Anonymous Attribute Certificates based on Traceable Signatures

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

Abstract

In Benjumea et. al (Benjumea, 2004) we introduced the concept of anonymous attribute certificates in order to integrate anonymity capabilities in the standardized X.509 attribute certificates. That solution was based on the use of fair-blind signatures (Stadler, 1995), but did not explore further possibilities of constructing similar data structures based on more advanced signature schemes. In this new work, we propose a new type of anonymous attribute certificates that is based on the more recently proposed traceable signature scheme (Kiayias, 2004a), providing a new anonymous authorization solution with interesting features that were not covered in the aforementioned scheme. Thus, this new solution allows users to make use of their attribute certificates in an anonymous way, but under certain circumstances it allows to disclose the users' identities, trace the transactions carried out by any specific user, or revoke any anonymous attribute certificate. An additional contribution of this work is that it pays special attention to the preservation of the unlinkability property between transactions, making impossible the creation of anonymous user profiles.

Keywords: authorization, PMI, anonymity, credential, X.509 attribute certificates, traceable signatures

1 Introduction

The number of remote transactions that takes place through the Internet is growing everyday. One of the problems of this situation is that malicious entities can record and process those transactions and, in a later stage, cross reference them. Such a behavior allows to obtain valuable information about many of the users' activities. Therefore, the more the number of remote transactions grows, the more convenient is that these transactions are done in an anonymous way.

However, when dealing with anonymity services we find that, in many cases, there are some requirements convenient to fulfill. Firstly, users should be able to make use of all the privileges assigned to them, without lack of anonymity. Secondly, most of times users' anonymity should not be perfect since this would provide a perfect framework for fraud and dishonest behavior (von Solms, 1992). In this sense, it should be possible to disclose the identities of anonymous users if an authority requests it, and only under certain conditions. Finally, it should be impossible to link anonymous transactions since this facilitates the creation of anonymous user profiles, what eventually can disclose users' identities.

Many anonymity schemes and cryptographic tools have been designed and developed throughout the years to provide anonymity services in many different ways. Chaum introduced the concept of *blind signature* in (Chaum, 1983; Chaum, 1985), and since then many signature schemes focusing on anonymity and its conditional revocation have been proposed (Chaum, 1991; Stadler, 1995; Kilian, 1998; Ateniense, 2000; Rivest, 2001; Kiayias, 2004a). In many cases, these theoretical schemes have not evolved towards practical solutions for a wide deployment. Our previous work (Benjumea, 2004) introduced a first approach to provide anonymity in standard (so, potentially widely deployed) X.509 attribute certificates (ITU-T Recommendation X.509, 2000), transferring a fair blind signature scheme to those certificates. The main outcome of that work was the definition of a new data object, the *Anonymous Attribute Certificate*, in which the holder's identity can be conditionally traceable depending on certain conditions.

In this paper, we explore the suitability of more recent signature schemes to provide anonymity in those ITU-T attribute certificates. More precisely, we elaborate on the use of the traceable signature scheme (Kiayias, 2004a), that provides a powerful tool-set to support anonymity in many different scenarios. It is a provably secure group signature scheme that provides efficient tools to support groups of anonymous users, though keeping the possibility to revoke a user's identity. We make use of this type of scheme in order to produce a new solution of anonymous attribute certificates that avoids the creation of anonymous user profiles, a problem that was not solved in (Benjumea, 2004).

The structure of this paper is as follows. In section 2, the underlying primitives used in our system will be briefly introduced. We also describe the standard X.509 attribute certificates proposed by ITU-T and how the framework that this type of attributes defines can be linked to other structures. Section 3 defines the data structure that supports anonymity when using a traceable signature scheme and that is used in the later sections. Section 4 shows a general overview of the system proposed, while section 5 describes the protocols to create and use those anonymous certificates. Section 6 discusses about the implementation of the system in real world applications. Finally, section 7 concludes the paper.

2 Background

2.1 Traceable signature primitives

Traceable signatures were proposed by Kiayias et al. (Kiayias, 2004a) as a group signature scheme with the capability of opening a signature. Additionally, it provides tracing capabilities, what makes it very suitable for real-world applications where such features are required for a broad acceptance of the model.

Group signatures (Ateniense, 2000) main features include the creation of virtual groups where users can join, with the particularity that any member of the group can prove that she belongs to it. The proof can be verified, but is indistiguishable from any other proof performed by the same or any other member. That is, given a proof, it can be verified whether it was performed by a member, but it can not be linked with any particular one nor with any other proof performed by any member or even by the same one. However, there is a special entity, the *Group Manager* (GM), who is able to identify the member who performed a given proof, allowing in this manner to revoke the anonymity that the group offers.

Traceable signatures offer, in addition to the aforementioned properties of group signatures, a tracing facility: the ability to identify, within a set of proofs, which ones were performed by a given member of the group. Moreover, a user can claim that a given proof was performed by herself.

In the following, we will briefly review the traceable signature primitives.

- The SETUP algorithm is executed by the GM. It gets a security level and produces a key pair (public and private) to be used in the next primitives.
  - $\langle GM_{publ}, GM_{priv}\rangle := \mathsf{TS\_Setup}(security\_level)$
- The JOIN protocol is run between a new user and the Group Manager when the user wants to join to the group managed by that GM. This primitive produces at the user's side of the protocol a private membership key that allows her to prove that belongs to such a group. At the GM's side produces some info related with the new member, that allows the GM to open or trace member activities. However, such information does not allow the GM to forge the member's proof.
  - $member\_ref_{priv} := \mathsf{TS\_Join}_{GM}(GM_{publ},\ GM_{priv})$
  - $member\_key_{priv} := \mathsf{TS\_Join}_{U}(GM_{publ})$
- The IDENTIFY protocol is run when a user wants to prove to an entity her membership to the specified group. It is an efficient zero knowledge proof that reveals nothing about the user. Nobody, except the GM, is able to relate the proof with the member that performed it, nor even with another proof performed by any member of the group.
  - $\mathsf{TS\_Identify}_{U}(GM_{publ},\ member\_key_{priv})$
  - $\langle ok, member\_proof\rangle := \mathsf{TS\_Identify}_{E}(GM_{publ})$
- The OPEN primitive is executed by the GM with the aim of knowing which member of the group issued a given membership proof. The result can be compared with the result of the JOIN protocol to identify the member of the group.
  - $member\_ref'_{priv} := \mathsf{TS\_Open}_{GM}(GM_{publ},\ GM_{priv},\ member\_proof)$
- The REVEAL primitive is executed by the GM too. Its goal is to get, for a given member of the group, her tracing trapdoor that allows to certain entities (tracers) to execute the TRACE algorithm and identify which proofs were issued by the given member.
  - $member\_trapdoor_{priv} := \mathsf{TS\_Reveal}_{GM}(GM_{publ},\ GM_{priv},\ member\_ref_{priv})$
- The TRACE primitive is executed by tracers with the aim of knowing if a given proof was issued by a designated member whose trapdoor is given.
  - $ok := \mathsf{TS\_Trace}_{T}(member\_trapdoor_{priv},\ member\_proof)$
- The CLAIM protocol is run when a user wants to prove to any entity that a given membership proof was performed by herself.
  - $\mathsf{TS\_Claim}_{U}(GM_{publ}, member\_key_{priv})$
  - $ok := \mathsf{TS\_Claim}_{E}(GM_{publ},\ member\_proof)$

2.2 X.509 Attribute Certificates

Public Key Certificates, as defined in ITU-T X.509v3 (ITU-T Recommendation X.509, 1997), can convey authorization information about its owner. The information can be encoded in one of the extension fields. However, most often that type of certificate is not the best vehicle to carry authorization information. This is the reason why the U.S. American National Standards Institute (ANSI) X9 committee developed the concept of *Attribute Certificate*. An attribute certificate is a data structure that binds some attribute values with identification information about its holder.

Attribute certificates have been incorporated into the most recent ITU-T X.509 Recommendation (ITU-T Recommendation X.509, 2000). Additionally, the Recommendation defines a framework that provides the basis upon which a *Privilege Management Infrastructure* (PMI) can be built. Precisely, the foundation of the PMI is the *Public Key Infrastructure* (PKI) framework previously defined in (ITU-T Recommendation X.509, 1997) by the same standardization body. In fact, a PKI and a PMI can be bound by the information contained in the identity and attribute certificates of any user, by assigning to the field *holder* (in the attribute one) the issuer and serial number contained in the fields of the user's identity certificate, as shown (and marked) in the corresponding ASN.1 specifications.

```
Certificate ::= SIGNED { SEQUENCE {
    version                 [0] Version DEFAULT v1,
    serialNumber            CertificateSerialNumber,
    signature               AlgorithmIdentifier,
    issuer                  Name,
    validity                Validity,
    subject                 Name,
    subjectPublicKeyInfo    SubjectPublicKeyInfo,
    issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier OPTIONAL,
    subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier OPTIONAL,
    extensions              [3] Extensions OPTIONAL
} }

AttributeCertificateInfo ::= SIGNED { SEQUENCE {
    version                 AttCertVersion,
    holder                  Holder,
    issuer                  AttCertIssuer,
    signature               AlgorithmIdentifier,
    serialNumber            CertificateSerialNumber,
    attrCertValidityPeriod  AttCertValidityPeriod,
    attributes              SEQUENCE OF Attribute,
    issuerUniqueID          UniqueIdentifier OPTIONAL,
    extensions              Extensions  OPTIONAL
} }

Holder  ::= SEQUENCE {
    baseCertificateID   [0] IssuerSerial     OPTIONAL,
        -- the issuer and serial number of the holder's Public Key Certificate
    entityName          [1] GeneralNames     OPTIONAL,
    objectDigestInfo    [2] ObjectDigestInfo OPTIONAL
}
```

All the possibilities for the binding can be concluded from the specification of the field *holder* in the attribute certificate. Because of its relevance for the solution that we provide in subsequent sections, it is important to specially consider the third of the possibilities, that is, to bind the attribute certificate to any object by using the hash value of that object. For instance, the hash value of the public key, or the hash value of the identity certificate itself, can be used.

Additionally, the last Recommendation introduces a new type of authority for the assignment of privileges, the *Attribute Authority* (AA), while a special type of authority, the *Source of Authority* (SOA), is settled as the root of delegation chains. The Recommendation defines a framework that provides a foundation upon which a PMI is built to contain a multiplicity of AAs and final users. That framework will be of help for our contribution, as we will show later.

3 Structuring an Anonymous Attribute Certificate

In this section we define the structure of a new type of anonymous attribute certificate, in this case based on traceable signatures. This data structure will allow the user to show to

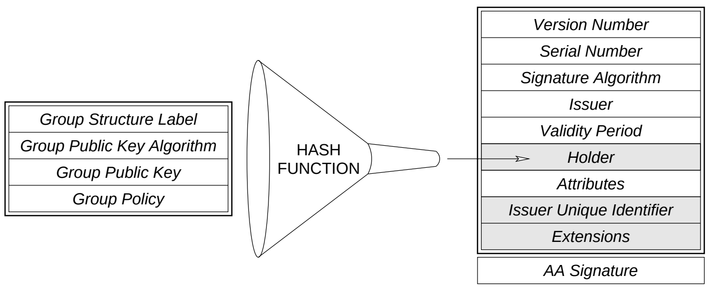


Figure 1: Anonymous Attribute Certificate

others that she holds certain attribute or privilege, but at the same time will not need to reveal her identity. However, whenever the user misuses that certificate and breaks the rules of a pre-specified policy of use, her real identity will become public.

We define the new type of anonymous attribute certificate (figure 1) as the composition of two certificates: a *group certificate* and a X.509 attribute certificate. Both are linked through the *holder* of the latter. Note, as pointed out in section 2.2, that the *holder* of the attribute certificate can contain the digest of any object, making possible to link both structures.

The group certificate structure contains all the information needed to check if a user belongs to the group of members owning a specific attribute for a period of time.

- *Group Structure Label:* A static field that allows to interpret the object as a proper group structure.
- *Group Public Key Algorithm:* Identifies the algorithm that will be used to check the membership to a group (by using the public key from the following field). In this work, we use the traceable signature scheme, previously introduced in 2.1.
- *Group Public Key:* The group manager public key created in the TS_Setup algorithm during the execution of the Group Manager Setup (sec. 5.3). It allows to any entity to check if a user belongs to the specified group.
- *Group Policy:* A reference to the policy that rules the requirements for joining the group, the disclosure of the user's identity and the revocation of certificates.

The AA creates and manages a group for each kind of anonymous attribute certificate that issues. Such group and certificate are valid for a specified period of time. If a user fulfills the requirements to enjoy the privileges associated with an anonymous attribute certificate, she joins to the corresponding group and gets the certificate and a unique private key that enables her to prove her membership to the group.

On the other side, the AA gets enough information to identify which member issued a given membership proof, or to trace all the membership proofs issued by a given member. However, the AA is considered a trusted party in the system and will only disclose restricted information if the policy governing the group is broken.

Note that the anonymous attribute certificate is the same for all members of the group, and that the membership proof is unlinkable with the member that issued it and with any other membership proof. These facts guarantee that any user that makes use of any anonymous attribute certificate to enforce her privileges remains anonymous, and furthermore, each anonymous use remains unlinkable with any other one. Therefore, it becomes impossible to create anonymous user profiles, something that could be done with our previous approach (Benjumea, 2004).

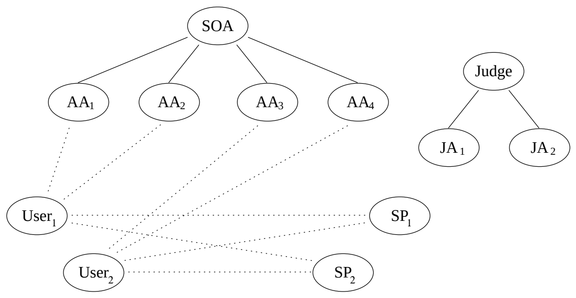


Figure 2: System Overview

4 Using the Anonymous Attribute Certificates: Overview of the System

Our scheme coexists with a standard PMI. The AA issues certificates about attributes that the users hold. Additionally, we suppose that some organizations (SP) provide services to users based on users' attributes. A number of AAs will have the special capacity to issue anonymous attribute certificates, which state that their anonymous owners hold such attribute. These AAs will be considered as trusted parties with respect to the anonymity of the attribute certificates that they issue (see figure 2).

The role that the different actors play in our solution can be roughly seen as follows. For each anonymous attribute certificate that a user wants to get, she collects all proofs needed to apply for a specific attribute certificate and sends the proofs, together with her identity, to the AA responsible of issuing that attribute certificate. If the set of proofs is complete, the anonymous attribute certificate is sent together with a unique private membership key that allows the user to anonymously prove that owns the certificate. As result, the user is joined to the group stated in the anonymous attribute certificate.

As the certificate is anonymous, it is not linked to any PKI. However, it contains a group specification (group public key and policy), and the users that can prove that belong to that group are considered to own such attribute. If the proof can be verified with the group public key then the user is considered to be a member of the group and, therefore, to own the anonymous attribute certificate.

The user anonymously makes use of the attribute certificate in order to enforce her privileges. Therefore, she presents her attribute certificate to a SP and anonymously proves that she is the owner (belongs to the group stated in the certificate), being granted for the service. The SP keeps a record with each presented proof linked with the service request.

Note again that the only information that can be extracted from the proof is that the user who issued it is a member of the group, but can not be linked with the originating member. Moreover, the proof can not be linked with any other proof, regardless they are performed by the same member or by different ones.

However, the AA, which is considered a trusted party, knows restricted information, and is able to identify which member of the group issued a particular proof, or is even able to identify, from a set of proofs, which ones were issued by a given member, with no need to disclose any other kind of information. These features can be used to disclose the identity of an anonymous user that breaks the group policy, to trace all the anonymous activities that a suspicion user has performed, or even to revoke an anonymous attribute certificate for a given user.

Nomenclature	Meaning
$A : act$	$A's$ action act
$A \rightarrow B : m$	m is sent from A to B
$m := (m_1, m_2)$	m is composed by m_1 and m_2
$\langle a, b, \cdots \rangle$	a tuple of objects
$c := E_z(m)$	m is encrypted with the symmetric key z
$m := D_z(c)$	c is decrypted with the symmetric key z
A_{publ}, A_{priv}	$A's$ asymmetric public and private keys
$c := E_A(m)$	m is encrypted with $A's$ asymmetric public key
$m := D_A(c)$	c is decrypted with $A's$ asymmetric private key
$h := H(m)$	$m's$ one way hash function
$s_m := \mathbb{S}_A(m)$	$m's$ message signature with $A's$ asymmetric private key $[\mathbb{S}_A(m) \Leftrightarrow E_{A_{priv}}(H(m))]$
$m_s := S_A(m)$	Signed message composed by the message m and its signature with $A's$ asymmetric private key $[S_A(m) \Leftrightarrow (m, \mathbb{S}_A(m))]$
$b := V_A^?(m_s)$	Verify the signed message m_s with $A's$ asymmetric public key $[V_A^?(m_s) \Leftrightarrow \left(H(m') \overset{?}{=} D_{A_{publ}}(\mathbb{S}_A(m))\right)]$ / $[m_s \equiv (m', \mathbb{S}_A(m))]$
$z := NSK()$	Create new symmetric key z
$A := NAK()$	Create new asymmetric key pair for A

Table 1: Cryptographic protocol nomenclature

5 Specification of Protocols for the use of Anonymous Attribute Certificates

This section explains the protocol to obtain an anonymous attribute certificate, how it can be used, and how the user's identity can be disclosed, traced or revoked. This protocol uses as fundamental construction block: the traceable signature scheme presented earlier in section 2.1.

In addition to the aforementioned notation for traceable signatures, the nomenclature used for the protocols is listed in Table 1.

5.1 Actors

SOA is the Source of Authority. Its public key (SOA_{publ}) is known by every actor in the system. It enables the AA to issue attribute certificates.

AA is the attribute authority that issues anonymous attribute certificates. It is considered as a trusted party regarding the identity of anonymous users of the system, although it has the capacity of disclosing such identity under certain circumstances.

U is the user of the system. She owns attribute certificates and anonymous attribute certificates, and uses them in order to enforce her privileges. Every user that owns an anonymous attribute certificate belongs to a group managed by the AA in charge of issuing such certificates. She makes use of such attribute by proving anonymously her membership to the group.

SP is a Service Provider that offers services to those anonymous users that have the appropriate attribute certificate(s).

J is the Judge, who can request the AA to disclose the identity of any user involved in any particular transaction. Moreover, he can ask the involved parties (AA and SP) to provide the information needed to trace all the activities performed by any given user in the whole system. He can order the AA to revoke the anonymity of a given user.

JA^j represents any of the Judge Agents ($j \in \{1, 2, 3, \cdots\}$), in charge of tracing the activities of any given user when the Judge orders it.

AACRL is the entity that manages the *Anonymous Attribute Certificate Revocation List* database (AACRL_DB), and holds the information needed to know if any anonymous attribute certificate has been revoked.

Group_Member_DB is a database where the AA stores information about the groups associated with the anonymous attribute certificates and their members. This information enables the AA to disclose the identity of any member that breaks the policy. Moreover, it provides the necessary information to trace tha activities of any member.

Member_Proof_DB is a database where the SP stores the membership proofs that anonymous users have provided to access the resources offered. It enables any entity with enough information to trace the anonymous activities performed by users under suspicion.

msg refers, throughout the protocols, to the last message received.

Attribute^i represents a specific attribute within the whole set of attributes in the system ($i \in \{1, 2, 3, \cdots\}$). Throughout the protocol, any object with an "i" superscripted represents an instance of the object for the specific attribute "i".

5.2 Attribute Authority General Setup

The AA presents the necessary proofs and requests the SOA the privileges to issue attribute certificates for a certain attribute ($Attribute^i$). The SOA, if the requirements are met, issues a certificate that enables the AA to do it. The policy that the AA will follow to issue certificates and to disclose/revoke identities is also published in this phase. See section 2.2 for an explanation of the fields in the Attr_Cert structure.

1. $AA : AA := NAK()$
2. $AA : STORE\,(AA_{priv})$
3. $AA \rightarrow SOA : E_{SOA}\left(S_{AA}\left(aa\_request, proofs, Attribute^i, AA_{publ}, Policy\right)\right)$
4. $SOA :$ `IF` $\left(\neg V^?_{AA}(msg) \;\vee\; \neg fulfill\_req\,(msg)\right)$ `THEN Abort`
5. $SOA : Attr\_Cert := S_{SOA}\left(Vers, Serial, Sig\_Alg, SOA, Val\_Period, AA, Attribute\_Authority^i \;\&\; Policy\right)$
6. $SOA \rightarrow AA : E_{AA}\,(Attr\_Cert)$
7. $AA :$ `IF` $\neg V^?_{SOA}\,(Attr\_Cert)$ `THEN Abort`
8. $AA : AA_{cert} := Attr\_Cert$
9. $AA : PUBLISH\,(AA_{cert})$

5.3 Group Manager Setup

The AA acts as a group manager for each type of anonymous attribute certificate that issues for each period of time. All users that own an anonymous attribute certificate will be members of the related group. The AA creates a new group and the related attribute certificate by running the following algorithm. The TS_Setup algorithm produces the group public key and the group private key. The public key is stored in the created anonymous attribute certificate, while the private key is stored to be used by the AA whenever needed. See sections 2.2 and 3 for a detailed explanation of structures Group_Struct and Attr_Cert.

1. $AA : \langle GM_{publ}, GM_{priv} \rangle := \mathbb{TS\_SETUP}\,(security\_level)$
2. $AA : Group\_Struct^i := \left(GS\_Label, GPK\_Alg, GM_{publ}, Policy\right)$
3. $AA : Attr\_Cert^i := S_{AA}\left(Vers, Serial, Sig\_Alg, AA, Val\_Period, H\left(Group\_Struct^i\right), Attribute^i\right)$
4. $AA : Anon\_Attr\_Cert^i := \left(Group\_Struct^i, Attr\_Cert^i\right)$
5. $AA : STORE\left(\langle Anon\_Attr\_Cert^i, GM_{priv} \rangle\right)$

5.4 Obtaining the Certificate

Whenever a user wants to get an anonymous attribute certificate, she presents the necessary proofs $(attr\_proofs^i_U)$ together with her identity to AA. If the requirements are fulfilled, the certificate issued in the "group manager setup" phase (section 5.3) is sent to the user and the TS_Join protocol is run. As a result, the user gets a unique private membership key $(member\_key_{priv})$ and the AA gets some information useful for revoking and tracing purposes $(member\_ref_{priv})$. A rough scheme is depicted in figure 3.

The AA stores, for each user, the revoking information together with her identity. The private membership key enables the user to anonymously prove that she belongs to the group.

Note: In the TS_Join protocol run, all messages interchanged will be encrypted with the recipient's public key.

1. $U \rightarrow AA : E_{AA}\left(S_U\left(attr\_req^i, user\_identity, attr\_proofs^i_U\right)\right)$
2. $AA :$ `IF` $\left(\neg V^?_U\,(msg) \;\vee\; \neg fulfill\_req\left(attr\_proofs^i_U\right)\right)$ `THEN Abort`
3. $AA \rightarrow U : E_U\left(Anon\_Attr\_Cert^i\right)$
4. $U :$ `IF` $\Big(\neg V^?_{AA}\left(Attr\_Cert^i\right)$
   $\vee\; H\left(Group\_Struct^i\right) \overset{?}{\neq} Holder\_Field\left(Attr\_Cert^i\right)\Big)$ `THEN Abort`
5. $U \;\&\; AA : \mathbb{TS\_JOIN}\,()$
   - $U : member\_key_{priv} := \mathbb{TS\_JOIN}_U\left(GM_{publ}\right)$
     $\&\; STORE\left(\langle Anon\_Attr\_Cert^i, member\_key_{priv} \rangle\right)$
   - $AA : member\_ref_{priv} := \mathbb{TS\_JOIN}_{GM}\left(GM_{publ}, GM_{priv}\right)$
     $\&\; STORE\_IN\left(group\_member\_db, \langle user\_identity, Anon\_Attr\_Cert^i, member\_ref_{priv} \rangle\right)$

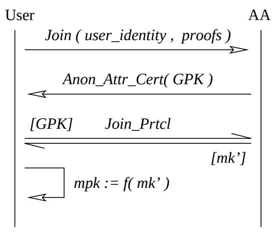


Figure 3: Obtaining a certificate

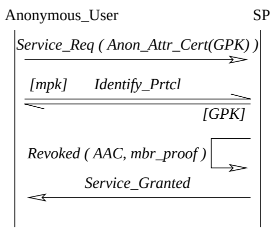

Anonymous_User
SP
Service_Req (Anon_Attr_Cert(GPK))
[mpk]
Identify_Prtcl
[GPK]
Revoked (AAC, mbr_proof)
Service_Granted

Figure 4: Using the certificate

5.5 Using the Certificate

The user makes use of her anonymous attribute certificate by showing it and, additionally, running the TS_Identify protocol to anonymously prove (by using $member\_key_{priv}$) that belongs to the group of users owning that attribute. It is very important to note that it is impossible to link several membership proofs, even if executed by the same user. They are efficient zero knowledge proofs. A rough scheme is depicted in figure 4.

Before granting the access to service, in step 5 of the protocol SP asks the AACRL (see sec. 5.9) if the membership proof corresponds with any member whose anonymous attribute certificate has been revoked (see sec. 5.8). The SP stores a transcript of the membership proof, thus the Judge will be able to disclose the identity of the anonymous user (see sec. 5.6) or even to trace all transactions achieved by users under suspicion (see sec. 5.7). The AA is able to provide information that allows to link every membership proof with the user that performed it, or to identity if a given membership proof has been performed by a specific member, thus providing the system with fair anonymity or conditionally traceable anonymity.

Note: In the TS_Identify protocol run, all messages interchanged will be encrypted with a symmetric session key (ssk).

1. $U : ssk := NSK()$
2. $U \rightarrow SP : E_{SP}\left(serv\_req, Anon\_Attr\_Cert^i, ssk\right)$
3. $SP :$ `IF` $\Big(\neg V^?_{AA}\left(Attr\_Cert^i\right) \;\vee\; H\left(Group\_Struct^i\right) \overset{?}{\neq} Holder\_Field\left(Attr\_Cert^i\right)$
   $\vee\; \neg fulfill\_req\left(Anon\_Attr\_Cert^i\right)\Big)$ `THEN Abort`
4. U & $SP : \mathbb{TS\_IDENTIFY}\,(ssk)$
   - $U : \mathbb{TS\_IDENTIFY}_U\left(GM_{publ}, member\_key_{priv}\right)$
   - $SP : \langle ok, member\_proof\rangle := \mathbb{TS\_IDENTIFY}_{SP}\left(GM_{publ}\right)$
     $\&\; STORE\_IN\left(member\_proof\_db, \langle Anon\_Attr\_Cert^i, member\_proof, timestamp, trans\_id\rangle\right)$
5. $SP :$ `IF` $\left(\neg\; ok \;\vee\; revoked\left(Anon\_Attr\_Cert^i, member\_proof\right)\right)$ `THEN Abort`
6. $SP \rightarrow U : E_{ssk}\left(S_{SP}\left(Service\_Granted, timestamp, trans\_id\right)\right)$
7. $U :$ `IF` $\neg V^?_{SP}\,(msg)$ `THEN Abort`

5.6 Disclosing the Identity of the User

If the service provider considers that the anonymous user has broken the policy, can request the Judge to disclose the user's identity to be prosecuted. If the proofs presented

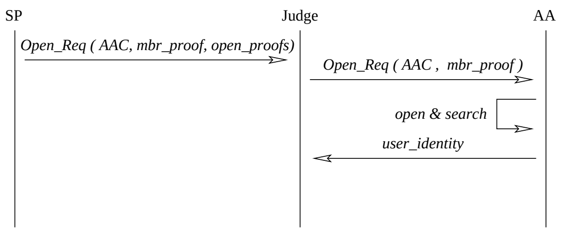

SP
Judge
AA
Open_Req (AAC, mbr_proof, open_proofs)
Open_Req (AAC , mbr_proof)
open & search
user_identity

Figure 5: Disclosing user's identity

(*opening_proofs*) are enough, the Judge requests the AA to provide the identity of the member that issued the given membership proof (*member_proof*). The AA uses the TS_Open to get reference to the member involved (*member_ref*). This reference is used to search the database and get the user's identity. A rough scheme is depicted in figure 5.

Note: *opening_proofs* can be whatever is stated in the AA policy.

1. $SP \rightarrow J : E_J \left(S_{SP} \left(open\_req, Anon\_Attr\_Cert^i, member\_proof, opening\_proofs\right)\right)$
2. $J :$ `IF` $\left(\neg V^?_{SP} \left(msg\right) \ \vee \ \neg fulfill\_req \left(opening\_proofs\right)\right)$ `THEN Abort`
3. $J \rightarrow AA : E_{AA} \left(S_J \left(open\_req, Anon\_Attr\_Cert^i, member\_proof, opening\_proofs\right)\right)$
4. $AA :$ `IF` $\left(\neg V^?_J \left(msg\right) \ \vee \ \neg fulfill\_req \left(opening\_proofs\right)\right)$ `THEN Abort`
5. $AA : member\_ref := \mathbb{TS\_OPEN}_{GM} \left(GM_{publ}, GM_{priv}, member\_proof\right)$
6. $AA : user\_id := SEARCH\_IN \left(group\_member\_db, Anon\_Attr\_Cert^i, member\_ref\right)$
7. $AA \rightarrow J : E_J \left(S_{AA} \left(Anon\_Attr\_Cert^i, member\_proof, user\_id\right)\right)$
8. $J :$ `IF` $\neg V^?_{AA} \left(msg\right)$ `THEN Abort`

5.7 Tracing the Anonymous Transactions of a Specific User

If a user is suspicious, the Judge can decide to trace all her anonymous activities, and requests to the AA to provide the member trapdoor (by means of the TS_Reveal primitive) that allows the Judge agents to identify (by means of the TS_Trace) which membership proofs were performed by the user under suspicion. Note that it is not necessary to disclose the identities of the members that performed the transactions (that would violate their rights, since they are not under suspicion). Only the transactions performed by the suspicious user are identified and the rest remain anonymous. A rough scheme is depicted in figure 6.

Note that *tracing_proofs* can be a Judge resolution, a collection of facts achieved by the user, or whatever thing that is stated in the AA policy.

It is important to note that even the Judge agents do not guess anything about the identity of the subject under study; they only collect her transactions.

1. $J \rightarrow AA : E_{AA} \left(S_J \left(trace\_req, user\_identity, Anon\_Attr\_Cert^i, tracing\_proofs\right)\right)$
2. $AA :$ `IF` $\left(\neg V^?_J \left(msg\right) \ \vee \ \neg fulfill\_req \left(tracing\_proofs\right)\right)$ `THEN Abort`
3. $AA : member\_ref := SEARCH\_IN \left(group\_member\_db, Anon\_Attr\_Cert^i, user\_identity\right)$
4. $AA : member\_trapdoor_{priv} := \mathbb{TS\_REVEAL}_{GM} \left(GM_{publ}, GM_{priv}, member\_ref\right)$
5. $AA \rightarrow J : E_J \left(S_{AA} \left(user\_identity, Anon\_Attr\_Cert^i, member\_trapdoor_{priv}\right)\right)$

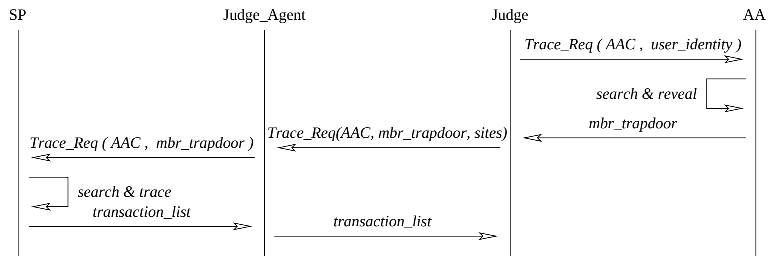

SP
Judge_Agent
Judge
AA
Trace_Req (AAC , user_identity)
search & reveal
mbr_trapdoor
Trace_Req(AAC, mbr_trapdoor, sites)
Trace_Req (AAC , mbr_trapdoor)
search & trace
transaction_list
transaction_list

Figure 6: Tracing anonymous transactions

6. J : IF $\neg V^{?}_{AA}\,(msg)$ THEN Abort
7. $J \rightarrow JA^j : E_{JA^j}\left(S_J\left(trace\_req, sites, Anon\_Attr\_Cert^i, member\_trapdoor_{priv}\right)\right)$
8. JA^j : IF $\neg V^{?}_{J}\,(msg)$ THEN Abort
9. JA^j :

```
FOREACH site IN sites DO
    FOREACH $proof \in Anon\_Attr\_Cert^i$ IN member_proof_db DO
        $ok := \mathbb{TS\_TRACE}\left(member\_trapdoor_{priv},\ proof\right)$
        IF ok THEN
            APPEND_TO($trans\_list$, $\langle proof, timestamp, trans\_id\rangle$)
        ENDIF
    ENDFOR
ENDFOR
```

10. $JA^j \rightarrow J : E_J\left(S_{JA^j}\,(trans\_list)\right)$

5.8 Revoking a Certificate

It is possible to revoke an anonymous attribute certificate for a given user in such a way that when the user uses such anonymous certificate, this will be detected (see section 5.9 and step 5 in section 5.5). A rough scheme is depicted in figure 7.

1. $J \rightarrow AA : E_{AA}\left(S_J\left(revoke\_req, revoking\_proofs, user\_identity, Anon\_Attr\_Cert^i\right)\right)$
2. AA : IF $\left(\neg V^{?}_{J}\,(msg) \ \vee\ \neg fulfill\_req\,(revoking\_proofs)\right)$ THEN Abort
3. $AA : member\_ref := SEARCH\_IN\,(group\_member\_db, user\_identity)$
4. $AA : member\_trapdoor_{priv} := \mathbb{TS\_REVEAL}_{GM}\left(GM_{publ}, GM_{priv}, member\_ref\right)$
5. $AA \rightarrow J : E_J\left(S_{AA}\left(user\_identity, member\_trapdoor_{priv}\right)\right)$
6. J : IF $\neg V^{?}_{AA}\,(msg)$ THEN Abort
7. $J \rightarrow AACRL : E_{AACRL}\left(S_J\left(revoke\_req, Anon\_Attr\_Cert^i, member\_trapdoor_{priv}\right)\right)$
8. $AACRL$: IF $\neg V^{?}_{J}\,(msg)$ THEN Abort
9. $AACRL : STORE\_IN\left(AACRL\_DB, \langle timestamp, Anon\_Attr\_Cert^i, member\_trapdoor_{priv}\rangle\right)$

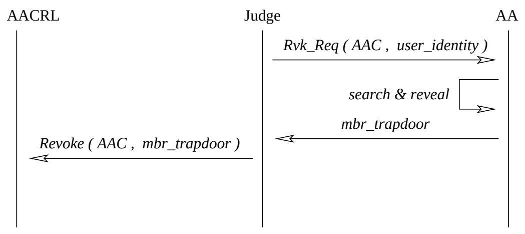

AACRL
Judge
AA
Rvk_Req (AAC , user_identity)
search & reveal
mbr_trapdoor
Revoke (AAC , mbr_trapdoor)

Figure 7: Revoking a certificate

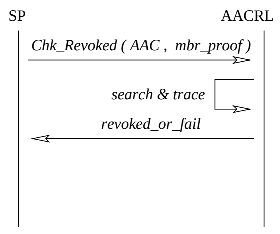


Figure 8: Checking revocation status

5.9 Checking of Revocation Status

Whenever an anonymous user presents an anonymous attribute certificate (sec. 5.5), SP checks if such certificate has been revoked for the actual anonymous user (see step 5 in section 5.5). A rough scheme is depicted in figure 8. The following protocol is played between SP and AACRL to do the checking.

1. $SP \rightarrow AACRL : E_{AACRL}\left(S_{SP}\left(\mathit{check\_revoke\_req}, \mathit{Anon\_Attr\_Cert}^{i}, \mathit{member\_proof}\right)\right)$
2. $AACRL :$ `IF` $\left(\neg V^{?}_{SP}(msg) \;\vee\; \neg \mathit{fulfill\_req}(msg)\right)$ `THEN Abort`
3. $AACRL :$ $\mathit{revoked} \;=\; \mathit{null}$
```
FOREACH $\mathit{member\_trapdoor} \;\in\; \mathit{Anon\_Attr\_Cert}^{i}$ IN $\mathit{AACRL\_DB}$ DO
    $ok := \mathbb{TS\_TRACE}\left(\mathit{member\_trapdoor}_{priv},\; \mathit{member\_proof}\right)$
    IF ok THEN
        $\mathit{revoked} \;=\; \mathit{timestamp}$
    ENDIF
ENDFOR
```
4. $AACRL \rightarrow SP : E_{SP}\left(S_{AACRL}(\mathit{revoked})\right)$
5. $SP :$ `IF` $\neg V^{?}_{AACRL}(msg)$ `THEN Abort`
6. $SP :$ `IF` $\mathit{revoked}$ `THEN` $\mathit{UPDATE\_IN}\left(\mathit{member\_proof\_db}, \langle \mathit{Anon\_Attr\_Cert}^{i}, \mathit{member\_proof}, \mathit{revoked}\rangle\right)$

5.10 Claiming Authorship of Use

It is possible for a user to claim that she used a specific anonymous attribute certificate to access to certain service. So, she identifies the transaction performed $(\mathit{timestamp}, \mathit{trans\_id})$ and then both play the TS_Claim protocol to prove if such transaction was peformed by the claiming user.

1. $U \rightarrow SP : E_{SP}\left(S_{U}\left(\mathit{claim\_req}, \mathit{user\_identity}, S_{SP}\left(\mathit{Service\_Granted}, \mathit{timestamp}, \mathit{trans\_id}\right)\right)\right)$
2. $SP : \langle \mathit{member\_proof}, \mathit{Anon\_Attr\_Cert}^{i}\rangle := \mathit{SEARCH\_IN}\left(\mathit{member\_proof\_db}, \mathit{timestamp}, \mathit{trans\_id}\right)$
3. $SP :$ `IF` $\mathit{member\_proof}\ \mathit{is\ revoked}$ `THEN Abort`
4. $U\ \&\ SP : \mathbb{TS\_CLAIM}\,()$
   - $U : \mathbb{TS\_CLAIM}_{U}\left(GM_{publ}, \mathit{member\_key}_{priv}\right)$
   - $SP : ok := \mathbb{TS\_CLAIM}_{SP}\left(GM_{publ}, \mathit{member\_proof}\right)$
5. $SP :$ `IF` $(\neg\ ok)$ `THEN Abort`
6. $SP \rightarrow U : E_{U}\left(S_{SP}\left(\mathit{Claim\_OK}, \mathit{timestamp}, \mathit{trans\_id}, \mathit{user\_identity}\right)\right)$
7. $U :$ `IF` $\neg V^{?}_{SP}(msg)$ `THEN Abort`

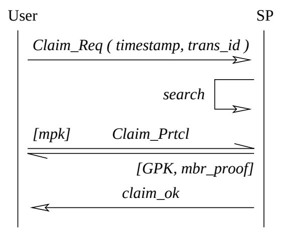


Figure 9: Claiming authorship

6 Considerations Regarding Real World Applications

6.1 Splitting the Functions of the Attribute Authority

As we have seen in previous sections, in our system the AAs are responsible for issuing the anonymous attribute certificates, and they are able to disclose the users' identities under certain circumstances. They are considered trusted parties. In the real world, private companies, such as banks, trade centers, etc., can have the role of AAs. However, they are in a business world where users' identities can result a very profitable information for cross-reference profiles. Hence, their fairness may become suspicious.

In these cases, the Attribute Authority could be divided into two entities. One of them responsible for checking if the user fulfills the requirements and the issuing of anonymous attribute certificates. The other entity should keep private information that makes compulsory its involvement in the disclosing, tracing or revoking operations aforementioned, in such a way that if such entity does not collaborate, then those operations can not be performed. This entity acts as a light weight trusted third party that can be managed by the Justice department. Only very light involvement is required because these infrequent operations.

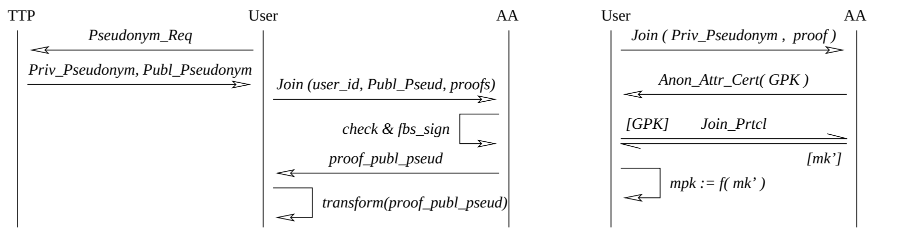
As result of this division, a private business company can still issue anonymous attribute certificates, but the power to revoke the anonymity resides in a trusted third party managed by the Justice department. This division can be incorporated to the proposed system following several ways. One of them could be adding a threshold scheme to the underlying traceable signature primitives, as in (Nguyen, 2004), where the private key of a trusted third party is required for the open primitive (however, surprisingly, it is not required for the trace primitive).

Another approach could be to use a "fair blind signature" scheme (Stadler, 1995) as in (Benjumea, 2004) where the trusted third party manages pseudonyms, which are composed of two parts: one public part that is related with the identity of a user, and a private part that is related with the member of the group. Only the trusted third party knows the relationship between the two parts.

Then the acquisition of an anonymous attribute certificate by a user roughly involves the following steps (see also figure 10):

6.1.1 Phase I

1. $U \rightarrow TTP : E_{TTP}\,(pseud\_req, ssk)$
2. $TTP : \langle Pseudonym_{priv}, Pseudonym_{publ}\rangle := Create\_Pseudonym()$
3. $TTP : STORE\_IN\,\big(Pseudonym\_DB, \langle Pseudonym_{priv}, Pseudonym_{publ}\rangle\big)$
4. $TTP \rightarrow U : E_{ssk}\,\big(S_{TTP}\,(Pseudonym_{priv}, Pseudonym_{publ})\big)$


TTP
User
AA
Pseudonym_Req
Priv_Pseudonym, Publ_Pseudonym
Join (user_id, Publ_Pseud, proofs)
check & fbs_sign
proof_publ_pseud
transform(proof_publ_pseud)
User
AA
Join (Priv_Pseudonym , proof)
Anon_Attr_Cert(GPK)
[GPK]
Join_Prtcl
[mk']
mpk := f(mk')

(a) Phase I (b) Phase II

Figure 10: Obtaining a certificate

5. $U : N := NAK()$
6. $U \rightarrow AA : E_{AA}\left(S_U\left(attr\_req^i, user\_identity, attr\_proofs_U^i, \mathsf{FBS\_Blind}\left(Pseudonym_{publ}, N_{publ}\right)\right)\right)$
7. $AA : \texttt{IF}\left(\neg V_U^?\left(msg\right) \vee \neg fulfill\_req\left(attr\_proofs_U^i\right)\right) \texttt{ THEN Abort}$
8. $AA : fair\_blind\_signature := \mathsf{FBS\_Sign}\left(AA_{priv}, \mathsf{FBS\_Blind}\left(Pseudonym_{publ}, N_{publ}\right)\right)$
9. $AA : STORE\_IN\left(Identity\_DB, \langle user\_identity, Pseudonym_{publ}\rangle\right)$
10. $AA \rightarrow U : E_U\left(S_{AA}\left(fair\_blind\_signature\right)\right)$
11. $U : S_{AA}\left(Pseudonym_{priv}, N_{publ}\right) := \mathsf{FBS\_Transform}\left(fair\_blind\_signature\right)$
12. $U : \cdots\cdots\cdots Sleep\ for\ decoupling\ both\ phases \cdots\cdots\cdots$

6.1.2 Phase II

1. $U \rightarrow AA : E_{AA}\left(S_N\left(attr\_req^i, S_{AA}\left(Pseudonym_{priv}, N_{publ}\right)\right)\right)$
2. $AA : \texttt{IF}\left(\neg V_N^?\left(msg\right) \vee \neg V_{AA}^?\left(S_{AA}\left(Pseudonym_{priv}, N_{publ}\right)\right)\right) \texttt{ THEN Abort}$
3. $AA \rightarrow U : E_U\left(Anon\_Attr\_Cert^i\right)$
4. $U : \texttt{IF}\Big(\neg V_{AA}^?\left(Attr\_Cert^i\right)$
   $\vee\ H\left(Group\_Struct^i\right) \overset{?}{\neq} Holder\_Field\left(Attr\_Cert^i\right)\Big) \texttt{ THEN Abort}$
5. $U\ \&\ AA : \mathbb{TS\_JOIN}()$
   - $U : member\_key_{priv} := \mathbb{TS\_JOIN}_U\left(GM_{publ}\right)$
     $\&\ STORE\left(\langle Anon\_Attr\_Cert^i, member\_key_{priv}\rangle\right)$
   - $AA : member\_ref_{priv} := \mathbb{TS\_JOIN}_{GM}\left(GM_{publ}, GM_{priv}\right)$
     $\&\ STORE\_IN\left(group\_member\_db, \langle Pseudonym_{priv}, Anon\_Attr\_Cert^i, member\_ref_{priv}\rangle\right)$

Note that AA is unable to link N_{publ} with the user identity, since in the first phase N_{publ} is blinded, and in the second phase neither the user identity nor $Pseudonym_{publ}$ appear. The FBS_Transform primitive in step 11 of the first phase converts a signature on the public part of the pseudonym of a blind N_{publ} in a signature on the private part of the pseudonym of a clear N_{publ}. Please, also note that in step 2 of the second phase, if $S_{AA}\left(Pseudonym_{priv}, N_{publ}\right)$ can be verified against AA_{publ}, it means that the anonymous user who knows N_{priv} fulfills the requirements to get the certificate, since such proofs were verified in step 7 of the first phase. Also, if the message received can be verified against N_{publ} in step 2 of the second phase, that means that the peer anonymous user is really one of those that fulfill the requirements.

As consequence of this new protocol, the member of the group is linked with the private part of a pseudonym, the public part is linked with the user's identity, and no one, except the TTP is able to relate both parts of the pseudonym. Therefore, the TTP will always be required for the disclosing, tracing or revoking operations, and with very little involvement guarantees that such operations are performed under the law requirements.

6.2 Constraining the Use of Certificates

Public key identity systems define the identity of a user and link such identity with the knowledge of a private key that enables her to prove that is really the person that is stated in the identity being checked. The security of the above scheme is based on the fact that the private key is only known by the person that owns the defined identity. Normally, in these systems, an identity brings about some privileges that the user may enjoy.

A sometimes underestimate problem in public key based systems is the fact that a user can share her knowledge of private keys with someone else, thus making possible to forge the identity of another user and enjoy in this case the privileges of the forged user.

In anonymous systems based on public key cryptography, the aforementioned problem is magnified since, in addition to the privilege forgery mentioned above, a collusion of anonymous users may cooperate. They may share the knowledge of private keys and create a "virtual super user" that is able to enjoy the privileges that no one, by herself, is able to enjoy.

How to avoid this problem in a general and feasible way still remains open.

7 Conclusions

We have presented an new approach to extend X.509 attribute certificates with anonymity capabilities, as well as a set of protocols to obtain and use certificates preserving user's identity by using a group signature scheme. We have made use of traceable signatures, that provide our system with a very suitable and powerful toolset for conditionally removing the user's anonymity in many ways, such as disclosure of the user's identity that made use of a certificate, and tracing of all certificate uses carried out by any given user without disclosing any other information about the system. A given user can claim her authorship in the use of an anonymous attribute certificate. It is also possible to revoke attribute certificates based on Anonymous Attribute Certificate Revocation Lists.

A very important feature of our solution is that any use of an anonymous certificate is completely unlinkable with any other use, even if performed by the same user under the same circumstances, what makes completely impossible the creation of anonymous user profiles by the entities after transaction cross-reference.

We have also presented some considerations regarding the deployment of the system in real world applications and pointed out some future work, like how to avoid that a user can make use of a certificate that does not belong to her.

References

Ateniense, G., Camenish, J., Joye, M. and Tsudik, G. (2000). A practical and provably secure coalition-resistant group signature scheme, *CRYPTO 2000: Advances in Cryptology*, Vol. 1880 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 255–270.

Benjumea, V., López, J., Montenegro, J. A. and Troya, J. M. (2004). A first approach to provide anonymity in attribute certificates, *in* F. Bao, R. H. Deng and J. Zhou (eds), *PKC 2004, 2004 International Workshop on Practice and Theory in Public Key Cryptography*, Vol. 2947 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 402–415.

Chaum, D. (1983). Blind signatures for untraceable payments, *in* D. Chaum, R. Rivest and A. Sherman (eds), *Advances in Cryptology–Crypto'82*, Plenum Press, Santa Barbara, CA USA, pp. 199–203.

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

Chaum, D. and Heyst, E. v. (1991). Group signatures, *in* D. W. Davies (ed.), *Advances in Cryptology - EuroCrypt '91*, Springer-Verlag, Berlin, pp. 257–265. Lecture Notes in Computer Science Volume 547.

ITU-T Recommendation X.509 (1997). Information Technology - Open systems interconnection - The Directory: Authentication Framework.

ITU-T Recommendation X.509 (2000). Information Technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks.

Kiayias, A., Tsiounis, Y. and Yung, M. (2004a). Traceable signatures, *EUROCRYPT 2004: Advances in Cryptology*, Vol. 3027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 571–589.

Kiayias, A., Tsiounis, Y. and Yung, M. (2004b). Traceable signatures, *IACR E-Print Cryptology Archive, Report 2004/007*, http://eprint.iacr.org/.

Kilian, J. and Petrank, E. (1998). Identity escrow, *CRYPTO 1998: Advances in Cryptology*, Vol. 1462 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 169–185.

Nguyen, L. and Safavi-Naini, R. (2004). Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings, *ASIACRYPT 2004: Advances in Cryptology*, Vol. 3329 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 372–386.

Rivest, R., Shamir, A. and Tauman, Y. (2001). How to leak a secret, *ASIACRYPT 2001: Advances in Cryptology*, Vol. 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 552–565.

Stadler, M. A., Piveteau, J. M. and Camenisch, J. L. (1995). Fair blind signatures, *in* L. C. Guillou and J. J. Quisquater (eds), *Advances in Cryptology–Eurocrypt'95*, Vol. 921 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 209–219.

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