Achieving Privacy in Trust Negotiations with an Ontology-Based Approach

Anna C. Squicciarini, Elisa Bertino, *Fellow*, *IEEE*, Elena Ferrari, *Senior Member*, *IEEE*, and Indrakshi Ray, *Member*, *IEEE*

Abstract—The increasing use of Internet in a variety of distributed multiparty interactions and transactions with strong real-time requirements has pushed the search for solutions to the problem of attribute-based digital interactions. A promising solution today is represented by automated trust negotiation systems. Trust negotiation systems allow subjects in different security domains to securely exchange protected resources and services. These trust negotiation systems, however, by their nature, may represent a threat to privacy in that credentials, exchanged during negotiations, often contain sensitive personal information that may need to be selectively released. In this paper, we address the problem of preserving privacy in trust negotiations. We introduce the notion of *privacy preserving disclosure*, that is, a set that does not include attributes or credentials, or combinations of these, that may compromise privacy. To obtain privacy preserving disclosure sets, we propose two techniques based on the notions of *substitution* and *generalization*. We argue that formulating the trust negotiation requirements in terms of disclosure policies is often restrictive. In this sense, we show trust negotiation requirements expressed as property-based policies that list the properties needed to obtain a given resource. To better address this issue, we introduce the notion of *reference ontology*, and formalize the notion of trust requirement. Additionally, we develop an approach to derive disclosure policies from trust requirements and formally stated some semantics relationships (i.e., equivalence, stronger than) that may hold between policies. These relationships can be used by a credential requestor to reason about which disclosure policies he/she should use in a trust negotiation.

Index Terms—need keywords from authors.

1 Introduction

Most of the interpersonal transactions, carried out in any application environment we may think of, are based on the disclosure of relevant *attributes* of the involved parties. Such attributes are used to certify some properties of the counterpart that are relevant for proceeding in the transaction (e.g., the possession of a valid credit card or the membership to a given organization). In the digital world, such interactions have been historically handled out-of-band using alternative means or simply avoided. However, the increasing use of the Internet in a variety of distributed multiparty interactions and transactions with strong real-time requirements has pushed the search for solutions to the problem of attribute-based digital interactions. A promising solution is today represented by automated trust negotiation systems [3], [19]. Trust negotiation systems allow subjects in different security domains to securely exchange protected resources and services [2], [22], [25]. This is achieved by first establishing trust through a bilateral, iterative process of requesting and disclosing user attributes, and policies. Attributes are actually exchanged through the disclosure of digital credentials, which can be considered the equivalent, in the digital world, of paper credentials. Digital credentials can collect several 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 on. The other key component of any trust negotiation system is represented by disclosure policies, protecting sensitive resources, credentials, and even other policies from unauthorized accesses. Disclosure policies express in a machine-understandable way the *trust requirements* of the involved parties, that is, the properties/characteristics needed to obtain a given resource, in terms of credentials and credential attributes.

Trust negotiation systems, however, by their nature, may represent a threat to privacy in that credentials, exchanged during negotiations, often contain sensitive personal information that may need to be selectively released. Also, a user may want to minimize the released information, thus enforcing the need to know principle in disclosing his/her credentials to other parties. In other situations, a user may want to carry out negotiations that cannot be linked to him; we refer to such a requirement as nonlinkability.

In this paper, we address the problem of preserving privacy in trust negotiations. In particular, we propose three orthogonal privacy preserving mechanisms that can be used in trust negotiations. The first is based on the idea that, given a trust requirement, the resource requester should be able to choose the credentials/attributes to submit to satisfy such a requirement, if several alternatives are possible. Multiple alternatives are possible when there are multiple disclosure policies implementing the same high-level trust requirement. For instance, the age of a

- *A.C. Squicciarini is with the Dipartimento di Informatica e Comunicazione, Universita degli Studi di Milano, via Comelico 139, 20135 Milano, Italy. E-mail: squiccia@dsi.unimi.it.*
- *E. Bertino is with CERIAS and the Computer Science Department, Purdue University, 905 University Street, West Lafayette, IN (zip code). E-mail: bertino@cerias.purdue.edu.*
- *E. Ferrari is with the Dipartimento di Scienze della Cultura, Politiche e Inforazione-22100 COMO, FACOLTA Di SCIENZE MM.FF.NN., SEDE DI COMO, Universita degli Studi dell'Insubria, Italy. E-mail: elena.ferrari@uninsubria.it.*
- *I. Ray is with the Computer Science Department, Colorado State University, 601 S. Howes Stree, Fort Collins, CO 80523-1873. E-mail: iray@cs.colostate.edu.*

Manuscript received 12 Oct. 2004; revised 2 Sept. 2005; accepted 13 Dec. 2005; published online 3 Feb. 2006.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0148-1004.

person can be verified either through a policy asking for the passport or the driving license. Having multiple policies implementing the same trust requirement has many advantages. First, it increases the chances of a trust negotiation succeeding. For instance, a user not having the credentials required in a disclosure policy can provide credentials required for an alternative disclosure policy that implements the same high-level trust requirement. More importantly, it gives the user the opportunity to preserve his/her privacy and, at the same time, satisfy the trust negotiation requirement. Finally, the use of trust requirements facilitates the task of policy specification and it also allows the credential requestor to choose, according with his/her privacy adopted strategy, which policies and in which form to reveal.

Supporting this approach requires, however, that trust requirements be expressed in terms of high-level semantics properties. To address this issue, we introduce the notion of *reference ontology*, and formalize the notion of trust requirement. Additionally, we develop an approach to derive disclosure policies from trust requirements and we formally state some semantics relationships (i.e., equivalence, stronger than) that may hold between policies. These relationships can be used by a credential requestor to reason about which disclosure policies he/she should use in a trust negotiation.

The second mechanism we propose to address privacy issues allows a subject to adopt strategies to make privacy-preserving the set of credentials/attributes he/she is going to release. By privacy-preserving, we mean that the set does not include attributes or credentials, or combinations of these, that may compromise privacy. To express subjects privacy preferences, we introduce the notion of *private concept groups*. A private concept group contains a set of concepts that should not be all released during the same negotiation session. Private concept groups are formed by taking into account not only the subject privacy preferences, but also the privacy practices of the counterpart.

It is, however, important to notice that it is not always possible to generate privacy preserving disclosure sets. To address this problem, we propose two techniques by which a credential submitter can substitute the privacy-sensitive credentials and attributes with alternative ones to preserve his/her privacy and satisfy, at the same time, the trust negotiation requirements. These techniques are based on the notions of *substitution* and *generalization*. Substitution is used when the privacy breach is due only to nonrequested attributes, that is, attributes that are not requested by the considered policies but that are disclosed because they belong to the same credential of the requested ones. Such credentials are replaced with some alternative credentials or attributes that implement the same property and that, however, do not result in privacy breaches. Generalization is instead used when privacy breaches arise because of requested attributes. In such situations, we try to replace the sensitive attribute/credential with a more general one, by taking advantage of the reference ontology. Generalization provides an information which is related to the information requested in a disclosure policy but which is not exactly the same. In other words, the provided information does not satisfy the original disclosure policy. However, it may satisfy an alternative disclosure policy that implements the same high-level trust requirement. This allows the trust negotiation to succeed without compromising privacy. In the paper, we provide algorithms for enforcing such techniques, we prove their soundness, and give some complexity results. To the best our knowledge our work is the first providing the above-mentioned features and systematically addressing privacy issues in trust negotiations.

The remainder of the paper is organized as follows: In the next section, we introduce a scenario that we use throughout the paper as a running example. Section 3 describes our notion of trust negotiation policies and their specification. Further, it describes our notion of concepts in ontologies and their use for specifying high-level trust requirements. Section 4 focuses on the notion of trust requirements and their representation through disclosure policies. In Section 5, we deal with privacy issues in trust negotiations. Section 6 introduces a system architecture supporting trust negotiation and describes how the proposed techniques can be combined to carry out a privacy preserving process. Finally, Section 7 gives the related work in this area, whereas Section 8 concludes the paper with pointers to future directions.

2 Running Example

In this section, we briefly introduce the scenario we will use throughout the paper to demonstrate the proposed techniques. The scenario we refer to is that of an online tour operator service, named $Summer$. $Summer$ sells trips to end customers as well as wholesale to travel agencies that then resell the trips by retail. Suppose that Fly is a travel agency wishing to place an order through the $Summer$ portal for a group of tickets for a vacation on an exotic island, and that the order is placed by $Alice$, an employee of Fly. Suppose that $Alice$, while placing the order, decides to buy a single trip by herself, exploiting a special discount for employees of $Summer$ affiliated agencies. Also, Alice wishes to take advantage of a special offer for women travelling alone. However, whereas Alice is very happy to enjoy the convenience of doing business over the Web, she wants to be sure that her privacy requirements are fully compliant with privacy practices of the counterpart. Further, she wants some of her personal properties to be revealed in a controlled and conscious way. For example, she does not want to reveal simultaneously her home address with her age and gender, and she does not want her profession to be linked with her identity. As the paper proceeds, we will use this scenario as a running example to show how Alice can complete those transactions without compromising her privacy requirements.

3 Credentials, Attributes, and Their Relationships

Since credentials and attributes form the core of a trust negotiation, we begin by providing some details about them. Credentials can be specified in several ways. However, to facilitate credentials interpretation and distribution, we consider credentials as instances of associated credential types. The notion of credential type provides a template for specifying credentials having a similar structure and collecting the same attributes. An attribute is a descriptive property, characterized by a name and a domain. The name of an attribute uniquely identifies it. The domain gives the set of permitted values for that attribute. We use subscript of A, such as A_i, to denote the name of an attribute, and D_{A_i} to denote the domain of A_i. Precisely, a credential type is defined by a pair: $< CT_i, AS_{CT_i} >$, where CT_i is the name of the credential type and AS_{CT_i} is a set of pairs conveying attribute names and the associated domains. Credentials are formally defined below.

Definition 3.1: Credential. *Let $< CT_i, AS_{CT_i} >$ be a credential type. A credential Cr_i, instance of $< CT_i, AS_{CT_i} >$, consists of a set of pairs of the form (A_x, V_{A_x}). For each*

attribute $A_k \in AS_{CT_i}$, Cr_i contains the pair (A_k, V_{A_k}), where V_{A_k} denotes the value of attribute A_k, and $V_{A_k} \in D_{A_k}$.

The set of credentials associated with a subject is referred to as *Profile* ($Prof$, for short). Typically, all the credentials conveyed in a Profile uniquely determine the subject. However, if some attributes are not released, we can either prevent identity disclosure of the subject or protect his/her privacy sensitive information. The credential requestor indeed is often interested in some, but not all, of the attributes of the requested credentials. Ideally, a credential submitter would want to provide information on a *need-to-know* basis and would be reluctant to disclose information that is not requested. In other words, it would prefer to selectively disclose attributes contained into a credential. The best approach currently available to allow partial disclosure of credentials relies on the use of the bit commitment technique [16], which enables users to communicate a value without revealing it. By exploiting this technique within digital credentials, it is possible to actually send credentials by revealing only the minimal set of attributes required during the negotiation.

Let us illustrate the process of blinding attributes. Given an attribute `a` with value `va` the operations needed for its blinding are: 1) generate a random string `r`, 2) compute `p=va|r`, that is, the concatenation of `va` with `r`, and 3) compute `v=hash(p)`, where `hash` is a one way multibit hash function. This process can be repeated to blind several attributes in a credential. Once the credential is ready, it is submitted to the credential authority, who verifies its contents and signs it. Now, if a negotiation requires some blinded attributes to be disclosed to the requestor, the submitter can send the requestor the original value `va` and the random value `r`. The requestor computes `va` using the hash function that is publicly known and verifies the attribute validity. The remaining sensitive attributes of a credential that are not relevant for the negotiation can be left hidden, and never be disclosed to the counterpart.

In the following, we assume some attributes of the credentials not to be blindable, such as the reference information about the corresponding credential type, the issuer, and its temporal validity. This set of information is indeed crucial for proving that the credential, besides its specific content, is a signed and valid digital document issued by an entity reputed trusted and should never be hidden to the receiver. Further, some attributes may not be blindable because of the certificate authority practices. For instance, some authorities may permit blinding only attributes considered sensitive, which may reasonably be a subset of the whole set of attributes carried in a credential. The ability to blind one or more attributes in a credential generates different *views* of the credential. Additionally, a credential may contain attributes that cannot be blinded because they convey the meaning of the credential itself. For instance, a password credential with the login-name attribute blinded is completely useless. Views of a same credential differ among each other by the number of hidden attributes. Precisely, let l be the number of attributes in a credential *cred*, and k ($k \leq l$) be the number of attributes that can be blinded. Then, there are 2^k possible views for *cred* [1]. When a credential requestor requests a credential *cred* without specifying the required attributes, one of the most blinded views of the credential is provided.[1] When a credential requestor requests an attribute *attr* contained in a credential *cred*, the submitter provides the view in which *attr* is not hidden and in which the maximum number of other attributes are blinded. Next, we give an example of different views associated with a credential. We denote with • • • • a blinded attribute value.

Proof. Consider a credential named Passport with the following attributes: Number, Surname, Name, Nationality, Date of birth, Sex, Place of birth, Date of issue, and Expiration Date. Suppose that the following four attributes cannot be hidden: Date of birth, Number, Date of issue, and Expiration Date. Consider now the following views of the Passport among the $2^5 = 32$ possible ones:

1. Passport = {Number: 512965C, Surname: White, Name: • • • •, Nationality: • • • •, Date of birth: 01.11.1977, Sex: • • •, Place of birth: • • • •, Date of issue: 05 May 2004, Expiration Date: 04 May 2014}.
2. Passport = {Number: 512965C, Surname: • • • •, Name: • • • •, Nationality: • • • •, Date of birth: 01.11.1977, Sex: • • • •, Place of birth: • • • •, Date of issue: 05 May 2004, Expiration Date: 04 May 2014}.

Suppose the submitter is asked for the attribute Passport.Date of issue.[2] Then, the second view is disclosed, since it hides the highest number of attributes. By contrast, if the submitter is asked for the attribute Passport.Surname, the first view is released because the second has this attribute encrypted. Finally, if the submitter is asked for the whole credential Passport, the second view is released because it has the highest number of blinded attributes.

Note that, credential types syntactically structure the information conveyed by the corresponding credentials, but they do not specify anything about the semantics of the attributes. This makes it impossible to automatically determine whether attributes belonging to different credentials are semantically related (e.g., they express the same information, one is the specialization of the other, and so on) or not. This information is crucial during a negotiation, because it allows one to select among possible different equivalent sets of credential attributes the one to be disclosed, thus enhancing flexibility and privacy. To deal with this issue, we borrow some ideas from work on ontologies [10], [14], [21]. Ontologies represent an essential tool for allowing communication and knowledge sharing among distributed users and applications by providing a common understanding of a domain of interest. An ontology consists of a set of concepts together with relationships defined among these concepts. Here, we use an ontology for credentials and attributes, that we refer to as *reference ontology*. Each *concept* in the ontology is associated with a name, a set of keywords, and a set of attributes and credential types names. The formal definition is given below.

Definition 3.2: Concept. *A concept, denoted by C_i, is a tuple $\langle Name_i, KeywordSet_i, LangSet_i \rangle$, where $Name_i$ is the concept name, $KeywordSet_i$ is a set of keywords associated with C_i, and $LangSet_i$ is a set of credential type and/or attribute names. $KeywordSet_i$ describes the set of all possible keywords used to describe concept C_i. Each element in $KeywordSet_i$ is a synonymous of $Name_i$. Each attribute*

1. There might be several views having the same number of attributes in clear and, thus, several alternative views for a given attribute may be available.

2. Here, and in the following, we use the dot notation to identify credential attributes.

or credential type in $LangSet_i$ implements concept $\mathcal{C}_i$.

$$\mathcal{C} = < gender, \{sex\}, \{Passport.gender, DrivingLicense.sex\} >$$

is an example of concept. In this case, the concept known as $gender$ or sex can be implemented by the attribute $Passport.gender$ or the attribute $DrivingLicense.sex$. In other words, a concept can be implemented by attributes of different credentials or by different credentials.

For any two distinct concepts $\mathcal{C}$ and $\mathcal{C}'$, where $\mathcal{C} =< Name, KeywordSet, LangSet >$ and

$$\mathcal{C}' =< Name', KeywordSet', LangSet' >,$$

the following conditions hold: $KeywordSet \cap KeywordSet' = \emptyset$ and $LangSet \cap LangSet' = \emptyset$. In other words, any keyword belongs to exactly one concept. Similarly, each credential type or attribute of a credential is associated with exactly one concept.

A reference ontology $\mathcal{O}$ is a partially ordered set of concepts $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$. The order relationship, denoted as $\prec$, represents a generalization relationship between concepts. $\mathcal{C}_i \prec \mathcal{C}_k$ if concept C_k is a generalization of concept C_i. This means that information conveyed by concept C_i can be used to infer information conveyed by concept C_k. For instance, the concept *country of residence* is a more general concept than *address* (denoted as $address \prec$ *country of residence*), since if we know the address, we also know his/her country of residence.

In our work, we assume the existence of the reference ontology as part of the Trust-$\mathcal{X}$ language. Also, we assume the ontology to be stored in a repository available to all the Trust-$\mathcal{X}$ subjects, for reference. However, we can also think of different ontologies to be used for the same domain of interest, as it actually happens in the P2P area. To enable information processing and content retrieval, appropriate matching techniques can be employed, to determine semantic mappings between concepts of different related ontologies [8].

4 Expressing Trust Requirements

In our approach, we express trust requirements at two different levels of abstraction. We refer to the abstract representation as *abstract trust requirements*, and to the concrete representation as *disclosure policies*. In this section, we describe these two representations and illustrate how to derive disclosure policies from trust requirements.

4.1 Expressing Abstract Trust Requirements

In the context of trust negotiations, each entity is interested in obtaining information and verifying properties about the counterpart to establish trust. A *trust requirement* lists the set of properties the counterpart has to provide to obtain a given resource. The formal definition appears below.

Definition 4.1: Abstract Trust Requirement. *Let $\mathcal{O}$ be a reference ontology, and R be a resource. A* trust requirement *(TR for brevity) for R with respect to $\mathcal{O}$ is a triple $\langle R, properties, conditions\rangle$,[3] where R denotes a target resource, $properties$ is a set of property names, and $conditions$ is a set of conditions defined over one or more properties in $properties$. $\forall\, p \in properties$, there exists a concept $C_i = \langle Name_i, KeywordSet_i, LangSet_i\rangle \in \mathcal{O}$ such that $p \in KeywordSet_i$.*

3. $properties$ and $conditions$ are optional.

Example 2. Examples of abstract trust requirements with respect to the running example presented in Section 2, for $Summer$ tour operator are:

$$\langle SingleTrip, \{Company\_Employee, gender, country\}, \{Company\_Employee \in \{Fly, Sun, Alpitour\}, gender = female, country = America\}\rangle$$

and

$$\langle TripPackage, \{Jobtitle, Company\_employee\}, \{JobTitle = AgentSeller, Company\_Employee \in \{Fly, Sun, Alpitour\}\}\rangle.$$

This means that in order to buy the $SingleTrip$, Alice has to qualify herself as an employee of $Summer$ affiliated agency, give proof that she is American, and reveal her gender. By contrast, to place the order for the agency she works for, she only has to qualify herself as an *AgentSeller*, besides proving that she works for an affiliated agency.

We show in the following sections how we can implement abstract trust requirements into a set of disclosure policies.

4.2 Expressing Disclosure Policies

To implement an abstract trust requirement $TR = \langle R, properties, conditions\rangle$, each property in $properties$ must be stated in terms of credentials or attributes, and each condition in $conditions$ must be translated into an appropriate condition on credential attributes. We refer to such conditions as *attribute conditions*. An attribute condition AC is a Boolean expression of the form A *op* k, where A denotes an attribute name and *op* is a comparison operator in $\{\neq, <, >, =, \leq, \geq, \in\}$. If *op* is a comparison operator in the set $\{\neq, <, >, =, \leq, \geq\}$, then k is either a constant or variable and $k \in D_A$. Alternatively, if *op* is the operator $\in$, then $k \subset D_A$.

A credential can thus be represented as a structured object composed of several items, corresponding to its attributes. The name of the credential and attribute conditions can be used to compose expressions to be used while defining disclosure policies. Disclosure policies are the *implementation* of trust requirements, as elaborated in the following sections. Disclosure policies are defined using terms, given below.

Definition 4.2: Term. *A term T is of the form $CT(C)$, where: CT is a credential type and C is a (possibly empty) set $\{C_1, \ldots, C_n\}, n \geq 1$, where each $C_i, i \in [1, n]$ is either an attribute name in the attribute set AS_{CT} associated with CT, or it is an attribute condition defined on some attributes in AS_{CT}.*

According to the above definition, terms are of two forms: credential types, denoted by $CT()$, and credential types with attribute names or attribute conditions, denoted by $CT(C)$. For terms of the form $CT()$, we assume that the possession of the credential is of interest. The information conveyed by the attributes of the credential are not required. If a term is of the form $CT(C)$, we assume that the attributes listed in C are of interest.

In what follows, a particular form of terms, that we refer to as *canonical terms*, is of interest.

Definition 4.3: Canonical Term. *A term is said to be in a canonical form if it is one of the following: 1) $CT()$ and 2) $CT(C_i)$, where CT is a credential type and C_i is a single attribute name or a condition defined over a single attribute.*

Each term not in canonical form can be decomposed into a set of canonical terms.

Example 3. The following are examples of terms:

$$\Diamond Id\_Card(Lastname, Residence = America),$$
$$\Diamond Work\_Badge(Company \in \{Fly, Sun, Atour\}).$$

The canonical form of the first term is: $Id\_Card(Lastname)$, $Id\_Card(Residence = America)$, while the second term is already in canonical form.

We are now ready to define disclosure policies.

Definition 4.4: Disclosure policy. *A* disclosure policy *(DP for brevity) is an expression of the form:*[4] *1) $R^4 \leftarrow T_1, \ldots, T_h, h \geq 1$, where: 1) R denotes a target resource, and 2) $T_1, \ldots, T_h$ are terms, defined according to Definition 4.2.*

Example 4. With respect to our running example, examples of disclosure policies for a SingleTrip are:

$DP1 : SingleTrip \leftarrow Id\_Card$
$(Lastname, Residence = America),$
$Work\_Badge(Company \in \{Fly, Sun, Atour\}),$
$Passport(Sex = female)$

$DP2 : SingleTrip \leftarrow Student\_card(Lastname),$
$ResidenceCertificate$
$(issuer \in \{Alaska, Alabama, Baltimora, etc..)\},$
$Work\_Badge(Company \in \{Fly, Sun, Atour\}),$
$DriverLicense(Sex = female)$

$DP3 : SingleTrip \leftarrow WorkCertificate(Lastname),$
$Id\_Card(country = America), Badge$
$(Company \in \{Fly, Sun, Atour\}),$
$Passport(Sex = female)$

$DP4 : TripPackage \leftarrow WorkCertificate$
$(Qualification = agentSeller,$
$Company \in \{Fly, Sun, Atour\})$

$DP5 : TripPackage \leftarrow WorkCertificate$
$(Qualification = agentSeller),$
$Payroll(Company \in \{Fly, Sun, Atour\})$

$DP6 : TripPackage \leftarrow Work\_Insurance$
$(Qualification = agentSeller),$
$WorkCertificate(Company \in \{Fly, Sun, Altour\}).$

4.3 Implementing Abstract Trust Requirements

Trust requirements for a resource can be implemented by a number of alternative disclosure policies. All of these disclosure policies achieve the same goal: to prove the properties required by the trust requirements. This is because the same conditions/properties stated in the TR can be expressed through alternative terms, using equivalent attributes/conditions and credentials. The techniques to be used for finding equivalent terms strictly depend on the kind of term being considered. For instance, when the TR does not contain conditions, finding equivalent terms for its implementation is quite straightforward, since this process can be driven by the reference ontology. (For instance, if the property sex is required, either term *passport(sex)* or *Id_Card(gender)* can be used). In general, given a property, each term built by using an element in the Langset component of the corresponding concept can be used for its implementation. By contrast, when the TR contains conditions, finding equivalent attribute conditions for their implementation is not always trivial.

For instance, consider the condition $age > 25$. This condition can be implemented by two equivalent terms: $Passport(age > 25)$ and

$$DrivingLicense(yearOfBirth < 1979).$$

In such a case, a set of predefined inference rules can be adopted for mapping among equivalent attribute conditions. We will further elaborate on this in Section 6. In the following, we use the notation $\Leftrightarrow$ to indicate semantically equivalent attribute conditions, whereas we use the notation $\Rightarrow$ to indicate that satisfaction of one condition implies satisfaction of another (e.g. $age > 45 \Rightarrow age > 20$).

The definition below captures how abstract trust requirements can be implemented by a set of disclosure policies. In the definition and throughout the paper, we denote with $\mathcal{C}_X$ the concept X belongs to, where X is either a property or an element in the Langset component of the concept.

Definition 4.5: Trust Requirement Implementation. *A disclosure policy* $\mathrm{DP} = R \leftarrow T_1, \ldots, T_n$, $n \geq 1$, *implements an abstract trust requirement*

$$TR = \langle R, properties, conditions \rangle$$

if the following conditions hold:

- $\forall (p\ op\ k) \in conditions$, *there exists a canonical term T_i of the form $CT(A_i\ op'\ k')$, obtained from a term in $\{T_1, \ldots, T_n\}$, such that $CT.A_i \in LangSet$, where $LangSet$ is the set of credential types and attributes associated with concept $\mathcal{C}_p$ and $A_i\ op'\ k' \Rightarrow p\ op\ k$.*
- $\forall p \in properties$, *there exists a canonical term T_i obtained from a term in $\{T_1, \ldots, T_h\}$, such that if T_i is of the form $CT()$, $CT(A_i)$, or $CT(A_i\ op\ k)$, then $CT() \in LangSet$ or $CT.A_i \in LangSet$, where $LangSet$ is the set of credential types and attributes associated with concept $\mathcal{C}_m$ and $\mathcal{C}_m \preceq \mathcal{C}_p$.*

The above definition allows a single trust requirement to be implemented by a number of different disclosure policies. This is possible because a property listed in an abstract trust requirement can be proved by different credentials/attributes. It may ask for attributes and credentials corresponding to the properties in the trust requirement or they can ask for attributes and credentials corresponding to more specialized properties. Indeed, as stated in the second condition of Definition 4.5, a disclosure policy can also be implemented using more specialized concepts than the requested ones. The next example illustrates disclosure policies implementing trust requirements.

4. An extended notation supporting the parametrization of R based on the attributes to be disclosed has been presented in [2]. For simplicity, in the presentation, we do not use such extended format here. Notice, however, that the use of a simpler format does not lead the generality of our results, as the attributes to be used as parameter criteria are always disclosed in

Example 5. The disclosure policies shown in Example 4 are examples of policies implementing the trust requirements of Example 2. In particular, DP1 and DP3 are trivial implementation of the trust requirement for $SingleTrip$, since the properties required in such TR are implemented by terms generated using elements of the LangSet of the concepts the properties belong to. DP2, instead, requires a more specific information than those required by the previous ones, in term

$$ResidenceCertificate$$
$$(issuer \in \{Alaska, Alabama, Baltimore, ..)\}.$$

By receiving the credential satisfying such term, however, the requestor can unambiguously infer American residence of the submitter, since concept $State$ is a specialization of $Country$. As such DP2 implements the same abstract trust requirement as DP1 and DP3.

Disclosure policies can thus be generated from trust requirements by considering the Langset components of each concept corresponding to a TR property, and using one of its elements for expressing the required property/condition through credentials/attributes or attribute conditions. The process is trivial if the TR does not contain any condition. Otherwise, conditions against the properties have to be properly translated into attributes conditions. Such process must be carefully managed since some conditions might be translated into a number of equivalent attribute conditions. In this paper, we assume the use of specific techniques to perform such mappings, based on different methods depending on the type of conditions being considered (see Section 6 for more details).

Intuitively, the process of generating disclosure policies may result in a large number of alternative policies implementing a single trust requirement. We refer to the set of all the possible policies implementing a trust requirement TR as *complete set* for TR. The credential requestor thus has the possibility of selecting the policies being used among a large variety, and select either one, a subset, or all the policies of the set, according to its way of approaching trust. In this paper, however, we do not discuss the criteria on which the requestor makes such a decision However, we plan to investigate this issue as future work. A further advantage of having several alternative policies is that if a credential submitter chooses not to release or does not have the credentials requested by a disclosure policy DP_i, it can give the credentials requested by a disclosure policy DP_j implementing the same trust requirement (admitted by the submitter) and continue with the trust negotiation. We further elaborate on this aspect in the next section.

4.4 Relationships between Disclosure Policies

Since different disclosure policies can implement the same abstract trust requirement, a credential requestor needs to know the relationships between these different disclosure policies. Two disclosure policies implementing the same high-level trust requirement can be related by *equivalence* or *stronger than* relation. To formalize these relationships, we need first to introduce the notion of equivalence of canonical terms.

Definition 4.6: Equivalence of canonical terms. *Let T_i and T_j be two terms in canonical form. T_i is equivalent to T_j, written $T_i \equiv T_j$, if one of the following conditions hold:*

- $T_i = CT(C), C = a\ op\ k$, $T_j = CT'(C'), C' = a'\ op'\ k'$, *and* $\mathcal{C}_{CT.a} = \mathcal{C}_{CT'.a'} \wedge a\ op\ k \Leftrightarrow a'\ op'\ k'$;
- $T_i = CT()$, $T_j = CT'()$, *and* $\mathcal{C}_{CT} = \mathcal{C}_{CT'}$;
- $T_i = CT(C), C = a$, $T_j = CT'(C'), C' = a'$, *and* $\mathcal{C}_{CT.a} = \mathcal{C}_{CT'.a}$;
- $T_i = CT()$, $T_j = CT'(C'), C' = a\ op\ k$ *and* $CT() \Leftrightarrow a\ op\ k$;
- $T_i = CT()$, $T_j = CT'(C'), C' = a$ *and* $\mathcal{C}_{CT()} = \mathcal{C}_{CT'.a}$.

Example 6. The terms $School\_badge(age > 18)$ and $Id\_Card(date\_of\_birth < 1986)$ are equivalent because the attributes $School\_badge.age$ and $Id\_Card.date\_of\_birth$ are in the $LangSet$ of the concept age, and $age > 18 \Leftrightarrow date\_of\_birth < 1986$.

Two disclosure policies implementing the same trust requirements can be related by the stronger than relationship. This happens when one of the two refers to more specialized terms.

Definition 4.7: Stronger than relation. *Let $DP : R \leftarrow T_1, \ldots, T_n$ and $DP' R' \leftarrow T'_1, \ldots, T'_k$ be two disclosure policies. Let $\mathbf{T}$ be the set of canonical terms generated from $T_1, \ldots, T_n$, and let $\mathbf{T}'$ be and the set of canonical terms generated from $T'_1, \ldots, T'_k$. DP is said to be stronger than DP', denoted by $DP \succeq DP'$, iff $R = R'$, and $\forall T'_j \in \mathbf{T}'$, $\exists T_i \in T$, such that either $T_i \equiv T'_j$ or $\mathcal{C}_{T_i}$ is a specialization of $\mathcal{C}_{T'_j}$, where $\mathcal{C}_{T_i}$ ($\mathcal{C}_{T'_j}$) denotes the concept to which term T_i (T'_j) refers to.*

The above definition says that DP is stronger than DP', if for each canonical term T'_j in DP' there is a term T_i in DP such that T_i is equivalent to T_j or T_i is associated with a concept $\mathcal{C}_{T_i}$ that is a specialization of the concept $\mathcal{C}_{T'_j}$ that is associated with T'_j.

Example 7. Consider the following disclosure policies:

$DP1 : CoupleTrip \leftarrow Id\_Card$
$(Lastname = Rossi, dateOfBirth > 01/01/1978, status = married)$,
$Passport(citizenship = American)$,
$DP2 : CoupleTrip \leftarrow Passport(citizenship = American)$,
$Id\_Card(dateOfBirth > 01/01/1978, status = married)$
$DP3 : CoupleTrip \leftarrow Id\_Card$
$(dateOfBirth 01/01/1978, city = FortCollins)$,
$MarriageCert()$
$DP3 \succeq DP1$ *as proved by the following relations* :
$Id\_Card(dateOfBirth 01/01/1978) = id\_card$
$(dateOfBirth 01/01/1978)$,
$Id\_Card(status = married) \equiv MarriageCert()$
$Id\_Card(city = FortCollins) \succeq Passport$
$(citizenship = American)$.

As shown, all terms of $DP1$ either have an equivalent or specialized term in $DP3$ (but the converse does not hold).

Definition 4.8: Equivalence relation. *Two disclosure policies DP and DP' are equivalent, denoted as $DP \equiv DP'$, iff DP is stronger than DP' and DP' is stronger than DP.*

Example 8. Consider the following disclosure policies for the Singletrip service of our running example:

$$DP_1 : SingleTrip \leftarrow credit\_card \\ (expiration\_date > current(), name = Mary), \\ Id\_Card(country = US), MarriageCert();$$

$$DP_2 : SingleTrip \leftarrow VISA \\ (expiration\_datecurrent(), name = Mary) \\ driving\_licence(country = US), \\ Id\_Card(status = married).DP_1 \equiv DP_2,$$

since

$$Id\_Card(country = US) \equiv driving\_licence \\ (country = US),\ credit\_card \\ (expiration\_date > current()) \equiv VISA \\ (expiration\_date > current()), \\ credit\_card(name = Mary) \equiv VISA(name = Mary),$$

and

$$MarriageCert() = Id\_Card(status = married).$$

As introduced before, a complete set for a TR contains all the disclosure policies implementing it. All such policies are related either by the equivalence or the stronger than relation.

To satisfy a disclosure policy, the submitter has to send the requestor a *disclosure set* (DSet), that is, a set of credential views conveying the required attributes and/or credentials. We assume that the submitter always selects the most blinded views of the credentials needed to satisfy the disclosure policy. Note that, it is possible that the most blinded view of the credential contains other attributes that are not requested by the disclosure policy, referred to as *nonrequested attributes*.

For simplicity, in the following examples, we represent a DSet as the set of nonblinded attributes contained in the corresponding views.

Example 9. Consider disclosure policy $DP1$ of Example 4, and suppose that the $Passport$ serial number cannot be blinded. The corresponding disclosure set, denoted by $DSet$, is

$$\{Id\_Card.Lastname, Id\_Card.Residence, \\ Work\_Badge.Company, Passport.sex, Passport.number\}.$$

Here, Passport.number is an example of nonrequested attribute, whereas the remaining are all requested attributes.

5 Privacy in Trust Negotiations

Privacy is a very subjective notion. Some users may consider a specific concept, say age, as private, whereas others may not. To ensure privacy, each user has to classify which concepts are private and which are nonprivate. Sometimes, several nonprivate concepts when combined together may disclose private information. For instance, the concept $name$ or the concept $salary$ when considered in isolation are not private. However, when they are considered together, they disclose some information which is likely to be considered private by most subjects. To address the problem of identifying concept combinations (and corresponding attributes/credentials) that may violate privacy, we propose the notion of *private concept groups* (PCgs for short), described in the following section.

5.1 Private Concept Groups

A private concept group contains a set of concepts that together result in a privacy breach. Note that a privacy concept group can be a singleton consisting of one concept only. In such a case, the concept is a *private concept*. To protect against the disclosure of private information, all concepts in a private concept group should not be released to the same counterpart.

Example 10. With respect to our running example, possible private concept groups for Alice might be:

$$\{PersonId, Residence, Sex\}, \\ \{Company\_Employee, EmployeeId\}, \{Age, Sex, \}.$$

With this background, we are ready to formalize privacy preserving disclosures.

Property 1: Privacy-preserving disclosures. *Let s be a submitter, $OldDSet$ be the set of all attributes and credentials already released to a requestor, and $DSet$ be the current disclosure set. Moreover, let AS_{DSet}, CT_{DSet} be the set of attributes and credential types appearing in $DSet$, and AS_{ODSet}, CT_{ODSet} be the set of attributes and credential type names appearing in $OldDSet$. Let $PCg\text{-}Set = \{PCg_1, \ldots, PCg_n\}$ be the set of private concept groups defined by s. $DSet$ is privacy-preserving with respect to s if the following condition holds:*

- $\forall PCg_j \in PCg\text{-}Set$, $\forall c_i \in PCg_j$, *if c_i is mapped into an attribute name, then $C_{c_i}.LangSet_{c_i} \cap (AS_{Dset} \cup AS_{ODSet}) = \emptyset$, else, if c_i is mapped into a credential type name, then $C_{c_i}.LangSet_{c_i} \cap (CT_{Dset} \cup CT_{ODSet}) = \emptyset$.*

The condition above ensures that $DSet$ does not contain any element that discloses any PCgs of the $PCg\text{-}Set$.

5.2 Forming Private Concept Groups

Having formalized privacy-preserving disclosures, we need to say how an entity can form its private concept groups. Since privacy is subjective, each subject is expected to provide its private concept groups. One possibility is to evaluate all possible private concept groups without taking into account the counterpart that the entity is dealing with during a specific negotiation. This approach has, however, a high overhead when dealing with a large number of private concept groups—all these groups must be monitored and controlled to prevent privacy breaches while negotiating trust. We choose the alternative approach of managing the private concept groups on the basis of the privacy practices of the counterpart. These privacy practices can be obtained from the counterpart's privacy policies.

Privacy policies, in general, state who the *recipients* will be for the user *data*, the *purpose* for which this data will be used, and how long the data will be *retained*. Data in a

privacy policy can belong to different granularity levels. In this work, we assume that privacy policies refer to data at a single level of granularity—the data element.[5] The data element refers to smallest granularity data. Examples of data element are social security number and lastname. In the following, we use the P3P syntax for describing privacy policies. In our context, data elements actually correspond to concepts. A privacy policy is a set of privacy statements of the form $< data, ret, recip, purp >$, where:

- $data$ denotes a data element,
- ret denotes the type of retention and assumes values in

$$\{no\text{-}retention, stated\text{-}purpose, legal\text{-}requirement, business\text{-}practice, indefinitely\}$$

  according to the P3P standard taxonomy,
- $recip$ is the legal entity, or domain, beyond the service provider and its agents where data may be distributed; $recip$ assumes values in $\{ours, legal, delivery, unrelated, \ldots\}$,
- $purp$ denotes purposes for data processing; $purp$ assumes values in

$$\{current, admin, tailoring, pseudoanalysis\}$$

  according to the P3P taxonomy.

Example 11. To sell package trips, $Summer$ needs to obtain certain information from the incoming customers, and store them until the reservation is not confirmed. Moreover, $Summer$ also adopts a privacy policy stating that it offers personalized trip recommendations for customer buying single trips, for which it collects customer personal information. Representation of such data practices are summarized in what follows:

$$< JobTitle, current, ours, stated\text{-}purpose >,$$
$$< company, current, ours, stated\text{-}purpose >$$
$$< gender, contact, same, business\text{-}practices >,$$
$$< name, contact, same, business\text{-}practices >$$
$$< country, contact, oursbusiness\text{-}practices > .$$

Since P3P has not been specifically conceived for negotiations, its syntax include data elements that are not of interest to trust negotiations, such as $click\text{-}stream$. In the following, we always limit our analysis to elements having a corresponding concept in our reference ontology. Such data elements can then be used to evaluate private concepts. In particular, we can classify private concept groups into two different categories—*strong* and *weak*—depending on the counterpart data practice. If a private concept group belongs to the strong category, a disclosure of this group will result in a serious privacy breach and should thus be prevented. By contrast, disclosing all concepts in a private concept group that belongs to the weak category results in breach of privacy that is not desirable, but is nevertheless acceptable.

The classification is based on how long the data will be retained by the counterpart and whether or not the data will be released to third parties. For instance, if the data is not released to third parties or has a shorter retention time, we may have weaker controls on the dissemination of private information. Otherwise, we need stronger controls. The data element in a privacy policy is marked as strong or weak on the basis of *purpose*, *recipient*, and *retention*, specified in the privacy policy. This is illustrated in Table 1. The classification can be customized by the user and modified when needed. The concept associated with the data element is classified as the data element itself. If any concept in a private concept group is classified as strong, then the entire group is classified as strong. Otherwise, the private concept group is marked as weak. If no privacy policies can be found for any concept of a private concept group, we assume the default mark of the PCg be strong.

We are now ready to show how a subject can classify its private concept groups:

1. P3P policies of interest are extracted and modified such that each policy refers to a single data element.
2. P3P data elements are filtered from the statements to remove data elements not related with any concept.
3. Each data element is labeled strong or weak, according to the categorization given in Table 1.
4. Correspondences are created between private concepts and the corresponding data elements.
5. A PC_g is marked as strong ('s', for short) if it contains at least one concept linked to a strong data element; otherwise, it is marked as weak ('w', for short).

Example 12. Consider the private concept groups of Example 10. According to the privacy policies sketched in Example 11, the first private concept group has to be labeled as strong, whereas the second is a weak private concept group.

5.3 Techniques for Achieving Privacy

Once a credential submitter has specified his/her private concept groups, he/she can verify whether the disclosure set that he/she is about to reveal is privacy preserving, as stated by Property 1. In case the disclosure set is not privacy preserving, the submitter can apply the methods we present in the following to find (if possible) an alternative privacy preserving disclosure set. To this purpose, we propose two techniques: generalization and substitution. The first should be applied when a privacy breach occurs due to requested attributes, whereas the substitution technique is applied when the privacy breach is due to non requested attributes.

5.3.1 Generalization

This technique is used when the disclosure set is not privacy preserving and contains one or more requested attributes whose disclosure causes a privacy breach, as shown by the following example.

Example 13. Suppose that Alice is about to disclose the following DSet:

$$\{Id\_Card.Lastname, Id\_Card.Residence, Work\_Badge.Company, Passport.sex\}.$$

5. We adopt the terminology of the P3P standard [15].

TABLE 1
Data Element Categorization on the Basis of Privacy Policies

type	label	purpose	recipient	retention
strong	s	pseudo-decision, contact, historical, telemarketing, other purposes	other-recipient, public unrelated	business-practices, indefinitely
weak	w	admin, develop, current, tailoring, pseudo-analysis	ours, same, delivery	legal-requirement, no-retention stated-purpose

These attributes belong to the LangSet of four concepts: PersonId, Residence, CompanyEmployee, and sex, denoted in the following as $\mathcal{C}_i$, $\mathcal{C}_j$, $\mathcal{C}_m$, and $\mathcal{C}_k$, respectively. According to Alice private concept group set (see Example 10) there is a PCg protecting the concepts related to identity, gender, and residence. Thus, disclosure of attributes $Id\_Card.Lastname$, $Id\_Card.Residence$, and $Passport.sex$, will result in a privacy breach. Thus, to protect privacy either $\mathcal{C}_i$ or $\mathcal{C}_j$, must be generalized. For instance, to generalize concept Residence, $Id\_Card.Residence$ can be replaced with Id$_Card.Country$ or $Id\_Card.Region$.

Generalization of an attribute or credential involves replacing it with an alternative one, such that the latter contains a less specific information than the former.

Generalization can be executed by using the reference ontology, where concepts are related by the generalization relationship. Additionally, the ontology keeps track of the credentials/attributes that may implement a given concept, and, thus, can be used to determine which is the attribute/credential that generalizes the given attribute/credential. However, generalization should be also driven by the profile of the involved entity in that the more general attribute/credential should be selected among those belonging to the profile. To support generalization, we thus make use of a graph structure, that we call *concept graph* (*C-Graph*, for short), which is built starting from the ontology and adding information on which attributes/credentials of a subject profile implements a concept. A C-Graph contains only concepts specified in private concept groups of the corresponding subject, because only these concepts may be involved in a generalization process. Note moreover that, since the ontology may include some unrelated concepts, we may have multiple concept graphs associated with a user. The formal definition of a C-Graph is as follows.

Defintion 5.1: Concept Graph. *Let $\mathcal{O}$ be a reference ontology, and $Prof$ be the profile of a subject s. Let $PCg\text{-}Set$ be the set of concept groups for s. A concept graph $\mathcal{CG} = \langle \mathcal{N}, \mathcal{E} \rangle$ for s is an acyclic directed graph built from the ontology $\mathcal{O}$ satisfying the following conditions:*

- *$\mathcal{N}$ is a set of nodes where each node n_i is associated with a concept $\mathcal{C}_i =< Name_i, KeywordSet_i, LangSet_i >$ in $\mathcal{O}$ and $\mathcal{C}_i \in PCg$, $PCg \in PCg\text{-}Set$. Each node n_i is labeled with $< Name_i, Prof_i >$. $Name_i$ identifies the concept $\mathcal{C}_i$ associated with node n_i. $Prof_i$ is the set of attributes and credentials belonging to s that describes concept $\mathcal{C}_i$.*
- *$\mathcal{E}$ denotes a set of directed edges. For each edge $e = (n_i, n_j) \in \mathcal{E}$, the concept $\mathcal{C}_j$ corresponding to node n_j is a specialization of concept $\mathcal{C}_i$ corresponding to node n_i in the reference ontology.*

Fig. 1 shows some examples of C-Graphs. Generalization is executed by traversing the graph from a node to its ancestor. When a node has more than one ancestor, a selection is performed to determine the best choice.

We now give the algorithms needed to perform generalization. In the algorithms and in the remainder of this section, we adopt the notations shown in Table 2. Generalization is executed by function $Generalization$ (presented in Algorithm 2 in Fig. 3) which uses the concept graph to perform the generalization. Such function takes as input a disclosure set *DSet* that has to be transformed into a privacy preserving disclosure set, the set of concepts that have been revealed in the past negotiations, the set of all private concept groups, denoted by PCg-Set, and the profile of the submitter. The function begins by extracting the requested attributes from the input Dset. Once such attributes have been determined, the function maps each requested attribute and credential composing the DSet into the corresponding concept in the input graph. The resulting set, called *concept disclosure set* ($CDSet$ for short), is then compared with the input private concept groups (PCgs) to check whether $CDSet$ contains elements that will reveal all the elements of a PCg. If this is the case, for each PCg that will be disclosed by elements of CDSet, a generalization process is executed to prevent privacy breach. The process of generalization consists of the following steps. First, function *WhichGeneralize* (see Algorithm 1 in Fig. 2) identifies the concept that will be generalized among the ones in the considered PCg-Set. Once the concept to be generalized has been identified, the second step, performed by function $WhichAncestor$ (Algorithm 4 in Fig. 6), identifies the ancestor node in the concept graph to be used for the generalization process. Then, the final step consists of selecting a suitable attribute or credential for implementing the generalized concept. This is executed by function $BestCandidate$. We now detail the functioning of such functions.

Selection is performed by Function $WhichGeneralize$, which drives the choice to those concepts that have not been released in the past negotiations and that minimize the number of attributes or credentials to be generalized. Let us explain this with an example.

Example 14. Consider the PCgs of Example 10. Since we have $\langle\{Company\_Employee, EmployeeId\}\rangle$, if Alice has previously revealed her $EmployeeId$ and her Age (or any of its specializations), the concept names $\{Company\_Employee\}$ and Sex are to be generalized. Second, it is desirable to generalize the concept name that occurs in multiple PCgs. Here, we have that concept *Sex* appears twice. In this case, *Sex* is the first concept the function tries to generalize. This is because only one generalization is sufficient to avoid the breach of two different PCgs. Note that if generalization is not possible, as for concept *Sex*, we try with another concept, such as *Residence*, in our example. If more than a candidate is possible for generalizing $Residence$, (ZIP and Street Name according to the Cgraph sketched in Fig. 1), the one less used in previous negotiations and

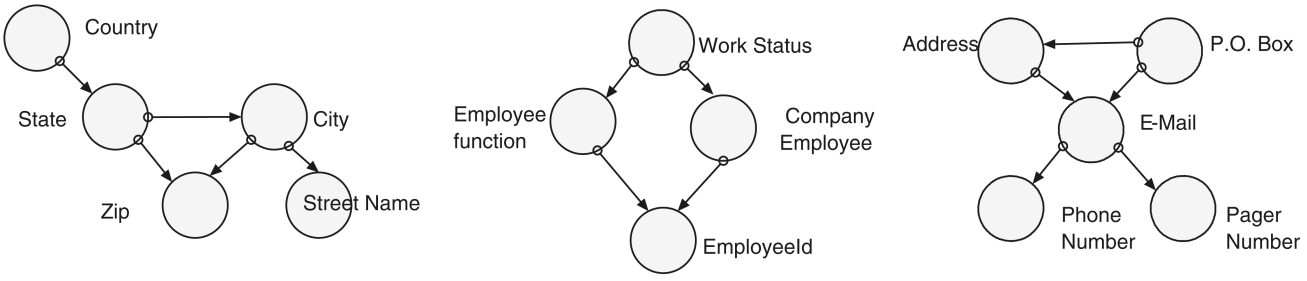

Fig. 1. Example of C-Graphs.

Algorithm 1 $WhichGeneralize(OldDiclosures, PCg\text{-}Set, PCg)$

Input: $OldDisclosures$: set of concepts already revealed
$PCg\text{-}Set$: set of PCgs of the submitter
PCg: a private concept group
Output: a: concept to be generalized (if it exists), $\emptyset$ otherwise

```
(1)  foreach c ∈ PCg
(2)      if c is marked
(3)          PCg = PCg − c
(4)  while PCg ≠ ∅
(5)      Let con be the concept appearing the maximum number times in PCg-Set
(6)      Let G_con be the C-Graph conveying con and let n_con be the node in G_con labelled with con
(7)      if HasAncestor(n_con, G_con) ∧ ((Descedendants(n_con, G_con) ∩ OldDisclosures) = ∅)
(8)          return c
(9)      return ∅
```

Fig. 2. Algorithm 1. $WhichGeneralize(OldDiclosures, PCg\text{-}Set, PCg)$.

not interfering with other PCgs is selected.

The task of choosing the ancestor to use for generalization is performed by the recursive function $WhichAncestor$. Function $WhichAncestor$ starts by checking if any of the concepts associated with the set of ancestors of the node to be generalized has already been revealed. Each time a concept is revealed it is marked in each PC-g it appears. The marking is done by function **Mark** (line 37 of Function Generalization), and denoted with symbol $*$ in our examples. Intuitively, when a concept has been revealed, all the attributes and/or credentials of the associated $LangSet$ can be freely disclosed, since there is no need to further protect that type of information. In such a case, the ancestor that has already been revealed is used to generalize. If this is not the case, the function returns the most privacy preserving ancestor, quantified by evaluating the $DynCost$ parameter associated with the ancestor. The concept of $DynCost$ is formalized by the following definition.

Definition 5.2: Dyncost. *Let $PCg\text{-}Set$ be a set of private concept groups. Let $\mathcal{C}_i$ be a concept appearing in at least one PCg of the $PCg\text{-}Set$. The Dyncost associated with $\mathcal{C}_i$ is computed as one less the minimum number of unmarked concepts in any private concept group of $PCg\text{-}Set$ containing $\mathcal{C}_i$.*

Example 15. With respect to our running example, consider the concept PersonId that is associated with a node of Alice C-Graph. Suppose that Alice has, in addition to the PCgs shown in Example 10, the following PCg:

$$\{\langle\{PersonId, residence^*, sex^*\}, s\rangle, \langle\{PersonId, medicalStatus\}, s\rangle\}.$$

The first PCg has one element not marked and the second has two. The minimum number of unmarked elements is one, as such DynCost for the above private concept group is 0.

Note that $DynCost$ of a concept equals to zero implies that disclosing the concept results in a privacy breach. This is because a private concept group consisting of all marked elements causes disclosure of private information. A higher value of $DynCost$ associated with a concept indicates that the PCgs related to that concept have more unmarked elements. In other words, selecting a concept having a high value of $DynCost$ reduces the need of generalization in future disclosures. For generalizing a concept, we choose the ancestor from the ancestor set having the highest value of $DynCost$. Suppose the $DynCost$ to be zero for all the ancestors in the set; then, it means disclosing any ancestor will result in a privacy breach. In such a situation, we recursively traverse the C-Graph to get the ancestors at the next level and repeat the process. Note that a suitable ancestor may not always exist. This may happen when the node does not have an ancestor in the concept graph or revealing any ancestor results in a privacy breach. In such a case, we do not generalize but evaluate whether disclosing the attribute or credential results in a serious privacy breach or in a less damaging privacy breach. Recall that a PCg is labeled either with "s" or "w" to indicate whether disclosing the corresponding attributes and credentials is acceptable or not. Now, if the concept associated with the attribute (or credential) is present only in PCgs labeled with a "w," then the attribute (or credential) can be released. Otherwise, the attribute (or credential) should not be released, as such, the function returns an empty set, causing the abort of the negotiation.

Finally, when more than one candidate can be used for generalization function *BestCandidate* is invoked. First, the function tries to select among the views already released. If more than one view have already been released, the

attribute or credential is randomly chosen among these views. By contrast, if no view has already been released, the most blinded view available (among the possible ones) is selected. The selected attribute or credential can replace the old one in the DSet. Once the DSet is ready to be disclosed a marking operation is executed. This operation involves the most blinded views that convey the elements of the DSet. For each view, the attributes in clear are selected and the corresponding concepts detected and marked in the PCgs of the PCg-Set they appear. Marking a concept in a PCg implies that the concept has been revealed to the counterpart. Thus, if all the concepts in a PCg are marked, a privacy breach has occurred. Our algorithms ensures that this does not happen.

Example 16. Consider the following views of the credentials Id_Card, $Work\_Certificate$, and $Passport$:

1.

$$Id\_Card = \{Lastname : Ross, Name : \bullet\bullet\bullet\bullet, Address : \bullet\bullet\bullet\bullet, Residence : USA, Date\ of\ Birth\ 17.11.1978\}.$$

2.

$$Work\_Badge = \{\bullet\bullet\bullet\bullet, Name : \bullet\bullet\bullet\bullet, CompanyName : Fly\}.$$

3.

$$Passport = \{Number\ 51243G, Lastname : Ross\ \bullet\bullet\bullet\bullet, Name : \bullet\bullet\bullet\bullet, Sex : Female, Residence : America\}.$$

Now, suppose that the views have been selected to respond to the disclosure policy DP1 of Example 4.

That implies that the concepts corresponding to the attributes revealed have to be marked in every PCg they appear. As such according to Alice PCg-Set of Example 10 the marking is as follows:

$$\{\{PersonalInformation^*, Residence^*, Gender^*\}, \{Company\_Employee^*, EmployeeId^*\}, \{Age^*, Sex^*\}\}.$$

Clearly, these private concept groups result in a privacy breaches and thus need to be generalized.

5.3.2 Substitution

For preventing privacy breaches resulting from nonrequested attributes, instead, the technique of substitution is used. Such technique is based on replacing views resulting in breaches because of their nonrequested attributes with views showing the same requested attributes and limiting any side effect deriving from non requested ones. The substitution process is performed by function *Substitution*, presented in Algorithm 3 in Fig. 4.

Function *Substitution* starts with an analysis of the views to be disclosed to check whether any of them has to be substituted. Precisely, given a set of views, the function first extracts from the views the nonrequested attributes. The resulting set is analyzed to verify whether some combinations of these attributes result in privacy breaches. If this is the case, the function first tries to replace the compromised view with a view of the same type showing a different combination of nonrequested attributes in clear, if possible. Otherwise, the view might also be replaced by an equivalent one, that is, a view conveying equivalent requested attributes, of the same or of a different credential types. The resulting set of views is again analyzed and the process iterated until there are no views resulting in privacy breaches due to non requested attributes. We make use of a function called $PrivacyBreach$ (lines 1 and 22) that, given a set of views and a PCg-Set, returns the PCg that is compromised by the disclosure of the set of views, if any. If more than one PCg is found, it returns the first one. $PrivacyBreach$ is invoked until there are no broken PCg by the set of views to be disclosed.

Example 17. Consider the views of Example 16. Attributes in clear such as passport number and date of birth are nonrequested attributes. A private concept group has been specified by Alice protecting concepts age and sex (see Example 10). As such, the views must go through a process of substitution. For instance, instead of using the credential Id_Card, the credential Passport could be used, if Alice has a view of Passport that conveys semantically equivalent requested attributes in clear, like the following one: {Number: $51243g$, Surname: Ross, Name: • • • •, Nationality: American, Date of birth: • • • •, Sex: M, Place of birth: • • • •, Date of issue: • • • •, Date of expiry: • • • •}. Note that the passport number may not be blindable (see Section 3), but its disclosure is not prevented by any private concept group, and thus can be freely disclosed.

5.4 Formal Results

Next, theorems prove correctness and give complexity results of the proposed algorithms. due to space limitations we do not report here formal proofs. Interested readers may find them in [?].

Theorem 5.1. *Let DSet be a disclosure set, PCg-Set be the set of private concept groups of the submitter, OldDisclosures be the set of concepts already revealed and Prof be the submitter profile. The disclosure set DSet′ generated by function Generalization satisfies Property 1.*

In the following theorems, we prove the efficiency of the proposed algorithms. First, we prove that *Generalization* function is polynomial with respect to the number of private concept groups associated with a subject. Then, we show that the algorithm performing the substitution technique is linear with respect to the cardinality of the submitter's Profile.

Theorem 5.2. *The computational complexity of function Generalization is $O(n^2pd)$, where n is the cardinality of the PCg-Set of the submitter, p is the number of ancestors of the node labeled with the concept to be generalized, and d is the width of the C graph used for the generalization process.*

Theorem 5.3. *The computational complexity of function Substitution is $O(tjn)$, where n is the cardinality of the PCg-Set of the submitter, j is the number of nonrequested attributes of the input DSet, and t is the cardinality of the submitter's Profile.*

Algorithm 2 $Generalization(DSet, OldDisclosure, PCg\text{-}Set, Prof)$

Input: $DSet$: disclosure set to be made privacy-preserving
$OldDisclosures$: set of concepts already revealed
$PCg\text{-}Set$: set of PCg of the submitter
$Prof$: submitter profile
Output: $DSet'$: set of disclosures preserving privacy or Abort if generalization is not possible

(1) Let $nrAttr$ be the set of non requested attributes of *DSet* with respect to the disclosure policy to satisfy
(2) $DSet' = DSet\text{-}Nr_{attr}$, $CDSet = \emptyset$
(3) **foreach** $e \in DSet'$
(4) $CDSet = CDSet \cup \mathcal{C}_e.KeywordSet_e$
(5) $AllPCgProcessed = False$
(6) **while not** $AllPCgProcessed$
(7) $PCg\text{-}Set'$ = $PCg\text{-}Set$
(8) **while** $PCg\text{-}Set' \neq \emptyset$
(9) get a PCg from the PCg-Set
(10) **if** $PCg \subseteq CDSet$
(11) $a = WhichGeneralize(OldDisclosures, PCg - Set, PCg)$
(12) **if** $a = \emptyset$
(13) **return** Abort
(14) **else**
(15) $CDSet = CDSet - a$
(16) $b = WhichAncestor(a, OldDisclosures)$
(17) **if** $b = \emptyset$
(18) **return** Abort
(19) **else**
(20) $CDSet = CDSet \cup b$
(21) $OldElement = \mathcal{C}_a.LangSet_a \cap DSet'$
(22) $DSet' = DSet' - OldElement$
(23) $NewElements = \mathcal{C}_b.LangSet_b \cap Prof$
(24) $e = BestCandidate(NewElements)$
(25) $DSet' = DSet' \cup e$
(26) **end while**
(27) **if** $PCg\text{-}Set' = \emptyset$
(28) $AllPCgProcessed = True$
(29) PCg-Set' = PCg-Set' - PCg
(30) $OldDisclosures = OldDisclosures \cup CDSet$
(31) **foreach** $e \in DSet'$
(32) Let v_e be the view conveying e
(33) **foreach** $attr \in v_e$
(34) $c = \mathcal{C}_{attr}.KeywordSet_{attr}$
(35) **foreach** $PCg \in PCg\text{-}Set$
(36) **if** $c \in PCg$
(37) **Mark** c in PCg if it is not marked

Fig. 3. Algorithm 2. $Generalization(DSet, OldDisclosure, PCg - Set, Prof)$

6 A System Supporting Privacy Preserving Trust Negotiations

In this section, we outline the architecture of a system supporting privacy preserving trust negotiations. In presenting the system we describe how the techniques presented so far can be combined for carrying out a privacy preserving negotiation process. The main components of the system and their interactions are presented in Fig. 7. As shown by the figure, besides the Policy Base and the local Profile of credentials, the main system components are the *Compliance Checker* module, the *Privacy Preserving* module, the *PCgs Classifier*, and the *Disclosure Policy Generator*. The Compliance Checker is the module in charge of checking whether the remote policies can be satisfied and transmitting the determined Disclosure Set to the Privacy Preserving module. The task of such module is processing the DSet to make it privacy preserving, if possible. Finally, the latter two modules are wizards used to classify private concept groups and create disclosure policies, respectively.

A trust negotiation always starts with a resource request submitted from a subject to the subject holding the resource. According to the general understanding of trust negotiations, we consider a trust negotiation as a peer to peer process and, thus, each subject is equipped with both policies and privacy rules, expressed through private concept groups.

Before starting the on line process, both negotiating subjects specify trust requirements for their resources. Further, each subject may also set the concept groups to be protected with respect to its own privacy preferences. Such preliminary operations are required but not essentials for proceeding with the negotiation. There might be several scenarios in which only one of the parties has private

Algorithm 3 $Substitution(DSet, PCg\text{-}Set, DP)$

Input: $DSet$: set of views to be made privacy-preserving
$Pcg - Set$: private concept group set, DP disclosure policy to satisfy
Output: $DSet$: set of privacy preserving views or an empty set

(1) **if** PrivacyBreach($DSet$, $Pcg\text{-}Set$)<> $\emptyset$
(2) $ViewSet' = ViewSet$
(3) Let $nrAttr$ be the set of non requested attributes of *DSet* with respect to the disclosure policy *DP* to satisfy
(4) SubnrAttr = $\emptyset$
(5) ContinueScanPCg= true
(6) **while** $PCg <> \emptyset$ and ContinueScanPCg= true
(7) Let c be a concept in a PCg
(8) $PCg = PCg - c$
(9) $SubnrAttr = SubnrAttr \cup C_c.LangSet_c \cap nrAttr$
(10) SubDSet = $\emptyset$
(11) **foreach** $a \in SubnrAttr$
(12) Let $view_a$ be the view conveying a in *ViewSet*
(13) $SubDSet = SubDSet \cup view_a$
(14) ContinueScanSvwSet=true
(15) **while** $SubDSet <> \emptyset$ and ContinueScanSvwSet=true
(16) $SubViewSet = SubDSet - view_a$
(17) Continue =true
(18) **while** *DSet* <> $\emptyset$ and Continue=true
(19) *% try to replace $view_a$ with an equivalent view of the same credential type that does not break any PCg*
(20) Let *view'* be a view in in *DSet*
(21) $DSet' = DSet - view'$
(22) **if** PrivacyBreach($DSet - view_a \cup view', PCg\text{-}Set) = \emptyset$
(23) $DSet = DSet - view_a \cup view'$
(24) Continue=false
(25) ContinueScanSvwSet=false
(26) ContinuePCg=false
(27) *%Comparison with the original view set*
(28) **if** $DSet = DSet'$ **then Return** $DSet'$
(29) **Return** *DSet*

Fig. 4. Algorithm 3. $Substitution(DSet, PCg\text{-}Set, DP)$

concepts to protect (this is the case, for instance, when one of the parties is a portal offering Web services) or others in which the resource requestor has no particular requirements over the credentials it is up to disclose. The case of both parties exchanging disclosure policies is nothing but a simple iteration of the process detailed below. Fig. 7 presents the main modules used in such a scenario by each of the involved subjects.

As soon as the connection between the two subjects is established, the resource requester retrieves the privacy policies of the counterpart and proceeds to label its private concept groups, as specified in Section 5.2. This operation is executed by the PCgs Classifier wizard of the credential submitter, shown in Fig. 7 (arrows 1, 2, and 3). The resource provider, instead, elaborates the request and extracts the related trust requirements. The trust requirements can then be used by the resource provider to dynamically generate the corresponding disclosure policies or to retrieve the corresponding disclosure policies, if they have been already generated. We can think of providers involved in a large number of negotiations for the same resources having the disclosure policies ready, whereas there might also be resource providers incurring in sporadic negotiations for certain resources. Such providers would thus just save the trust requirements and generate the disclosure policies on the fly, using the *Disclosure Policy Creator* module of Fig. 7. Generating disclosure policies from trust requirements implies mapping the concepts onto the trust requirements in the corresponding terms, using the associated $LangSet$ for determining which attributes/credentials are needed. Many options may be possible, since different equivalent terms may be used. As introduced in Section 4.3, the complexity of generating equivalent terms depends on the kind of terms involved. The most complex case is that of terms conveying conditions to be mapped using attributes having different domains, or cases where the mapping is not one to one but one to many (and vice versa). To address this issue, we assume that the ontology is complemented with some *inference rules* that help in generating equivalent terms. Let us illustrate this with a simple example. Consider the term $Passport(issuer = Europe)$. This term can be translated into two different conditions: if $Y \in EU$ and $Passport(issuer = Y)$. We assume such mapping can be successfully executed by using the appropriate inference rule. We are currently investigating this topic, and we will further elaborate on this and other related issues in our future work.

Once the disclosure policies have been generated (or extracted), they can be disclosed to the counterpart. We recall that, given a trust requirement, the disclosure policies that a resource provider may use for negotiating can be either the complete set of DPs (see Section 4.3), or a subset

Algorithm 4 $WhichAncestor(a, OldDisclosures)$

Input: a: concept to be generalized
$OldDisclosures$: set of concepts already revealed
Output: $b$generalized concept, $\emptyset$ generalization is not possible

```
(1)   Let AncSet be the set of ancestors of node n_a
(2)   if AncSet = ∅
(3)       return ∅
(4)   else
(5)       foreach s_b ∈ AncSet
(6)           if b ∈ OldDisclosures
(7)               return b
(8)       b = ∅
(9)       MaxDynCost = 0
(10)      foreach s_c ∈ AncSet
(11)          DynCost = CalcDynCost(c)
(12)          if DynCost ≠ 0 ∧ DynCost > MaxDynCost
(13)              MaxDynCost = DynCost
(14)              b = c
(15)      if b ≠ ∅
(16)          return b
(17)      %All the ancestors have DynCost = 0
(18)      if Sign(a) = "w"
(19)          % Sign(a) returns the sign of a's concept group
(20)          b = a
(21)          return b
(22)      else
(23)          foreach s_a ∈ AncSet : WhichAncestor(a, OldDisclosures)
```

Fig. 5. Algorithm 4. $WhichAncestor(a, OldDisclosures)$

of it. Note that given a trust requirement, the complete set of policies is characterized by all the possible policies providing a concrete representation of it. It includes not only the the policies implementing the properties required in the original trust requirement but also policies that are stronger than those ones. The resource provider can use the relationships of equivalence or stronger than to select the disclosure policies to be used, or it can select any of them according to minimality criteria (for instance, it can give priority to policies requiring the minimum number of credentials).

Upon receiving the disclosure policies, the *Compliance Checker* module of the resource requester (see Fig. 7) determines the corresponding disclosure sets.

If at least one of the determined disclosure sets is privacy preserving, it can be disclosed as it is and the negotiation can proceed. By contrast, if the privacy preserving property is violated by any DSet, an iterative process of generalization and/or substitution (see Sections 5.3.1 and 5.3.2) is applied to the DSets. The goal of such process is to determine an alternative privacy preserving DSet able to still satisfy the counterpart policies. Such functions are executed by the *Privacy Preserving* module. More in details, the generalization/substitution process is organized as follows. First, Function Generalization checks whether a privacy breach occurs due to requested attributes. If this is the case, the attributes are generalized. If the process fails, another DSet is considered. Otherwise, the DSet returned by the generalization function is checked to be privacy preserving. If the check fails, this means that privacy breaches occur due to non requested attributes, the substitution technique is applied. If such a process fails for all DSets, the negotiation fails. If the process of generalization and/or substitution successfully ends, the resulting disclosure set is sent to the counterpart. Note that, when more than one DSet can be generated, different criteria may be adopted for selecting the disclosure set to reveal. In general, the credential submitter is interested in sending information on a need-to-know basis. One further criterion that he/she may use is *credential minimality*—send the set that requests the minimum number of credentials. Alternatively, he/she may use the *attribute minimality property* and send the set having a minimum number of attributes.[6]

The disclosure can be executed either immediately or postponed at the end of the negotiation, if other disclosure policies protecting the credentials and attributes in the DSet have to be disclosed. The final part of the process thus consists of the credential receiver evaluating whether or not the received disclosure set satisfies the submitted disclosure policy or an alternative policy implementing the same trust requirement. The possibility of succeeding is not null, even if the disclosure set is not exactly the one satisfying the originally received policy. This is because the disclosure policy to satisfy might be a policy stronger than the minimal one and, thus, the substitutions and generalizations applied might not involve the attributes actually required for the trust requirement satisfaction. Further, as previously seen, if the policy was requiring more specialized attributes than those actually needed the generalization can be safely applied. Note that we always have assumed the resource requestor to have no knowledge of the underlying trust requirement. Other variations of this approach might also be considered, such, for instance, that of a requestor not relying on disclosure policies, and negotiating on the basis

6. In this paper, we do not further elaborate on such criteria, since the minimality properties are quite intuitive and easy to implement.

Algorithm 5 $PrivPresN(DP, PCg\text{-}Set, OldDisclosure)$

Input: DP: disclosure policy of the form $R \leftarrow \mathcal{T}(n_1) \ldots \mathcal{T}(n_k), k \geq 1$
$PCg\text{-}Set$: set of PCgs
$OldDisclosure$: set of concepts already revealed
Output: $DSet$: Dset or Abort if a failure occurs

```
(1)   Let Prof be the local Profile
(2)   while (DP=GetDpFromRequestor()<> ∅)
(3)     Let TSet_DP be the set of terms of DP
(4)     DSet = ∅
(5)     Continue = true
(6)     while (TSet_DP <> ∅ and Continue=true)
(7)       Let T_i = CT_i(C_i) be a term in TSet_DP
(8)       TSet_DP = TSet_DP − T_i
(9)       if ∃ view ∈ Prof of type CT_i satisfying C_i
(10)        Viewset = Viewset∪ view
(11)      else
(12)        GetNewDp=true
(13)        Continue= False
(14)    if GetNewDp=False
(15)      Dset= Generalization(DSet,PCg-Set, OldDisclosure, Prof)
(16)      if Dset<> ∅
(17)        DSet=Substitution(DSet, PCg-Set, DP)
(18)        if DSet <> ∅
(19)          Return DSet
(20)        else
(21)          Return Abort
(22)        else
(23)          Return Abort
```

Fig. 6. Algorithm 5. $PrivPresN(DP, PCg\text{-}Set, OldDisclosure)$

only of trust requirements. We will explore this and other approaches in our future work.

The above process can be iteratively repeated, if both parties have trust requirements for protecting the involved resources. If this is the case, the DSet to evaluate in order to establish the success of the negotiation will be replaced by a sequence of DSets alternatively belonging to the parties.

Correctness, termination, and efficiency of the approach presented here are proved by the following theorems. Formal proofs can be found in [?].

Theorem 6.1. *Let R be a resource requested by a requester subject Req from a resource provider, $Prov$. 1) Function PrivPresN terminates and 2) if there exists a privacy-preserving Disclosure set of Req satisfying any disclosure policy controlling access to R, then function PrivPresN determines it.*

Theorem 6.2. *The complexity of a privacy preserving negotiation for R is $0(n^2)$, where n is the cardinality of the PC-g set of the submitter.*

7 Related Work

Trust negotiation for Web-based applications has been recognized as an interesting and challenge research area to explore, and it has been the subject to intensive work in the recent years. As a result, a variety of systems and prototypes have been recently developed [11], [23], [24], [26].

Among those approaches the closest to ours are $PSPL$, $TrustBuilder$ and $Trust\text{-}\mathcal{X}$. However, to the best of our knowledge, none of the existing approaches to trust negotiations deals with credential language ontologies nor reason about semantic equivalence of different attribute credentials. and gives mechanism to prevent from privacy breaches.

PSPL [5] is part of a uniform framework to formulate and reason about information release on the Web. It is a language for expressing access control policies for services and release policies for client and service portfolios. PSPL also includes a mechanism for filtering the policies, to provide compact policy disclosures and to protect privacy during policy disclosures. Like our work, the authors provide the notion of equivalence among policies syntactically different. However, they compare policies on the basis of the syntax of terms, conditions, and credential types. Our notion of equivalence is not based on syntax but on the semantics of the underlying policies. Further, they do not address issues pertaining to privacy.

Trust Builder [26] is one of the most significant proposals. It provides a set of negotiation protocols that define the ordering of messages and the type of information the messages will contain, and strategies for controlling the exact content of messages. A variety of strategies are defined to allow strangers to establish trust through the exchange of digital credentials and the use of access control policies that specify what combinations of credentials a stranger must disclose in order to gain access to each local service or credential. Our work instead focuses on the process of choosing a disclosure set for a given policy during a single round of a negotiation. We have delineated desirable properties that an individual may use as criteria for selecting among the disclosure sets satisfying a single policy. The approach adopted by the user in a single step iterated at each round of a negotiation process actually results in new strategy for carrying out a negotiation preserving privacy. We plan to further explore this aspect in our future work and formally define a class of strategies and compare them with the ones presented in [26].

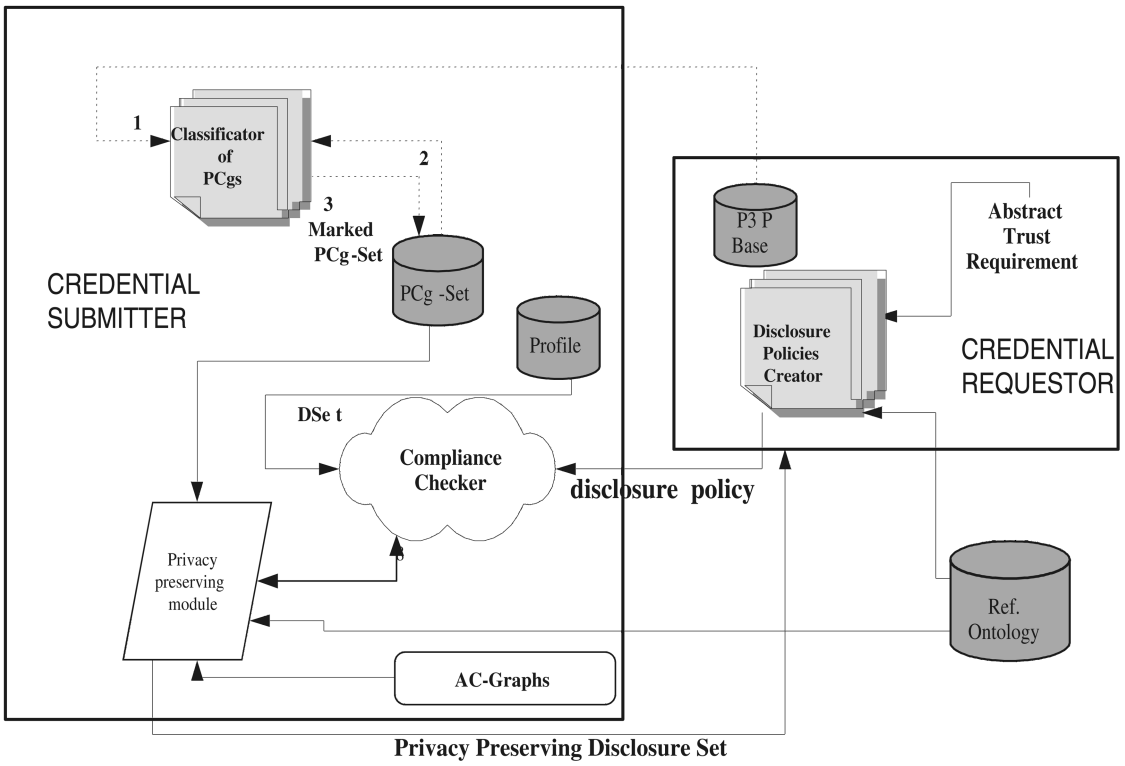

Fig. 7. System architecture.

The idea of selectively disclosing credential attributes is not new [7], [13], [17]. However, this technique has never been thoroughly explored, especially in trust negotiations. The only work close to this topic is from Holt et al. [12]. This work focuses on hidden credentials features, and on how hidden credentials can be constructed from identity-based crypto systems which satisfies Credential Indistinguishability. The authors show how transactions which depend on the fulfillment of policies described by monotonic Boolean formulae can take place in a single round of messages. Our focus, differently from [12], is to deeply analyze the impact of protected attribute credentials on trust negotiations, and devise new techniques to compare different credential views in order to select the one that limit privacy breaches.

A formal framework for trust negotiations has been recently proposed by Winsborough and Li in [22]. The authors provide an approach for safe enforcement of policies that focus on credential exchange. A formal notion of safety in automated trust negotiations is given which is based on the possibility by third parties of inferring information on negotiating parties profiles. Our approach to trust negotiations is compliant with respect to the authors notion of safety. Additionally we focus on other aspects related to trust negotiations, such as, for instance, the development of a credential and policy language based on the use of ontology. Such aspects are not considered in [22], nor the authors mention them as future work. Furthermore, the notion of private concept groups we present in this paper addresses the need of protecting combinations of attributes considered sensitive. This is also a crucial aspect of safe negotiations, and it is defined as *Attribute-combination hiding* in [22]. However, the goal of the approach we present in this paper is to protect attribute-combinations in a much effective manner, since we do not only focus on attribute combinations, but we consider the underlying semantic of the information to be released and protect disclosure of all equivalent (or more specialized) combinations of attributes. We will further explore this analysis and formally evaluate our approach with respect to the author's proposed notion of safety. Further, Li et al. work is based on the notion of *ack policies*, which are used to prevent information leakage during the negotiation. Such policies require to be specified for both possessed and nonpossessed attributes. The trust requirements introduced in Section 4 may be usefully integrated to Winsborough and Li's work in this sense. The use of high-level expressions that can be mapped at credential level, indeed, can be used to automate the ack policy specification operation and make, in general, negotiation policies simpler and effective.

Finally, the work presented in this paper has been developed in the context of the Trust-$\mathcal{X}$ [1], [2] system. Trust-$\mathcal{X}$ [2] is a comprehensive framework for trust negotiations, providing both a language for encoding policies and certificates, and a system architecture. We have defined several different approaches to carry out negotiations, also achieving protection of disclosure policies. Recently, in [1], we also have extended the system by adding techniques for preserving privacy, such as the selective disclosure of credentials and the integration with the P3P platform. Like in current work, we have defined a logical formalism for defining credentials and protection needs for the release of a resource, denoted as *disclosure policies*. However, in our previous work, we did not have the notion of trust requirements nor the notion of reference ontologies and we discussed privacy under a different perspective.

The problem of releasing data so that individuals who are the subjects of the data cannot be identified has been explored by work on k-*anonymity* [18], [20], statistical databases [9] and deductive databases [4]. Most of this work focuses on limiting the information that can be released in

response to multiple queries. These schemes require history information to be maintained so that multiple interactions with the same parties can be correlated. We adapt from the work on k-anonymity the notion of attribute suppression and generalization. However, we have revisited these techniques to adapt them in the trust negotiation scenarios, where releasing credentials may cause disclosure of private concepts. Moreover, we focused on controlled release of information from the perspective of the single individual who may not have the view of the data collected by the counterpart. Other techniques similar to ours are related with perturbation techniques [6]. These techniques have been widely used for statistical databases. The idea is that, rather than releasing the value of an attribute, the value is changed. The result is that the value itself is privacy preserved. In our context, this technique is not directly adoptable as it is, since it is worthless for users to submit wrong data. Wrong data does not satisfy any policy as it cannot be proved by any signed credential. However, some variations of this technique might be adopted. We will further explore possible applications in our future work.

8 Conclusion

In this paper, we have addressed the problem of preserving privacy in trust negotiations. We have introduced the notion of *privacy preserving disclosure*. A privacy preserving disclosure is a set that does not include attributes or credentials, or combinations of these, that may compromise privacy. Privacy preserving disclosures are in turn based on the notion of *private concept groups*.

To obtain privacy preserving disclosure sets we have proposed two techniques based on the notions of *substitution* and *generalization*. Substitution is used when only nonrequested attributes in the disclosure set result in privacy breaches. In such a case, we replace the associated requested attributes with some alternative credential or attribute that implement the same concept and that, however, do not result in privacy breaches. Generalization is instead used when privacy breaches arise because of the concepts specified in trust requirements and consists of substituting attributes/credentials with others corresponding to more generalized concepts.

The mechanism we proposed for privacy can also be effectively used for enforcing user anonymity. Privacy sensitive attributes can be suppressed or generalized before any disclosure. A credential submitter can adopt private concept groups for controlling disclosure of information revealing his/her identity.

We have argued that formulating the trust negotiation requirements in terms of disclosure policies is often restrictive. To better address this issue, we have introduced the notion of *reference ontology*, and formalized the notion of trust requirement. Additionally, we have developed an approach to derive disclosure policies from trust requirements and formally stated some semantics relationships (i.e., equivalence, stronger than) that may hold between policies. These relationships can be used by a credential requestor to reason about which disclosure policies he/she should use in a trust negotiation.

A lot of work still remains to be done. In the current paper we have assumed that one disclosure policy is chosen and sent to the counterpart without specifying how such choice is made. To make this choice much effective, we will design mechanisms to select disclosure policies among a set of equivalent ones. Further, in the current approach, we control release of concepts during the same negotiation and check previous concept disclosures without distinguishing whether the counterpart is the same or not. We will explore how to associate the released concepts with an entity, so to focus on the set of concepts that actually need to be protected. There might be cases in which breaking privacy concept groups is not avoidable. For instance, if the attribute required is not further generalizable or it cannot be replaced, privacy has to be compromised, unless the negotiation fails. However, we will further extend our study to better define which are the actual limitations of the proposed approach and whether it can be extended to ensure success in any possible context. Moreover, we plan to further explore the use of ontologies in trust negotiation systems. In particular, we are currently developing mechanisms and techniques for mapping equivalent terms using inference rules. We are investigating the logical relationship between predicates and designing algorithms that derive one from the other. In the current work, we have assumed the use of a shared ontology known to each subject. We will explore the possibility of negotiating with parties referring to different ontologies and investigate the conflicts and issues that may arise. Future work also include a more precise mathematic modelling of the proposed techniques based on distribution functions to ensure the effectiveness of the algorithms. Finally, an implementation of the proposed approach has to be done. We have already a prototype system supporting trust negotiations. We will extend such a system in order to fully support the features proposed in the paper.

Acknowledgments

The work reported in this paper has been partially supported by the US National Science Foundation under the ITR Grant No. 0428554, "The Design and Use of Digital Indentities," and by the sponsors of CERIAS.

References

[1] E. Bertino, E. Ferrari, and A. Squicciarini, "Privacy Preserving Trust Negotiations," *Proc. Fourth Int'l Workshop Privacy Enhancing Technologies,* 2004.

[2] E Bertino, E. Ferrari, and A. Squicciarini, "*Trust-$\mathcal{X}$*—A Peer to Peer Framework for Trust Establishment," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 7, pp. 827-842, Apr. 2004.

[3] E. Bertino, E. Ferrari, and A. Squicciarini, "Trust Negotiations: Concepts, Systems, and Languages," *IEEE Computing in Science and Eng.,* vol. 6, no. 4, pp. 27-34, 2004.

[4] P. Bonatti and S. Kraus, "Foundations on Secure Deductive Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 7, no. 3, pp. 406-422, 1995.

[5] P. Bonatti and P. Samarati, "Regulating Access Services and Information Release on the Web," *Proc. Seventh ACM Conf. Computer and Comm. Security,* 2000.

[6] R. Brand, "Microdata Protection through Noise Addition," *Inference Control in Statistical Databases, From Theory to Practice,* pp. 97-116. London: Springer-Verlag, 2002.

[7] S. Brands, *Rethinking Public Key Infrastructure and Digital Credentials.* MIT Press, 2000.

[8] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, *Ontology Matching: Machine Learning Approach,* publisher? 2003.

[9] J. Domingo-Ferrer, *Inference Control in Statistical Databases from Theory to Practice,* vol. 2316. Springer, 2002.

[10] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition,* vol. 5, no 2, pp. 199-220, 1993.

[11] A. Herzberg et al., "Access Control meets Public Key Infrastructure, or: Assigning Roles to Strangers," *Proc. IEEE Symp. Security and Privacy,* 2000.

[12] J. Holt, R. Bradshaw, K.E. Seamons, and H. Orman, "Hidden Credentials," *Proc. Second ACM Workshop Privacy in the Electronic Soc.* 2003.
[13] R.D. Jarvis, "Selective Disclosure of Credential Content during Trust Negotiation," master of science thesis, Brigham Young Univ., Apr. 2003.
[14] L. Khan, D. McLeod, and E. Hovy, "Retrieval Effectiveness of an Ontology-Based Model for Information Selection," *The VLDB J.*, vol. 13, no. 1, pp. 71-85, 2004.
[15] M. Marchiori, L. Cranor, and M. Langheirich, "The Platform for Privacy Preferences 1.0 (p3p1.0) Specification," W3C Reccomandation, Apr. 2002, http://www.w3.org/P3P/brochure.html.
[16] M. Naor, "Bit Commitment Using Pseudorandomness," *Advances in Cryptology-89,* vol. 435, Lecture Notes in Computer Science, New York, 1990.
[17] I. Visconti and P. Persiano, "User Privacy Issues Regarding Certificates and the TLS Protocol," *Proc. ACM Conf. Computer and Comm. Security,* 2000.
[18] P. Samarati and L. Sweeney, "Generalizing Data to Provide Anonymity When Disclosing Information," *Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* June 1998.
[19] K.E. Seamons, M. Winslett, and T. Yu, "Requirements for Policy Languages for Trust Negotiation," *Proc. Third IEEE Int'l Workshop Policies for Distributed Systems and Networks,* 2002.
[20] L. Sweeney, "*k*-anonymity: A Model for Protecting Privacy," *Int'l J. Uncertainty, Fuziness and Knowledge-Based Systems,* vol. 10, no. 5, pp. 557-570, 2002.
[21] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods, and Applications," *Knowledge Eng. Rev.,* vol. 11, no. 2, pp. 93-155, 1996.
[22] M. Winsborough and N. Li, "Safety in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* 2004.
[23] W.H. Winsborough and N. Li, "Protecting Sensitive Attributes in Automated Trust Negotiation," *Proc. ACM Workshop Privacy in the Electronic Soc.,* 2002.
[24] W.H. Winsborough, K.E. Seamons, and V. Jones, "Automated Trust Negotiation," *Proc. DARPA Information Survivability Conf. and Exposition,* vol. I, pp. 88-102, 2000.
[25] T. Yu and M. Winslett, "A Unified Scheme for Resource protection inAutomated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* 2003.
[26] T. Yu, M. Winslett, and K.E. Seamons, "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation," *ACM Trans. Information and System Security,* vol. 6, no. 1, 2003.

Anna C. Squicciarini received a degree in computer science from the University of Milan in July 2002 with full marks and is a candidate PhD degree at the University of Milan, Italy. Since Autumn of 2003, she has been a visiting researcher at the Swedish Institute of Computer Science, Stockholm. In the Spring of 2004, she was a visiting researcher at Colorado State University, where she explored anonymity issues in trust negotiations. She also visited Purdue University during 2005. Her main research interests span from trust negotiations, privacy models and mechanisms for privilege and contract management in virtual organizations to grid systems, and Federated Identity Management. In 2006, she joined Purdue University as a post doctorate student.

Elisa Bertino is professor of computer science and electrical and computer engineering at Purdue University and serves as research director of the Center for Education and Research in Information Assurance and Security (CERIAS). Previously, she was a faculty member in Department of Computer Science and Communication at the University of Milan where she has been department chair and director of the DB&SEC Laboratory. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation, at Rutgers University, and at Telcordia Technologies. Her main research interests include security, privacy, database systems, object-oriented technology, multimedia systems. In those areas, Professor Bertino has published more than 250 papers in all major refereed journals, and in proceedings of international conferences and symposia. She is a coauthor of the several books and is a coeditor in chief of the *Very Large Database Systems* (*VLDB*) Journal. She serves also on the editorial boards of several scientific journals. She has been consultant to several Italian companies on data management systems and applications and has given several courses to industries. She has been involved in several projects sponsored by the EU. She has served as a program committee member of several international conferences, as program cochair and chair several conferences. She is a fellow of the IEEE and the ACM and has been been named a Golden Core Member for her service to the IEEE Computer Society. She received the 2002 IEEE Computer Society Technical Achievement Award for "For outstanding contributions to database systems and database security and advanced data management systems."

Elena Ferrari received the MS degree in computer science from the University of Milano, Italy, in 1992. In 1998, she received the PhD degree in computer science from the same university. She is a full professor of computer science at the University of Insubria, Como, Italy. From 1998 until January 2001, she was an assistant professor in the Department of Computer Science at the University of Milano, Italy. During summer of 1996, she was a visiting researcher in the Department of Computer Science of George Mason University, Fairfax (Virginia). During the summers of 1997 and 1998, was a visiting researcher at Rutgers University, Newark, New Jersey. Her research activities are related to various aspects of data management systems, including Web security, access control and privacy, multimedia databases, and temporal databases. On these topics, she has published more than 100 scientific publications in international journals and conference proceedings. She gave several invited lectures and tutorials in Italian and at foreign universities as well as on international conferences and workshops. Dr. Ferrari has served as program chairm cochair, and demonstration chair of several conferences. She has also served as program committee member of several international conferences and workshops. Professor Ferrari is on the editorial board of the *VLDB Journal* and *the International Journal of Information Technology (IJIT)*. She is a member of the ACM and a senior member of the IEEE.

Indrakshi Ray received the PhD degree from George Mason University. She is an assistant professor in the Computer Science Department at Colorado State University. Prior to joining Colorado State, she was a faculty member at the University of Michigan-Dearborn. Her research interests include security and privacy, database systems, e-commerce, and formal methods in software engineering. She has published several refereed journal and conference papers in these areas. She served as the program chair for the IFIP WG 11.3 Conference on Data and Applications Security 2003 and SACMAT '06. She has also been a member of several program committees such as EDBT, SACMAT, ACM CCS, and EC-Web. She is a member of the ACM and the IEEE.