.

- **Anonymity constraints**

  In final step, we specify whether each anonymity service is conditional or not, and what are those constraints if there is any.

So we can specify the anonymity requirement or properties as a tuple <EA, MA> where EA is the set of Entity Anonymity services that each of its members is a tuple as follows:

<AE, AO, Atype, C Authenticable, Repliable>AE is the Anonymous Entity, AO is the Anonymity Observer entity, AType is type of this anonymity service, C is a set of constraints; if it is empty, then the anonymity service is applied absolutely, and Authenticable and Repliable are Boolean parameters that indicate whether entity anonymity has these abilities or not.

MA defines the set of Message Anonymity services that each of its members is a tuple as follows:

<SE, RE, AO, Atype, C >

In which SE and RE are communicated entities and other parameters are the same as EA.

3.4 Case study

In this section we use our model to study some real cases including two anonymity protocols and two anonymity applications. As we told earlier in section 3.2, we did not increase the granularity of Idinfos to make detailed combinations of anonymity types. So although in some cases we achieve same anonymity type for two anonymity services, if details are considered, these services may have different anonymity types.

3.4.1 *Mix net protocol*

A number of anonymity protocols like Webmixes [20], ISDN-Mixes [21], the Java Anon Proxy [22], Stop-and- Go-Mixes [23] and others have been based on David Chaum's anonymous email solution called mix net [24].

In this protocol, each mix has a public key which senders use to encrypt messages to that mix. The mix accumulates a batch of these encrypted messages, decrypts them, and delivers them to next receiver. Because a decrypted output message looks nothing like the original encrypted input message, and because the mix collects a batch of messages and then sends out the decrypted messages in a rearranged order, an observer cannot learn which incoming message corresponds to which outgoing message.

This protocol uses methods like batch sending, dummy message, adding random delay and so on to protect from traffic analyzing.

It provides the following kinds of anonymities:

- Anonymity of communication between sender and receiver from:

  - Global observer
  - Local observer

These two kinds of anonymities are of type MA7. In addition, these anonymities are applied absolutely.

- Anonymity of sender from receiver:

If content of message has no sender identity, this anonymity is of type EA7, otherwise anonymity is of type EA5. This anonymity is applied absolutely too.

Regarding this analysis, we can specify the anonymity of this protocol as tuple <EA, MA> in which EA is as follows:

EA = {<Sender, Receiver, EA5, null, False, False >}

MA is also as follows:

MA = {< Sender, Receiver, global observer, MA7, null >,

< Sender, Receiver, local observer, MA7, null >}

There are some versions of this protocol which in the anonymous seder is repliable. So in these versions EA is as follows:

EA = {<Sender, Receiver, EA5, null, False, True >}

3.4.2 *Onion routing protocol*

In onion routing protocol presented by Reed ‘Syverson and Goldschlag [9], sender and receiver know each other. Goal of this protocol is protecting relation of sender and receiver from others and preventing from traffic analyzing. In other words it makes a private channel in a public network.

This protocol uses layered cryptography as mix net but unlike mix net, the route from sender to receiver is established in the beginning of communication. Then this route is sent as a layered encrypted message to routers so that each router saves the address of next and previous router. After establishing the route, message is sent layered and encrypted to routers. So these messages are untraceable.

We specify anonymity of this protocol as tuple <EA, MA> in which MA is as follows:

MA = {< Sender, Receiver, global observer, MA7, null >,

< Sender, Receiver, local observer, MA7, null >,

< Sender, Receiver, helper routers, MA7, null >}

Assuming lack of any identification information in messages, EA is specified as follows:

EA = {<Sender, Receiver, EA6, null , False, False >}

3.4.3 *Email Service*

Suppose we need an email service which is used to present medical consolation to patients by preserving their privacy. Requirements of this service are as follows:

- Patient must be anonymous from every one
- Anonymity of patient must be continuous.
- Messages of patient must be untraceable.
- Because of important role of patient’s medical background in consolation, it must be possible to understand the relation between messages from the same patient.
- Doctors can reply to anonymous patients.

Based on the above requirements, we specify this service as tuple <EA, CA> that MA is an empty set and EA is as follows:

EA = {<Patient, Doctor, EA4, null, False, True >,

<Patient, global observer, EA7, null, False, False >,

<Patient, local observer, EA7, null, False, False >}

3.4.4 E-Payment Service

Suppose we need a payment service such that buyers want to be anonymous. Much as we use cryptographic protocols, behavioral pattern of buyer, her/his preferences, and so on remains clear.

Unlike email service, in this service, anonymity can not be absolute to prevent from fraud. So if buyers disobey from payment rules, he/she must be traceable. Also in this service buyer can be authenticated.

Based on above requirements, we illustrate this service as tuple <EA, CA> that MA is an empty set and EA is as follows:

EA = {<Buyer, Seller, EA7, {Paid on time}, True, True>,

< Buyer, global observer, EA7, null >,

< Buyer, local observer, EA7, null >}

4. ANONYMITY PRIMITIVES

Anonymity services have similar functionalities especially in communication layer. By extracting these reusable functionalities, we achieve anonymity primitives.

Some advantage of using anonymity primitives are as follows:

- They are reusable in different applications.
- They can be composed to achieve more anonymity functionalities.
- Using them, we can separate application layer development from anonymity provider layer development.

We divide anonymity primitives to the following classes based on layers they are going to be applied:

- anonymity primitives in communication layer
- anonymity primitives in application layer
- anonymity primitives usable in both layer

Anonymity primitives in application layer are generally complex. They are complete solutions which can provide the required anonymity in application without composing with other anonymity techniques.

Anonymity primitives in communication layer are generally simple and must be composed with each other to make a complete solution to provide the required anonymity.

In next section we describe various techniques of three mentioned classes. Then in last part of this section, we analyze two anonymity protocols based on these techniques.

4.1 Communication layer primitives

1- Padding

Members of anonymizer in route of sender to receiver prevent from traffic analysis based on size of messages by modifying messages size.

2- Dummy messages

Members of anonymizer prevent from tracing real message between entities by inserting dummy messages to their outputs.

3- Reordering messages

Members of anonymizer prevent from traffic analysis based on order of messages by reordering messages before sending them to their outputs.

4- Batching

Members of anonymizer cause different delay in sending messages by batch sending of messages. So they prevent from traffic analysis based on order and delay of messages.

5- Adding random delay

Members of anonymizer wait some moment randomly before sending input messages to their output. So they prevents from traffic analysis based on order and delay of messages.

6- Dining Cryptography

Every entity in network shares a 1-bit key with next neighbor entity. To send a message, all entities send sum of bits they share. If some one wants to transfer real message, he/she sends reverse of sum. If no one send real message, result is zero else there is a message from an anonymous sender. To send message greater that 1-bit, each pair share chain of bits.

Using this technique, network entities can send their massages anonymously.

7- Broadcasting

To send a message to anonymous sender, message can be send to all member of the group. Structure of message must be in such a format which only real receiver can read it.

8- caching

Members of anonymizer system cache replies of receivers to requests. Every time a member receives a request similar to its cached requests, send corresponding response without forwarding request to its receiver. So tracing messages becomes harder.

9- Filtering

Members of anonymizer filter messages that include identification or other properties of anonymous entity. So this technique prevents from flaw of identity information.

10- Multiplexing

Members of anonymizer system send several messages as one message so traffic analysis becomes harder.

4.2 Common primitives

1- Encryption

By encrypting/decrypting input messages before sending them to output, correspondences between input and output messages in members of anonymizer is covered.

2- Compressing

By compressing/decompressing input messages before sending them to output, correspondences between input and output messages in members of anonymizer is covered.

3- Impersonation

By replacing real identity of entities with unreal identity, we can conserve identity information of them.

4- Pseudonymous

Using pseudonyms as entities identity, we can conserve identity information of them.

5- Secret sharing[25]

The goal of this technique is to prove authorization of an entity for using a service without flaw of its identity information. In this technique an authorized entity (such as system manager) use share key schema and divides secret authentication message to *N* parts. Server has *t-1* parts and other parts are divided between entities who want get service from server. Threshold for secret message retrieval is t that means when an entity sends a request to server, if it has part *t* of secret, entity is authorized. Yet it remains anonymous because server doesn't know which part of *n-t-1* parts has been received.

4.3 Application layer primitives

1- Generalization

In this technique values of some properties in database of entities information are replaced with more general values in such a way that statistical information remains valid. For example by replacing zip code with street in the database, achieving entities identity from their information becomes harder.

2- Tuple suppression

K-anonymity method is a combination of Generalization technique and Tuple suppression technique. In Tuple suppression technique some of records in database are deleted to decrease generalization degree needed to anonymize entities.

3- Blind signature[26]

Through this technique, an entity can get sign of other entity on its message without flaw of any information about content of its message. This technique can be used in applications which we need authorization in addition to anonymity.

4- Fair blind signature

In blind signature technique there is no relation between original readable message and corresponding unreadable message. Applications that need controllable and conditional anonymity must have a way to link between two messages whenever they need. Fair blind signature technique provides this ability through trusted third party.

5- Partially blind signature[27]

Using this technique, an entity can get sign of other entity on its message without flaw of information about content of message. Difference between this technique and blind signature technique is that in this technique some content of message is readable for signer.

6- Group signature

In this technique, the signer of message remains anonymous. This technique helps members of a group to sign a message anonymously. Sign verifier can not indicate what member of group signed the message.

7- Zero knowledge proof[28]

This technique helps an entity to prove to another entity that it is aware of a secret without flaw of information about that secret. It can be used to provide authorization in addition to anonymity.

4.4 Case studies

In this section we decompose two anonymity protocols into their anonymity primitives which were presented in previous sections. Components of each protocol presented as Flowchart. In these Flowcharts, gray boxes are anonymity primitive blocks.

4.4.1 Mix net protocol

An abstract of operation of this protocol was presented in previous sections. Building blocks of this protocol are as Fig. 4.

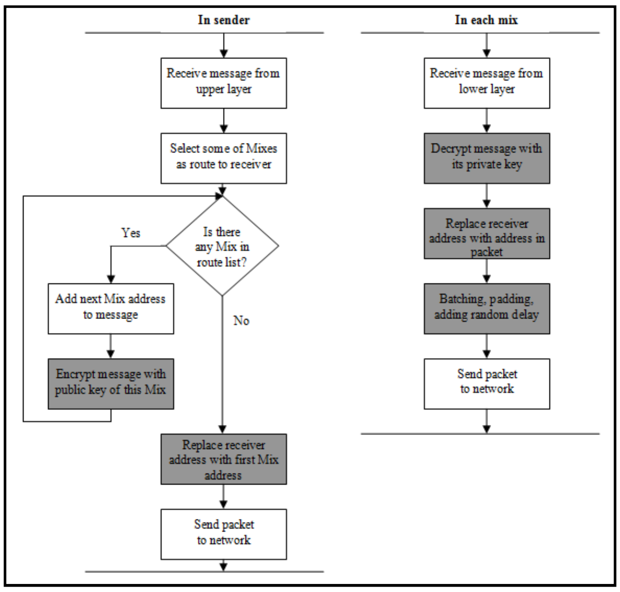
In sender
Receive message from upper layer
Select some of Mixes as route to receiver
Is there any Mix in route list?
Yes
No
Add next Mix address to message
Encrypt message with public key of this Mix
Replace receiver address with first Mix address
Send packet to network
In each mix
Receive message from lower layer
Decrypt message with its private key
Replace receiver address with address in packet
Batching, padding, adding random delay
Send packet to network

Figure 4. Mix net protocol flowchart

4.4.2 Crowd protocol

In crowd [14], presented by Reiter and Rubin, each user must be member of Crowd. Requests of users to a web server pass from random number of Crowd members to server.

Each member which wants to send message to web server, select a member randomly, encrypt message by their share key and send message to that member. After receiving a message, anonymizer member decrypts it, select another member or web server randomly, encrypt message by their share key and send message to selected member.

Flowchart of this protocol is as Fig. 5.

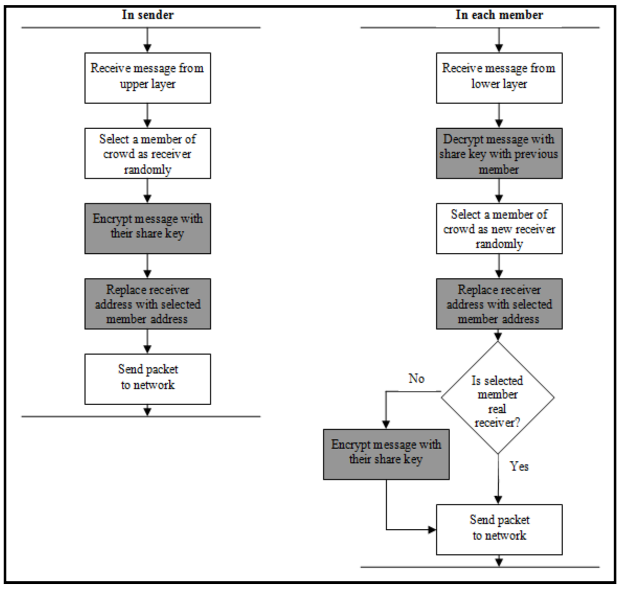
In sender
Receive message from upper layer
Select a member of crowd as receiver randomly
Encrypt message with their share key
Replace receiver address with selected member address
Send packet to network
In each member
Receive message from lower layer
Decrypt message with share key with previous member
Select a member of crowd as new receiver randomly
Replace receiver address with selected member address
Is selected member real receiver?
No
Yes
Encrypt message with their share key
Send packet to network

Figure 5. Crowd protocol flowchart

5. CONCLUSION

Although different methods and protocols have been developed so far to provide anonymity, each of them has been developed by different ad hoc approaches. In this paper we presented a conceptual framework that makes designing and developing anonymity applications more systematic. To do this, we first presented a conceptual model of anonymity which can be used in clear description of anonymity requirements of applications. Then we extracted anonymity primitives from existing anonymity providing methods. These primitives can be composed to form anonymity services which provide specified anonymity requirements of the system. We are trying to bind the conceptual model to anonymity primitives. This way, using these primitives in addition to the conceptual model and the layered anonymity provider architecture, we can establish an anonymity application development methodology which is our next goal.

6. REFERENCES

[1] Qing Zhang, A fair and anonymous schema , PhD thesis , university of London, 2007.

[2] Dahlia Malkhi , Anonymity-Advanced Course in Computer and Network Security, The Hebrew University, Jerusalem, 2002

[3] Goldreich, O.,Micali, S. , Wigderson, A , Proofs that yield nothing but their validity , ACM,1991.

[4] S´ebastien Canard , Aline Gouget , Divisible E-Cash Systems Can Be Truly Anonymous , International Association for Cryptology Research 2007

[5] Joris Claessens , Bart Preneel†, Joos Vandewalle , ANONYMITY CONTROLLED ELECTRONIC PAYMENT SYSTEMS , 20th Symposium on Information Theory IEEE, 1999.

[6] G. Medvinsky, B. C. Neuman. NetCash: A design for practical electronic currency on the Internet. In ACM-CCS'93, 1993.

[7] M. Jakobsson, D. M'Ra¨ıhi, Mix-based Electronic Payments. Fifth Annual Workshop on Selected Areas in Cryptography (SAC'98), Queen's University, Kingston, Ontario,Canada, August 1998.

[8] Kristen LeFevre, David J. DeWitt, Raghu Ramakrishnan, Incognito: Efficient Full Domain K-Anonymity, University of Wisconsin Madison Madison, WI 53706.

[9] H Tillwick and MS Olivier, Bridging the gap between anonymous e-mail and anonymous Web browsing, Online Information Review, 32, 1, 22-34 2008.

[10] M. Freedman, Design and Analysis of an Anonymous Communications Channel for the Free Haven Project, BS Thesis, MIT, 2000.

[11] IWT, APES: Anonymity and Privacy in Electronic Services, Requirement study of different applications, Deliverable 2, 2001.

[12] George Danezis, Better Anonymous Communications, Ph.D. Thesis, University of Cambridge, January 2004.

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

[14] M. K. Reiter and A. D. Rubin., Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security, 1(1):66-92, November 1998.

[15] M. Reed, P. Syverson, and D. Goldschlag, Anonymous Connections and Onion Routing, IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection, 1998.

[16] D. Chaum., The Dining Cryptographers Problem: Unconditional Sender and Recipient untraceability, Journal of Cryptography, 1(1):65- 75, 1988

[17] R. Sherwood, B. Bhattacharjee, and A. Srinivasan.,P5: A Protocol for Scalable Anonymous Communication, In Proc. 2002 IEEE Symposium on Security and Privacy, May 2002

[18] Joris Claessens, Bart Preneel, Joos Vandewalle, anonymity controlled electronic payment systems, 20th Symposium on Information Theory in the Benelux Haasrode, Belgium, May 27-28, 1999.

[19] Giuseppe Ateniese , Breno de Medeiros, Anonymous E-Prescriptions , ACM , 2002.

[20] O. Berthold, H. Federrath, and M. Kohntopp. Project anonymity and unobservability in the internet. In Computers Freedom and Privacy Conference 2000 (CFP 2000) Workshop on Freedom and Privacy by De-sign, April 2000.

[21] A. P_tzmann, B. P_tzmann, and M. Waidner. Isdnmixes: Untraceable communication with very small bandwidth overhead. In GI/ITG Conference: Communication in Distributed Systems, February 1991.

[22] H. Federrath. Jap: A tool for privacy in the internet. http://anon.inf.tu-dresden.de/index en.html.

[23] D. Kesdogan, J. Egner, and R. Buschkes. *Stop-and-go mixes providing probablilistic anonymity in an open system*. In Information Hiding, April 1998.

[24] Aneta Zwierko1 and Zbigniew Kotulski, *Mobile agents: preserving privacy and anonymity*, IMTCI2004, Warsaw 2004

[25] Chaum, D. L. *Blind signature for Untraceable payment*, Cryptography proceedings of CRYPTO'82 1982.

[26] M. Abe and E. Fujisaki, *How to date blind signatures*, Proc. Advances in Cryptology-ASIACRYPTO '96, LNCS 1163, pp 244– 251, 1996

[27] Goldreich, O.,Micali, S. , Wigderson, A , *Proofs that yield nothing but their validity* , ACM,1991.