Privacy in the Electronic Society

Sabrina De Capitani di Vimercati and Pierangela Samarati

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
26013 Crema - Italy
samarati@dti.unimi.it

Abstract. In today's systems, the protection of privacy is an increasing concern. Users are often required to provide a vast amount of information about themselves on which the restrictions to be enforced may come from different input requirements, possibly under the control of different authorities. In addition, users have often little control over their personal information once it has been disclosed to third parties. Secondary usage regulations are therefore increasingly demanding attention.
In this paper, we present the emerging trends in the data protection field to address the new needs and desiderata of today's systems.

1 Introduction

Nowadays, a global information infrastructure connects remote parties worldwide through the use of large scale networks, relying on application level protocols and services such as the World Wide Web. Human activities are increasingly based on the use of remote resources and services [34], and on the interaction between different, remotely located, and unknown parties. The vast amounts of personal information thus available has led to growing concerns about the privacy of their users: effective information sharing and dissemination can take place only if the users have some assurance that, while releasing information, disclosure of sensitive information is not a risk.

Unfortunately, users' privacy is often poorly managed and sometimes abused. For instance, it is well known how personal information is often disclosed to third parties without the consent of legitimate data owners or that there are professional services specialized on gathering and correlating data from heterogeneous repositories, which permit to build user profiles and possibly to disclose sensitive information not voluntarily released by their owners. In such a scenario, protecting privacy requires the investigation of different aspects, including:

- *data protection requirements composition* to take into consideration requirements coming from the data owner, the data holder, and possible privacy law. These multiple authorities scenario should be supported from the administration point of view providing solutions for modular, large-scale, scalable policy composition and interaction [11, 47].

- *security and privacy specifications and secondary usage control* to identify under which conditions a party can trust others for their security and privacy. Trust models are one of the techniques be evaluated [9, 10]. In particular, *digital certificates* (statements certified by given entities) can be used to establish properties of their holder (such as identity, accreditation, or authorizations) [12, 22, 32, 38, 58]. Moreover, since users often have no idea on how their personal information may be used subsequently, it must also be given a mechanism to specify whether or not to consent to the future use of that information in secondary applications [6].
- *inference and linking attacks protection* that is often impossible, if not at the price of not disclosing any information at all. Among the techniques used to protect the released data, k-anonymity promises to be a successful solution towards increasing privacy.

In this paper, we discuss these problems and illustrate some current approaches and ongoing research. The remainder of this paper is organized as follows. Section 2 addresses the problem of combining authorization specifications that may be independently stated. We describe the characteristics that a policy composition framework should have and illustrate some current approaches and open issues. Section 3 addresses the problem of defining policies in open environments such as the Internet. We then describe current approaches and open issues. Section 4 addresses the problem of protecting released data against inference and linking attacks. We describe the k-anonymity concept and illustrate some related current approaches and open issues. Finally, Section 5 concludes the paper.

2 Policy composition

Traditionally, authorization policies have been expressed and managed in a centralized manner: one party administers and enforces the access control requirements. In many cases however, access control needs to combine restrictions independently stated that should be enforced as one, while retaining their independence and administrative autonomy. For instance, the global policy of a large organization can be the combination of the policies of its independent and geographically distributed departments. Each of these departments is responsible for defining access control rules to protect resources and each brings its own set of constraints. To address these issues, a *policy composition framework* by which different component policies can be integrated while retaining their independence should be designed. The framework should be flexible to support different kinds of composition, yet remain simple so to keep control over complex compound policies. It should be based on a solid formal framework and a clear semantics to avoid ambiguities and enable correctness proofs.

Some of the main requirements that a policy composition framework should have can be summarized as follows [11].

- *Heterogeneous policy support.* The composition framework should be able to combine policies expressed in arbitrary languages and possibly enforced by

different mechanisms. For instance, a datawarehouse may collect data from different data sources where the security restrictions autonomously stated by the sources and associated with the data are stated with different specification languages, or refer to different paradigms (e.g., open vs closed policy).

- *Support of unknown policies.* It should be possible to account for policies that may be not completely known or even be specified and enforced in external systems. These policies are like "black-boxes" for which no (complete) specification is provided, but that can be queried at access control time. Think, for instance, of a situation where given accesses are subject, in addition to other policies, to a policy P enforcing "central administration approval". Neither the description of P, nor the specific accesses that it allows might be available; whereas P can respond yes or no to each specific request. Runtime evaluation is therefore the only possible option for P. In the context of a more complex and complete policy including P as a component, the specification could be partially compiled, leaving only P (and its possible consequences) to be evaluated at run time.
- *Controlled interference.* Policies cannot always be combined by simply merging their specifications (even if they are formulated in the same language), as this could have undesired side effects. The accesses granted/denied might not correctly reflect the specifications anymore. As a simple example, consider the combination of two systems P_{closed}, which applies a closed policy, based on rules of the form "*grant access if* $(s, o, +a)$", and P_{open} which applies an open policy, based on rules of the form "*grant access if* $\neg(s, o, -a)$". Merging the two specifications would cause the latter decision rule to derive all authorizations not blocked by P_{open}, regardless of the contents of P_{closed}. Similar problems may arise from uncontrolled interaction of the derivation rules of the two specifications. Besides, if the adopted language is a logic language with negation, the merged program might not be stratified (which may lead to ambiguous or undefined semantics).
- *Expressiveness.* The language should be able to conveniently express a wide range of combinations (spanning from minimum privileges to maximum privileges, encompassing priority levels, overriding, confinement, refinement etc.) in a uniform language. The different kinds of combinations must be expressed without changing the input specifications (as it would be necessary even in most recent and flexible approaches) and without ad-hoc extensions to authorizations (like those introduced to support priorities). For instance, consider a policy P_1 regulating access to given documents and the central administration policy P_2. Assume that access to administrative documents can be granted only if authorized by both P_1 and P_2. This requisite can be expressed in existing approaches only by explicitly extending all the rules possibly referred to administrative documents to include the additional conditions specified by P_2. Among the drawbacks of this approach is the rule explosion that it would cause and the complex structure and loss of controls of two specifications; which, in particular, cannot be maintained and managed autonomously anymore.

– *Support of different abstraction levels.* The composition language should highlight the different components and their interplay at different levels of abstraction. This is important to: *i)* facilitate specification analysis and design; *ii)* facilitate cooperative administration and agreement on global policies; *iii)* support incremental specification by refinement.
– *Support for dynamic expressions and controlled modifications.* Mobile policies that follow (*stick with*) the data and can be enriched, subject to constraints, as the data move.
– *Formal semantics.* The composition language should be declarative, implementation independent, and based on a solid formal framework. The need of an underlying formal framework is widely recognized and in particular it is important to *i)* ensure non-ambiguous behavior, and *ii)* reason about and prove specifications properties and correctness [27]. In our framework this is particular important in the presence of *incomplete* specifications.

2.1 Overview of ongoing work

Various models have been proposed to reason about security policies [1, 21, 24, 36]. In [1, 24] the authors focused on the secure behavior of program modules. McLean [36] proposed a formal approach including combination operators: he introduced an algebra of security which enables to reason about the problem of policy conflict that can arise when different policies are combined. However, even though this approach permits to detect conflicts between policies, it did not propose a method to resolve the conflicts and to construct a security policy from inconsistent sub-policies. Hosmer [21] introduced the notion of meta-policies (i.e., policies about policies), an informal framework for combining security policies. Subsequently, Bell [8] formalized the combination of two policies with a function, called *policy combiner*, and introduced the notion of *policy attenuation* to allow the composition of conflicting security policies. Other approaches are targeted to the development of a uniform framework to express possibly heterogeneous policies [25, 29, 51]. Recently, Bonatti et al. [11] proposed an algebra for combining security policies together with its formal semantics. Following Bonatti et al.'s work, Jajodia et al. [47] presented a propositional algebra for policies with a syntax consisting of abstract symbols for atomic policy expressions and composition operators. The basic idea of these proposals is to define a set of policy operators used for combining different policies. In particular, in [11] a policy is defined as a set of triples of the form (s,o,a), where s is a constant in (or a variable over) the set of subjects S, o is a constant in (or a variable over) the set of objects O, and a is a constant in (or a variable over) the set of actions A. Here, complex policies can then be obtained by combining policy identifiers, denoted P_i, through the following *algebra operators*.

– *Addition* (+) merges two policies by returning their set union. For instance, in an organization composed of different divisions, access to the main gate can be authorized by any of the administrator of the divisions (each of them knows users who needs the access to get to their division). The totality of the

accesses through the main gate to be authorized would then be the union of the statements of each single division. Intuitively, additions can be applied in any situation where accesses can be authorized if allowed by any of the component (operand) policies.

- *Conjunction* (&) merges two policies by returning their intersection. For instance, consider an organization in which divisions share certain documents (e.g., clinical folders of patients). Access to the documents is to be allowed only if all the authorities that have a say on the document agree on it. Intuitively, while addition enforces maximum privilege, conjunction enforces minimum privilege.
- *Subtraction* (−) restricts a policy by eliminating all the accesses in the second policy. Intuitively, subtraction specifies exceptions to statements made by a policy and it encompasses the functionality of negative authorizations in existing approaches, while probably providing a clearer view of the combination of positive and negative statements. The advantages of subtraction over explicit denials include a simplification of the conflict resolution policies and a clearer semantics. In particular, the scoping of a difference operation allows to clearly and unambiguously express the two different uses of negative authorizations, namely *exceptions to positive statements* and *explicit prohibitions*, which are often confused in the models or requires explicit ad-hoc extension to the authorization form. The use of subtraction provides extensible as the policy can be enriched to include different overriding/conflict resolution criteria as needed in each specific context, without affecting the form of the authorizations.
- *Closure* (∗) closes a policy under a set of inference (derivation) rules. Intuitively, derivation rules can be thought of as logic rules whose head is the authorization to be derived and whose body is the condition under which the authorization can be derived. Examples of derivation rules can be found in essentially all logic based authorization languages proposed in the literature, where derivation rules are used, for example, to enforce propagation of authorizations along hierarchies in the data system, or to enforce more general forms of implication, related to the presence or absence of other authorizations, or depending on properties of the authorizations [25].
- *Scoping restriction* (ˆ) restricts the application of a policy to a given set of subjects, objects, and actions. Scoping is particularly useful to "limit" the statements that can be established by a policy and, in some way, enforcing authority confinement. Intuitively, all authorizations in the policy which do not satisfy the scoping restriction are ignored, and therefore ineffective. For instance, the global policy of an organization can identify several component policies which need to be merged together; each component policy may be restricted in terms of properties of the subjects, objects and actions occurring in its authorizations.[1]

[1] A simple example of scoping constraint is the limitation of authorizations that can be stated by a policy to a specific portion of the data system hierarchy [25].

– *Overriding* (o) replaces part of a policy with a corresponding fragment of the second policy. The portion to be replaced is specified by means of a third policy. For instance, consider the case where users of a library who have passed the due date for returning a book cannot borrow the same book anymore *unless* the responsible librarian vouchers for (authorizes) the loan. While the accesses otherwise granted by the library are stated as a policy P_{lib}, black-list of accesses, meaning triples (`user`, `book`, `loan`) are stated as a policy P_{block}. In the absence of the *unless* portion of the policy, the accesses to be allowed would simply be $P_{lib} - P_{block}$. By allowing the librarian discretion for "overriding" the black list, calling P_{vouch} the triples authorized by the librarians, we can express the overall policy as $o(P_{lib}, P_{vouch}, P_{block})$.
– *Template* (τ) defines a partially specified policy that can be completed by supplying the parameters. Templates are useful for representing partially specified policies, where some component X is to be specified at a later stage. For instance, X might be the result of further policy refinement, or it might be specified by a different authority.

To fix ideas and make concrete examples, consider a drug-effects warehouse that might draw information from many hospitals. We assume that the warehouse receives information from three hospitals, denoted h_1, h_2, and h_3, respectively. These hospitals are responsible for granting access to information under their (possibly overlapping) authority domains, where domains are specified by a scoping function. The statements made by the hospitals are then unioned meaning that an access is authorized if any of the hospital policy states so. In term of the algebra, the warehouse policy can be represented as an expression of the form $P_1\hat{}[o \leq \mathtt{O}_{\mathtt{h}_1}] + P_2\hat{}[o \leq \mathtt{O}_{\mathtt{h}_2}] + P_3\hat{}[o \leq \mathtt{O}_{\mathtt{h}_3}]$, where P_i denotes the policy defined by hospital h_i, and the scope restriction $\hat{}[o \leq \mathtt{O}_{\mathtt{h}_i}]$ selects the authorizations referred to objects released by hospital h_i.[2] Each policy P_i can then be further refined. For instance, consider policy P_1. Suppose that hospital h_1 defines a policy P_{drug} regulating the access to drug-effects information. Assume also that the drug-effects information can be released only if the hospital's researchers obtain a patient's consent; $P_{consents}$ reports accesses to drug-effects information that the patients agree to release. We can then express P_1 as $P_{drug} \& P_{consents}$.

2.2 Open issues

We briefly describe some open issues that need to be taken into consideration in the future development of a policy composition framework.

– Investigate different *algebra operators and formal languages* for enforcing the algebra and proving properties. The proposed policy composition frameworks can be enriched by adding new operators. Also, the influence of different rule languages on the expressiveness of the algebra has to be investigated.

[2] We assume that the information collected from the hospitals can be organized in abstractions defining groups of objects that can be collectively referred to with a given name. Objects and groups thereof define a partial order that naturally introduces a hierarchy, where $\mathtt{O}_{\mathtt{h}_i}$ contains objects obtained from hospital h_i.

- *Administrative policies and language* with support for multiple authorities. The proposed approaches could be enriched by adding administrative policies that define who can specify authorizations/rules (i.e., who can define a component policy) governing access control.
- *Policy enforcement.* The resolution of the algebraic expression defining a policy P determines a set of ground authorization terms, which define exactly the accesses to be granted according to P. Different strategies can be used to evaluate the algebraic expression for enforcing access control: materialization, run-time evaluation, and partial evaluation. The first one allows a one-time compilation of the policy against which all accesses can be efficiently evaluated and which will then need to be updated only if the policy changes. The second strategy consists in enforcing a run-time evaluation of each request (access triple) against the policy expression to determine whether the access should be allowed. Between these two extremes, possibly combining the advantages of them, there are partial evaluation approaches, which can enforce different degrees of computation/materialization.
- Incremental approaches to enforce *changes to component policies*. When a materialization approach is used to evaluate the algebraic expression for enforcing access control, incremental approaches [43] can be applied to minimize the recomputation of the policy.
- *Mobile policies.* Intuitively, a *mobile policy* is the policy associated with an object and that follows the object when it is passed to another site. Because different and possibly independent authorities can define different parts of the mobile policy in different time instants, the policy can be expressed as a policy expression. In such a context, there is the problem on how ensure the obedience of policies when the associated objects move around. Within the context of mobile policies we can also classify the problem of providing support for handling "sticky" policies [13], that is, policies that remain attached to data as they move between entities and are needed to enforce secondary use constraints (see Section 3). Mobile policies encompass also the problem of digital right management (DRM) as they also require constraints of the owner to remain attached to the data.

3 Access control in open systems

Open environments are characterized by a number of systems offering different resources/services. In such a scenario, interoperability is a very important issue and traditional assumptions for establishing and enforcing policies do not hold anymore. A server may receive requests not just from the local community of users, but also from remote, previously unknown users. The server may not be able to authenticate these users or to specify authorizations for them (with respect to their identity). Early approaches that attempt to solve these issues, PolicyMaker [10] and KeyNote [9], basically use credentials to describe specific delegation of trusts among keys and to bind public keys to authorizations. Although early trust management systems do provide an interesting framework for

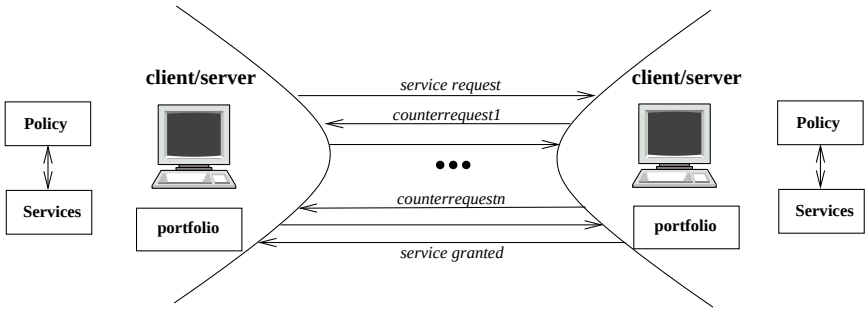

Fig. 1. Client/server interaction

reasoning about trust between unknown parties, assigning authorizations to keys may result limiting and make authorization specifications difficult to manage.

A promising direction to overcome such a disadvantage is represented by *digital certificates*. A digital certificate is basically the on-line counterparts of paper credentials (e.g., drivers licenses). Digital certificates can be used to determine whether or not a party may execute an access on the basis properties that the requesting party may have. These properties can be proven by presenting one or more certificates [22, 32, 38, 58]. The development and effective use of credential-based models require tackling several problems related to credential management and disclosure strategies, delegation and revocation of credentials, and establishment of credential chains [19, 33, 41, 42, 46, 50, 56].

Figure 1 depicts the basic scenario we consider. We are given different parties that interact with each other to offer services. A party can act both as a server and a client and each party has *i)* a set of services it provides and *ii)* a *portfolio* of properties (attributes) that the party enjoys. Access restrictions to the services are expressed by policies that specified the properties that a requester should enjoy to gain access to the services. The services are meant to offer certain functionalities that depend on the input parameters supplied by its users. Often input parameters must fulfill certain conditions to assure correct behavior of a service. We identified the following requirements for specifying credential-based access control.

- *Attribute interchange.* A server should be able to communicate to the client the requirements it need to satisfy to get access. Also, a client should be able to prove its eligibility for a service. This communication interchange could be performed in different ways (e.g., the involved parties can apply different strategies with respect to which properties are submitted).
- *Support for fine-grained reference to attributes within a credential.* The system should allow the selective disclosure of credentials which is a requirement that is not usually supported because users attributes are defined according to functional needs, making it easier to collect all credentials in a row instead of iteratively asking for the ones strictly necessary for a given service only.

- *Support for hierarchical relationships and abstractions on services and portfolio.* Attribute-based access control policies should be able to specify accesses to collection of services based upon collection of attributes processed by the requester.
- *Expressiveness and flexibility.* The system must support the specification of complex access control requirements. For instance, consider a service that offers telephone contracts and requires that the customer is at least 18 years of age. The telephone selling service has two input parameters, namely `homeAddress` and `noticePeriod`. The `homeAddress` must be a valid address in Italy and `noticePeriod` must be either one or three months. Further, the service's access control policy requires that contracts with one month notice period and home address outside a particular geographical region are closed only with users who can prove their AAA membership. Hence, we see that the access control requirements of a service may require more than one interaction between a client and a server.
- *Purpose specific permission.* The permission to release data should relate to the purpose for which data are being used or distributed. The model should prevent information collected for one purpose from being used for other purposes.
- *Support for meta-policies.* The system should provide meta-policies for protecting the policy when communication requisites. This happens when a list of alternatives (policies) that must be fulfilled to gain the access to the data/service is returned to the counterpart. For instance, suppose that the policy returned by the system is "citizenship=EU". The party can decide to return to the client either the policy as it is or a modified policy simply requesting the user to prove its nationality (then protecting the information that access is restricted to EU citizens).
- *Support for secondary use specifications and control.* The information owner should be able to control further dissemination and use of personal information. This represents a novel feature that is no simply concerned with authorizing the access to data and resources but also with defining and enforcing the way data and resources are subsequently managed.

3.1 Overview of ongoing work

The first proposals investigating the application of credential-based access control regulating access to a server were made by Winslett et al. [41, 50]. Here, access control rules are expressed in a logic language and rules applicable to an access can be communicated by the server to clients. In [49, 56] the authors investigated trust negotiation issues and strategies that a party can apply to select credentials to submit to the opponent party in a negotiation. In [12] the authors proposed a uniform framework for regulating service access and information disclosure in an open, distributed network system like the Web. Like in previous proposals, access regulations are specified as logical rules, where some predicates are explicitly identified. Certificates are modeled as *credential expressions* of the form "credential_name(attribute_list)", where credential_name is the attribute

credential name and attribute_list is a possibly empty list of elements of the form "attribute_name=value_term", where value_term is either a ground value or a variable. Besides credentials, the proposal also allows to reason about declarations (i.e., unsigned statements) and user-profiles that the server can maintain and exploit for taking the access decision. Communication of requisites to be satisfied by the requester is based on a filtering and renaming process applied on the server's policy, which exploits partial evaluation techniques in logic programs. Yu et al. [56–58] developed a service negotiation framework for requesters and providers to gradually expose their attributes. In [56] the *PRUdent NEgotiation Strategy* (PRUNES) has been presented. This strategy ensures that the client communicates its credentials to the server only if the access will be granted and the set of certificates communicated to the server is the minimal necessary for granting it. Each party defines a set of *credential policies* that regulates how and under what conditions the party releases its credentials. The negotiation consists of a series of requests for credentials and counter-requests on the basis of the parties' credential policies. The credential policies established can be graphically represented through a tree, called *negotiation search tree*, composed of two kinds of nodes: *credential nodes*, representing the need for a specific credential, and *disjunctive nodes*, representing the logic operators connecting the conditions for credential release. The root of a tree node is a service (i.e., the resource the client wants to access). The negotiation can therefore be seen as a backtracking operation on the tree. The backtracking can be executed according to different strategies. For instance, a *brute-force* backtracking is complete and correct, but is too expensive to be used in a real scenario. The authors therefore proposed the PRUNES method that prunes the search tree without compromising completeness or correctness of the negotiation process. The basic idea is that if a credential C has just been evaluated and the state of the system is not changed too much, than it is useless to evaluate again the same credential, as the result will be exactly as the result previously computed. The same research group proposed also a method for allowing parties adopting different negotiation strategies to interoperate through the definition of a *Disclosure Tree Strategy* (DTS) family [57]. The authors show that if two parties use different strategies from the DST family, they are able to establish a negotiation process. The DTS family is a closed set, that is, if a negotiation strategy can interoperate with any DST strategy, it must also be a member of the DST family.

In [55] a *Unified Schema for Resource Protection* (UniPro) has been proposed. This mechanism is used to protect the information in policies. UniPro gives (opaque) names to policies and allows any named policy P_1 to have its own policy P_2 meaning that the contents of P_1 can only be disclosed to parties who have shown that they satisfy P_2. Another approach for implementing access control based on credentials is the *Adaptive Trust Negotiation and Access Control* (ATNAC) [39]. This method grants or denies access to a resource on the basis of a *suspicion level* associated with subjects. The suspicion level is not fixed but may vary on the basis of the probability that the user has malicious intents. In [45] the authors proposed to apply the automated trust negotiation

technology for enabling secure transactions between portable devices that have no pre-existing relationship. In [58] the authors presented a negotiation architecture, called *TrustBuilder*, that is independent from the language used for policy definition and from the strategies adopted by the two parties for policy enforcement. Other logic-based access control languages based on credentials have been introduced. For instance, D1LP and RT [30, 31], the SD3 language [26], and Binder [17]. In [25, 51] logic languages are adopted to specify access restrictions in a certificate-based access control model.

Few proposals have instead addressed the problem of how to regulate the use of personal information in secondary applications. In [5], the authors proposed an XML-based privacy preference expression language, called *PReference Expression for Privacy* (PREP), for storing the user's privacy preferences with Liberty Alliance. PREP allows users to specify, for each attribute, a *privacy label* that is characterized by a purpose, type of access, recipient, data retention, remedies, and disputes. The *Platform for Privacy Preferences Project* (P3P) [53] is another XML-based language that allows service providers and users to reach an agreement on the release of personal data. Basically, a service provider can define a P3P policy, which is an XML document, where it is possible to define the recipient of the data, desired data, consequence of data release, purpose of data collection, data retention policy, and dispute resolution mechanisms. Users specify their privacy preferences in term of a policy language, called APPEL [52], and enforce privacy protection through a user agent: the user agent compares the users' privacy policy with the service provider P3P policy and checks whether the P3P policy conforms to the user privacy preferences. Although P3P is a good starting point, it is not widely adopted by the service providers and presents some limitations on the user side [4]. The main limitation is that the definition of simple privacy preferences is a complex task and writing APPEL preferences is error prone. For this reason, Agrawal et al. [4] proposed a new language, called XPref, for user preferences. However, both APPEL and XPref are not sufficiently expressive because, for example, they do not support negotiation and contextual information, and they do not allow the definition of attribute-based conditions. Another important disadvantage of these approaches is that users have a passive role: a service provider defines a privacy policy that users can only accept or reject. In [6] a new type of privacy policy, called *data handling policy*, that regulates the secondary use of a user's personal data has been discussed. A data handling policy regulates how Personal Identifiable Information (PII) will be used (e.g., information collected through a service will be combined with information collected from other services and used in aggregation for market research purposes), how long PII will be retained (e.g., information will be retained as long as necessary to perform the service), and so on. Users can therefore use these policies to define how their information will be used and processed by the counterpart.

3.2 Open issues

Although current approaches supporting attribute-based policies are technically mature enough to be used in practical scenarios, there are still some issues that need to be investigated in more detail to enable more complex applications. We summarize these issues as follows [12].

- *Ontologies.* Due to the openness of the scenario and the richness and variety of security requirements and attributes that may need to be considered, it is important to provide parties with a means to understand each other with respect to the properties they enjoy (or request the counterpart to enjoy). Therefore, common languages, dictionaries, and ontologies must be developed.
- *Access control evaluation and outcome.* Users may be occasional and they may not know under what conditions a service can be accessed. Therefore, to make a service "usable", access control mechanisms cannot simply return "yes" or "no" answers. It may be necessary to explain why authorizations are denied, or - better - how to obtain the desired permissions. Therefore, the system can return an undefined response meaning that current information is insufficient to determine whether the request can be granted or denied. For instance, suppose that a user can use a particular service only if she is at least eighteen and provides a credit card. According to this policy, two cases can occur: *i)* the system knows that the user is not yet eighteen and therefore returns a negative response; *ii)* the user has proved that she is eighteen and the system returns an undefined response together with the request to provide the information of a credit card.
- *Privacy-enhanced policy communication.* Since access control does not return only a "yes" or "no" access decision, but it returns the information about which conditions need to be satisfied for the access to be granted ("undefined" decision), the problem of communicating such conditions to the counterpart arises. To fix the ideas, let us see the problem from the point of view of the server (the client's point of view is symmetrical). A naive solution consists in giving the client a list with all the possible sets of credentials that would enable the service. This solution is however not feasible due to the large number of possible alternatives. Also, the communication process should not disclose "too much" of the underlying security policy, which might also be regarded as sensitive information.
- *Negotiation strategy.* Credentials grant parties different choices with respect to what release (or ask) the counterpart and when to do it, thus allowing for multiple trust negotiation strategies [57]. For instance, an *eager* strategy, requires parties to turn over all their credentials if the release policy for them is satisfied, without waiting for the credentials to be requested. By contrast, a *parsimonious* strategy requires that parties only release credentials upon explicit request by the server (avoiding unnecessary releases).
- *Composite services.* In case of a composite service (i.e., a service that is decomposable into other services called component services) there must be

a semi-automatic mechanism to calculate the policy of a composite service from the policies of its component services.
- *Semantics-aware rules.* Although attribute-based policies allow the specifications of restrictions based on generic attributes or properties of the requestor and the resources, they do not fully exploit the semantic power and reasoning capabilities of emerging web applications. It is therefore important to be able to specify access control rules about subjects accessing the information and about resources to be accessed in terms of rich ontology-based metadata (e.g., Semantic Web-style ones) increasingly available in advanced e-services applications [16].

4 Privacy issues in data collection and disclosure

Internet provides unprecedented opportunities for the collection and sharing of privacy-sensitive information from and about users. Information about users is collected every day, as they join associations or groups, shop for groceries, or execute most of their common daily activities. Consequently, users have very strong concerns about the privacy of their personal information and they fear that their personal information can be misused. Protecting privacy requires therefore the investigation of many different issues including the problem of *protecting released information against inference and linking attacks*, which are becoming easier and easier because of the increased information availability and ease of access as well as the increased computational power provided by today's technology. In fact, released data too often open up privacy vulnerabilities through, for example, data mining techniques and record linkage. Indeed, the restricted access to information and its expensive processing, which represented a form of protection in the past do not hold anymore. In addition, while in the past data were principally released in tabular form (macrodata) and through statistical databases, many situations require today that the specific stored data themselves, called microdata, be released. The advantage of releasing microdata instead of specific pre-computed statistics is an increased flexibility and availability of information for the users. At the same time however microdata, releasing more specific information, are subject to a greater risk of privacy breaches. To this purpose, the main requirements that must be taken into account are the following.

- *Identity disclosure protection.* Identity disclosure occurs whenever it is possible to identify a subject, called *respondent*, from the released data. It should therefore be adopted techniques for limiting the possibility of identifying respondents.
- *Attribute disclosure protection.* Identity disclosure protection alone do not guarantee privacy of sensitive information because all the respondents in a group could have the same sensitive information. To overcome this issue, mechanisms that protect sensitive information about respondents should be adopted.
- *Inference channel.* Given the possibly enormous amount of data to be considered, and the possible inter-relationships between data, it is important that

the security specification and enforcement mechanisms provide automatic support for complex security requirements, such as those due to inference and data association channels.

To protect the anonymity of the respondents to whom the released data refer, data holders often remove, encrypt, or code identity information. Identity information removed or encoded to produce anonymous data includes names, telephone numbers, and Social Security Numbers. Although apparently anonymous, however, the de-identified data may contain other quasi-identifying attributes such as race, date of birth, sex, and geographical location. By linking such attributes to publicly available databases associating them with the individual's identity, data recipients can determine to which individual each piece of released data belongs, or restrict their uncertainty to a specific subset of individuals. This problem has raised particular concerns in the medical and financial fields, where microdata, which are increasingly released for circulation or research, can be or have been subject to abuses, compromising the privacy of individuals.

SSN	Name	Race	Date of birth	Sex	ZIP	Marital status	Disease
		asian	71/07/05	F	20222	Single	hypertension
		asian	74/04/13	F	20223	Divorced	Flu
		asian	74/04/15	F	20239	Married	chest pain
		asian	73/03/13	M	20239	Married	Obesity
		asian	73/03/18	M	20239	Married	hypertension
		black	74/11/22	F	20238	Single	short breath
		black	74/11/22	F	20239	Single	Obesity
		white	74/11/22	F	20239	Single	Flu
		white	74/11/22	F	20223	Widow	chest pain

(a)

Name	Address	City	ZIP	DOB	Sex	Status
.....	.....	.....	.....	.....	.....	.....
Susan Doe	Eye street	Washington DC	20222	71/07/05	F	single
.....	.....	.....	.....	.....	.....	.....

(b)

Fig. 2. An example of private table PT (a) and non de-identified public available table

To better illustrate the problem, consider the microdata table in Figure 2(a) and the non de-identified public available table in Figure 2(b). In the microdata table, which we refer to as private table (PT), data have been de-identified by suppressing names and Social Security Numbers (SSNs) so not to explicitly disclose the identities of respondents. However, the released attributes `Race`, `Date of birth`, `Sex`, `ZIP`, and `Marital status` can be linked to the public tu-

ples in Figure 2(b) and reveal information on `Name`, `Address`, and `City`. In the private table, for example, there is only one single female (F) born on 71/07/05 and living in the 20222 area. This combination, if unique in the external world as well, uniquely identifies the corresponding tuple as pertaining to "Susan Doe, 20222 Eye Street, Washington DC", thus revealing that she has reported hypertension. While this example demonstrates an exact match, in some cases, linking allows one to detect a restricted set of individuals among whom there is the actual data respondent.

Among the microdata protection techniques used to protect de-identified microdata from linking attacks, there are the commonly used approaches like sampling, swapping values, and adding noise to the data while maintaining some overall statistical properties of the resulting table [15]. However, many uses require the release and explicit management of microdata while needing truthful information within each tuple. This "data quality" requirement makes inappropriate those techniques that disturb data and therefore, although preserving statistical properties, compromise the correctness of single tuples [15]. k-anonymity, together with its enforcement via *generalization* and *suppression*, has been proposed as an approach to protect respondents' identities while releasing truthful information [40].

The concept of k-anonymity tries to capture, on the private table to be released, one of the main requirements that has been followed by the statistical community and by agencies releasing the data, and according to which the *released data should be indistinguishably related to no less than a certain number of respondents*.

The set of attributes included in the private table, also externally available and therefore exploitable for linking, is called *quasi-identifier*. The requirement above-mentioned is then translated in the k-anonymity requirement [40]: *each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents*. Since it seems impossible, or highly impractical and limiting, to make assumptions on the datasets available for linking to external attackers or curious data recipients, essentially k-anonymity takes a safe approach requiring that, in the released table itself, the respondents be indistinguishable (within a given set) with respect to a set of attributes. To guarantee the k-anonymity requirement, k-anonymity requires each quasi-identifier value in the released table to have at least k occurrences. This is clearly a sufficient condition for the k-anonymity requirement: if a set of attributes of external tables appears in the quasi-identifier associated with the private table PT, and the table satisfies this condition, the combination of the released data with the external data will never allow the recipient to associate each released tuple with less than k respondents. For instance, with respect to the microdata table in Figure 1 and the quasi-identifier `Race`, `Date of birth`, `Sex`, `ZIP`, `Marital status`, the table satisfies k-anonymity with $k = 1$ only, since there are single occurrences of values over the quasi-identified (e.g., "asian, 71/07/05, F, 20222, single").

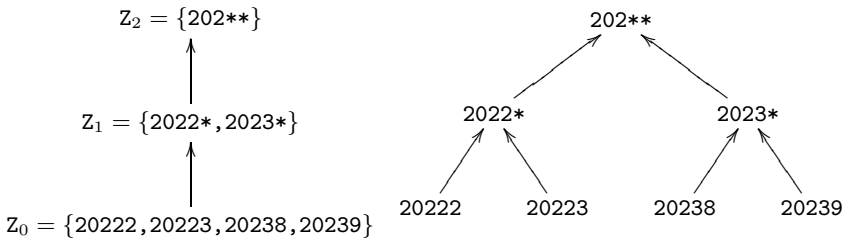

Fig. 3. An example of domain generalization hierarchy for attribute ZIP

4.1 Overview of ongoing work

As above-mentioned, k-anonymity proposals focus on *generalization* and *suppression* techniques. Generalization consists in representing the values of a given attribute by using more general values. This technique is based on the definition of a *generalization hierarchy*, where the most general value is at the root of the hierarchy and the leaves correspond to the most specific values. Formally, the notion of *domain* (i.e., the set of values that an attribute can assume) is extended by assuming the existence of a set of *generalized domains*. The set of original domains together with their generalizations is referred to as Dom. Each generalized domain contains generalized values and there exists a mapping between each domain and its generalizations. This mapping is stated by means of a *generalization relationship* $\leq_D$. Given two domains D_i and $D_j \in \mathsf{Dom}$, $D_i \leq_D D_j$ states that values in domain D_j are generalizations of values in D_i. The generalization relationship $\leq_D$ defines a partial order on the set Dom of domains, where each D_i has at most *one* direct generalization domain D_j, and all values in each domain can always be generalized to a single value. The definition of a generalization relationship implies the existence, for each domain $D \in \mathsf{Dom}$, of a totally ordered hierarchy, called *domain generalization hierarchy*, denoted DGH_D. As an example, consider attribute ZIP code and suppose that a step in the corresponding generalization hierarchy consists in suppressing the least significant digit in the ZIP code. Figure 3 illustrates the corresponding domain generalization hierarchy. In this case, for example, if we choose to apply one generalization step, values 20222, 20223, 20238, and 20239 are generalized to 2022* and 2023*. A generalization process therefore proceeds by replacing the values represented by the leaf nodes with one of their ancestor nodes at a higher level. Different generalized microdata tables can be built, depending on the amount of generalization applied on the considered attribute.

Suppression is a well-known technique that consists in protecting sensitive information by removing it. The introduction of suppression can reduce the amount of generalization necessary to satisfy the k-anonymity constraint.

Generalization and suppression can be applied at different levels of granularity. Generalization can be applied at the level of single column (i.e., a generalization step generalizes all the values in the column) or single cell (i.e., for a specific column, the table may contain values at different generalization lev-

SSN	Name	Race	Date of birth	Sex	ZIP	Marital status	Disease
		asian	74/04/	F	202**	divorced	Flu
		asian	74/04/	F	202**	married	chest pain
		asian	73/03/	M	202**	married	obesity
		asian	73/03/	M	202**	married	hypertension
		black	74/11/	F	202**	single	short breath
		black	74/11/	F	202**	single	obesity
		white	74/11/	F	202**	single	flu
		white	74/11/	F	202**	Widow	chest pain

Fig. 4. An example of a 2-anonymized table for the private table PT in Figure 2(a)

els). Suppression can be applied at the level of row (i.e., a suppression operation removes a whole tuple), column (i.e., a suppression operation obscures all the values of a column), or single cells (i.e., a k-anonymized table may wipe out only certain cells of a given tuple/attribute). The possible combinations of the different choices for generalization and suppression (including also the choice of not applying one of the two techniques) result in different k-anonymity proposals and different algorithms for k-anonymity.

Note that the algorithms for solving k-anonymity aim at finding a k-minimal table, that is, one that does not generalize (or suppress) more than it is needed to reach the threshold k. As an example, consider the microdata table in Figure 2(a) and suppose that the quasi-identifier is {`Race`, `Date of birth`, `Sex`, `ZIP`}.

Figure 4 illustrates an example of 2-anonymous table obtained by applying the algorithm described in [40], where generalization is applied at the column level and suppression is applied at the row level. Note that the first tuple in the original table has been suppressed and attribute `Date of birth` has been generalized by removing the day and attribute `ZIP` has been generalized by applying two generalization steps along the domain generalization hierarchy in Figure 3.

In [14] we defined a possible taxonomy for k-anonymity and discussed the main proposals existing in the literature for solving the k-anonymity problems. Basically, the algorithms for enforcing k-anonymity can be partitioned into three main classes: *exact*, *heuristic*, and *approximation* algorithms, respectively. While exact and heuristic algorithms produce k-anonymous tables by applying attribute generalization and tuple suppression and are exponential in the size of the quasi-identifier [7, 20, 48, 28, 40, 44, 23], approximation algorithms produce k-anonymous tables by applying cell suppression without generalization or cell generalization without suppression [2, 3, 37]. In these case, exact algorithms are not applicable because the computational time could be exponential in the number of tuples in the table.

Samarati [40] presented an algorithm that exploits a binary search on the domain generalization hierarchy to avoid an exhaustive visit of the whole generalization space. Since the k-anonymity definition is based on a quasi-identifier,

the algorithm works only on this set of attributes and on tables with more than k tuples (this last constraint being clearly a necessary condition for a table to satisfy k-anonymity). Bayardo and Agrawal [7] presented an optimal algorithm, called *k-Optimize*, that starts from a fully generalized table (with all tuples equal) and specializes the dataset in a minimal k-anonymous table, exploiting ad-hoc pruning techniques. LeFevre, DeWitt, and Ramakrishnan [28] described an algorithm that uses a bottom-up technique and a priori computation.

Iyengar [23] presented genetic heuristic algorithms and solves the k-anonymity problem using an incomplete stochastic search method. The method does not assure the quality of the solution proposed, but experimental results show the validity of the approach. Winkler [48] proposed a method based on simulated annealing for finding locally minimal solutions, which requires high computational time and does not assure the quality of the solution. Fung, Wang and Yu [20] presented a top-down heuristic to make a table to be released k-anonymous. The algorithm starts from the most general solution, and iteratively specializes some values of the current solution until the k-anonymity requirement is violated. Each step of specialization increases the information and decreases the anonymity.

Meyerson and Williams [37] presented an algorithm for k-anonymity, which guarantees a $O(k \log(k))$-approximation. Aggarwal et al. [2, 3] illustrated two approximation algorithms that guarantee a $O(k)$-approximation solution. Note that although both heuristics and approximation algorithms do not guarantee the minimality of their solution, and we cannot perform any evaluation on the result of a heuristic, an approximation algorithm guarantees near-optimum solutions.

k-anonymity is also currently the subject of many interesting studies. In particular, these studies aim at: studying efficient algorithms for k-anonymity enforcement; using k-anonymity as a measure on information disclosure due to a set of views [54]; extending its definition to protect the released data against attribute, in contrast to identity, disclosure (ℓ-diversity) [35]; supporting fine-grained application of generalization and suppression; and investigating additional techniques for k-anonymity enforcement [18].

4.2 Open issues

We now summarize the main open issues in developing a k-anonymity solution.

- *Extensions and enrichment of the definition.* k-anonymity captures only the defence against identity disclosure attacks, while remaining exposed to attribute disclosure attacks [40]. Some researchers have just started proposing extensions to k-anonymity [35] to capture also attribute disclosure, however research is still to be done.
- *Protection against utility measures.* As we can imagine the more the protection, the less precise or complete the data will be. Research is needed to develop measures to allow users to assess, besides the protection offered by the data, the utility of the released data. Clearly, utility may be different

depending on the data recipients and the use intended for the information. Approaches should be therefore devised that maximize information utility with respect to intended uses, while properly guaranteeing privacy
- *Efficient algorithms.* Computing a table that satisfies k-anonymity guaranteeing minimality (i.e., minimal information loss or, in other words, maximal utility) is an NP-hard problem and therefore computationally expensive. Efficient heuristic algorithms have been designed, but still research in needed to improve the performance. Indexing techniques could be exploited in this respect.
- *New techniques.* The original k-anonymity proposal assumed the use of generalization as suppression since, unlike others, they preserve truthfulness of the data. The k-anonymity property is however not tied to a specific technique and alternative techniques could be investigated.
- *Merging of different tables and views.* The original k-anonymity proposal as well as most subsequent work assume the existence of a single table to be released with the further constraints that the table contains at most one tuple for each respondents. Work is needed to release these two constraints. In particular, the problem of releasing different tables providing anonymity even in presence of join that can allow inferring new information needs to be investigated.
- *External knowledge.* k-anonymity assumes the data recipient has access to external database linking identities with quasi identifiers; it did not however model external knowledge that can be further exploited for inference and expose the data to identity or attribute disclosure. Work is needed to allow modeling external knowledge and taking it into account in the process of computing the table to be released.

5 Conclusions

This paper discussed aspect related to the protection of information in today's globally networked society. We outlined the needs for providing means to: combine security specifications to enforce security (and privacy) in the dynamic interactions among different, possibly unknown, parties; protect privacy in the dissemination of data; protect privacy of location information. For these contexts, we discussed ongoing researches and open challenges.

6 Acknowledgements

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

1. M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages*, 14(4):1–60, October 1992.

2. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. of the 10th International Conference on Database Theory (ICDT'05)*, pages 246–258, Edinburgh, Scotland, January 2005.
3. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, paper number 20051120001, 2005.
4. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An xpath based preference language for P3P. In *Proc. of the 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.
5. Gail-J. Ahn and J. Lam. Managing privacy preferences in federated identity management. In *Proc. of the ACM Workshop on Digital Identity Management (In conjuction with 12th ACM Conference on Computer and Communications Security)*, Fairfax, VA, USA, November 2005.
6. C.A. Ardagna, S. De Capitani di Vimercati, and P. Samarati. Enhancing user privacy through data handling policies. In *Proc. of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Sophia Antipolis, France, August 2006.
7. R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st International Conference on Data Engineering (ICDE'05)*, pages 217–228, Tokyo, Japan, April 2005.
8. D.E. Bell. Modeling the multipolicy machine. In *Proc. of the New Security Paradigm Workshop*, August 1994.
9. M. Blaze, J. Feigenbaum, J. Ioannidis, and A.D. Keromytis. *The KeyNote Trust Management System (Version 2)*, internet rfc 2704 edition, 1999.
10. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. of the 17th Symposium on Security and Privacy*, Oakland, California, USA, May 1996.
11. P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, February 2002.
12. P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.
13. M. Casassa Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *Proc. of the 14th International Workshop on Database and Expert Systems Applications*, Prague, Czech, September 2003.
14. V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-anonymity. In *Security in Decentralized Data Management*. Springer, 2006. (to appear).
15. V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. Microdata protection. In *Security in Decentralized Data Management*. Springer, 2006. (to appear).
16. E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati. Extending policy languages to the semantic web. In *Proc. of the International Conference on Web Engineering*, Munich, Germany, July 2004.
17. J. DeTreville. Binder, a logic-based security language. In *Proc. of the 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2002.
18. J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.
19. C.M. Ellison, B. Frantz, B. Lampson, R.L. Rivest, B.M. Thomas, and T. Ylonen. SPKI certificate theory. RFC2693, September 1999.

20. B. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005.
21. H. Hosmer. Metapolicies ii. In *Proc. of the 15th National Computer Security Conference*, 1992.
22. K. Irwin and T. Yu. Preventing attribute information leakage in automated trust negotiation. In *Proc. of the 12th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, November 2005.
23. V. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the Eigth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288, Edmonton, Alberta, Canada, 2002.
24. T. Jaeger. Access control in configurable systems. *Lecture Notes in Computer Science*, 1603:289–316, 2001.
25. S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):18–28, June 2001.
26. T. Jim. SD3: A trust management system with certified evaluation. In *Proc. of the 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001.
27. C. Landwehr. Formal models for computer security. *Computing Surveys*, 13(3):247–278, September 1981.
28. K. LeFevre, D.J. DeWitt., and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of the 24th ACM SIGMOD International Conference on Management of Data*, pages 49–60, Baltimore, Maryland, USA, June 2005.
29. N. Li, J. Feigenbaum, and B. Grosof. A logic-based knowledge representation for authorization with delegation. In *Proc. of the 12th IEEE Computer Security Foundations Workshop*, pages 162–174, July 1999.
30. N. Li, B. Grosof, and Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, February 2003.
31. N. Li and J. Mitchell. Datalog with constraints: A foundation for trust-management languages. In *Proc. of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003)*, New Orleans, LA, USA, January 2003.
32. N. Li, J.C. Mitchell, and W.H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management. *Journal of the ACM*, 52(3):474–514, May 2005.
33. N. Li, W.H. Winsborough, and J.C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.
34. P. Liu, P. Mitra, C. Pan, and V. Atluri. Privacy-preserving semantic interoperation and access control of heterogeneous databases. In *ACM Symposium on InformAtion, Computer and Communications Security*, Taipei, Taiwan, March 2006.
35. A. Machanavajjhala, J. Gehrke, and D. Kifer. ℓ-diversity: Privacy beyond k-anonymity. In *Proc. of the International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, USA, April 2006.
36. J. McLean. The algebra of security. In *Proc. of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, USA, April 1988.
37. A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems*, pages 223–228, Paris, France, 2004.
38. J. Ni, N. Li, and W.H. Winsborough. Automated trust negotiation using cryptographic credentials. In *Proc. of the 12th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, November 2005.

39. T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K.E. Seamons. Adaptive trust negotiation and access control. In *Proc. of the 10th ACM Symposium on Access Control Models and Technologies*, Stockholm, Sweden, June 2005.
40. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, November 2001.
41. K. E. Seamons, W. Winsborough, and M. Winslett. Internet credential acceptance policies. In *Proc. of the Workshop on Logic Programming for Internet Applications*, Leuven, Belgium, July 1997.
42. K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Monterey, CA, June 2002.
43. V.S. Subrahmanian, S. Adali, A. Brink, J.J. Lu, A. Rajput, T.J. Rogers, R. Ross, and C. Ward. *Hermes: heterogeneous reasoning and mediator system.* http://www.cs.umd.edu/projects/hermes.
44. L. Sweeney. Achieving k-anonynity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.
45. T.W. van der Horst, T. Sundelin, K.E. Seamons, and C.D. Knutson. Mobile trust negotiation: Authentication and authorization in dynamic mobile networks. In *Proc. of the Eighth IFIP Conference on Communications and Multimedia Security*, Lake Windermere, England, September 2004.
46. L. Wang, D. Wijesekera, and S. Jajodia. A logic-based framework for attribute based access control. In *Proc. of the 2004 ACM Workshop on Formal Methods in Security Engineering*, Washington DC, USA, October 2004.
47. D. Wijesekera and S. Jajodia. A propositional policy algebra for access control. *ACM Transactions on Information and System Security*, 6(2):286–325, May 2003.
48. W.E. Winkler. Masking and re-identification methods for public-use microdata: Overview and research problems. In J. Domingo-Ferrer, editor, *Privacy in Statistical Databases 2004*. Springer, New York, 2004.
49. W. Winsborough, K. E. Seamons, and V. Jones. Automated trust negotiation. In *Proc. of the DARPA Information Survivability Conf. & Exposition*, Hilton Head Island, SC, USA, January 25-27 2000.
50. M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring security and privacy for digital library transactions on the web: Client and server security policies. In *Proc. of the ADL '97 — Forum on Research and Tech. Advances in Digital Libraries*, Washington, DC, May 1997.
51. T.Y.C. Woo and S.S. Lam. Authorizations in distributed systems: A new approach. *Journal of Computer Security*, 2(2,3):107–136, 1993.
52. World Wide Web Consortium. *A P3P Preference Exchange Language 1.0 (APPEL1.0)*, April 2002. http://www.w3.org/TR/P3P-preferences/.
53. World Wide Web Consortium. *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*, July 2005. http://www.w3.org/TR/2005/WD-P3P11-20050701.
54. C. Yao, X.S. Wang, and S. Jajodia. Checking for k-anonymity violation by views. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB'05)*, Trondheim, Norway, August 2005.
55. T. Yu and M. Winslett. A unified scheme for resource protection in automated trust negotiation. In *Proc. of the IEEE Symposium on Security and Privacy*, Berkeley, California, May 2003.

56. T. Yu, M. Winslett, and K.E. Seamons. Prunes: An efficient and complete strategy for automated trust negotiation over the internet. In *Proc. of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.
57. T. Yu, M. Winslett, and K.E. Seamons. Interoperable strategies in automated trust negotiation. In *Proc. of the 8th ACM Conference on Computer and Communications Security*, Philadelphia, PA, USA, November 2001.
58. T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensitive policies trough interoperable strategies for automated trust. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, February 2003.