V. Benjumea, J. Lopez, and J. M. Troya, "Specification of a Framework for the Anonymous Use of Privileges", *Telematics and Informatics*, vol. 23, pp. 179-195, 2006.
NICS Lab. Publications: https://www.nics.uma.es/publications

Specification of a Framework for the Anonymous Use of Privileges*

Vicente Benjumea Javier Lopez Jose M. Troya
Computer Science Department, University of Malaga, Spain
{benjumea,jlm,troya}@lcc.uma.es

Abstract

In this paper we have defined an open framework to support open distributed applications where anonymous transactions based on user privileges play an important role. The goal of the framework is to provide a basis to the application level, and is presented from an open and general perspective where many different implementation schemes can fit. Moreover, we have presented a set of requirements that implementation schemes must fulfill to conform a fully anonymous privilege system, which guarantees to supported applications that anonymity will be preserved in remote transactions. Finally, we present an application scenario using the services provided by the framework in order to better show the possibilities of what this type of systems offers.

Keywords: authorization, privilege, anonymity, credential, framework

1 Introduction

The increment in the number of remote transactions carried out through the Internet has a drawback: a big amount of user information can be collected, processed and cross referenced. This facilitates the creation of user profiles that store their preferences and activities. For this reason, privacy and anonymity are becoming emergent research topics in both the information security and cryptography areas. More than ever before they are becoming challenging topics with an important goal, the protection of users' sensitive information without restricting their capacity of performing remote transactions.

Beginning with Chaum's seminal work [6], many studies have focused on the support of privacy and anonymity services in many different ways [6, 7, 9, 8, 10, 18, 11, 14, 3, 1, 5, 4, 13, 12]. All proposals addressing anonymous systems [14, 3, 5, 4, 12] focus essentially on defining protocols and algorithms that try to solve the aforementioned problems. Additionally, many of those proposals include the definition of what an anonymous system is. However, to the best of our knowledge, there is no proposal trying to define a wide, open and general anonymity framework, independent of any implementation/protocol design, and based only on the requirements that such a system should fulfill

*Work partially supported by the Spanish Ministry of Science and Technology under the Project PRIVILEGE (TIC2003-08184-C02-01)

and on the operation primitives that should offer to the application level. To be more precise, the work presented in [14], and [5] seems to be the only work that focuses and defines the problem from a general point of view, though very influenced by the solution they propose.

Our paper tries to close this gap. That is, the present work collects a reasonable set of requirements that such a wide, open and general anonymity framework should fulfill. We also define the set of operation primitives that the framework should offer to the applications. Additionally, by making use of an application scenario, we show the possibilities that this kind of framework can offer.

2 Requirements that an anonymous system must fulfill

An anonymous system [6, 7, 9, 11, 14, 3, 5, 4, 12, 2] can be defined as a distributed system where the identities of some of the parties involved in the remote transaction remain unknown. Because these transactions are carried out among untrusted parties, perfect anonymity can not be accepted [19] since, in case of the abuse or misuse of the anonymous capability by any of the parties, the real identity of the dishonest party must be disclosed for eventual prosecution. A system must fulfill the following requirements to be considered fairly anonymous.

- No information can be revealed that can help any party to guess the identity of users involved in the transaction.
- The anonymous user must be able to enforce his privileges in the system without lack of anonymity. Thus, the set of anonymous transactions that the user can perform is compelled by his set of privileges.
- The real identity of the anonymous user that performed a given transaction can be disclosed under certain circumstances.
- No entity in the system can disclose by himself the real identity of a user that took part in the transaction. Such eventual disclosure must be done in collusion with several entities, where one of them is a *trusted third party* (TTP) that guarantees that the disclosure is achieved only when certain conditions are fulfilled.
- An anonymous user can not share his privileges with another anonymous user. That is, the later can not benefit from the privileges of the first one.
- As in any cryptographic system, it must be impossible to forge any operation in the system without the knowledge of the correspondent secret information. No entity, even a TTP or any type of Authority, or a collusion of them, can act on behalf of any other entity.
- No information will be revealed if this can help any entity to guess secret information used by involved parties in performed transactions.
- No anonymous party involved in a transaction can deny such involvement.

Additionally, an anonymous system must fulfill the following requirements to be broadly accepted.

- The anonymous transactions performed in the system must be unlinkable, thus avoiding the creation of anonymous user profiles.
- No information about the system can be disclosed whenever the real identity of an anonymous user that took part in a transaction is disclosed.
- Under certain circumstances, it will be possible to trace all the anonymous transactions that a specific user has performed. That is, it will be possible to distinguish the transactions performed by a specific user among all the anonymous transactions.
- When tracing the anonymous transactions in a system, no information will be revealed except for identifying those ones performed by a specific user.
- No entity in the system will be able to trace the anonymous transactions performed by another user, except if several entities (a TTP among them) collude.
- When conditions are met, it will be possible to revoke the privileges of a given user without revealing anything about the system. A TTP should be involved in the process.

The following requirements are also desirable in an anonymous system:

- A user will be able to show that he has performed a given anonymous transaction.
- If the system supports a payment method, the user will be able to make use of it without losing anonymity.

It should be noted that this set of requirements has been inspired by those appeared in the literature [14, 3, 5, 4, 12, 13]. However, we have specially adapted them in order to create a general and open anonymity framework since the requirements presented in the aforementioned literature are normally biased, to a greater or lesser extent, to fit in the corresponding proposal. To the best of our knowledge, they have never been collected and presented from a broad and general perspective and independently of any proposed scheme.

3 Anonymous identities versus anonymous privileges

An anonymous system can be defined in many different ways. However, focusing on the functionality that they provide to the applications, it is important to distinguish two different approaches.

In specific applications where all users involved have the same privileges and it is not necessary to distinguish them to decide the kind of service to be granted, it is sufficient to provide an anonymous system where the real identities of the users are hidden. Only under certain circumstances identity of users will

be disclosed. The real identities of the users are not necessary to perform anonymous transactions. Instead, the only requirement is to belong to the system, what makes possible to disclose the user's real identity under certain circumstances. We refer to these systems as *anonymous identity systems*, and can be supported by schemes that provide anonymity with respect to users' identities, such as anonymous PKI [12], anonymous communication channels [6, 8, 15, 16, 17], and others.

However, there are applications where the parties involved have different privileges and the kind of transactions that they perform depends on their individual privileges. Also, there are scenarios where certain infrastructures provide a framework where general applications can interact based on the privileges that different users have. In these two cases, the anonymous identity systems do not provide the necessary functionality. Contrarily, it is necessary a system that provides anonymity with respect to the privileges that the users have. We refer to these systems as *anonymous privilege systems* [14, 3, 5, 4, 2].

In this paper, we focus on the anonymous privilege systems, as they provide a more flexible and broader range of possibilities. Note that an anonymous identity system can be supported by an anonymous privilege system where only one privilege exists and it is available to all users.

4 A framework to use anonymous privileges: A functional perspective

This section introduces a framework focussing on the support of an anonymous privilege system. The framework will be studied from a functional perspective, that is, we will discuss the entities that compose the system, the primitives offered to the application level, and the operations that an entity in the application layer has to provide to the framework in order to behave coherently.

The framework is presented as an open and general scheme where different implementation schemes can fit. It does not impose any restriction, and provides the basis where many different application schemes based on anonymous privileges can be developed independently of a real implementation scheme. More precisely, in the framework, independent anonymous based applications and anonymous scheme implementations can work together to form a fully functional system. More restricted implementation schemes can also fit in the framework as a particularization with reduced functionality.

It is important to note that there is no guarantee that a scheme fitting into the framework necessarily fulfills the requirements aforementioned (see 2). It is up to the implementation scheme to guarantee such fulfillment, and should specify in which degree the requirements are met.

4.1 Overview of the framework

The framework is composed of two main sets of entities (figure 1). *External entities* make use of the services provided by the framework to perform anonymous transactions based on privileges. These external entities also manage the assignment of privileges, and provide a legal support for some actions needed by the framework.

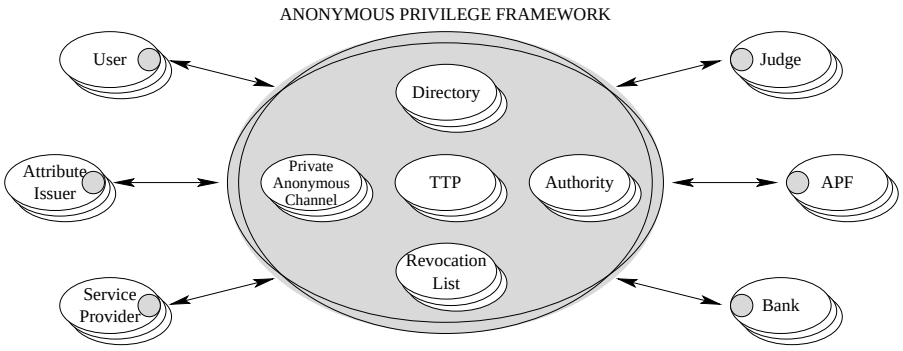

Figure 1: Anonymous privilege framework

On the other hand, *internal entities* provide the mechanisms to support the anonymous transactions, based on privileges, performed by the external entities. Moreover, they provide the mechanisms that make possible the revocation of the anonymity in the transactions.

Internal entities belong to the framework, and their main goal is precisely to support it. External entities have their own behavior and interest. Any external entity needs to contain a module that will provide the framework functionality, enabling that entity to make use of the services provided by the framework. Every module will provide, on the entity side, the cryptographic protocols that are part of the services provided by the framework. Eventually, a module will request the host to perform some actions and provide some information that will allow the module to complete its job.

A general overview of the anonymous privilege framework is as follows. The user requests from the framework the services and resources that this one and the external entities provide. Then, the user collects from the attribute issuer entities a set of credentials that contain information stating that he has certain privileges (implying that he fulfills certain conditions). Afterwards, he accesses the service provider entity anonymously and enforces his privileges by presenting the corresponding credentials. If a transaction requires a previous payment, the bank entity will support such anonymous payment. These are the normal operations in the system. However, under certain circumstances, such as anonymity abuse or misuse, the judge entity will order the disclosure of the the user's identity that performed the "suspicious" anonymous transactions. Finally, the framework must be able to connect and route anonymous transactions with other frameworks, thus allowing the creation of networks of anonymous privilege systems.

4.2 Actors

As stated before, the framework is composed of two kinds of entities: internal and external. There can be as many actors of each kind as defined by the implementation scheme. However, there should be at least one of each type in order to have full functionality. As a special case, if the system does not support anonymous payment, then the bank entity is not necessary.

Although the proposed framework is open and general enough to fit different

implementation schemes, any of them, from a functional perspective, should provide at least the following internal actors.

- The *Authority*, among other things, enrolls into the system the external entities that request it.
- The *Directory* keeps information about the system, the entities that compose it, the services offered, resources available, public key certificates, entities' policies, etc. It allows the entities to access to the public information that they need.
- The *Trusted Third Party* (TTP) is a lightweight entity in charge of assuring that disclosures, traces and revocations are achieved with fairness. Therefore, its active involvement in such operations is mandatory.
- The *Revocation List* holds information about the attributes that have been revoked in the system. Entities are allowed to request that information.
- The *Private Anonymous Channel* (PAC) guarantees that the physical communication endpoints remain unknown in anonymous transactions. All the anonymous transactions performed in the system are to be carried out through *private anonymous channels* [6, 8, 15, 16, 17], thus guaranteeing that the communication physical endpoints remain unknown. As a special case, and only when the user requests it, the framework can provide a *private channel* which allows direct, fast and private communication with no anonymity at the communication physical endpoints.

The external entities make use of the services provided by the framework, and use them to provide other services to other entities of the system.

- The *User* collects attributes that enable him to enforce his privileges when performing anonymous transactions through the framework; that is, when anonymously accessing the services and resources offered by the system providers.
- The *Service Provider* (SP) offers its services and resources to anonymous users that prove to hold the necessary privileges. The policy of the service or resource will specify which are the attributes necessary for a user to hold.
- The *Attribute Issuer* (AI) entity issues attributes to users that prove fulfillment of certain conditions.
- The *Bank* provides payment support to the anonymous transactions that require it. Additionally, a bank can act as a service provider entity to offer its services to other entities.
- The *Judge* acts as a law enforcement entity that authorizes or denies the disclosure, trace and revocation actions performed in the system when the conditions required are met.
- It is possible to interconnect *Anonymous Privilege Frameworks* (APF) together to form hierarchies of APFs or even more complicated topologies. In these cases, the interconnected APFs will act as routers with respect to the anonymous transactions.

It is possible that different entities in the framework are represented by the same entity in the real world (or in a virtual world). For instance, an "internet online bank" can act as a Bank entity in the framework, but also can act as an Attribute Issuer, issuing attributes for workers of the bank. In fact, it also can act as a service provider which offers some resources to anonymous users.

4.3 Primitives provided

The primitives that the framework provides will be discussed in this subsection from a functional perspective again, and the different implementation schemes should provide them, guaranteeing the fulfillment of the requirements mentioned in section 2. Also, we will show in this section the operations that the external entities have to provide to enable their interaction with the framework primitives.

The set of primitives are inspired by those appeared in the literature [14, 5, 13, 2]. However, here again, we have adapted them to fit in a general and open anonymous privilege system. As for the requirements, the reason is that the primitives presented in the literature are normally biased to fit in the corresponding schemes.

In the following explanation, an underlined operation means a primitive provided by an external entity that will be invoked by the framework module to provide the necessary information to run the implementation protocol. From the primitive perspective, input information is denoted as a parameter marked with a down-arrow (⇓) and output information is denoted as a parameter marked with an up-arrow (⇑). However, from the invoker perspective the meaning is the opposite.

4.3.1 Management primitives

Management primitives are related to the framework setup and the way entities join and leave it. The directory databases will be updated to contain these transactions, as well as all the information needed for the entities to cooperate in the framework.

- The general setup of the framework and internal entities is provided by the following primitive. It will create a key pair. The private key will be used by the framework manager. The public key will be kept in the directory databases, together with the framework and internal entities information.

  `APF_Framework_Setup(⇓`*Apf_Id*`, ⇓`*Setup_Info*`,`
  `⇑`*Ok*`, ⇑`*Apf_SK*`, ⇑`*Setup_Report*`)`

- The following primitive allows the external entities (User, Attribute Issuer, Service Provider, Judge, Bank, external APF) to join the framework. It creates a key pair that enables the entity to identify itself. The public key certificate and the entity information will be stored in the directory databases.

  `APF_Join(⇓`*Entity_Id*`, ⇓`*Join_Info*`,`
  `⇑`*Ok*`, ⇑`*Entity_SK*`, ⇑`*Join_Report*`)`

- An entity leaves the framework by using the following primitive. This event will be recorded in the directory databases too.

```
APF_Resign(⇓Entity_Id, ⇓Entity_SK,
           ⇑Ok, ⇑Resignation_Report)
```

- The following primitive allows the entities to access the directory databases, that will hold information about external APFs, system entities, offered services, available resources, public key certificates, entities' policies, etc.

```
APF_Get_Directory_Info(⇓Info_Request, ⇑Directory_Info)
```

4.3.2 User oriented primitives

User oriented primitives provide a means for the user to access the services provided by the framework and external entities.

- This primitive allows the user to obtain an information credential guaranteeing his attributes. He will use the credential in order to enforce the related privileges. The user requests the credential to an Attribute Issuer entity, which will check if the user fulfills the requirements needed to posses it, and in such case, it will issue the credential. A TTP should be involved in the process in order to fulfill the anonymity requirements. No entity alone will be able to relate the credentials issued with the identity of the users that own them.

  1. The following primitive will start the process. The user will identify himself and will provide the proofs that enable him to acquire the credential guaranteeing the attributes. The user will obtain the credential as well as the secret key that enables him to prove anonymously his ownership.

```
APF_Attribute_Request(⇓User_Id, ⇓User_SK, ⇓PAC_Id, ⇓TTP_Id,
                      ⇓Attr_Issuer_Id, ⇓Attr_Id, ⇓Attr_Proof_List,
                      ⇓Attr_Proof_SK_List,
                      ⇑Ok, ⇑Attribute, ⇑Attr_SK)
```

  2. As a consequence of the primitive action, the framework will request the specified Attribute Issuer (by invoking the next primitive provided by the attribute issuer entity) to verify if the user fulfills the requirements to own such attribute. If so, the framework will be enabled to verify the ownership of the presented proof list.

```
AI_Verify_Attr_Req(⇓User_Id, ⇓Attr_Id, ⇓Attr_Proof_List,
                   ⇑Ok, ⇑Attr_Req_Id, ⇑Attr_Issuer_SK)
```

  3. If the user fulfills the requirements to own such attribute, and once the the proof list ownership have been checked, the framework will request the Attribute Issuer to provide the information needed to issue the attribute credential through the following action.

```
AI_Provide_Attribute(⇓Attr_Req_Id, ⇓Attr_Req_Report,
                     ⇑Ok, ⇑Attribute_Info, ⇑Attr_Issuer_SK)
```

- The main aim of the framework is to allow the users to perform anonymous transactions with service providers. This is the goal of the next primitive.

  1. The user will request a service offered by a service provider through the following primitive. He will identify the entity that provides the service and the anonymous channel that will route the communication. Then, he will provide the attribute list that enables him to

access the offered service. If the set of presented attributes is enough and the ownership is proved, the user will get a transaction identification and the requested service.

```
APF_Service_Request(⇓User_Id, ⇓User_SK, ⇓Service_Provider_Id, ⇓PAC_Id,
                    ⇓Service_Id, ⇓Attribute_List, ⇓Attr_SK_List,
                    ⇑Ok, ⇑Transaction_Id, ⇑Service)
```

2. As a consequence of the primitive action, the framework will request the Service Provider entity to verify if the anonymous user owns enough attributes to access the requested service. If so, the service provider will enable the framework to check if the presented attributes are valid (they have not been revoked and the validity period is right) and to anonymously verify the ownership of the attributes presented.

```
SP_Verify_Service(⇓Service_Id, ⇓Attribute_List,
                  ⇑Ok, ⇑Transaction_Id, ⇑Service_Provider_SK)
```

3. Once the attribute ownership has been proved, the framework will request the service provider to provide the requested service through the following action.

```
SP_Provide_Service(⇓Service_Id, ⇓Transaction_Report,
                   ⇑Ok, ⇑Service, ⇑Service_Provider_SK)
```

- Many services offered in the system require, in addition to owning some attributes, a payment before accessing the service. The following primitive support the access to services that require a previous payment. Here again, the framework will ensure that the attributes involved are valid and not revoked.

1. The user will access the service as before, but now he has to provide a payment information and a secret key to enable such payment. At last, the user accesses the service and gets an evidence of the payment done.

```
APF_Payment_Service_Request(⇓User_Id, ⇓User_SK, ⇓Payment_Info,
                            ⇓Payment_SK, ⇓Service_Provider_Id,
                            ⇓PAC_Id, ⇓Service_Id, ⇓Attribute_List,
                            ⇓Attr_SK_List,
                            ⇑Ok, ⇑Transaction_Id, ⇑Service,
                            ⇑Payment_Evidence)
```

2. As before, the service provider will be requested to verify if the user owns enough attributes to access the service and, if so, it will provide information about the way the charge can be done.

```
SP_Verify_Payment_Service(⇓Service_Id, ⇓Attribute_List,
                          ⇑Ok, ⇑Transaction_Id, ⇑Charge_Info,
                          ⇑Service_Provider_SK)
```

3. The Bank entity will be requested to do the following action in order to support the payment transaction, which will be achieved anonymously. It will provide a payment evidence that will be communicated to involved parties.

```
BK_Charge_Payment_Service(⇓Charge_Req, ⇓Payment_Req,
                          ⇑Ok, ⇑Payment_Evidence, ⇑Bank_SK)
```

4. Once the service provider gets the payment evidence, it provides the requested service.

SP_Provide_Payment_Service(⇓*Service_Id*, ⇓*Transaction_Report*,
⇓*Payment_Evidence*,
⇑*Ok*, ⇑*Service*, ⇑*Service_Provider_SK*)

- Sometimes the user needs to justify that he performed a certain transaction. The following primitive supports such action.

  1. The user has to identify the transaction for which he is claiming the authorship. Moreover, he needs to provide the secret keys used to prove his ownership in the original transaction. Finally, he will get a report stating that the user performed such anonymous transaction.

     APF_Authorship_Claim(⇓*User_Id*, ⇓*User_SK*, ⇓*Service_Provider_Id*,
     ⇓*PAC_Id*, ⇓*Service_Id*, ⇓*Transaction_Id*,
     ⇓*Attr_SK_List*,
     ⇑*Ok*, ⇑*Authorship_Report*)

  2. The Service Provider entity will be requested to support such operation by providing the necessary information to the framework.

     SP_Provide_Authorship(⇓*Service_Id*, ⇓*Transaction_Id*,
     ⇑*Ok*, ⇑*Transaction_Info*, ⇑*Service_Provider_SK*)

- The framework could offer simply anonymous transactions through the Private Anonymous Channel entities to allow the external entities to communicate anonymously. This is a low level primitive to support anonymous transactions where privileges are not involved.

  1. The origin entity will perform the desired anonymous transaction by using the following primitive

     APF_Anonymous_Transaction(⇓*Origin_Id*, ⇓*Origin_SK*, ⇓*Destination_Id*,
     ⇓*PAC_Id*, ⇓*Transaction_Id*, ⇓*Transaction_Info*,
     ⇑*Ok*, ⇑*Transaction_Result*)

  2. The destination entity will be requested to perform the following action in order to process the anonymous transaction.

     XX_Process_Anonymous_Transaction(⇓*Transaction_Id*, ⇓*Transaction_Info*,
     ⇑*Ok*, ⇑*Transaction_Result*,
     ⇑*Destination_SK*)

4.3.3 Special case primitives

Special case primitives include all primitives related with the revocation, in different flavors, of anonymity. Additionally, they include a primitive oriented to persuade the user for not sharing the ownership of his attributes with other users.

- The following primitive allows, if the requirements are met, to disclose the identity of the user that performed a given anonymous transaction. This primitive is offered to any entity in the system. However, a judge must authorize such action.

  1. Any entity in the system can request the disclosure of the real identity of the user that performed a given transaction. Such operation will be achieved if the conditions are met. The requester must provide the proofs necessary to check if the disclosure requirements are met and, if so, the user identity will be disclosed.

APF_Disclosure_Req(⇓*Requester_Id*, ⇓*Requester_SK*, ⇓*Transaction_Id*,
⇓*Disclosure_Proof_List*,
⇑*Ok*, ⇑*User_Identity*)

2. The Judge entity will be requested to evaluate the presented proofs and to agree (or disagree) on the requested action.

JG_Verify_Disclosure_Req(⇓*Requester_Id*, ⇓*Transaction_Id*,
⇓*Disclosure_Proof_List*,
⇑*Ok*, ⇑*Judge_Resolution*, ⇑*Judge_SK*)

3. If the Judge agrees on the disclosure, the framework will request the involved entities (Service Provider, Attribute Issuer, TTP, Bank, etc.) to provide the necessary information to disclose the real identity of the user.

XX_Process_Judge_Resolution(⇓*Info_Request*, ⇓*Judge_Resolution*,
⇑*Ok*, ⇑*Requested_Info*, ⇑*Entity_SK*)

- The following primitive allows, if the requirements are met, to trace all the anonymous transactions that a given user has performed in the system. This primitive is offered to any entity in the system, however, a Judge must authorize such action.

1. Any entity in the system can request to trace the transactions performed by certain user. Such operation will be achieved if some conditions are met. The requester must provide the proofs necessary to check if the tracing requirements are met and, in that case, a list with the identification of the anonymous transactions performed by the addressed user will be generated.

APF_Tracing_Req(⇓*Requester_Id*, ⇓*Requester_SK*, ⇓*User_Id*,
⇓*Tracing_Proof_List*,
⇑*Ok*, ⇑*Transaction_Id_List*)

2. The Judge entity will be requested to evaluate the proofs presented and to agree (or disagree) on the requested action.

JG_Verify_Tracing_Req(⇓*Requester_Id*, ⇓*User_Id*, ⇓*Tracing_Proof_List*,
⇑*Ok*, ⇑*Judge_Resolution*, ⇑*Judge_SK*)

3. If the Judge agrees on tracing the anonymous transactions, the framework will request the involved entities (Service Provider, Attribute Issuer, TTP, Bank, etc.) to provide the necessary information to trace the transactions performed by the addressed user.

XX_Process_Judge_Resolution(⇓*Info_Request*, ⇓*Judge_Resolution*,
⇑*Ok*, ⇑*Requested_Info*, ⇑*Entity_SK*)

- The framework allows to revoke an attribute owned by certain user if certain conditions are met. In this case, the user will not be able to further use that attribute in any following anonymous transactions. A Judge must authorize that action.

1. Any entity in the system can request the revocation of an attribute of certain user when some conditions are met. The requester must provide the proofs necessary to check that the revocation has to be done.

APF_Revoke_UAttr_Req(⇓*Requester_Id*, ⇓*Requester_SK*, ⇓*User_Id*, ⇓*Attr_Id*,
⇓*Revoking_Proof_List*,
⇑*Ok*, ⇑*Revoking_Report*)

2. The Judge entity will be requested to evaluate the proofs presented and to agree (or disagree) on the requested action.

   JG_Verify_UAttr_Revoking_Req(⇓*Requester_Id*, ⇓*User_Id*, ⇓*Attr_Id*, ⇓*Revoking_Proof_List*, ⇑*Ok*, ⇑*Judge_Resolution*, ⇑*Judge_SK*)

3. If the judge agrees on revoking such attribute, the framework will request the involved entities (Service Provider, Attribute Issuer, TTP, Bank, etc.) to provide the necessary information to revoke that attribute. The internal databases must reflect such event, and provide mechanisms to make it public to interested parties.

   XX_Process_Judge_Resolution(⇓*Info_Request*, ⇓*Judge_Resolution*, ⇑*Ok*, ⇑*Requested_Info*, ⇑*Entity_SK*)

- The framework allows to revoke an attribute used in an anonymous transaction if certain conditions are met. In this case, the owner will not be able to further use such attribute in any latter anonymous transaction. A judge must authorize such action.

  1. Any entity in the system can request the revocation of an attribute used in an anonymous transaction. The requester must provide the proofs necessary to check if the revoking requirements are met.

     APF_Revoke_TAttr_Req(⇓*Requester_Id*, ⇓*Requester_SK*, ⇓*Transaction_Id*, ⇓*Attr_Id*, ⇓*Revoking_Proof_List*, ⇑*Ok*, ⇑*Revoking_Report*)

  2. The Judge entity will be requested to evaluate the presented proofs and to agree (or disagree) on the requested action.

     JG_Verify_TAttr_Revoking_Req(⇓*Requester_Id*, ⇓*Transaction_Id*, ⇓*Attr_Id*, ⇓*Revoking_Proof_List*, ⇑*Ok*, ⇑*Judge_Resolution*, ⇑*Judge_SK*)

  3. If the judge agree on revoking such attribute, the framework will request the involved entities (Service Provider, Attribute Issuer, TTP, Bank, etc.) to provide the necessary information to revoke such attribute. The internal databases must reflect such fact, and provide mechanisms to make it public to interested parties.

     XX_Process_Judge_Resolution(⇓*Info_Request*, ⇓*Judge_Resolution*, ⇑*Ok*, ⇑*Requested_Info*, ⇑*Entity_SK*)

- The following primitive is oriented to persuade users for not sharing the ownership of their attributes with other users. It follows the approach proposed by [14] of linking sensitive information that the user does not want to share, with the capability of using certain attribute. In our case, we link the capability of using an attribute with the capability to perform an action that, thought profitable for the user of the attribute, it is *unprofitable* for the owner of the attribute.

  1. The dissuading primitive is as follows.

     APF_Dissuading_Primitive(⇓*User_Id*, ⇓*User_SK*, ⇓*Charge_Info*, ⇓*PAC_Id*, ⇓*Attribute*, ⇓*Attr_SK*, ⇑*Ok*, ⇑*Payment_Evidence*)

  2. The Bank entity will be requested to perform the following action in order to support the primitive action.

     BK_Charge_Dissuading_Payment(⇓*Charge_Info*, ⇓*Payment_Info*, ⇑*Ok*, ⇑*Payment_Evidence*, ⇑*Bank_SK*)

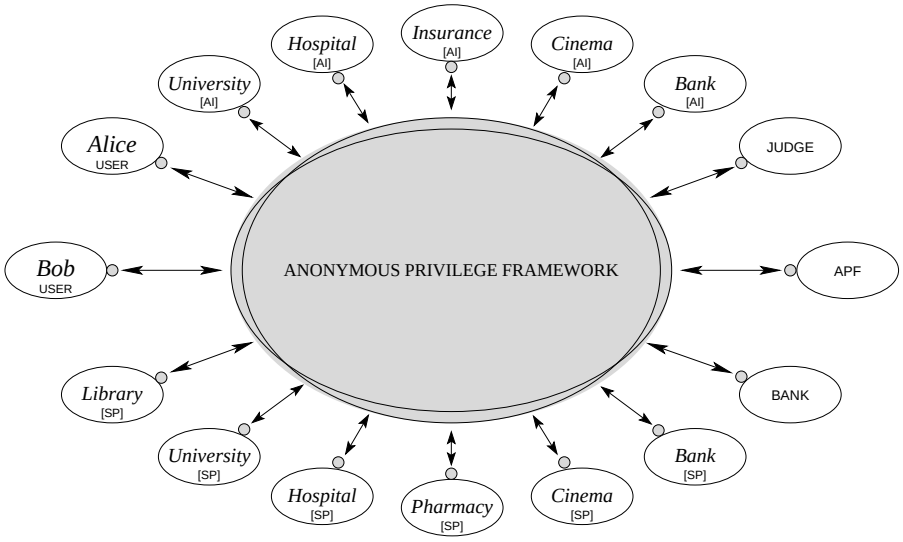

Figure 2: An anonymous privilege scenario

5 Towards an anonymous "cyberworld"

This section describes a scenario where the potential of this technology can be put into perspective. It tries to resemble what an anonymous world could be[1].

In our world (figure 2) there are several entities: a University, an insurance company, a bank, a hospital, a pharmacy and a cinema, in addition to the judge and the interconnected APFs. The different functionalities will be divided among the entities. Thus, for example, the university will act as an attribute issuer entity, issuing credentials guaranteeing the relationship between the users and the university, and will act as a service provider offering documents (books, journals, papers, etc). Additionally, it will act as a service provider managing the payment of salaries. The bank itself also will have several functionalities. It will act as a bank entity supporting payment in transactions. Additionally, it will act as an attribute issuer entity to issue credentials guaranteeing the information related to users' bank accounts. Finally, it will act as a service provider entity offering the capability of anonymous bank transfers.

In our world there also exists two users that will use the framework to perform anonymous transactions in their daily life. *Alice* is a professor and *Bob* is a student at the university. Moreover, more different entities and applications can join the system to become part of an anonymous "cyberworld" as complete as the real one.

Note that this anonymous world can represent a virtual world interconnected through the Internet mixed with a real world where the users personally access the resources. The access control is achieved between the user's mobile computer (PDA, mobile cell phone, etc) and the service provider computer through a wireless connection.

When the entities join the framework, the directory databases are updated

[1]The scenario health part has been inspired by [14]

entity_class	entity	resource	policy	pkey_cert	info
directory	default	system_info	access_enabled	dir_pkc	dir_info
authority	default	system_mgnt	system_policy	auth_pkc	auth_info
ttp	default	system_ttp	ttp_policy	ttp_pkc	ttp_info
pac	default	system_pac	pac_policy	pac_pkc	pac_info
pac	direct	system_pac	pac_policy	pac_pkc	pac_info
rev_list	default	system_rvl	rvl_policy	rvl_pkc	rvl_info
judge	judge	law&fairplay	judge_policy	judge_pkc	judge_info
bank	bank	trans_payment	bank_policy	bank_pkc	bank_info
ext_apf	eapf_1	ext_apf	eapf_policy	eapf_pkc	eapf_info
attr_issuer	university	status_mgnt	univ_policy	univ_pkc	univ_info
attr_issuer	university	job_mgnt	univ_policy	univ_pkc	univ_info
serv_provider	university	salary_mgnt	univ_policy	univ_pkc	univ_info
serv_provider	library	book_repository	univ_policy	univ_pkc	univ_info
attr_issuer	bank	account_mgnt	bank_policy	bank_pkc	bank_info
serv_provider	bank	transfer_mgnt	bank_policy	bank_pkc	bank_info
attr_issuer	hospital	health_trnt	hospital_policy	hospital_pkc	hospital_info
serv_provider	hospital	health_trnt	hospital_policy	hospital_pkc	hospital_info
attr_issuer	insurance	insurance_mgnt	ins_policy	ins_pkc	ins_info
serv_provider	pharmacy	medicine_repository	pharmacy_policy	pharmacy_pkc	pharmacy_info
serv_provider	cinema	film_repository	cinema_policy	cinema_pkc	cinema_info
user	alice		user_policy	alice_pkc	alice_info
user	bob		user_policy	bob_pkc	bob_info

Figure 3: Directory

with information related to them, like resources and services offered, public keys certificates, policies, etc. Thus, for example, the directory databases can hold information like that shown in figure 3. In that figure, *system_policy* states the rules that define the fair behavior in the system, how entities are enrolled and resigned in the system, and under what circumstances the real identities will be disclosed, anonymous transactions will be traced and privileges revoked.

Alice joins the framework and provides a bank account certificate and a statement saying that accepts any charge performed by using the `APF_Dissuading_Primitive` with any of her anonymous attributes (which means that she has shared her secret information with other user).

Because Alice is professor at the university, she will request an attribute stating such relationship to the University attribute issuer. This entity will verify the set of proofs presented by Alice, consulting if necessary its internal databases, and will issue the corresponding credential to Alice. As a result, Alice gets a credential guaranteeing that its anonymous owner is a professor at the university. She will use it anonymously to enforce her privileges anywhere that accepts such credential. Additionally, she will also request an anonymous credential where her duties are specified.

As a worker of the University, Alice owns a medical insurance, therefore she will request a credential stating such relationship, allowing her to enforce her medical privileges with no need of identification. Thus, nobody, even the doctors, will be able to relate herself with any of the actions that she performs.

During Alice's daily life, she anonymously presents her professor credential

to access the restricted area in the library to get any paper she is interested in. Bob is a student at the university, and therefore is only allowed to access the student area in the library.

When Alice feels ill, she goes to the hospital, and accesses the services provided by a doctor there. She does not have to pay for such service, since she has anonymously presented her insurance credential. After examination, the doctor issues an anonymous attribute stating that its owner suffers the specified illness. Additionally, he also issues an anonymous credential stating the prescriptions for such illness. If this does not allow Alice to work, the doctor also issues an anonymous credential stating the period of sick leave for its owner.

Alice will anonymously present her sick leave and duty attributes to inform the university management department that she will not be able to work for the specified period of time. She will anonymously use the medical prescription credential to get the specified medicines from the pharmacy by means of the `APF_Payment_Service_Request`. Also, she will anonymously present her insurance and professor credentials if they provide any discount in the price. Once the prescription credentials have been used, they will be revoked. If she goes again to the hospital, she can anonymously present the attribute stating her illness to require a second diagnostic.

Whenever Alice goes back to work, she anonymously presents her professor credential and gets a credential stating that its anonymous owner performed that work. Alice will get her salary after presenting her professor and duty credentials together with her work receipt and sick leave attributes. The salary will be calculated accordingly, and payed through the payment service. The work receipt and sick leave attributes will be revoked after this. If the anonymous user does not fulfill her duties, her real identity can be disclosed and her attributes revoked.

Alice can anonymously buy tickets for the cinema at the university. If she gets any discount for being a professor, she will anonymously present her professor credential when paying for the tickets. The ticket will be revoked after its use.

If Alice shares with Bob any of her anonymous attributes, she will allow Bob enjoying privileges that does not correspond to him. However, she will also allow him to use the `APF_Dissuading_Primitive` on her behalf, which is a tempting action since Bob will legally get the money that he wants from Alice bank account. Most of cases, this fact will prevent Alice from sharing her secret information with other users.

Having a broader look at this anonymous cyberworld, we can envision a world where citizens that pay their taxes get anonymous credentials that enables them to use the resources that the Government provides. Citizens that have a driving license and have a car insurance will hold anonymous credentials that the car could check to allow them to drive or not. A policeman can ask the driver to prove the ownership of the anonymous driving license attribute, and if the driver infringes the law, the penalty can be paid with no need of disclosing his identity. However, if the penalty requires the disclosure of the identity or the revocation of the license, this action can be performed.

It can be seen that this cyberworld can resembles all the complexities of the real world with the added value that all the transactions can be done anonymously, and under certain circumstances the identity can be disclosed, the attributes can be revoked, and the anonymous actions traced.

6 Conclusions

We have defined an open framework to support open distributed applications where anonymous transactions based on user privileges play an important role. The framework is presented from an open and general basis where many different implementation schemes can fit.

Moreover, we have presented a set of requirements that implementation schemes must fulfill to conform a fully anonymous privilege system, which guarantees to supported applications that anonymity will be preserved in remote transactions.

The presented framework is the connection point between open applications and different implementation schemes supporting anonymous privileges by defining the playing rules and the requirements that both sides should fulfill.

Finally, we have described a scenario where the potential of this kind of systems can be seen. The scenario is presented as an application example, and it is far from being exhaustive.

To the best of our knowledge, it is the first time that this kind of systems is approached from the point of view shown in this paper.

References

[1] G. Ateniense, J. Camenish, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000: Advances in Cryptology*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.

[2] V. Benjumea, J. López, J. A. Montenegro, and J. M. Troya. A first approach to provide anonymity in attribute certificates. In F. Bao, R. H. Deng, and J. Zhou, editors, *PKC 2004, 2004 International Workshop on Practice and Theory in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 402–415. Springer-Verlag, Mar. 2004.

[3] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates Building in Privacy.* The MIT Press, Aug. 2000.

[4] J. Camenisch and E. V. Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proc. of 9th ACM Conference on Computer and Communications Security (CCS)*, Washington D.C., Nov. 2002. ACM, Academic Press.

[5] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer-Verlag, 2001.

[6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, Feb. 1981.

[7] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.

[8] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptography*, 1(1):65–75, 1988.

[9] D. Chaum and J. H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In A. M. Odlyzko, editor, *Advances in Cryptology - Crypto '86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–170, Berlin, 1986. Springer-Verlag.

[10] D. Chaum and E. v. Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology - EuroCrypt '91*, pages 257–265, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 547.

[11] L. Chen. Access with pseudonyms. In E. Dawson and J. Golic, editors, *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, 1995.

[12] D. Critchlow and N. Zhang. Security enhanced accountable anonymous PKI certificates for mobile e-commerce. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45(4):483–503, July 2004.

[13] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT 2004: Advances in Cryptology*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer-Verlag, 2004.

[14] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[15] P. S. M. Reed and D.Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection*, 1988.

[16] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, nov 1988.

[17] C. Shields and B. Levine. A protocol for anonymous communication over the internet. In *Proc. 7th ACM Conference on Computer and Communication Security*, nov 2000.

[18] M. A. Stadler, J. M. Piveteau, and J. L. Camenisch. Fair blind signatures. In L. C. Guillou and J. J. Quisquater, editors, *Advances in Cryptology–Eurocrypt'95*, volume 921 of *Lecture Notes in Computer Science*, pages 209–219. Springer-Verlag, 1995.

[19] S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers & Security*, 11:581–583, 1992.