k-anonymous Attribute-Based Access Control[⋆]

Anna Squicciarini[1], Alberto Trombetta[2], Abilasha Bhargav-Spantzel[1], and Elisa Bertino[1]

[1] Computer Science Department, Purdue University, West Lafayette IN, US
[2] Dipartimento di Informatica e Comunicazione, Universitá degli Studi dell'Insubria, Varese, Italy

Abstract. Access control in a distributed system can be achieved by requesting digital credentials of the entity wanting to access the system. Credentials contain attributes that attest information concerning a given subject. Because such information can be sensitive, uncontrolled disclosure of such sensitive attributes may result in privacy breaches. Previous research efforts have shown that, even if one discloses only non-sensitive attributes, these attributes can still be linked to specific individuals. In this work, we propose attribute-based authorizations to satisfy the k-*anonymity property*: the set of credentials submitted by a subject during an access control operation should be equal to at least k other such sets received by the counterpart during earlier access controls operations. We thus propose a protocol that ensures k-anonymity for users accessing services in a distributed setting. Our protocol has a number of important features. First, anonymity is user-centric, in that anonymity degree k is defined by the credential submitter itself. A credential submitter before submitting its set of credentials has the assurance that its set will be identical to at least k other sets already stored at the counterpart. This assurance is provided in a privacy-preserving fashion, so that no actual information is leaked about the two negotiating parties data used to build such knowledge. Second, we provide a cryptographic protocol ensuring that the credentials submitted by the submitter during different transactions cannot be linked to each other. Third, we ensure that the critical data exchanged during the protocol are valid.

1 Introduction

Access control systems based on credentials' exchange (such as trust negotiation systems [5, 7, 6]) and recent digital identity management systems [3] have the goal of replacing conventional authentication and access control approaches (typically based on login and password mechanisms) with the more general notion of attribute-based authentication and access control so to allow subjects outside a local security domain to securely access protected resources and services. A key component of such approaches is represented by the notion of *identity attributes* (attributes, for short), that encode properties about subjects that

⋆ This work is partially funded by IP IST-015964 Aeolus are relevant for authentication and access control. An important issue is related to the authentication of such attributes. An approach addressing such issue is based on the use of *digital credentials*, which can be considered the equivalent, in the digital world, of paper credentials. Credentials often contain multiple attributes, for example, the name and the birth date of an individual, and can be used to verify identification information, professional qualifications, association memberships, and so forth. Usually, credentials are digitally signed by an issuing authority. The use of public key encryption guarantees that these credentials are both unforgeable and verifiable. The other relevant component of any access control system is represented by policies, which are statements specifying under which conditions an access request can be satisfied. Such conditions are often expressed in terms of attributes and their corresponding values. When dealing with systems based on the use of identity attributes, privacy is a crucial issue. Attributes may encode sensitive personal information or information that when combined with other information available to the receiving party may lead to the disclosure of sensitive information. In addition, a user may want to carry out multiple negotiations without having them linked to her/him. A feature that should be provided in order to achieve the prevent from information leakage is anonymization of identity attributes.

Addressing this problem is **not** straightforward, as several solutions need to be combined in order to achieve an high assurance privacy. In this paper, we consider an approach which is based on solutions developed for dealing with anonymity in the context of database systems. The key idea underlying our approach is to require that the attributes submitted to satisfy an access policy be k*-anonymous* [2]. In our context such property requires that the set of attributes a subject is about to submit must be indistinguishable with respect to k other sets that the counterpart received in earlier negotiations. Achieving k-anonymity in the context of database systems is relatively easy. These systems are closed systems and thus the set of individuals to whom the data are related is known and the application domains well-understood. Suppression and generalization techniques are used to ensure that the database has at least k tuples that are indistinguishable from each other. In open distributed systems ensuring k-anonymity is a much harder problem. In such systems, the set of participating subjects are not known in advance and cannot be trusted. Thus, any information obtained from the counterpart must be verified to ensure that the information is accurate and has not expired; information must thus be appropriately encrypted and digitally signed. Suppression and generalization techniques applied to this digitally signed information will result in information tampering and can invalidate the information. In this paper, we propose a protocol that provides k-anonymity in attribute-based access systems by addressing such issues.

An informal description of our protocol begins with the *Policy Enforcer* sending its disclosure policy to the *Credential Submitter*. The *Policy Enforcer* specifies its access control requirements using *disclosure policies*. stating what kind of credentials are needed in order to obtain the resources. The counterpart, that is, the *Credential Submitter*, satisfies the disclosure policies by providing the requested credentials. We assume that such credentials are released by by a certification authority, which we call *Credential Issuer*. Since the notion of privacy is very subjective, we let the *Credential Submitter* choose an integer k (denoting a suitable level of anonymity) and forward it to the *Policy Enforcer*. The *Credential Submitter* before forwarding its disclosure set must ensure that this set is k-anonymous with respect to the other disclosure sets stored at the *Policy Enforcer*. In order to check for k-anonymity, the *Credential Submitter* receives information from the *Policy Enforcer* and executes a private matching protocol. If the *Credential Submitter* is satisfied with the results, it forwards the disclosure set to the *Policy Enforcer*. Otherwise it requests more information from the *Policy Enforcer*. During the protocol sensitive data are exchanged between the *Policy Enforcer* and the *Credential Submitter*. Notice that we do *not* require the intervention of an external trusted third party at any time, but we still ensure that each party can verify that the received data is *valid*, that is, data that is accurate and has not expired. To accomplish this, we assume that the *Credential Submitter* has the right of signing the data it sends to the *Policy Enforcer* on behalf of the *Credential Issuer* which originally signed all the exchanged credentials. For instance, the *Credential Submitter* interacts with the *Policy Enforcer* to verify that the data exchanged during the private matching protocol corresponds to actual credentials stored at the *Policy Enforcer*. Similarly, the *Policy Enforcer* interacts with the *Credential Submitter* in order to check whether the data submitted by the latter correspond to valid credentials or not.

A particularly challenging issue arises when the *Policy Enforcer* does not hold a sufficient number of credentials for carrying on our k-anonymous access control protocol, upon the reception of a request from the *Credential Submitter*. This typically happens in the initial protocol running phase, which we call the *bootstrapping* phase. We overcome this difficulty by letting other credential submitters – having similar credentials to those of the*Credential Submitter* – issue similar requests towards the *Policy Enforcer* . This is done with the online involvement of the *Crdential Issuer*, which knows who are the credential submitters having disclosure sets equal to the one of the *Credential Submitter*.

Our protocol satisfies a number of important properties. First, before submitting its disclosure set, the *Credential Submitter* has the assurance that its set will be identical to other k disclosure sets already stored with the *Policy Enforcer*. Second, the protocol ensures that the credentials submitted by the *Credential Submitter* cannot be linked to each other. Third, we ensure that the critical data exchanged during the protocol is *valid*; as we already said, this property is achieved without the help of a trusted third party (other than those required for signing on the *Credential Issuer*'s behalf[3].

The rest of the paper is organized as follows. Section 2 formalizes the problem we address in the paper. Section 3 introduces the notion of attribute assertions and gives an high level description of our protocol, along with a discussion on how to cope with the bootstrapping problem. In Section 4 we conclude the paper. due to space limits, we do not provide the backgorund on the cryptographic tools

[3] This can be done in a setup phase, detached from the other steps of the protocol.

needed by our protocols neither the formal proofs about their security properties. These can be found in the extended version of this work [4].

2 Problem Statement

Our problem involves two entities executing an access control protocol: the *Policy Enforcer* and the *Credential Submitter*, (*PE* and *CS* in what follows). The *PE* lists the disclosure policies which the *CS* must satisfy in order to get access to some resource. We denote a disclosure policy as $P(x_1, \ldots, x_q)$, where $x_1, \ldots, x_q$ denote the names of attributes present in credentials or boolean conditions defined on the attributes. For ease of presentation, we assume that $x_1, \ldots, x_q$ refer to names of attributes.

A *CS* may satisfy disclosure policies by submitting a *disclosure set* of digital attributes of his possession. The disclosure set of attributes is extracted from set $\{\texttt{Cred}_1, \ldots, \texttt{Cred}_n\}$ where each $\texttt{Cred}_i, i \in [1, \ldots, n]$, is a credential. Each credential is represented as a data structure of the form $\texttt{Cred} = \langle \texttt{Attr}_1^{id} = v_1^{id}, \ldots, \texttt{Attr}_u^{id} = v_u^{id}, \texttt{Attr}_1^s = v_1^s, \ldots, \texttt{Attr}_w^s = v_w^s \rangle$, composed by *identifying* attributes $\texttt{Attr}^{id}$ and *sensitive* attributes $\texttt{Attr}^s$. Identifying attributes uniquely identify the credential's submitter, and include attributes such as the issuer's reference and the submitter's unique id. On the contrary, sensitive attributes carry information that is non-identifying, when considered in isolation. Typical examples of sensitive attributes are the submitter's age, address, job, citizenship, qualification, etc. Additionally, we assume that credentials are signed from trusted third parties, referred to as *Credential Issuers* (*CI*).*CI*s sign credentials of a same type, and a same subject typically possesses a set of credentials issued from different issuers. In the paper we assume without loss of generality that all users' credential are signed by a same *CI*.

The *disclosure set* of a submitter CS with respect to a policy $P(x_1, \ldots, x_q)$ is the set of relevant attributes for satisfying one or more attribute requests of policy P and is denoted by $DSet_{cs}$. The set of values of all and only sensitive attributes contained in a disclosure set $DSet$ is denoted as $Val(DSet^s) = \{v_1^s, \ldots, v_u^s\}$. Subscripts and superscripts are omitted when they are clear from the context.

A *CS*, to remain anonymous, should prevent disclosure of the value of any identifying attribute. For stronger assurance, it may require some specific conditions to hold at the *PE* end. It may want to be assured that at least k other sets $DSet_1, DSet_2, \ldots DSet_k$ with the same combinations of sensitive attribute values are stored at *PE*, so that the disclosure of his attribute values $DSet_{cs}$ cannot be uniquely linked to him.

Definition 1. (k-anonymity). *Let* PE *be the party in charge of enforcing policies and* CS *be the credential submitter. Let $P(x_1, \ldots, x_q)$ be a disclosure policy of* PE*, and $CredB_{PE}$ be the credentials stored by* PE*. Let $DSet_{cs}$ be the disclosure set of non-identifying attribute values that* CS *submits in response to $P(x_1, \ldots, x_q)$. $DSet_{cs}$ is k-anonymous with respect to $CredB_{PE}$, if and only if* *there exist at least k disclosure sets $DSet_1, DSet_2, \ldots DSet_k \in CredB_{PE}$ such that they have equal values for the sensitive attributes, that is, $Val(DSet^s_{cs}) = Val(DSet^s_1) = \ldots = Val(DSet^s_k)$.*

PE may run similar negotiations with different subjects, and collects over time the received disclosure sets. The values of $DSet_{cs}$ thus need to be matched with such sets stored at *PE*. If k identical sets are stored, then – even if *PE* knows the values stored in the sensitive attributes $DSet_{cs}$ – it should not be able to link such values to less than k credentials' owners.

However, checking k-anonymity on sensitive attributes of credentials alone is not sufficient in order to carry on successful anonymous trust negotiations. The following problems arise when dealing with trust negotiations aiming at preserving the anonymity of *CS*.

- *PE* has to be assured about the validity of attributes stored in the *CS*'s disclosure set.
- *CS* has to be assured about the validity of the disclosure sets stored by *PE*.
- Upon receiving the same disclosure set from the same *CS* more than once, *PE* should *not* be able to link multiple trust negotiation sessions to such submitter. We call this last requirement *credential unlinkability*. This requirement is crucial to make k-anonymity effective.
- The disclosure sets stored by *PE* should remain confidential at all times from any user (besides the actual credential owners).

Example Scenario Throughout the paper we will use an example scenario for the case of medical services provided to individuals. Asthma Research Foundation delivers symptoms relief medication for free to individuals' who suffer of asthma diseases. The medications are tailored to the users, based on their characteristics, like the geographic location, weight, age etc. As this medication is for symptoms relief it does not require that past usages be controlled or issues regarding overdose, and it can be requested by the same individual multiple times. Alice wants to take advantage of the offer, but she wants to keep her health condition confidential. Consequently, she wants to ensure that she is not identifiable based on her specific asthma disease or geographic location. Accuracy and authenticity of the data is crucial to ensure safety of the individuals, and from the foundation standpoint, to avoid distributing medications to ineligible individuals. Thus, collected data cannot be altered. Alice is willing to participate and disclose the required information only if at least she has the assurance that the same medication has been provided k other times, so that her own entry will not uniquely identify her. In this scenario, Alice is the *CS* and the *PE* corresponds to the ARF portal, which collects records of users from different locations, to explore the relations among the asthma disease and the environment and the age of the affected individuals. The credential issuer is played by the HIPAA hospital association, issuing health certificates, referred to as HealthCert, and patients id cards, referred to as IdCard.

3 *k*-Anonymous Access Control Protocol

In this section we illustrate the protocol for carrying out k-anonymous access protocols. We start by describing the structure of attribute assertions required to exchange credential attributes in an anonymous fashion. We then provide a step by step description of our k-anonymous access protocol.

3.1 Attribute Assertions

Credentials by their very nature cannot be modified, blinded, or forged without invalidating the *CI*'s signature. To achieve anonymity during access operations, identifying attributes should never be revealed in clear. However, the credential's signature is typically computed over the whole credential, and not on single attributes. Unless credentials are encoded using specific cryptographic techniques, selective disclosure of specific attributes is not possible.
Our protocol for k-anonymous access to services tackles this problem by using ad-hoc assertions, which preserve the same security properties of digital credentials, and are generated from attributes belonging to 'ordinary' credentials. This feature is crucial for ensuring that our protocol can exploit ordinary credentials, and it does not constraint the interacting parties nor credential issuers to adopt specific credential encoding. Specifically, given a disclosure $DSet_{cs}$, a corresponding *attribute assertion* can be created to combine information about identifying and sensitive attributes in such a way that it is impossible for unauthorized users to infer identifying information while having the assurance that such information is valid.

Example 1. Consider the scenario discussed in Section 2. A possible (simplified) policy of the ARF portal distributing the medication on asthma disease is:
P(IdCard.zipcode, IdCard.dateOfBirth, HealthCert.weight, HealthCert.asthma, HealthCert.bloodType).
The corresponding sensitive attributes in the disclosure set (i.e., $DSet$) to be submitted by Alice are: IdCard.zipcode = 47805, IdCard.dateOfBirth = 10/11/1980, HealthCert.weight=170lb, HealthRecord.asthma = AlternariaAlternata, HealthRecord.bloodType = AB+. The resulting set of values is thus
Val(DSet)={47805,10/11/1980, 170, AlternariaAlternata, AB+}.

An attribute assertion is computed by the CS itself and is signed on behalf of the CS using a *proxy signature*.

The structure of an attribute assertion is defined below. Let $H()$ be a cryptographic hash function. Let $Commit()$ denote a cryptographic commitment scheme. Finally, let $PSig_z()$ denotes a proxy digital signature scheme with signing key equal to z.

Given a disclosure set $DSet$ and its corresponding value set $Val(DSet) = \{v_1^{id}, \ldots, v_u^{id}, v_1^s, \ldots, v_w^s\}$ the *attribute assertion* of $DSet$ is defined as the pair $AA = \langle hash, \texttt{psig} \rangle$ where:
- $hash = H(ConcVal(DSet^s)||commit)$,

with $commit = Commit(v_1^{id}, \ldots, v_u^{id})$ and $ConcVal(DSet^s) = (v_1^s || \ldots || v_w^s)$[4]
• `psig` $= PSig_{proxyKey}(hash)$
The commitment is executed on the identifying components of the credentials which the attributes in $DSet$ originally belong to. Hence, identifying attributes, once disclosed, can be used to prove the actual existence of the specific credential from which the sensitive attributes were extracted.

Example 2. Consider Alice's $Val(DSet)$ of the example above. Alice has to send attribute values from two credentials, the IdCard and Health certificate. To disclose her values Alice can create a single attribute assertion as follows:
1) Commits identifiers: 124 and Bohrty, and obtains
$commit_{Alice} = Commit(124, Bohrty)$
2) Hashes sensitive attributes, concatenated with $commit_{Alice}$:
$hash = H(47805||10/11/1980||170||AlternariaAlternata||AB+||commit_{Alice})$
3) Signs the hashed value using HIPAA's signing key `psig`$= PSig_{HIPAA}(hash)$
4) Compose $\langle hash, \texttt{psig} \rangle$.

As shown by the example above, the commitment is concatenated with the actual values and hashed, so that credential values are completely randomized. Attributes assertions thus have the property of *unlinkability* of the disclosure sets, which means that an adversary is unable to link pairs of attribute assertions even with the same attribute values and of the same individual. This is because of the semantically secure commitment of the identifying attribute values hashed with the sensitive values. The secure commitment is embedded so that the receiving entity can, if desired, check the existence and the validity of the credentials $\{\texttt{Cred}_1, \ldots, \texttt{Cred}_n\}$ originating the received AA.

3.2 The Protocol: an High Level Overview

We now present a detailed description of the steps composing the k-anonymous access protocol. We assume that previous k-anonymous[5] accesses have been carried out by the same *PE*, using the same protocols, and it thus possesses a non-empty set attribute assertions.
Protocol 31
(i) **Setup**:
• The *CS* is asked to satisfy policy $P(x_1, \ldots, x_q)$ by submitting a set of attribute credentials, denoted as $DSet$. The *CS* then establishes its anonymity requirement, by choosing an integer k, and asks for $n, n \geq k$, disclosure sets to match.
• The *PE*, based on the attribute assertions collected during past negotiations involving the same policy, sends the *CS* a list of attribute assertions $HList = \langle AA_1, \ldots, AA_n \rangle$. Each AA_i appearing in $HList$ is an attribute assertion encoded as illustrated in Section 3.1 and it collects all and only attribute values for the attribute names of $DSet$.

[4] We assume the attributes to be sorted in a lexicographic order.
[5] k is defined by *CS* each time the protocol is started.

(ii) **Private Matching**:
• The *CS* and the *PE* run a private matching protocol, to verify the existence at the *PE*'s end of k disclosure sets having the same sensitive values as its own sensitive values. The private matching is thus computed by matching $Val(Dset^s_{cs})$ against $Val(DSet^s_1), \ldots, Val(DSet^s_n)$. By running a private matching protocol, the *CS* does not learn any additional information about the *PE*'s disclosure sets used. However, it learns the indexes -the position in the $HLIST$ list- of the *PE*'s disclosure sets matching $DSet_{cs}$. The set of indexes is referred to as the *k-set index Index* and identifies a subset of attribute assertions in $HList$ to be verified in the following steps of the protocol.
(iii) **Disclosure Set validation**:
• If the cardinality of k-set index I is strictly smaller than k, then the *CS* aborts the protocol, or asks for extra disclosure sets to match (from now on, we assume without loss generality that I has cardinality equal to k). Otherwise, the *PE* chooses the attribute assertions corresponding to the elements in the k-set index I and proceeds by sending the corresponding commitments $commit_{I_1}, \ldots, commit_{I_k}$ to the *CS*.
• The *CS*, by using commitments $commit_{I_1}, \ldots, commit_{I_k}$, verifies the content of the corresponding attribute assertions, by checking for each element AA_{I_j}, with $1 \leq j \leq k$, that the value $hash_{I_j}$ in AA_{I_j} is equal to $H(Val(Dset^s_{cs})||commit_{I_j})$.
• The *CS* sends to the *PE* the tuple $\langle AA_{cs}, DSet^s_{cs}, commit_{cs}\rangle$. The *PE* verifies that the value $hash_{cs}$ matches with $H(Val(DSet^s_i)||commit_{cs})$, where i can be any value in the k-set I and that the proxy signature sig_{cs} is valid. Additionally, it can challenge the *CS* on the verification in a zero-knowledge fashion of the commitment $commit_{cs}$ for assurance on the attributes originally used for generating the attribute assertion AA_{cs} .

A key aspect of the proposed protocol is that, upon receiving of the $HList$, at step (i), the *CS* has assurance of the existence of valid disclosure sets of the *PE*, obtained during previous negotiations. The *CS* at that stage cannot open those attribute assertions, but it can still verify that they are signed with a valid proxy signature. Thus, in the disclosure set validation, the *CS* only has to verify the attributes assertions of interest, by matching the hash values.

Example 3. With respect to our running example, consider the protocol carried out by Alice to obtain the medications. Alice is willing to submit her personal data, as long as she has assurance of being at least 5-anonymous. As such, the protocol is executed as follows. Alice logs in into the ARF portal and starts filling the request form. She is asked for credentials as a means for revealing her data, and she counter replies asking for disclosure sets result of other 8 past transactions with other users. Note that the past transactions carried out by ARF and the HIPAA's delegation represent the precondition for the engaged protocol. ARF, in response, sends Alice the corresponding list of attribute assertions, which are listed in the left end side column of Table 1, while the corresponding attribute values appear in the first four columns of the table. The values are properly concatenated and used for running the Private Matching protocol. Alice's disclosure set conveys values:

$47805, 1980, 170, AlternariaAlternata, AB+$. As such, the private matching returns the following set of k-set indexes $= \{1, 3, 4, 8, 9\}$. Since I's cardinality is 5, the k-anonymity requirement of Alice can be potentially satisfied. Alice sends ARF the request to provide the information - that is, the commitments - to open the hash values in the corresponding assertions which corresponding records are in the positions of the values in I. At this point, Alice has all the information to check the validity and the actual content of the attribute assertions with position $\{1, 3, 4, 8, 9\}$. As the validation does not require opening the identity attributes commitments, Alice does not learn anything about the identity of the owners of the attribute assertions. If the hash comparisons prove the assertions to be valid, Alice can prepare the attribute assertions to send. The creation of the assertion is executed as described in Example 2.Upon receiving the assertion the ARF server can check directly the validity of the signature, and the content. Alice can complete her survey being sure that her k-anonymity requirements are satisfied.

Val(DSet)					Attribute Assertion fields	
IdC Zip-Code	**IdC Yearof-Birth**	**HealthC Weight**	**HealthC Disease**	**HealthC Blood-Type**	**Commit(HealthC.ID, IdC.LastName)**	**Hash(ZipCode, YearofBirth, BloodTe)**
47805	1980	170	Alternaria Alternata	AB+	W5VBHT76	SQUCV25Y
59448	1979	148	Psilocybe Cubensis	B+	54NH8GUJ	DRY3QTB
47805	1980	170	Alternaria Alternata	AB+	TREDRFV	GTVVRFF
47805	1980	170	Alternaria Alternata	AB+	P4fNHUJ	DRVYTJ
29785	1980	150	Psilocybe Cubensis	AB+	59IGBHUJ	DRS9SS
20052	1984	120	Penicillium citrinum	B-	DRD9DTJ	RDJ4EDA
20152	1981	175	Penicillium citrinum	B-	54fNHUJ	DRVY21Q
47805	1980	170	Alternaria Alternata	AB+	54fNHUJ	NMR678T
47805	1980	170	Alternaria Alternata	AB+	4BBASQE	GBTNT025

Table 1. Credential attributes for the running example and attribute assertions commitments.

3.3 The Protocol: a Detailed View

In this section we provide a detailed description of the private matching of negotiators' attribute sets and the validity credentials' checking phase.

Private Matching of Assertions When two parties want to determine whether they have common elements in their private datasets, they can use a private matching protocol. Private matching consists of secure intersection of inputs of two parties, based on semantically-secure homomorphic encryption. In our context, the *CS* and the *PE* want to determine whether there are at least k disclosure sets at the *PE*'s side having the same sensitive values of the *CS*'s disclosure set. For our purposes, we deploy the private matching protocol presented in [1]. For completeness, we present a version of such protocol adapted to our framework.

Protocol 32 (i) **Setup**:

1. The *PE* provides the set $\Delta = \{\zeta_1, \ldots, \zeta_z\}$, where each ζ_i is a concatenation of all sensitive values of the *PE*'s disclosure set $Val(Dset_i)$, for $1 \leq i \leq z$. That is, $\zeta_i = (v^s_{i_1}|| \ldots ||v^s_{i_u})$.
2. The *CS* specifies the integer k, denoting the desired level of anonymity.
3. the *CS* computes the concatenation of the sensitive values in $DSet_{cs}$ $\delta = (v^s_1|| \ldots ||v^s_u)$.

(ii) **Hash and Commit**:

1. The *CS* chooses the secret-key parameters for a semantically-secure homomorphic encryption scheme, and publishes its public keys and parameters.
2. The *CS* uses interpolation to compute the coefficients of the polynomial $P(\zeta) = \sum_{u=0}^{t}(\lambda^u \zeta^u)$ of degree t with only one root δ.
3. The *CS* encrypts each of the $(t + 1)$ coefficients by the semantically-secure homomorphic encryption scheme and sends to the *PE* the resulting set of ciphertexts, $\{\mathsf{E}(\lambda_0), \ldots, \mathsf{E}(\lambda_t)\}$.

(iii) **Commitments' Exchange**:
EncSet denotes the set of encrypted values, initially equal to the empty set.

1. The *PE* for all $\zeta_i \in \Delta$ computes $\mathsf{E}(P(\zeta)) = \mathsf{E}\ (\sum_{u=0}^{1} \lambda_u \zeta^u)$. The *PE* chooses a random value r and computes $Enc = \mathsf{E}(rP(\zeta) + \zeta)$. The *PE* computes $EncSet$=$EncSet \cup Enc$.
2. The *PE* sends the k sorted elements to the *CS*.

(iv) **Cardinality Checking**:

1. The *CS* computes for all Enc_j in *EncSet*, with $j \in [1, n]$, $\mathsf{D}(Enc_j)$ and locally outputs all values δ for which there is a corresponding decrypted value.
2. If δ matches $\mathsf{D}(Enc_j)$ the *CS* adjoins index j to the index set *Index*.
3. If $|Index| \geq k$, the protocol ends successfully.

Disclosure Sets Signature and Validation Once that the *CS* and *PE* have checked – executing Protocol 32 – that the latter has at least k disclosure sets matching the former's disclosure set on the sensitive values, they have to be assured on the fact that such values are from credentials owned by legitimate users. This should be executed without disclosing the values of identifying attributes. This task is accomplished by the *disclosure set signature and validity checking protocol*, that we illustrate in the following paragraphs.

Protocol 33 (i) **Setup**: the *CS* generates the public parameters starting from a integer number w (the security parameter): a prime p having bitlength equal to w, two base groups G_1, G_2, along with their generators g_1 and g_2, and a target group G_T, all of order p. A computable isomorphism ϕ between G_1 and G_2 and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$.

(ii) **Keys Generation**:
• The *CS* chooses a random number $v \in \mathbb{Z}_p$ and computes its *public key* g_2^v. Its corresponding *secret key* is v. Note that $g_2^v \in G_2$.
• The *CS* computes its public and secret keys $\langle pk_{cs}, sk_{cs} \rangle$, respectively. Such key pair can be computed using any public key scheme and will be involved only in the communication between the *CS* and *CS*.
Summarizing, the public parameters are $\langle G_1, G_2, G_T, g_1, g_2, e, \phi \rangle$, and the public keys pk_{cs} of the *CS*. Note that these last public key is deployed only in communications between the *CS* and the respective owner. As we will see, our protocol offer no way to the other party to link a public key to the respective owner.

(iii) ***CI*'s Signature Delegation**: the *CI* performs the following steps in order to let the *CS* sign on its behalf:
  (a) The *CI* chooses a random number $r \in \mathbb{Z}_{p-1}/\{0\}$, and computes $r_{cs} = g_2^r$. Then, the *proxy signature key* is defined as $\sigma_{cs} = v + r \cdot r_{cs} \mod p-1$.
  (b) The *CI* creates a *warrant* $\texttt{warr}_{cs}$ for constraining the *CS*'s delegation rights.
  (c) The *CS* computes its (original) signature on $\texttt{warr}_{cs}$. Such signature is computed with the co-GDH signature scheme $\mathcal{S}$ (for cryptographic background please refer to [4]) using the the original *CI*'s signing key sk_{ci}: first, the *CI* applies the cryptographic hash function H to $\texttt{warr}_{cs}$, thus obtaining the value $h = H(\texttt{warr}_{cs})$; then, the *CS*'s original signature is set equal to $cert_{cs} = h^v$. Note that $cert_{cs} \in G_1$. We refer to such signature as the *CS's (delegation) certificate*.
  (d) The *CI* encrypts with the *CS* public key pk_{cs} the tuple $\langle \texttt{warr}_{cs}, cert_{cs}, \sigma_{cs}, r_{cs} \rangle$, obtaining the encrypted value $del_{cs} = \mathcal{E}_{pk_{cs}}(\langle \texttt{warr}_{cs}, cert_{cs}, \sigma_{cs}, r_{cs} \rangle)$.
  (e) Finally, the *CI* sends del_{cs} to the *CS*.

Note that Setup, Key Generation and Signature's Delegation steps can be performed in advance, before the start of the trust negotiation process. In general, the delegation rights depend on the warrant's content. For example, the warrant could specify what kind of (or, part of) credentials the delegated party can sign and the time span in which the the delegated sign is valid. We discuss this issue in more detail in Section 3.3.

(iv) ***PE*'s Attribute Assertions**: the *PE* sends the list $HList = \{AA_1, \ldots, AA_n\}$ to the *CS*.

(v) **Commitment of *CS*'s Identifying Attributes' Values**: consider the tuple of identifying attributes' values $\langle v_1^{id}, v_2^{id}, \ldots, v_u^{id} \rangle$ contained in the $DSet_{cs}$. The *CS* commits their values using a commitment scheme as the following: generators $g_1, \ldots, g_u$ of G_1, an element $h \in G_1$ and a random element $\hat{r} \in \mathbb{Z}_p$. Then, the commitment of the identifying attributes' values is defined by $commit_{cs} = \prod_{i=1}^{u} g_i^{v_i^{id}} \cdot h^{\hat{r}}$. Again, note that $commit_{cs} \in G_1$.

(vi) **Proxy Signature of *CS*'s Attribute Assertion**: the *CS* signs its attribute assertion AA_{cs} on behalf of the *CI*. It does this by using the encrypted value del$_{cs}$ it received from the *CI* . The *CS* decrypts it with its private key, obtaining $\mathcal{E}_{sk_{cs}}^{-1} = \langle \texttt{warr}_{cs}, cert_{cs}, \sigma_{cs}, r_{cs} \rangle$ First, the *CS* computes the hash value $hash_{cs} = H(Val(DSet^{s}_{cs})||commit_{cs})$. Then, the *CS* signs with σ_{cs} the values pk_{ci} and $hash_{cs}$ with the co-GDH signature scheme $\mathcal{S}$, obtaining $sign_{cs} = \mathcal{S}_{\sigma_{cs}}(pk_{ci}||hash_{cs})$. The *CS* defines as a *proxy signature* of hash value $hash_{cs}$ the tuple $\texttt{psig}_{cs} = \langle \texttt{warr}_{cs}, pk_{ci}, cert_{cs}, sig_{cs}\rangle$. Finally, the *CS* sends its attribute assertion $AA_{cs} = \langle hash_{cs}, \texttt{psig}_{cs}\rangle$ to the *PE*.

(vii) ***PE*'s Commitments**: the *PE* sends to the *CS* values $commit_{I_1}, \ldots, commit_{I_k}$ of the identifying attributes' values of the disclosure sets as identified by running Protocol 32.

(viii) **Proxy Signatures' Verification**: the *Policy Enforcer* verify the validity of the signatures on the committed identifying values it receives from the *CS*, without disclosing such values.

(ix) ***PE*'s Verification**: recall that the proxy signature $\texttt{psig}_{cs}$ is defined as the tuple
$\langle \texttt{warr}_{cs}, pk_{ci}, cert_{cs}, sig_{cs}\rangle$. The *PE* has to check the two signatures $cert_{cs}$, sig_{cs}. According to the co-GDH signature scheme, this amounts to perform the following steps:

– **Verification of** $cert_{cs}$: the *PE* computes $h' = H(pk_{cs}||\texttt{warr}_{cs})$. Then, $\langle g_2, pk_{ci}, h', \texttt{warr}_{cs}\rangle$ is checked to be a valid co-GDH tuple.
– **Verification of** $sign_{cs}$: the *PE* computes $h'' = H(pk_{ci}||commit_{cs})$. As in the previous case, $\langle g_2, pk_{cs}, h'', commit_{cs}\rangle$ is checked to be a valid co-GDH tuple.

If both steps succeed, then the *PE* accepts the proxy signature $\texttt{psig}_{cs}$. Then, the *PE* computes the value $H(Val(DSet^{s}_{i})||commit_{cs})$ and checks whether such value is equal to value $hash_{cs}$.

(x) ***CS*'s verification**: the *CS* computes the values $H(Val(DSet^{s}_{cs})||commit_{I_i})$, where $\leq i \leq k$. Then, he checks whether such values match the corresponding values in $HList$, sent by the *PE* to the *CS* at the end of the run of Protocol 32.

(xi) In the case that both signatures and hash values are positively checked, the *CS* sends the pair
$\langle commit_{cs}, DSet^{s}_{cs}\rangle$ to the *PE*.

The Warrant and its Structure The structure of the warrant constraints the ability of the proxy signer to sign on behalf of the original issuer. The warrant does not have a fixed nor pre-established structure and can be used by any *CI* to specify the exact conditions under which the signing key can be used. In general, the information to carry on any warrant includes the `temporal validity` and `credential type`. The time interval indicating the temporal validity is typically short, to ensure that the delegation rights are periodically renewed by the *CS*. Finally, the credential type denotes the specific type of credentials that the *CS* is allowed to sign.[6] The signer may be allowed to sign a single credential type,

[6] Recall that credentials have an associated type, which is part of the identifying attributes.

or it may be delegated to sign a list of different credential types. At the time of verification, the *CS* cannot maliciously provide incorrect information regarding the signature on the attribute values to *PE*, as this is guaranteed by the proxy signature scheme employed. On the other side, the *PE* cannot forge the signatures as well.

3.4 The Problem of Bootstrapping

PE should have availability of several attribute assertions to execute protocol 31. A natural question is how to make this possible in the early stage of *PE* activity, when it has carried out an insufficient number of transactions with other peers in the system. We have devised two strategies to address the problem of bootstrapping of the k-anonymous protocol. One additional benefit of these strategies is that they allow *PE* to run negotiations with small data sets.

A first strategy is based on a *hold* mechanism and on the threshold k selected by *CS*. Upon service request, *PE* prioritizes its incoming access requests and first satisfies the ones with lower values for k. *CS* waits, either online or off line, until *PE* has first processed other *CS*'s smaller values of k. This helps *PE* in populating its data set of attribute assertions for similar access control requests and for re-use them for subsequent protocol runs.

A refined version of this strategy is based on ideas from the *crowds* protocols for anonymous networks. In this case, we request cooperations of other peers in the system, which we refer to as CS_C_1,..., CS_C_n (for Credential submitters crowd) and from *CI*, as well. *PE* can contact CS_C_1,..., CS_C_n, through *CI*, which knows the other peers having similar credentials. When *CS* submits a k-anonymous request to service R, *PE* contacts the credential submitters crowd. *CI* request each *CS* to submit its disclosure set as if it were asking for the same service R. By collecting the responses to such requests, a *crowd* is formed and *PE* can accumulate enough attribute assertions to match them with the k-anonymous request of the current *CS*. Then, *PE* forwards the collected assertions $AA_1, \ldots, AA_n$ to *CS* to carry out Protocol 31. If the private matching is successful, step (iii) requires the involvement of the attribute assertions holders which match the *CS*'s one. Such CS_C will submit their commitment components to *CS* which will perform the same verification operations of step (ii).

As we have seen, the proposed bootstrap protocol requires the on-line intervention of *CI* as well as of the members of the fake credential submitters set CS_C. This may result in a undesiderable communication overload. Note however that this state of affairs has limited duration, depending on how large the anonymity level k is. Further note that the bootstrap protocol is particularly effective if some form of incentive is available for the participating peers, such as for instance acquisition of privileges with respect to *PE* services or a higher priority when asking for services.

A final strategy brings together aspects of the above presented ones. It consists of having a crowd formed by the peers which are actually requesters, and

Algorithm 1 Hold and Crowd Protocol based Bootstrapping

```
Require: $CS$ submitting credentials belonging to $DSet_0$
 1: Instantiate ordered CS_Waitlist = { $CS_1, \ldots, CS_n$ }
    {List of $CS$'s which have not submitted their credentials belonging to $DSet_0$.
    Each $CS_i$ has a k-anonymity requirement $K_i$ such that if $CS_i$ precedes $CS_j$ then
    $K_i < K_j$}
 2: $CS_{new}$ = GetNewCS($DSet_0$,$K_{new}$)
 3: counter = 0
 4: for $1 \leq i \leq n$ do
 5:   $response_i$ = PE.Contact($CS_i$) {Ask if $CS_i$ are willing to submit if $CS_{new}$ sub-
      mits}
 6:   Increment(counter)
 7: end for
 8: if (counter > $K_{new}$) then
 9:   crowd.add($CS_{new}$) {Now add the rest of the $CS_i$ that volunteered to the crowd
      list}
10:   for $1 \leq i \leq n$ do
11:     if $response_i$ = true then
12:       crowd.add($CS_i$)
13:       CS_Waitlist.remove($CS_i$)
14:     end if
15:   end for
16:   for each person in the crowd list do
17:     Execute Protocol 3.1
18:   end for
19: else
20:   CS_Waitlist.add($CS_{new}$)
21: end if
```

their requests are scheduled based on their chosen anonymity level k. The pseudo code for such strategy is reported in Algorithm 1.

4 Conclusions and Future Work

In this paper we have proposed a protocol that provides a user-centric approach for anonymity in trust negotiations. We have shown how a subject can verify whether its disclosure set preserves its customized degree of anonymity, without inferring any information about the counterpart knowledge before the actual disclosure. We will further elaborate on possible threats for our protocols.

References

1. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

2. L. Sweeney. k-anonymity: a model for protecting privacy.
3. Microsoft Passport Network. Passport review guide. http://www.passport.com/.
4. Anna Squicciarini, Alberto Trombetta, Abilasha Bhargava-Spantzel, and Elisa Bertino. k-anonymous attribute-based access control. Technical report, 2007.
5. W. H. Winsborough and N. Li. Protecting sensitive attributes in automated trust negotiation. *ACM Workshop on Privacy in the Electronic Society*, November 2002. Washington, DC.
6. W. H. Winsborough and N. Li. Safety in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, May 2004. Oakland, CA.
7. T. Yu and M. Winslett. A Unified Scheme for Resource protection in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, May 2003. Oakland, CA.