Anonymity Preserving Techniques in Trust Negotiations

Indrakshi Ray[1], Elisa Bertino[2], Anna C. Squicciarini[2] and Elena Ferrari[3]

[1] Computer Science Department,
Colorado State University, Fort Collins, Co, USA.
iray@cs.colostate.edu
[2] CERIAS and Computer Science Department Purdue University,
West Lafayette, IN, USA
{bertino, squiccia}@cs.purdue.edu
[3] Dipartimento di Scienze della Cultura, Politiche e Informazione,
Universitá degli Studi dell'Insubria, Como.
elena.ferrari@uninsubria.it

Abstract. Trust negotiation between two subjects require each one proving its properties to the other. Each subject specifies disclosure policies stating the types of credentials and attributes the counterpart has to provide to obtain a given resource. The counterpart, in response, provides a disclosure set containing the necessary credentials and attributes. If the counterpart wants to remain anonymous, its disclosure sets should not contain identity revealing information. In this paper, we propose anonymization techniques using which a subject can transform its disclosure set into an anonymous one. Anonymization transforms a disclosure set into an alternative anonymous one whose information content is different from the original one. This alternative disclosure set may no longer satisfy the original disclosure policy causing the trust negotiation to fail. To address this problem, we propose that trust negotiation requirements be expressed at a more abstract level using *property-based policies*. Property-based policies state the high-level properties that a counterpart has to provide to obtain a resource. A property-based policy can be implemented by a number of disclosure policies. Although these disclosure policies implement the same high-level property-based policy, they require different sets of credentials. Allowing the subject to satisfy any policy from the set of disclosure policies, increases not only the chances of a trust negotiation succeeding but also the probability of ensuring anonymity.

1 Introduction

Most of the interpersonal transactions, carried out in any application environment we may think of, are contingent upon relevant attributes of the involved parties, like nationality, age, job function, financial resources. In the digital world, such interactions have been historically handled out-of-band using alternative means or simply avoided. The increasing use of Internet in a variety of distributed multi-party interactions and transactions with strong real-time requirements has however pushed the search for solutions to the problem of

attribute-based digital interactions. A promising solution to this problem is represented by automated trust negotiation systems [1, 9, 15, 16].

A trust negotiation system addresses the problems associated with classical authentication and authorization schemes by allowing subjects outside a local security domain to securely access protected resources and services [2, 15, 16]. It makes it possible for two parties to carry on secure transactions by first establishing trust through a bilateral, iterative process of requesting and disclosing digital credentials and policies. Digital credentials can be considered the equivalent, in the digital world, of paper credentials. Credentials often contain multiple attributes, for example, the name and the birth date of an individual, and can be used to verify identification information, professional qualifications, and association memberships, etc. Credentials are digitally signed by an issuing authority and assert the veracity of certain attributes of the owner. The use of public key encryption guarantees that these credentials are both unforgeable and verifiable. The other relevant component of any trust negotiation system is represented by policies, protecting sensitive resources, and even other policies from unauthorized access. By specifying necessary credentials a party must possess, and attribute conditions a party must verify in order to access a specific resource, policies provide a means by which any subject may be granted or refused access to a resource in real-time. Such policies are referred to as disclosure policies.

Trust negotiation systems, however, by their very nature may represent a threat to privacy. Credentials, exchanged during negotiations, often contain sensitive personal information that thus needs to be selectively released. Also, a user may want to minimize the released information, thus enforcing the need to know principle in disclosing his 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 non-linkability.

Even though a comprehensive solution to the problem of privacy in trust negotiation systems is quite articulated and requires the combination of different techniques, we believe that a feature that should be provided as part of a privacy-preserving trust negotiation is the anonymization of disclosed information. Such feature is crucial in order to address the non-linkability requirement. The goal of this paper is to develop a solution supporting anonymization in trust negotiation systems. To the best of our knowledge this is the first time such concept is proposed in the framework of trust negotiation systems.

We argue that specifying trust requirements using disclosure policies is too restrictive for trust negotiations requiring anonymity. Since these policies are expressed in terms of specific credential types and attributes, failure to provide the requested credentials and attributes causes the negotiation to fail. We propose that trust negotiation requirements should be specified in terms of high level

properties needed to obtain a given resource. These property-based policies can be translated into a number of disclosure policies, each of which requires different disclosure sets containing different credentials. Failure to provide a specific disclosure set no longer causes the negotiation to fail. If a specific disclosure set compromises anonymity, the subject can provide an alternative one. If this anonymous disclosure set satisfies an alternative disclosure policy that implements the same property-based policy as the original one, then the trust negotiation can proceed.

Ideally, each credential and/or attribute disclosed should reveal only the crucial information required to satisfy the corresponding policy without compromising anonymity. Unfortunately, this cannot be always realized in practice. To reach our goal we revisit substitution and generalization techniques [8, 11, 12] presented in previous work and adapt them for use in the trust negotiation context. We propose a novel technique for substitution and generalization of data conveyed in credentials and attributes by use of an ad-hoc data structure, called a concept graph. The concept graph is able to capture semantic relationships among data conveyed in different credentials. The rest of the paper is organized as follows. Section 2 describes our notion of trust negotiation policies and their specification. Section 3 introduces the anonymity property and illustrates how generalization and specialization techniques can be used to ensure it. Section 4 discusses the related work. Section 5 concludes the paper with pointers to future directions. The appendix presents the details of the algorithms for achieving anonymity.

2 Specification of Trust Negotiation Policies

Trust negotiation requirements can be expressed at different levels of abstraction. In what follows, we refer to the subject who requests the credential as *requester* and the subject who submits the credential as *submitter*.[4] The requester begins by expressing its high-level trust requirements in the form of property-based policies. These property-based policies are then refined into disclosure policies. Before discussing these, we introduce our notion of credentials and attributes because they form the basis of trust negotiation requirements.

2.1 Credentials and Attributes

A credential associated with a subject is a digitally signed document containing attributes that describe properties of the subject. Examples of attributes that may

[4] In trust negotiation a subject may act as a submitter in one step of the negotiation and as a requester in another step.

be contained in a credential are birth date, name, professional qualifications, and association memberships. Since credentials are encrypted and digitally signed by an issuing authority, they are unforgeable and verifiable. By providing the credentials listed in the disclosure policies, the submitter proves the properties required by the requester.

Like previous work on trust negotiations [2], we consider credentials as instances of credential types. The notion of credential type provides a template for encoding credentials having a similar structure and collecting the same attributes. We denote credential types using the notation CT_i, CT_j, etc. Each credential type CT_i contains a set of attributes denoted as AS_{CT_i}.[5] A credential contains a number of attributes together with values defined for each of these attributes. Often the requester is interested in some, but not all, of the attributes of the requested credentials. Ideally, a 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 like to selectively disclose attributes contained in a credential. One approach currently available to allow partial disclosure of credentials relies on the use of the bit commitment technique [10], which enables users to communicate a value without revealing it. By exploiting this technique on digital credentials it is possible to actually send credentials by revealing only the minimal set of attributes required during the negotiation. The ability to blind one or more attributes in a credential generates different *views* of the credential. Views of the same credential differ with respect to the number of hidden attributes. A requester might be interested in some attributes contained in the credential or the possession of the credential itself. For proving credential possession, we assume that the submitter must provide some attributes contained in the credential that indicates credential possession. When a credential requester requests an attribute *attr* contained in credential *cred*, the submitter will provide the view in which *attr* is not hidden but a maximum number of other attributes are blinded. When a requester requests credential *cred* without mentioning the attributes, then a view of the credential is provided in which the attributes indicating credential possession are not blinded but most of the other attributes are hidden. Henceforth, we will not distinguish between requested attributes and requested credentials. The difference lies in the specific attributes of the credentials that are of interest.

Credential types provide a syntactic structure of information but do not specify anything about the interpretation of the attributes contained in the credential types. This makes it impossible to automatically detect relationships between attributes belonging to different credentials. To solve this problem of semantic

[5] Typically, credential types will have other information, such as, digital signatures. But these are not relevant for our present discussion.

conflicts, we borrow some ideas from the work on ontologies [7, 14]. An ontology consists of a set of concepts together with relationships defined among the concepts. The concepts and their relationships are described using a formal language. We propose an ontology for credentials and attributes and express a *concept* as follows.

Definition 1 [Concept] *A concept C_i is a tuple of the form $< KeywordSet_{C_i}$, $LangSet_{C_i} >$, where $KeywordSet_{C_i}$ is a set of keywords and $LangSet_{C_i}$ is a set of attributes. $KeywordSet_{C_i}$ is the set of all possible keywords used to describe concept C_i. Each attribute in $LangSet_{C_i}$ implements concept C_i.*

For each concept C, we require that the $KeywordSet_C$ and $LangSet_C$ should be non-empty and finite. For any two distinct concepts C and C', $KeywordSet_C \cap KeywordSet_{C'} = \{\}$ and $LangSet_C \cap LangSet_{C'} = \{\}$. In other words, any keyword belongs to exactly one concept. Similarly, each attribute of a credential is associated with exactly one concept. We use the notation C_{ak} to indicate the concept associated with *ak* where *ak* denotes an attribute or a keyword. We assume that each concept is unique. However, concepts may be related using generalization/specialization relationships. We use the notation $C_i \subset C_j$ to indicate that the concept C_i is a generalization of concept C_j and the notation $C_i \subseteq C_j$ to indicate that C_i either equals C_j or is its generalization. For instance, the concept *address* is a specialization of the concept *country of residence*. We specify this as *country of residence* $\subset$ *address*. We assume that there are a finite number of such well-defined concepts in the ontology. An example of a concept is $C =< \{sex, gender\}, \{passport.gender, drivingLicense.sex\} >$. The concept known as *sex* or *gender* can be implemented by the attribute $passport.gender$ or the attribute $drivingLicense.sex$. Thus, a concept can be implemented by attributes of different credentials. The different attributes implementing a particular concept are semantically equivalent. The attributes in $LangSet_{C_i}$ are semantically equivalent but they may have different domains. To compare the values of two semantically equivalent attributes, we need functions that convert the value of one attribute to a corresponding value for the semantically equivalent attribute. Similarly, a condition specified over an attribute may need to be translated to a condition defined over a semantically equivalent attribute. These requirements motivate us to propose the notion of translation functions.

Definition 1. [Translation Function] *The translation function associated with a concept C_i, denoted as Π_{C_i}, is a total function that takes as input a condition A_{pq} op k ($A_{pq} \in LangSet_{C_i}$) and an attribute A_{rs} ($A_{rs} \in LangSet_{C_j}$) and produces an equivalent condition defined over attribute A_{rs}. This is formally expressed as follows. $\Pi_{C_i} : Cond_{C_i} \times LangSet_{C_j} \rightarrow Cond_{C_j}$ where $Cond_{C_i}$ is the set of all valid conditions specified over the attributes in $LangSet_{C_i}$ and $C_i \subseteq C_j$.*

Since the translation function is total, for every given valid condition and attribute there exists an equivalent condition defined on the given attribute. Several steps are involved in developing the translation function. Let us illustrate this with an example. To express $A_{pq}\ op\ k$ in terms of A_{rs}, we need to first convert the value k to an equivalent value that is in the domain of A_{rs}. This step is performed by conversion functions which converts the value of one attribute to an equivalent value of another attribute. The second step is to convert the operator op into an equivalent operator op' that is suitable for the domain of A_{rs}. The definition of the conversion function together with the domain of the attribute can determine how the operator must be changed. The details of the translation functions are domain dependent but an example will help to illustrate how they can be specified. Consider the two attributes *passport.age* and *driversLicense.yearOfBirth*. Suppose we want to translate $passport.age > 25$ to an equivalent condition defined over *driversLicense.yearOfBirth*. The first step is to convert $passport.age = 25$ to an equivalent value defined over *driversLicense.yearOfBirth*. Converting *passport.age* to *driversLicense.yearOfBirth* is done by the function: *driversLicense.yearOfBirth = currentYear – passport.age*. For *passport.age* $= 25$, this function returns *driversLicense.yearOfBirth* $= 1974$. Since $driversLicense.yearOfBirth$ and $passport.age$ are inversely related (that is, $passport.age$ increases as $driversLicense.yearOfBirth$ decreases) the operator $>$ is inverted to obtain $<$. The results obtained by the Π function in this case will be *driversLicense.yearOfBirth* < 1979. We use the $\Rightarrow$ operator to indicate that one condition implies another. For instance *driversLicense.yearOfBirth* $< 1979 \Rightarrow$ *passport.age* > 25 and *passport.age* $> 25 \Rightarrow$ *driversLicense.yearOfBirth* < 1979.

2.2 Property-based Policies

In a trust negotiation each entity is interested in obtaining information and verifying properties about the counterpart. Requestors may adopt different strategies for obtaining this information. One such strategy is the *open strategy*. In this strategy, the information requested from the counterpart who wants some resource is specified in the form of *property-based policies*. A property-based policy, specified at a higher level of abstraction, lists the properties the counterpart has to provide and the conditions it must satisfy in order to obtain some resource.

Definition 2. [Property-Based Policy] *A property-based policy (PbP for brevity) for a resource R is a pair* $(R, properties, conditions)$*, where R denotes a target resource, and properties is the set of property names, conditions is the set of conditions defined over one or more properties listed in properties.*

An example of property-based policy is *(Drug, {age, residence, person identifier}, {age > 25})*. This states that the counterpart has to prove that its age is above 25, and give its residence and identifier information in order to obtain the resource *Drug*. In a property-based policy, the requester needs to enumerate all the properties it is interested in. Sometimes it may not be willing to divulge such information to the counterpart. In such cases, the counterpart adopts the *closed strategy* and expresses only *disclosure policies*.

2.3 Disclosure Policies

A disclosure policy lists the attributes and credential types needed to obtain a given resource. Thus, they do not directly reveal the properties that a requester is interested in. Disclosure policies also speed up the process of trust negotiation because the submitter knows precisely the requested credentials and attributes.

Definition 3. [Disclosure policy] *A* disclosure policy *is expressed as* $R \leftarrow T_1, T_2, \ldots T_n, n \geq 1$, *where:*

1. *R denotes a target resource;*
2. T_i *(*$1 \leq i \leq n$*) denotes an expression, called term, of the form* $CT_i()$, $CT_i(A_{ij})$, *or* $CT_i(A_{ij}\ op\ k)$ *where* CT_i *refers to a credential type,* A_{ij} *is an attribute contained in* CT_i, $A_{ij}\ op\ k$ *is a condition on attribute* A_{ij} *that is contained in the credential type* CT_i*;*
3. $\forall T_i, T_j \in \{T_1, T_2, \ldots, T_n\}$ *where* $i \neq j$, $C_{T_i} \not\subseteq C_{T_j}$ *and* $C_{T_j} \not\subseteq C_{T_i}$.

According to such a formalization, because of condition 3, we have that concepts corresponding to the terms in a disclosure policy cannot be equal or related by a generalization/specialization relationship. This condition ensures that no duplicate information is being requested. The goal of the requester is to formulate disclosure policies that implement some property-based policy.

Definition 4. [Implement disclosure policy] *A disclosure policy* $DP : R' \leftarrow T_1, \ldots, T_h$ *is said to* implement *a property-based policy* $PbP : (R, properties, conditions)$ *if the following holds:*

1. *R = R'*
2. $\forall p \in properties \bullet (\exists T_i \in \{T_1, \ldots, T_h\} \mid T_i = CT_i()\ or\ CT_i(A_{ij})\ or\ CT_i(A_{ij}\ op\ k)$ $\bullet\ (CT_i()\ or\ CT(A_{ij}) \in LangSet_{C_x} \wedge C_p \subseteq C_x))$
3. $\forall (p\ op\ k) \in conditions \bullet (\exists T_i \in \{T_1, \ldots, T_h\} \mid T_i = CT_i(A_{ij}\ op\ k) \bullet (CT_i(A_{ij}) \in LangSet_{C_x} \wedge C_p \subseteq C_x \wedge A_{ij}\ op'\ k' \Rightarrow p\ op\ k)$

Condition 1 states that the disclosure policy *DP* and the property-based policy *PbP* must refer to the same resource. Condition 2 states that each property p in

PbP should be implemented by a credential or attribute in *DP* and the concept corresponding to the credential or attribute should be equal to or a specialization of the concept corresponding to the property *p*. Condition 3 states that each condition in *PbP* should be translated into an appropriate condition on the corresponding attribute in *DP*. Not all property-based policies can be implemented by disclosure policies. A property-based policy is implementable if there exists one or more disclosure policies that implements it. We next define the conditions required of property-based policies that make them implementable. The first is that the conditions in the property-based policy should not contradict each other. The second is that each property *p* in *PbP* must be associated with some concept *y*. Since the *LangSet* of a concept is non-empty, this ensures that each property can be implemented by some attribute. The third states that each condition in *PbP* should be translated into a condition on the corresponding attribute such that the attribute condition implies the condition stated in the *PbP*. This ensures that there is an attribute condition corresponding to every condition listed in the *PbP*.

Definition 5. [Property-based policy implementability] *A property-based policy* PbP: (R, properties, conditions) *is implementable if the following holds:*

1. $conditions = \{c_1, c_2, \ldots, c_n\} \Rightarrow c_1 \wedge c_2 \wedge \ldots c_n \neq \phi$
2. $\forall p \in properties \bullet (\exists C_y \bullet p \in KeywordSet_{C_y})$
3. $\forall (p\ op\ k) \in conditions \bullet (\exists A_{ij} \in LangSet_{C_x} \bullet (C_p \subseteq C_x \wedge A_{ij}\ op'\ k' \Rightarrow p\ op\ k))$

A single property-based policy can be implemented by a number of disclosure policies as the following example illustrates. Let *PbP: (loan,* {*MaritalStatus, Country*}, {*country=USA*}*)* be a property-based policy. Let {*MarriageCertificate (possess), HealthInsurance(MaritalStatus), id_card(maritalStatus)*} and {*id_card (country), ResidenceCert(country), drivingLicense(country)*} be the corresponding *LangSet* components of concepts corresponding to *MaritalStatus* and *Country*, respectively. Some disclosure policies implementing *PbP* are: (1) *loan* $\leftarrow$ *MarriageCertificate(), id_card(country=USA)*, (2) *loan* $\leftarrow$ *id_card(maritalStatus), id_card (country = USA)*, (3) *loan* $\leftarrow$ *HealthInsurance(MaritalStatus), HealthInsurance(Provider), id_card(country = USA)*. These disclosure policies require different sets of credentials. This is possible because a property listed in a property-based policy can be proved by different credentials. For example the property *married* can be demonstrated by giving the credential *MarriageCertificate()* or the attribute *Driving_License(maritalstatus=married)*. Thus a credential submitter not willing to disclose a particular credential can satisfy an alternate disclosure policy and continue with the negotiation process. Next we

formalize which alternative disclosure policies will be accepted by the requester. For this we need the notion of the stronger than relation for disclosure policies.

Definition 6. [Stronger than relation] *Let* $DP1 : R \leftarrow T_a, T_b, \ldots, T_n$ *and* $DP2 : R' \leftarrow T_p, T_q, \ldots, T_y$ *be two disclosure policies.* $DP1$ *is said to be stronger than* $DP2$*, denoted by* $DP1 \succ DP2$*, if the it satisfies the following:*

1. $R = R'$
2. $\forall T_j \in \{T_p, T_q, \ldots, T_y\} \bullet (\exists T_i \in \{T_a, T_b, \ldots, T_n\} \bullet C_{T_j} \subseteq C_{T_i})$
3. $\forall T_j \in \{T_p, T_q, \ldots, T_y\} \mid T_j = CT_j(A_{jm}\ op\ p) \bullet (\exists T_i \in \{T_a, T_b, \ldots, T_n\} \mid T_i = CT_i(A_{in}\ op'\ q) \bullet A_{in}\ op'\ q \Rightarrow A_{jm}\ op\ p)$

Condition 1 says that $DP1$ and $DP2$ must refer to the same resource. Condition 2 says that for each term T_j in $DP2$ there is a term T_i in $DP1$ such that the concept associated with T_i is equal to or a specialization of the concept associated with T_j. Condition 3 says that for each term T_j in $DP2$ that contains an attribute condition $A_{jm}\ op\ p$, there is a term in T_i in $DP1$ that has some attribute condition $A_{in}\ op'\ q$, such that the attribute condition $A_{jm}\ op\ p$ can be derived from the attribute condition $A_{in}\ op'\ q$.

Theorem 1 *For any two given disclosure policies,* $DP1$ *and* $DP2$*, evaluating whether* $DP1$ *is stronger than* $DP2$ *is decidable.*

Definition 7. [Equivalence relation] *Two disclosure policies* $DP1$ *and* $DP2$ *are said to be equivalent, denoted* $DP1 \equiv DP2$*, if* $DP1 \succ DP2$ *and* $DP2 \succ DP1$.

For two disclosure policies $DP1$ and $DP2$ to be equivalent, each term T_i in $DP1$ must have a corresponding term T_j in $DP2$, such that $C_{T_i} = C_{T_j}$.

The following example helps explain these relations. Consider the following disclosure policies: (i) *DP1: loan ← Marriage_Cert(), id_card(age > 25), id_card(country=USA),* (ii) *DP2: loan ← Marriage_Cert(), id_card(age > 25),* (iii) *DP3: loan ← id_card(MaritalStatus=married), id_card(age > 25),* (iv) *DP4: loan ← Marriage_Cert(), id_card(age > 25), id_card(city=Fort Collins, USA).* The following relations hold on the disclosure policies: $DP2 \equiv DP3$, $DP1 \succ DP2$, $DP4 \succ DP1$. Note that, all these disclosure policies implement the same property-based policy *(loan,* {*maritalstatus,age*}, {*maritalstatus=married,age >25*}*).*

We use the notation $DP2 \succeq DP1$ to denote that $DP2$ is either stronger than or equivalent to $DP1$. Suppose the requester requires a disclosure policy $DP1$, and the submitter provides credentials that satisfy an alternate disclosure policy $DP2$. If $DP2 \succeq DP1$, then the submitter has the assurance that the trust negotiation will proceed. This is formally proved by the following theorem.

Theorem 2 *For any two disclosure policies, DP*1 *and DP*2*, that are related by DP*2 $\succeq$ *DP*1*, any property-based policy PbP that is implemented by DP*1 *will also be implemented by DP*2*.*

2.4 Disclosure Sets

Depending on the trust negotiation strategy, the requester either provides a disclosure policy implementing a property-based policy for the negotiated resource or the property-based policy itself. The submitter, in turn, has to provide credentials to satisfy such a request. Each submitter has a *profile* of credentials. The submitter consults its profile to create a disclosure set which it submits to the requester. Disclosure set is a set of credentials, some of which may contain attributes that are blinded. The trust negotiation can proceed if the disclosure set completely or partially satisfies a disclosure policy or a policy that is stronger than or equivalent to the given one. But first, we define what it means for a disclosure set to completely satisfy a disclosure policy.

Definition 8. [Disclosure policy satisfaction] *Let DP be a disclosure policy of the form* $R \leftarrow T_1, T_2, , T_n,\ n \geq 1$*. The disclosure set DSet, consisting of unblinded attributes given by* AS_{DSet}*,* satisfies *DP if* $\forall\ T_i \in \{T_1, .., T_n\}$ *one of the following conditions hold depending on the form of* T_i*.*

case $T_i = CT_i()$**:** $\exists\, CR_{ij} \in DSet$
case $T_i = CT_i(A_{ik})$**:** $CR_{ij}.A_{ik} \in AS_{DSet}$
case $T_i = CT_i(A_{ik}\ op\ p)$**:** $CR_{ij}.A_{ik} \in AS_{DSet} \ \wedge\ CR_{ij}.V_{A_{ik}}\ op\ p$ *where* $CR_{ij}.V_{A_{ik}}$ *denote the value of attribute* A_{ik} *in credential* CR_{ij}*.*

The definition requires that each term specified in *DP* be satisfied by at least one credential or attribute in the set *DSet*. Intuitively a disclosure policy is satisfied if the submitter provides credentials that are instances of credential types listed in the disclosure policy, and the attributes of the credentials satisfy the conditions specified in the disclosure policy. Note that if a disclosure policy specifies credentials, attributes or attribute conditions, we provide the most blinded view of the credential that will meet the requirements of the disclosure policy. In some cases, a disclosure set provides some, but not all, of the credentials requested in a disclosure policy. We then say that the disclosure set *partially satisfies* the disclosure policy. In these circumstances, the requester, instead of rejecting the request, can ask for the remaining credentials and attributes needed to completely satisfy the policy and proceed with the negotiation.

A disclosure set *DSet* satisfying a disclosure policy $DP : R \leftarrow T_1, T_2, \ldots, T_n$ contains two kinds of attributes: *requested* and *non-requested*. A requested attribute A_{jk} is one which is mentioned in some term $T_j = CT_j(A_{jk})$ where $1 \leq j \leq$

n in the disclosure policy *DP*. An attribute that is not explicitly requested in the disclosure policy but is present because it cannot be blinded is a non-requested attribute.

Consider the disclosure policy: $R \leftarrow$ *Marriage_Certificate(), id(age* $>$ *25)*. To satisfy this disclosure policy, the subject can either provide the disclosure set $DSet_1 = \{$*Marriage_Certificate, id.age, id.country*$\}$ or it can provide $DSet_2 = \{$*Marriage_Certificate, id.age*$\}$. The subject will provide $DSet_2$ if it can blind all other attributes of *id*. The subject may provide $DSet_1$ if the most blinded view containing age also reveals *id*.*country*. In this case *id*.*age* is a requested attribute and *id*.*country* is a non-requested one.

A disclosure set complies with a property-based policy if it satisfies any disclosure policy implementing the property-based policy.

Definition 9. [Property-based policy compliance] *A disclosure set DSet* complies with *(satisfies) a property-based policy PbP if there exists a disclosure policy DP implementing PbP such that DSet satisfies DP*.

Note that a property-based policy is considered satisfied only when the disclosure set (completely) satisfies the corresponding disclosure policy. If a disclosure policy is partially satisfied, then several rounds of negotiation may be needed to satisfy the underlying property-based policy. When this occurs we say that the negotiation has succeeded.

3 Ensuring Anonymity of Disclosure Sets

As trust negotiations often occur between strangers, anonymity may represent an important requirement for negotiating subjects. For instance, Alice may not want to reveal her identity while bidding in an online auction. To formalize anonymity, we need the concept of identity disclosure. An identity disclosure is said to occur for the submitter if the data released to the counterpart contain attributes and credentials that uniquely identify him/her. Intuitively, identity disclosure happens when either the identity of an individual is directly revealed or it can be derived from the released data. For instance, if the released data contains the social security number, then the identity is directly revealed. If the released data contains name and address, then the identity of the individual can be inferred.

Other researchers, such as Samarati and Sweeney [11], have addressed the issue of protecting one's anonymity in the context of database systems. They classify attributes into three categories: (i) *identifiers* – attributes containing explicitly identifying information, (ii) *quasi-identifiers* – attributes that do not contain explicit identifying information, but that can be linked with other attributes to cause identity disclosure, and (iii) attributes that do not contain any

identifying information. Identifiers and quasi-identifiers can be automatically determined in the context of databases, where data content is available. There are several aspects in which we differ from Sweeney's work. First, we define identifiers and quasi-identifiers not on the basis of attributes but on the basis of concepts. Second, we need the notion of *set of quasi-identifier groups*. A quasi-identifier group is a set of quasi-identifiers with the following property: the release of all quasi-identifiers in a quasi-identifier group results in identity disclosure. In Sweeney's work, each table is associated with only one quasi-identifier group and so this concept is not needed. But in the context of trust negotiation, we may have different quasi-identifier groups. Third, the submitter trying to protect its anonymity has no knowledge about the information possessed by the requester. Thus, it may be impossible for the submitter to exactly determine the set of attributes that cause identity disclosure for a given case.

Examples of quasi-identifier groups are $\{$*employeeId, company name*$\}$ and $\{$*lastname, address*$\}$. In the first set, *employeeId* by itself does not reveal the identity of the individual, but *employeeId* together with the company name does. In the second set the last name does not uniquely identify the subject but when linked to its address, it does. $\{employeeId, lastname\}$ is not a quasi-identifier group because disclosing both of them do not breach anonymity.

Definition 10. [Anonymity-preserving disclosures] *Let DSet be a set of disclosures, and CS_{DSet} be the set of concepts associated with DSet. Let $Id = \{I_1, \ldots, I_k\}$ be a set of identifiers and let $Q\_Id = \{Q\_I_1, \ldots, Q\_I_n\}$ be a set of quasi-identifier groups. DSet is* anonymity-preserving *if the following holds:*

- $\forall I \in Id \bullet (\ I \notin CS_{DSet})$
- $\forall Q\_I \in Q\_Id \bullet (\ \exists I \in Q\_I \bullet (I \notin CS_{DSet}))$

Condition 1 ensures that identifiers are not present in the set of concepts associated with the disclosure set. Condition 2 ensures that for every quasi-identifier group, there is at least one element that is not present in the set of concepts associated with the disclosure set.

3.1 Concept Graphs

Our anonymization techniques make use of a data structure called *concept graph*. This is a directed acyclic graph in which each node n_i corresponds to a concept and each edge (n_i, n_j) indicates that the concept represented by node n_j is a generalization of the concept represented by node n_i. Since concepts may be unrelated, we may have multiple concept graphs. Unrelated concepts correspond to nodes in different concept graphs. Each concept corresponds to only one node

in the set of concept graphs. Figure 1 gives an example of a concept graph. We denote the concept graph associated with concept C_i as CG_{C_i}.

Definition 11. [Concept Graph]
A concept graph $CG = \langle N, E \rangle$ of a subject having the profile $Prof$ is a directed acyclic graph satisfying the following conditions.

- *N is a set of nodes where each node n_i is associated with a concept C_i and is labeled with $Prof_{C_i}$. $Prof_{C_i}$ is the credentials belonging to the user that contain unblinded attributes describing the concept C_i. Note that $Prof_{C_i} = LangSet_{C_i} \cap Prof$.*
- *E denotes a set of directed edges. For each edge $(n_i, n_j) \in E$, the concept C_j corresponding to node n_j is a generalization of concept C_i corresponding to node n_i.*

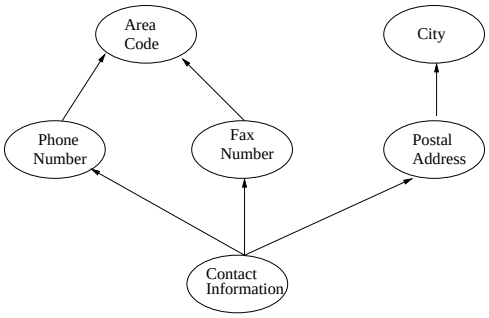

Fig. 1. Example of a Concept Graph

3.2 Using Substitution and Generalization to Achieve Anonymity

A disclosure set *DSet* satisfying a disclosure policy *DP* may cause a breach of anonymity. This may happen if the credentials or attributes contained in the disclosure set releases an identifier or quasi-identifiers. Identity disclosure may occur because of requested or non-requested attributes contained in *DSet*. If a requested attribute causes an identity disclosure, this attribute must be generalized. Alternately, if a requested attribute does not cause an identity disclosure but some other attribute contained in the same credential does, then the requested attribute needs to be substituted with an alternate one.

Consider the disclosure set *DSet* =$\{$*id.age, id.country*$\}$ where *id.age* is a requested attribute and *id.country* is a non-requested one. Recall that *id.country* is

present in the disclosure set because it cannot be blinded. Assume that *id.country* is a quasi-identifier and disclosing it will reveal the identity of the subject. To ensure anonymity we must remove *id.country* from *DSet*. This is only possible if the credential containing $id.age$ is removed from the *DSet*. Since $id.age$ is a requested attribute, this will cause the trust negotiation to fail. In such a scenario, we need to substitute $id.age$ with an alternate attribute, say *birthCert.dob*, such that $C_{id.age} = C_{birthCert.dob}$. We also need to ensure that all attributes visible in the credential *birthCert* do not cause an identity disclosure.

The process of substitution replaces each requested attribute contained in an identity revealing credential with an alternative equivalent attribute contained in an anonymous credential. Since each alternative attribute is equivalent to the original replaced attribute, it satisfies the same property of the property-based policy as the original one. Substitution, if successful, not only guarantees anonymity but also ensures that the underlying property-based policy will be satisfied.

Sometimes the disclosure policies request attributes or credentials, the disclosure of which causes identity disclosure. In such cases, substituting the attribute with an alternative one belonging to the same concept is not useful because the alternative attribute reveals the same concept as the original one. For such cases, we use the technique of generalization. In generalization we also choose an alternative attribute. However, unlike substitution, this alternative attribute belongs to the language set of the concept that is a generalization of the concept corresponding to the original attribute. Let us explain this with an example. Suppose $id.address$ is a requested attribute that causes an identity disclosure. The generalization technique will replace the requested attribute $id.address$ with an alternative attribute, say $id.city$ where $id.city$ belongs to the concept *city* that is a generalization of the concept *address*. Since we are not disclosing $id.address$, anonymity is preserved. On the other hand since the alternative attribute contains some but not all information about $id.address$, the negotiation may or may not succeed with the alternative attribute. The negotiation will succeed if the underlying property in the corresponding property-based policy corresponds to the generalized concept *city*. The negotiation will not succeed if the underlying property in the property-based policy corresponds to the concept associated with *address*.

4 Related Work

In this section we briefly review related approaches, which fall in two categories: trust negotiation systems and techniques for information disclosure control. Researchers have investigated trust negotiations for web-based applications and

have developed a number of systems and prototypes [2, 3, 5, 15, 16, 13]. However, to the best of our knowledge, these approaches do not consider different levels of abstractions for trust negotiation policies. Nor do they focus on anonymity. Winsborough and Li[15] have also addressed how sensitive credentials can be protected during trust negotiation. They formalize the notion of safety in the context of automated trust negotiations. The definition of safety is based upon third parties ability to infer information about the profiles of the negotiating parties. They do not address any issues pertaining to anonymity in particular. The problem of releasing data so that individuals who are the subjects of the data cannot be identified has been explored by works on *k-anonymity* [11, 12], statistical databases [6] 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 borrow the notion of identifier and quasi-identifier from Sweeney's work [12]. However, as outlined in Section 3, there are several aspects in which we differ.

5 Conclusion

Trust negotiation is a promising approach for establishing trust among subjects in open systems. Each subject specifies disclosure policies that list the credentials and attributes necessary to obtain its protected resources. The counterpart, in response, provides a disclosure set containing the necessary credentials. If the counterpart wants to remain anonymous, its disclosure set should not contain identity revealing information. We show how a subject can verify whether the disclosure-set preserves anonymity, and, if not, how it can use generalization and substitution techniques to transform its disclosure set to an anonymous one. The anonymous disclosure set may no longer satisfy the original disclosure policy. However, if it satisfies an alternate disclosure policy that implements the same property-based policy as the original one, the trust negotiation can proceed.

In future, we plan to propose the notion of *k*-anonymity-safe disclosure which ensures that the disclosure set submitted by a user is indistinguishable from the disclosure sets of *k* other subjects. We plan to develop crowd formation protocols that minimize the reliance on trusted third parties, and explore the use of incentives to obtain disclosure sets from other subjects.

References

1. E. Bertino, E. Ferrari, and A. Squicciarini. Trust Negotiations: Concepts, Systems and Languages. To appear in IEEE -CISE, Computing and Science Engineering.

2. E. Bertino, E. Ferrari, and A. Squicciarini. Trust-X a Peer to Peer Framework for Trust Establishment. To appear in IEEE TKDE, Transactions on Knowledge and Data Engineering.
3. E. Bertino, E. Ferrari, and A. C. Squicciarini. Privacy Preserving Trust Negotiations. In *4th International Workshop on Privacy Enhancing Technologies, Toronto, Canada*, May 2004.
4. P. Bonatti and S. Kraus. Foundations on Secure Deductive Databases. *IEEE TKDE, Transactions on Knowledge and Data Engineering, 7(3)*, pages 406–422, 1995.
5. P. Bonatti and P. Samarati. Regulating Access Services and Information Release on the Web. *7th ACM Conference on Computer and Communications Security, Athens, Greece*, November 2000.
6. J. Domingo-Ferrer. *Inference Control in Statistical Databases from Theory to Practice*. Volume 2316. Springer, 2002.
7. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
8. V. S. Iyengar. Transforming Data to Satisfy Privacy Constraints. *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada*, July 2002.
9. T. Yu K.E. Seamons, M. Winslett. Requirements for Policy Languages for Trust Negotiation. *Third IEEE International Workshop on Policies for Distributed Systems and Networks, Monterey, CA*, June 2002.
10. M. Naor. Bit Commitment Using Pseudorandomness. *Advances in Cryptology- 89*, 435, 1990.
11. P. Samarati and L. Sweeney. Generalizing Data to Provide Anonymity when Disclosing Information. In *Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, Washington*. ACM Press, June 1998.
12. L. Sweeney. *k*-anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuziness and Knowledge-based Systems, 10(5)*, pages 557–570, 2002.
13. K. E. Seamons T. Yu, M. Winslett. Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation. *ACM Transactions on Information and System Security, (6)1*, feb 2003.
14. M. Uschold and M. Gruninger. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review 11(2)*, pages 93–155, 1996.
15. M. Winsborough and N. Li. Safety in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, May 2004. Oakland, CA.
16. T. Yu and M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, May 2003. Oakland, CA.

Appendix – Algorithms for Ensuring Anonymity

In this appendix we give detailed algorithms pertaining to anonymity. Table 1 gives a table that lists the notations that we use in our algorithms.

Notation	Meaning
ANC_{C_i}	Set of ancestors of C_i obtained from CG_{C_i}
AS_{DSet}	Unblinded attributes in *DSet*
$C_{A_{ij}}$	Concept associated with attribute A_{ij}
CG_{C_i}	Concept graph containing node associated with concept C_i
CR_{ij}	Credential of type CT_i
CS_{DSet}	Concept set associated with unblinded attributes in *DSet*
$CS_{CR_{ij}}$	Concept set associated with unblinded attributes in CR_{ij}
DSet	Disclosure Set
IDSet	Set of identifier concepts
PCSet	Set of previously disclosed concepts
$Prof_{C_i}$	Credentials in profile containing unblinded attributes corres. to C_i
QIDSet	Set of quasi-identifier groups

Table 1. Notations used in algorithms

Algorithm 1 *Get Attributes Causing Identity Disclosure*
Input: *(i) DSet – the disclosure set that must be evaluated for identity disclosure, (ii) IDSet – set of identifier concepts, and (iii) QIDSet – set of Q_Id_Groups*
Output: *IdDiscAttr – set of attributes causing identity disclosure*

Procedure $GetIdDiscAttr(DSet, IDSet, QIDSet)$
begin
  $IdConDisc = CS_{DSet} \cap IDSet$
  for *each* $Q\_Id\_Group \in QIDSet$
    if $Q\_Id\_Group \cap CS_{DSet} = Q\_Id\_Group$
      $IdConDisc = IdConDisc \cup (Q\_Id\_Group \cap CS_{DSet})$
  for *each* $C_i \in IdDiscCon$
    $IdDiscAttr = AS_{DSet} \cap LangSet_{C_i}$
  return $IdDiscAttr$
end

The algorithm *GetIdDiscAttr* checks whether a disclosure set *DSet* contains attributes that cause identity disclosure. The first step finds the identifier concepts contained in CS_{DSet} which is the set of concepts associated with *DSet*. The second step finds the quasi-identifier concepts in CS_{DSet} that cause identity disclosure. The final step is to find the attributes corresponding to the identifier and the quasi-identifier concepts found in the earlier two steps. These set of attributes cause identity disclosure and are returned to the caller.

Algorithm 2 *Anonymize Disclosure Set Using Generalization/Substitution*
Input: *(i) DSet – the original disclosure set that must be made anonymous, (ii) IDSet – set of identifier concepts, and (iii) QIDSet – set of Q_Id_Groups*

Output: *returns $DSet'$ – anonymous disclosure set or an empty set if anonymity cannot be achieved*

Procedure *Anonymize(DSet, IDSet, QIDSet)*
begin
$IdDiscAttr = GetIdDiscAttr(DSet, IDSet, QIDSet)$
for *each attribute* $A_{ij} \in IdDiscAttr$
$DSet' = DSet - \{CR_{im}\}$
for *each requested attribute* A_{ik} *of* CR_i
if $A_{ik} \in IdDiscAttr$ */* A_{ik} caused identity disclosure */*
generalize = true
for *each element* $p \in AS_{DSet'}$
if $C_{A_{ik}} \subseteq C_p$ *or* $C_p \subseteq C_{A_{ik}}$
generalize = false
if *generalize*
$CR_{rs} = SelectAncCred(DSet', IDSet, QIDSet, C_t)$
if $CR_{rs} \neq NULL$
$DSet' = DSet' \cup \{CR_{rs}\}$
else
return *NULL*
else */* A_{ik} did not cause identity disclosure */*
substitute = true
for *each element* $p \in AS_{DSet'}$
if $C_{A_{ik}} \subseteq C_p$
substitute = false
if *substitute*
$CR_{rs} = SelectBestCand(DSet', IDSet, QIDSet, C_t)$
if $CR_{rs} \neq NULL$
$DSet' = DSet' \cup \{CR_{rs}\}$
else
return *NULL*
return $DSet'$
end

The algorithm *Anonymize* works as follows. It first gets the attributes causing identity disclosure by calling *GetIdDiscAttr*. For each attribute A_{ij} causing an identity disclosure, the corresponding credential CR_{im} is removed from the disclosure set. This causes the requested attributes A_{ik} contained in CR_{im} to be removed as well. If A_{ik} caused an identity disclosure, then A_{ik} must be generalized unless an attribute corresponding to a generalized or specialized or same concept is already present in the remaining disclosure set. To generalize, we

call the function *SelectAncCred* that returns an anonymity preserving credential corresponding to one of its ancestor concepts. If such a credential cannot be found, then the algorithm returns with a null value. On the other hand if the requested attribute A_{ik} did not cause an identity disclosure but some other attributes contained in CR_{im} did, then A_{ik} does not need to be generalized. In such cases, if an attribute exists in remaining $DSet'$ that belongs to the same or more specialized concept as A_{ik}, we do not have to look for a substitution for A_{ik}. If no such attributes exist, we call *SelectBestCand* that selects an anonymity preserving credential that contains an unblinded attribute corresponding to the concept $C_{A_{ik}}$. If no such credential can be found, the function returns with a null value. The process is repeated for every attribute causing identity disclosure. The function returns the anonymized disclosure set at the end or null if anonymization is not possible.

Algorithm 3 *Selecting the Best Candidate from a Concept*
Input: *(i) DSet – the disclosure set to which the new credential must be added, (ii) IDSet – set of identifier concepts, and (iii) QIDSet – set of Q_Id_Groups, (iv)* C_j *– the concept from which the best credential must be selected.*
Output: *returns* CR_r *– the most suitable credential or null if none can be found*

Procedure $SelectBestCandidate(DSet, IDSet, QIDSet, C_j)$
begin
  $minm = infinity$
  $min = NULL$
  for *each* $CR_{im} \in Prof_{C_j}$
    if $GetIdDiscAttr(CR_i \cup DSet, IDSet, QIDSet) = \{\}$
      $QIDrelease_{im} = QIDSet \cap CS_{CR_{im}}$
      if $minm > QIDrelease_{im}$
        $minm = QIDrelease_{im}$
        $min = CR_{im}$
  return *min*
end

The algorithm *SelectBestCandidate* selects a credential present in the subject's profile corresponding to a given concept. The objective of the algorithm is to select a candidate credential from the set of credentials in $Prof_{C_j}$. To qualify for a candidate the credential together with the given disclosure set (*DSet*) should not cause an identity disclosure. From the set of candidates, we use a heuristic to determine the best choice. The heuristic chooses the candidate that will cause minimum number of quasi-identifiers to be revealed.

Algorithm 4 *Selecting an Ancestor Concept Credential*
Input: *(i) DSet – the original disclosure set that must be made anonymous, (ii) IDSet – set of identifier concepts, (iii) QIDSet – set of Q_Id_Groups, and (iv)* C_i *– the concept whose ancestor credential must be selected.*
Output: *returns DSet′ – anonymous disclosure set or an empty set if anonymity cannot be achieved*

Procedure $SelectAncCred(DSet, IDSet, QIDSet, C_i)$
begin
    minm = infinity
    cred = NULL
    for *each* $t \in ANC_{C_i}$
        $CR_{rm} = SelectBestCandidate(DSet', IDSet, QIDSet, C_t)$
        if $CR_{rm} \neq NULL$
            $count_t = QIDSet \cap CS_{CR_{rm}}$
            if $count_t < minm$
                $minm = count_t$
                cred = CR_{rm}
    return *cred*
end

The above algorithm selects a credential corresponding to an ancestor concept. From all the ancestors, it tries to select the best credential that minimizes the number of disclosure of quasi-identifiers.