Traceable and Automatic Compliance of Privacy Policies in Federated Digital Identity Management

Anna Squicciarini, Abhilasha Bhargav-Spantzel, Alexei Czeskis, and Elisa Bertino

Computer Science Department, Purdue University
{squiccia,bhargav,aczeskis,bertino}@cs.purdue.edu

Abstract. Digital identity is defined as the digital representation of the information known about a specific individual or organization. An emerging approach for protecting identities of individuals while at the same time enhancing user convenience is to focus on inter-organization management of identity information. This is referred to as *federated identity management*. In this paper we develop an approach to support privacy controlled sharing of identity attributes and harmonization of privacy policies in federated environments. Policy harmonizations mechanisms make it possible to determine whether or not the transfer of identity attributes from one entity to another violate the privacy policies stated by the former. We also provide mechanisms for tracing the release of user's identity attributes within the federation. Such approach entails a form of accountability since an entity non-compliant with the users original privacy preferences can be identified. Finally, a comprehensive security analysis details security properties is also offered.

1 Introduction

Digital identity is defined as the digital representation of the information known about a specific individual or organization. As such, it encompasses not only login names (often referred to as *nyms*), but many additional pieces of information, referred to as *identity attributes* or *identifiers*, about users. Managing identity attributes raises a number of challenges. On one hand, these attributes often need to be shared among several parties in order to speed up and facilitate user authentication and access control, and thus enhancing usability of digital identities. On the other hand, identity attributes need to be protected because they may convey sensitive information that individuals may not be willing to share unless specific conditions are satisfied. An emerging approach for protecting identities of individuals while at the same time enhancing user convenience is to focus on inter-organization management of identity information. This is referred to as *federated identity management*. Specifically, the goal of a federated approach to digital identity management is to provide users with protected environments enabling identity attribute sharing. As such, federations provide a controlled method by which federated service providers (SP's) can provide more integrated and complete services to a qualified group of individuals within certain sets of business transactions. To date several on-going initiatives are developing standard protocols and platforms for the federated management of digital identities (see Table 5 in Appendix C for a summary of these initiatives).

Although federating identities greatly simplifies the task of collecting and distributing user attributes in the federation, no satisfying mechanisms are currently provided to protect users privacy and for privacy policy matching in collaborative environments. As SP's in a federation correspond to independent entities, they may adopt privacy practices that are not homogeneous. Uncontrolled identity information sharing may result in privacy breaches and threats like identity theft or phishing, and in the lack of compliance with respect to the privacy policies advertised by the various SP's.

A suitable solution to the problem of privacy in a federated environment should satisfy two important requirements. The first requirement is to provide mechanisms for facilitating privacy policies matching and harmonization among federated SP's. Such mechanisms would make it possible to determine whether or not the transfer of identity attributes from one SP to another would violate the privacy policies stated by the former. Notice that allowing a SP to transfer identity information to another SP is important in order to maximize user convenience and extend the notion of single-sign on to encompass a large variety of identity attributes. The second requirement is to provide mechanisms making it possible for users to trace their identity information across the federation, and verify whether it has been managed according to their privacy preferences. Privacy conscious users may in fact have their own preferences concerning the use of their identity attributes.

In this paper, we address these requirements by developing an approach that supports the privacy controlled sharing of identity attributes and the harmonization of privacy policies based on the notion of *subsumption*. Subsumption is used on policies defined over equal or similar class of data in order to determine if they are in conflict or if one implies the other. To facilitate policy harmonization in a federation, we assume some predefined policy templates to be available for policy specification. The SP's may either exploit the templates or may specify customized policies describing their own practices.

We base our approach on a rich privacy vocabulary rather than on the vocabulary provided by P3P[1]. We employ EPAL[2] vocabulary hierarchies to address the limited expressive power of the original P3P vocabulary. Moreover, we make use of an ontology to establish a common vocabulary for attributes, credentials, and data produced and exchanged across the federation. The use of an ontology makes it possible for the interacting parties to automatically detect semantic relationships among different attributes and reason about policy subsumption. To help users in verifying whether their privacy preferences have been enforced as required, we provide mechanisms for tracing the release of user's identity attributes. Our policy tracing is a method determining if such information has been transmitted from one SP to another along a path without violating the user's privacy preferences. We assume a tamper proof logging system to be in place. As a result, our protocol is also effective in the presence of malicious parties. In addition, if the policy tracing algorithm is executed a sufficient number of times, the user can check the enforcement of his/her privacy requirements over the whole federation

We cast our discussion in the context of the FAMTN (*Federated Attribute Management and Trust Negotiation*) system [3]. FAMTN is characterized by two types of entities: FAMTN SP's (FSP's for brevity) and users. A FSP is an entity providing a service to a user, if the user satisfies the policy requirements of the service. In addition, FSP's also manage and collect identity related information of federated users. As such a user will register at his/her own local FSP and then he/she will submit other identity attributes and credentials while interacting with FSP's to gain access to specific services or data. As no centralized identity provider exists, such information is not be stored at a unique server but is distributed among the various respective FSP's the user has visited. At the end of each interaction, the user obtains a receipt, referred to as *trust ticket*, that keeps track of relevant information concerning the interaction, like the purpose, the involved FSP and a time stamp. FSP's, besides interacting with users to provide them with services, also interact among each other in order to support the federated management of digital identities. FSP's, and more in general SP's in an IdM (Identity Management) system, exchange user attributes and credentials to automatically authorize users to access services and resources and so to avoid requiring multiple submissions of these attributes and credentials from users. It is important to notice that even though our approaches are cast in the context of FAMTN, they can be easily applied to other federated systems.

To the best of our knowledge, no previous approaches exist that provide privacy policy harmonization and tracing in federated environments for digital identity management. In the paper we also present a scenario motivating the development of the outlined techniques and we discuss how these techniques can be applied to specific domains characterized by a broad disclosure of sensitive information across federated domains.

The remainder of the paper is organized as follows. In the next section we introduce the motivating scenario that will be used throughout the paper. In Section 3 we provide preliminary concepts and definitions concerning ontologies and privacy policies. In Section 4 we illustrate the different mechanisms to policy specification and describe our algorithms for policy subsumption and tracing. In particular, Section 4.3 we illustrate the policy tracing algorithm and in Section 4.4 we present a detailed security analysis. In Sections 5 we discuss related work. Finally, we conclude the paper in Section 6 with pointers to future work.

2 Motivating Scenario

Health industry payers and providers maintain large volumes of confidential health information along with other sensitive personal and financial data and conduct many transactions electronically. In this arena an individual's digital identity includes his/her medical history, which is made up of (often disjoint) medical records from different health institutions. Privacy of medical records and medical-related identity information requires particular attention. To analyze the specific requirements and challenges of this environment we consider the example of an on-line federation of hospitals and organizations collaborating with each other, named *Trusted_Health*. We assume each federated organization being composed by service providers collecting users' information and interacting with users and other service providers through negotiations. In particular, we refer to the scenario of a user Alice who is a student of Purdue University.

We start from Alice getting an X-Ray performed at a city clinic called Lafayette-Health, part of the *Trusted_Health* federation. The resulting X-Ray report is stored with the privacy preferences of Alice at Lafayette-Health itself. Lafayette-Health collects medical records of its patients according to some privacy policies publicly available. Alice's report (along with her privacy preferences) is subsequently sent to her insurance company MedInsure, for filing her claim. *Trusted_Health* federation promotes privacy practices harmonization within the various institution by providing templates for possible policies describing different approach to data practices. Both Lafayette-Health and Medinsure specify policies using such templates. As such upon transmission of data between the two entities, MedInsure can easily verify whether its applied privacy policy is subsumed by the Lafayette-Health one. At a later date, Purdue Health Clinic requests the X-Ray information from Lafayette-Health for a routine check up and update of her health information. Purdue Health Clinic has all health related information of Alice. After three weeks Alice visits another university, State-U, and finds an X-Ray as a study sample in one of their biology classes. Even though the Alice's personal identifying information, such as name, SSN, and Purdue Identification Number, have been suppressed from the record provided with the X-Ray, such record still provides medical data, such as abnormalities seen in the x-rays, and supporting general data, such as gender, age, race, height, weight. Alice finds that this information perfectly fits her.

Therefore, how can Alice make sure that her privacy policy with respect to the X-Ray was not violated as this information was shared among the different institutions? Can Alice know which entity has managed her own data and according to which privacy practices?

3 Preliminary Notions

Our approach relies on the two important notions of ontology and privacy policies. In what follows we provide background information about these notions that is relevant for the subsequent discussion in the paper.

3.1 Ontologies

To properly apply and enforce privacy policies in a federation, interacting entities need to share a common vocabulary to facilitate communication and sharing of information. In particular, the meaning of a given attribute is to be understood in an unambiguous manner, so that other possibly related attributes are also automatically protected. In fact, same information can be often expressed through different attributes and be a generalization or a specialization of other attributes. With respect to our example of Section 2, the X-Ray report is the main data related to Alice in $Trusted\_Health$. This information may be referred to as *medical document* in the Insurance company MedInsure and *Bone Sample* in the biology department. Here the *medical document* may not have all the details of the original X-Ray report and might differ from the features of attribute used in the *Bone Sample*.

To model semantic relationships, we borrow ideas from work on ontologies [4,5,6]. In our work, we consider an ontology as a set of *concepts* together with relationships among these concepts. Specifically, the ontology assigns semantics to attributes, credentials and other identity related data, by defining two main classes. The first is the general class of identity related attributes, that are independent from any specific domain. Attributes like name, address and job position, fall in this class. The second class represents identity information that is specific to a given federated domain. In our scenario, this class includes information dealing with health state of an individual, his/her medical record information, blood type, diagnosis and so forth. For simplicity, we assume that the two class of information are disjoint, that is, there are no attributes which fall in both classes.

Each *concept* in the ontology is associated with a name, a set of keywords, a set of general purpose attributes names and a set of domain dependent attribute names. The formal definition is given below.

Definition 1. [Concept] *A concept, denoted by $\mathcal{C}_i$, is a tuple $\langle Name_i, KeywordSet_i, D\_Id\_Attr_i, Dom\_Attr_i \rangle$, where $Name_i$ is the concept name, $KeywordSet_i$ is a set of keywords associated with $\mathcal{C}_i$, $D\_Id\_Attributes_i$ is a set of credential type and/or attribute names and Dom_Attr_i is the set of domain related attribute names. $KeywordSet_i$ describes the set of all possible keywords used to describe concept $\mathcal{C}_i$. Each element in $KeywordSet_i$ is a synonymous of $Name_i$. Each attribute or credential type in $D\_Id\_Attr_i$ implements concept.* □

$\langle Xray, \{\}, xray, \{xray,\ medicaldocument,\ bonesample\}\rangle$ is an example of concept.

For any two distinct concepts $\mathcal{C}$ and $\mathcal{C}'$, where $\mathcal{C} = \langle Name_i, KeywordSet_i, D\_Id\_Attr_i, Dom\_Attr_i \rangle$ and $\mathcal{C}' = \langle Name_i, KeywordSet'_i, D\_Id\_Attr'_i, Dom\_Attr'_i \rangle$, the following conditions hold: $KeywordSet \cap KeywordSet' = emptyset$ and $D\_Id\_Attr_i \cap D\_Id\_Attr'_i = emptyset$. As such, any keyword belongs to exactly one concept. Similarly, we assume each attribute to be associated with exactly one concept.

An ontology is a partially ordered set of concepts $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$. The order relationship, denoted by $\prec$, represents a generalization relationship between concepts. $\mathcal{C}_i \prec \mathcal{C}_k$ if concept C_k is a generalization of concept C_i. This means that information conveyed by concept C_k can be used to infer information conveyed

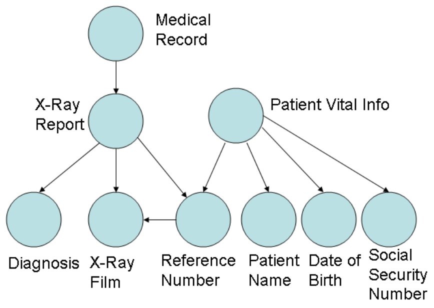

Fig. 1. Example of a Concept-Graph of an X-Ray Medical Report

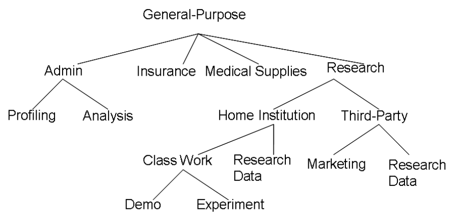

Fig. 2. Example of a purpose hierarchy for generic medical data

by concept C_i. For instance, the concept *Patient Vital Info* is a more general concept than *Patient Name* (denoted as *Patient Vital Info* $\prec$ *Patient Name*), since the Patient Vital Info imply the knowledge of his/her name. As an example, in Figure 1 we report graphical representation of concepts Medical Report, X-Ray, Diagnosis etc. In Figure, the parent node is a generalization of the child.

We assume the existence of an ontology which is shared and agreed upon by the various FSP's in a FAMTN system. Note that such an ontology is in most cases obtained through an integration process taking into account ontologies possibly existing at the various FSP's. A large number of integration techniques and methodologies have been developed for semantic ontologies [7]. For example matching techniques have been developed to determine semantic mappings between concepts of different related ontologies [5], that can be used in our context. Also, we assume the ontology to be stored for reference in a repository available to all the federated users. In what follows, we refer to the ontology shared in the federation as *federated ontology* in order to distinguish it from ontologies that are local to the various FSP's.

3.2 Privacy Policies

Privacy policies state who the *recipients* will be for the user *data*, the *purpose* for which this data will be used, and how long the data will be *retained*. Data in a privacy policy can be represented at different levels of granularity. They can refer to aggregate data, or they can refer to more specific piece of information, such as, last name or social security number. In our work, we adopt the terminology of the P3P standard [1]. The data element refers to smallest granularity data. Examples of data elements are social security number and last name. In our context, data elements actually correspond to ontological concepts. The current vocabulary adopted by the P3P standard is, however, not adequate for automatically and efficiently matching policies. We need to operate on a more articulated dictionary, using which we can compare and relate different values assigned to a same element of the policy. In particular, it is important to extend and define semantics relationships among elements in the *purpose* element and in the *recipient* elements. To achieve this goal, we consider the hierarchy developed for APPEL [8], a very well known language supporting the specification of privacy preferences by users. The referred data schema for the purpose is illustrated in Figure 2. In Appendix A we report details about the P3P syntax and the APPEL language.

4 Matching Privacy Policies in a Federation

FSP's can exchange user attributes and credentials to automatically authorize users without asking them to submit the same information multiple times. Further, in a medical environment, FSP's may need to access medical records to perform internal activities, such as evaluation of the health state of a patient or definition of patient eligibility to a given exam. We also notice here that sharing patient records may provide important benefits to the patient themselves, in that a physician may have available all information concerning a given patient and therefore perform a more informed diagnosis.

To enable secure information sharing across FSP's, we must assert that the privacy policies of all the FSP's that receive information pertaining to a given individual comply with the privacy preferences of this individual. In a system, like FAMTN, a compliance check can be executed between two FSP's when one FSP (referred to as FSP1) requests one or more user attributes from another FSP (referred to as FSP2). Instead of matching policies against the user preferences, FSP2 can more easily verify whether or not its policies subsume FSP1's. Subsumption reasoning is used on policies defined over equal or similar class of data in order to determine if they conflict.[1] To enhance flexibility and facilitate the task of policy specification of federated providers, we consider two different ways of specifying privacy policies: using *policy templates* or specifying *customized policies*. We assume a profile of policy templates to be pre-defined and available for privacy policy specification. We also assume privacy policy templates to be defined by the federated entities as preliminary agreement of the possible practices of the entities. We will further elaborate on this aspect in our future work.

A FSP may choose to use one of the available templates or can specify its own, customized privacy policies. Similarly, users can specify privacy requirements according to available specific pre-defined templates or they can specify their own requirements. The same FSP can specify policies using templates for some data and specify a customized privacy policy for other. Whether or not the enforced policies are instances of a template, a conservative approach is taken whereby if a request has even the slightest possibility of violating a privacy preference or policy due to some ambiguity, the request will be denied. Essentially, there are only two cases for each interaction: success or failure. A value like *Incompatible* or similar is never returned; instead only *Not Found* is returned. This generic

[1] Note that information about the availability of user attributes at FSP2 site is known because of the usage of trust tickets, as illustrated in Section 1.

reply is returned to avoid leak of information from FSP2 to FSP1 based on FSP1 reply (e.g., FSP2 learns that FSP1 has some data if an *Incompatible* reply is given).

4.1 Policy Templates

As introduced, FSP's can simplify the task of policy specification by using policy templates. Each template has a predefined set of values and is standardized across the federation. Each FSP can choose a template T_i[2] from the set $\{T_1, \dots, T_n\}$ of available templates. The templates in such set are totally ordered based on the strictness approach that will be followed for data disclosure. Specifically, templates are in descending order, then T_i defines practices that are stricter than those defined by policy template T_k, if $k > i$. In other words, T_k subsumes T_i. To simplify the process of policy specification, templates can be used to specify privacy practices for whole records, attributes or user credentials. In order for information to be released between two FSP's, the associated policies must be compatible. Here, by compatible policies we mean that if data is being released from FSP2 to FSP1, then privacy policy enforced by FSP1's policy should be equal or stricter than the policy applied by FSP2.

As suggested by [9], an example of set of policy templates ordered according the strictness is: {*Strict, Cautious, Moderate, Flexible, Casual* }. Adopting the notation adopted by [9] for the P3P syntax, we provide examples of such policy templates in Tables 1, 2, 3.

Table 1. Sample Strict Policy

Element	**Value**
Purpose	current
Access	all
Recipient	ours
Retention	stated-purpose, legal-requirement

Table 2. Sample Moderate Policy

Element	**Value**
Purpose	current, pseudo-analysis, contact
Access	all
Recipient	ours, same
Retention	stated-purpose, legal-requirement

The understanding of the policy illustrated in Table 1 is that data may be used only for the current activity and cannot be shared with others. Element *Recipient* is set to *ours*, meaning that the owner has full access to data and (as by *retention* element) data is kept only as long as the purpose requires or as mandated by law.

The policy template shown in Table 2 is a possible moderate policy and is to be interpreted as follows. The data it refers to may be used for this activity and can be shared with others having the same business practices. Statistical

[2] In what follows we refer to T_x as identifiers uniquely identifying the templates, while x denotes the position in the ordering.

records may be kept only with non-identifying information. The understanding of the *Access* element is that owner can be contacted with suggestions concerning treatments or drugs. As in the example of Table 1, owner has full access to data. Data is kept only as long as purpose requires or as mandated by law.

Table 3. Sample Casual Policy

Element	**Value**
Purpose	current, contact, other-purpose
Access	none
Recipient	ours, other-recipient, unrelated
Retention	indefinitely

Table 4. Policy supporting the privacy requirements described in Example 1

Element	**Value**
Purpose	current, pseudo-analysis
Access	all
Recipient	ours
Retention	stated-purpose, legal-requirement

In Table 3 a template for a casual policy is reported. The translation on such policy in natural language is as follows. Data may be used for virtually any activity, as stated by the *Purpose* element. Information may be shared with any unrelated entity irrespective of their policies. Owners can be contacted with suggestions concerning treatment or pharmaceutical. Owners may not be able to access or correct data. Finally, as reported by the *Retention* element, data may be kept indefinitely.

If both parties use pre-defined policy templates, policy comparison is straightforward: pre-defined policy templates are totally sorted based on the requirements that need to be met in order to release data. Policy subsumption reasoning is defined by Algorithm 1, encoding the protocol that performs local matching from the perspective of the *FSP1*, which is servicing a request for an attribute A from another service provider *FSP2*. Note that both parties are using policy templates totally sorted in descending order, thus T_k subsumes T_i if $k > i$. Assume that templates $\{T_k, T_i\}$ represent $\{Pol1, Pol2\}$ respectfully, then $isMoreStrict(Pol1, Pol2)$ at line 13 can be performed by checking if $k \leq i$.

It is important to note that the definition of policy templates is to be agreed upon by the federation members. When all entities in a federation use the policy template approach, it is simple to perform policy matching. However, policy templates inherently lack flexibility, and limit the range of preferences and intentions that users and FSP's can express, as illustrated by the following example.

Example 1. *Consider our motivating scenario. Alice might want to use a* Strict *policy as in Table 1, but might also want her data to be shared for statistical/research purposes as long as it cannot be linked to her. She is not able to use a* Cautious *or* Modest *policy because they are not strict enough. Therefore, she*

Algorithm 1. FSP1 services FSP2's request

```
Require: $Request$
 1: $Attr \Leftarrow Request.Attribute$
 2: $userID \Leftarrow Request.userID$
 3: $Policy \Leftarrow Request.getPolicyOf(Attr)$
 4: $myPolicy \Leftarrow this.userID.getPolicyOf(Attr)$
 5: if $Attr \notin this.userID.AttrList$ then
 6:     $this.log.Add(Request, notFound, time)$
 7:     return $notFound$
 8: end if
 9: if $isMoreStrict(Policy, myPolicy)$ then
10:     $this.log.Add(Request, tooStrict, time)$
11:     return $notFound$
12: end if
13: $this.log.Add(Request, released, time)$
14: return $this.userID.getAttribute(Attr)$
```

is not able to completely express her preferences because the privacy preferences are preset for each policy.

4.2 Customized Privacy Policies

Customized privacy policies are designed by FSP which can arbitrarily create a rules that describe how data will be managed. These policies give FSP's a flexible and expressive method for defining their privacy preferences and practices. However, customized policies are more difficult to specify, match and, typically, to enforce. Moreover, this flexibility increases the difficulty of policy matching. It is fair to assume that federation members may refer to similar terms with different names. For example, FSP1 and FSP2 may refer to the same group of people as *faculty* or *staff*. In order to determine the relationship between two different terms while performing local matching we make use of the federated ontology introduced in Section 3. It would indeed be misleading and error-prone to enforce a controlled vocabulary across a federation of disjoint entities without the help of the ontology.

The algorithm for performing local matching between customized FSP policies is identical to Algorithm 1. However, determining the relative policy strictness is a more articulated process. This is reflected by modifications to the *isMoreStrict()* function in order to use the ontology as in Algorithm 2.

An example of a possible customized policy is reported in Table 4, which solves the problem presented by Example 1. The policy states that data may be used only for this activity and cannot be shared with others. Statistical records may be kept only with non-identifying information. Data is kept only as long as purpose requires or according to the length mandated by law.

An explanation of Algorithm 2 follows. To evaluate the relationship between two given policies, $Pol1$ and $Pol2$, associated respectively with the requester

Algorithm 2. isMoreStrict(Pol1, Pol2)

```
Require: $Pol1, Pol2$ are objects
 1: //For all data elements of Pol1
 2: for all $E1 \mid E1 \in Pol1.dataElements$ do
 3:    //Get corresponding element from Pol2
 4:    $E2 \Leftarrow getElement(Pol2, E1.name)$;
 5:    if $E2 == NULL$ then
 6:       return $NO$
 7:    end if

 8:    //For all purposes of Pol1.E1
 9:    for all $P1 \mid P1 \in Pol1.getPurps(E1.name)$ do
10:       $P2 \Leftarrow getPurp(Pol2, E2.name, P1.name)$;
11:       if $P2 == NULL \parallel P1 \subseteq P2$ then
12:          return $NO$
13:       end if
14:    end for

15:    //For all retentions of Pol1.E1
16:    for all $Ret1 \mid Ret1 \in Pol1.getRets(E1.name)$ do
17:       $Ret2 \Leftarrow getRets(Pol2, E2.name, Ret1.name)$;
18:       if $P2 == NULL \parallel Ret1 \subseteq Ret2$ then
19:          return $NO$
20:       end if
21:    end for

22:    //For all recipients Pol1.E1
23:    for all $Rec1 \mid Rec1 \in Pol1.getRecs(E1.name)$ do
24:       $Rec2 \Leftarrow getRets(Pol2, E2.name, Rec1.name)$;
25:       if $Rec2 == NULL \parallel Rec1 \subseteq Rec2$ then
26:          return $NO$
27:       end if
28:    end for
29: end for
30: return $YES$
```

and the data holder, it is sufficient to analyze the purposes, recipients, and retentions for all data being requested from *Pol*1. Therefore, at line 2, every data element that is being requested by holder of *Pol*1 is evaluated, to determine whether the requester's intended use of the data element is subsumed by those in *Pol*2. Note that (at line 4) we exploit the federated ontology to determine an equal data element or if an equal one does not exist the closest generalization in *Pol*2. Next the algorithm proceeds by examining each purpose in *Pol*1 pertaining to this data element, checking if it is a subset of the purposes pertaining to the same data element from *Pol*2. The same comparisons are then performed for the retention and recipient conditions of the policies. As shown, comparison of purposes, recipients, and retentions are based on the semantic

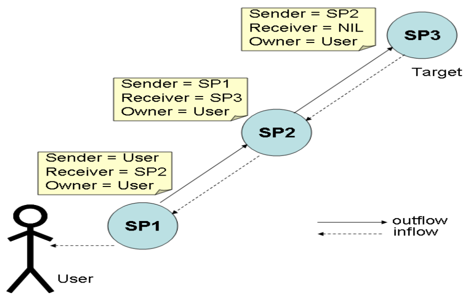

Fig. 3. Example of Policy Tracing execution

hierarchical nature of our vocabulary mentioned in Section 3 (proposed by [2]). Finally, if the purposes, retentions, and recipients of the requesting policy are all subsets of the servicing policy, a positive result is returned. Otherwise the result is negative.

4.3 Policy Tracing Algorithm

Policy tracing is a method for verifying if data have been transmitted from FSP_1 to FSP_k in a path that did not violate the user's privacy preferences. A trace can be initiated by a user who wishes to verify whether his/her privacy preferences have been properly applied as the data was passed to a target FSP (FSP_k in this case). Our solution for tracing is accomplished by a *Policy Tracing* algorithm, illustrated in Algorithm 3. Algorithm *Policy Tracing* is defined in terms of a series of message handlers which provide the necessary functionalities. Intuitively, the algorithm provides a tour of local match verifications, and policy compliance at each step starting from the sink peer FSP_k. The compliance depends on the transcript of the matching assumed to be stored in a tamper proof device at each FSP. Messages for the traces are propagated from the sink FSP to the user. Depending on the direction of the message, represented as an arrow in Figure 3, we define a prover FSP which is at the head of the arrow and a verifier which is at the tail. At each step, compliance is checked by the verifier, which retrieves the prover's logs. The GM_TRACEFAIL method ensures that if a point of failure is found the failure message returns to the target FSP. In case no point of failure is found, the trace message is propagated until it reaches the user. Behavior to signal success or failure could be implemented as a return to the calling algorithm.

To ease the presentation, we provide a detailed example illustrating the main steps of the algorithm (please refer to Figure 3). Alice finds her data at the FSP called SP3. As an initial step Alice checks if her privacy preferences matches the policies of SP3 using the local matching algorithm provided in Section 4.

The algorithm starts with the GM_START module which will spawns off a trace thread. The *inflow* thread goes from SP3 through SP2 and SP1 until it reaches Alice. For each thread instance the GM_TRACE function is called. SP3 first retrieves the sender of the data (which is also the head of the arrow representing the flow) from the log associated with this data. We assume that each such log contains the *sender, receiver* and the *owner* of the data. In the example, SP2 is the sender of the data contained at SP3 and Alice is the owner Now SP3 verifies that the transcript generated at the time of matching at SP2 is correct and does satisfy the privacy policy constraints. If this is true then GM_TRACE is called recursively until the message reaches the user. Otherwise, a GM_TRACEFAIL module is initiated by the verifier which recursively sends a failure message hop by hop to the target. GM_TRACEFAIL will reach the target because the entities in the GM_TRACEFAIL path have been verified to be honest.

Note that if executed multiple times from various sink FSP's a user can control the data related to him shared within the federation and match his/her privacy preferences with respect to all FSP's policies.

4.4 Security Analysis

The policy tracing algorithm is resistant to several types of attacks. Specifically, we address the cases of semi-honest, single malicious, and colluding malicious parties. By semi-honest, we mean parties that will follow the tracing protocol, but try to learn as much information as they can during the interaction. Malicious parties, in our case, correspond to parties that have not performed the local matching correctly and have released user data to parties whose rules for use of data violate the data owner's privacy requirements. Malicious parties may not follow the trace-back protocol and will attempt to circumvent being caught. Colluding malicious parties are multiple parties who can freely exchange user data meanwhile possibly violating the data owner's privacy requirements. One colluding party will never reveal information indicating policy violation by any other colluding party.

The following cases highlight the security features of our tracing algorithm:

Case 1. *All semi-honest parties.*

If all the parties are semi-honest, the initial local matching of the sink and user will always be compliant and the policy matching will be executed successfully.

This is because of the strict subsumption property of the local matching performed at each step when the data is released between any FSP's.

Case 2. *A single malicious party.*

A single malicious party in a trace is identified efficiently by a single execution of the tracing algorithm. The tracing begins at the information sink, that is, the FSP at which user found his/her data. We assume that the sink party is semi-honest (as such, it will follow the protocol)[3]. Let us assume FSP_{k-j}, $j < k$, be the malicious party and FSP_k be the sink FSP. As

[3] At this stage further release of the users data from the sink is not investigated. We trust the sink until proof of the contrary is found.

Algorithm 3. Policy Tracing Message Handlers

```
Require: TargetFSP T, DataObject D {sender, receiver, owner }
 1: GM_START:
 2: PostMessage(GM_START, inflow)
 3: if (T.containsData(D)) then
 4:    PassMessage(GM_TRACE (T,D,inflow), D.sender)
 5: else
 6:    PostMessage(GM_FAILURE)
 7: end if
 8: GM_TRACE(Target T, DataObject D, flowtype f):
 9: if (f == inflow and I am D.owner)  then
10:    PostMessage(GM_SUCCESS)
11: else
12:    if (f==inflow) then
13:       sender = D.sender
14:       if (localMatchVerify(sender, my ID, data) == TRUE) then
15:          PassMessage(GM_TRACE(T,D,f),sender)
16:       else
17:          PassMessage(GM_TRACEFAIL(T,D,my ID, outflow)
18:       end if
19:    end if
20:    if (f==outflow) then
21:       receiver = D.receiver
22:       if (localMatch(my ID, receiver, data) == TRUE) then
23:          PassMessage(GM_TRACE(T,D,f),receiver)
24:       else
25:          PassMessage(GM_TRACEFAIL(T,D,my ID, inflow)
26:       end if
27:    end if
28: end if

29: GM_TRACEFAIL(Target T, DataObject D, FailurePoint P, flowtype f):

30: if (f == inflow and I am D.owner) or (f == outflow and I am T ) then
31:    signal FALSE
32:    PostMessage(GM_FAILURE || at point P)
33: else
34:    if (f==inflow) then
35:       sender = D.sender
36:       PassMessage(GM_TRACEFAIL(T,D,P,f)
37:    end if
38:    if (f==outflow) then
39:       receiver = D.receiver
40:       PassMessage(GM_TRACEFAIL(T,D,P,f)
41:    end if
42: end if
```

the trace continues a recursive procedure is called such that FSP_k checks FSP_{k-1}, FSP_{k-1} checks FSP_{k-2}, and so on. This procedure continues until $\mathrm{FSP}_{k-(j-1)}$ is reached. Note that the trace has been executed by honest parties till this point. More precisely, each party's transcript is checked by an honest party before it is delegated the task of checking its parent. Then as $\mathrm{FSP}_{k-(j-1)}$ checks FSP_{k-j}'s transcript and finds an error, it will send a GM_TRACEFAIL message towards the sink such that the resulting error is notified at the sink.

Case 3. *Non-consecutive malicious parties.*
Once the first malicious party is found, as described above, there could still be other parties along the data path who are also dishonest. Again, assume FSP_{k-j}, $1 \leq j < k$, be the malicious party and FSP_k be the sink FSP. After FSP_{k-j} is identified as malicious (as in above example), FSP_{k-j-1} which is the FSP from whom FSP_{k-j}, received data is assigned as the next sink. In this case FSP_{k-j-1} is assumed to be trusted to not have released the data incorrectly. If FSP_{k-j-1}, is found non compliant with the users original policy then the only possibility is that other parties in the rest of the path are malicious. Hence, the tracing mechanism is called repeatedly in order to catch multiple malicious parties along the original trace of the data. In case FSP_{k-j-1} is compliant then there the release of data till this point has been executed correctly and no further trace is needed.

Case 4. *Consecutive or colluding malicious parties.*
The presence of colluding parties can be detected if the sink's policy is not subsumed by the user's policy, and after the trace no GM_TRACEFAIL message is propagated to the user. In this case we require the transcript verification run at each node by a trusted third party (TTP) in a brute force manner. This TTP can also be the sink or the user itself. TTP follows the same trace protocol with the only difference that it does the verifications. Once an error is found, the GM_TRACEFAIL message is propagated exactly the same as the original protocol.

5 Related Work

Our work is originally motivated from the existing initiatives related to federated digital identity management whose goal is to provide a controlled and protected environment for managing identities of federated users. In this section we first explore the most relevant federated digital identity management initiatives and then overview work related to privacy policy specification and enforcement.

In the corporate world there are several emerging standards for identity federation like Liberty Alliance [10] (LA) and WS-Federation. Because the projects are very similar we describe the former in more detail. LA is based on SAML and provides open standards for SSO with decentralized authentication. SSO allows a user to sign-on once at a Liberty-enabled site in order to be seamlessly signed-on when navigating to another site without the need to authenticate again. This group of Liberty-enabled sites is a part of what is called a *circle of trust*, which is a federation of SP's and identity providers having business relationships based on the Liberty architecture. The identity provider is a Liberty-enabled entity that creates, maintains and manages identity information of users and gives this information to the SP's. As compared to LA which has the identity provider as the only identity provisioning entity, our approach can protect sharing even when the provisioning is being done from the service providers that the user has visited. Such an approach provides privacy, flexibility and usability to the identity system. This is especially useful in the context of health data where the leakage of such information can have serious consequences.

Shibboleth [11] is an initiative by universities that are members of Internet2. The goal of such initiative is to develop and deploy new middleware technologies that can facilitate inter-institutional collaboration and access to digital contents. It uses the concept of federation of user attributes. When a user at an institution tries to use a resource at another, Shibboleth sends attributes about the user to the remote destination, rather than making the user log into that destination, thus enabling a seamless access. The receiver can check whether the attributes satisfy its own policies. Our approach differs with respect to Shibboleth in that we do not rely on a central identity provider providing all user attributes. User attributes in our framework are distributed within the different federation members, each of which can effectively be an identity provider. We also provide a mechanism for local and global matching which have not been well defined in federated identity management systems.

Regarding privacy, lots of researchers are actively working on privacy policy specification. The Platform for Privacy Preferences Project (P3P) is an attempt to provide a standardized, XML based policy specification language that can be used to specify an organizations privacy practices in a way that can be parsed and used by policy-checking agents on the users behalf [12].

Many user software agents are currently available for use (e.g. Privacy Bird2, Privacy Companion3, Internet Explorer 6.04), which handle policy checking and invoke the required actions that need to occur when a websites policy is found to conflict with the user's preferences. These actions range from blocking a particular webpage from being displayed, to placing a warning icon in the users browser status bar.

E-P3P[13] is a privacy policy language for expressing an enterprise-wide privacy policy. Its goals are different than P3P, in that it is geared towards internal policy enforcement and business practices, rather than expression of a policy to a user agent. As such, it supports enterprise-defined user roles, purposes, and arbitrary conditions and obligations that must be fulfilled. E-P3P expresses a privacy policy in abstract user role and data categories. The association of these with actual data and users or user groups in a system is outside the scope of E-P3P. E-P3P assumes an enterprise-wide policy, where users can opt-in or opt-out.

Finally, a work more closely related to ours is represented by IBMs Enterprise Privacy Authorization Language (EPAL) [2]. EPAL's authors propose an approach to achieve machine enforceable policies [14]. Like P3P, EPAL is an XML-based privacy policy specification language, specifically designed for organizations to specify internal privacy policies. EPAL policies can be used internally and amongst the organization and its business partners to ensure compliance different purposes and scopes, but to evaluate each languages expressiveness for specifying natural language privacy policies. In our work, we do not propose a new language. Rather, we focus on the deployment of protocols to facilitate privacy policy harmonization within federated entities. Further, differently from EPAL, we propose an approach to check users' privacy preferences compliance based on policy traceability.

6 Conclusion

In this paper we address the problem of privacy in a federated environment. In particular, we attempt to satisfy two important requirements. The first requirement is to provide mechanisms for facilitating privacy policies matching and harmonization among federated SP's. The second requirement is to provide mechanisms making it possible for users to trace their identity information across the federation, and verify whether it has been managed according to their privacy preferences.

To achieve such goals we have developed an approach that supports the privacy controlled sharing of identity attributes and the harmonization of privacy policies based on the notion of subsumption. This approach relies on the P3P language and federated ontology. Two well defined ways of specifying privacy policies have been proposed, that is, by use of pre-defined policy templates or by defining customized policies. We have also devised two main protocols to provide harmonization of the privacy policies at the local and global levels respectively. Our approach entails a form of accountability since an entity non-compliant with the users original privacy preferences can be identified. A comprehensive security analysis details security properties offered by our approach.

An interesting challenge that must be addressed to achieve effective privacy protection is maintaining data management consistency, as privacy practices and preferences might change over time. In the current work we do not take into account update of such policies after the data has been released. We will address such an issue in our future work.

In addition, we plan to extend this work along several other directions. The first concerns the conservative approach we took while determining the subsumption criteria for the local matching to avoid inference. We will further explore other inference problems to allow flexible subsumption criteria. Second, in our current work we assume privacy policy templates to be defined by the federated entities as preliminary agreement of the possible practices of the entities. We plan to define reasonable templates and their extension for customized policies. Third we are developing more articulated conflict resolution techniques in the tracing algorithm, taking into account the exact mismatch that occurred due to which the policies were non-compliant. We believe this will provide a mechanism to extend the search of multiple malicious and colluding parties such that all non-compliant entities are held accountable.

Acknowledgement

The work reported in this paper has been sponsored by NSF under the ITR Project 0428554 "The Design and Use of Digital Identities". Intel has provided the fellowship to fund Alexei Czeskis.

References

1. http://www.w3.org/TR/P3P/: (The Platform for Privacy Preferences 1.0 (P3P1.1) specification)
2. http://www.zurich.ibm.com/security/enterprise privacy/epal/: (EPAL 1.0 Specification)
3. Spantzel, A.B., Squicciarini, A.C., Bertino, E.: Integrating federated digital identity management and trust negotiation. In: Review for the IEEE Security and Privacy Magazine. (2005)
4. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2) (1993) 199–220
5. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology Matching: A Machine Learning Approach (2003)
6. Uschold, M., Gruninger, M.: Ontologies: Principles, Methods, and Applications. Knowledge Engineering Review 11(2) (1996) 93–155
7. Maedche, A., Motik, B., NunoSilva, Volz, R.: MAFRA – a MApping FRAmework for distributed ontologies. Lecture Notes in Computer Science **2473** (2002) 235–241
8. http://www.w3.org/TR/P3P preferences/: (P3P Preference Exchange Language 1.0 (APPEL1.0))
9. Alliance, L.: Liberty architecture framework for supporting privacy preference expression languages (ppel's) (2003)
10. http://www.projectliberty.org: (Liberty Alliance Project)
11. http://shibboleth.internet2.edu: (Shibboleth, Internet2)
12. Cranor, L.F.: P3P: Making privacy policies more useful. Volume 1. (2003) 50–55
13. Paul Ashley, Satoshi Hada, G.K., Schunter, M.: E-P3P Privacy Policies and Privacy Authorization. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES). (2001)
14. Stufflebeam, W.H., Antón, A.I., He, Q., Jain, N.: Specifying privacy policies with P3P and EPAL: lessons learned. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES). (2004) 35
15. http://www.switch.ch/aai/documents.html: (Switchaai Federation)
16. http://www.incommonfederation.org/: (InCommon Federation)
17. http://www.csc.fi/suomi/funet/middleware/: (HAKA Federation Finland Federation)

18. Overhage, S., Thomas, P.: Ws-specification: Specifying web services using uddi improvements. In: Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, London, UK, Springer-Verlag (2003) 100–119

A P3P Policy Language

A P3P privacy policy is specified by one policy element that includes the following major elements: *entity, access, extension*, and *statement.* The *entity* element identifies the legal entity making the representation of privacy practices contained in the policy. The access element indicates whether the site allows users to access the various kind of information collected about them. The *extension* is an optional element describing a website's self defined extension to the P3P specification. One or more *statement* elements are defined in a policy. A statement is the core of the policy as it defines the data and the data categories collected by the site, as well as the purposes, recipients and retention of that data. Each statement contain the following:

– *data* denotes a data element. In P3P each DATA element has a set of categories associated with it. Some categories are implicitly specified by the base P3P data schema whereas some others are defined by the policy itself;
– *purp* denotes purposes for data processing; *purpose* element assumes on or more pre-defined value in {*current*, *admin*, *tailoring*, *pseudoanalysis*}.
– *retention* denotes the type of retention and assumes values in {*no-retention, stated-purpose, legal-requirement, business-practice, indefinitely*} according to the P3P standard taxonomy;
– *recipient* is the legal entity, or domain, beyond the service provider and its agents where data may be distributed; *recipient* can assume one value in {*ours*, *legal*, *delivery*, *unrelated*, ...};

Example 1. Consider the following P3P policy:

```
<STATEMENT>
  <PURPOSE>
   <individual-decision required=
    "opt-out"/> </PURPOSE>
  <RECIPIENT><ours/></RECIPIENT>
  <RETENTION><stated-purpose/>
  </RETENTION>
  <DATA-GROUP>
    <DATA ref="#user.name.given"/>
    <DATA ref="#dynamic.cookies">
      <CATEGORIES><preference/>
       <uniqueid/>
      </CATEGORIES>
    </DATA>
  </DATA-GROUP>
</STATEMENT>
```

Since P3P has not been specifically conceived for negotiations within federation, its syntax include data elements that are not of interest to trust negotiations, such as *click-stream*. In the following, we always limit our analysis to elements having a corresponding concept in our reference ontology. Such data elements can then be used to evaluate privacy concepts.

B APPEL Preference Language

With respect to APPEL(ACCENT Project Policy Environment/Language) the privacy preferences are expressed in as a list of RULEs [8]. These rules are matched against a policy in the order in which they appear. A rule consists of two parts:

– Rule behavior (Rule head): Specifies the action to be taken if the rule fires. The behavior can be request, implying that the policy conforms to preferences specified in the rule body. It can be block, implying that the policy does not respect user's preferences.
– Rule body: Provides the pattern that is matched against a policy. The format of a pattern follows the XML structure used in specifying privacy policies described earlier.

An APPEL rule is satisfied by matching its constituent expressions and recursively their subexpressions. Every APPEL expression has a connective attribute that defines the logical operators between its subexpressions. An example of an APPEL policy is as follows:

```
<appel:RULESET>
 <appel:RULE behavior="block">
   <POLICY>
     <STATEMENT>
      <PURPOSE appel:connective="or">
      <contact/>
      <telemarketing/>
      </PURPOSE>
     </STATEMENT>
   </POLICY>
 </appel:RULE>

<appel:RULE behavior="request"/>
  <appel:OTHERWISE/>
  </appel:RULE>
</appel:RULESET>
```

C Federation Examples

Table 5. Federation Examples

SWITCHaai Federation [15]	The SWITCHaai Federation is a group of organizations like universities, hospitals and libraries, that have agreed to cooperate regarding inter-organizational authentication and authorization. They operate a Shibboleth-based authentication and authorization infrastructure (AAI).
InCommon [16]	By using Shibboleth authentication and authorization technology, InCommon intends to make sharing of protected resources easier, enabling collaboration between InCommon participants which protects privacy. Access decisions to protected resources are based on user attributes contributed by the user's home institution. InCommon became operational on 5 April 2005.
HAKA Federation [17]	The HAKA Federation in Finland entered its production phase in late 2004. The Federation was set up in 2003, currently including 2 (of 20) universities and 1 (of 29) polytechnics as Identity Providers, and 4 service providers, including the National Library Portal (Nelli). In Finland, the libraries in higher education traditionally co-operate widely in licensing electronic journals. It is based on Shibboleth.
Microsoft, IBM, WS* [18]	In April 2002, Microsoft and IBM published a joint whitepaper outlining a roadmap for developing a set of Web service security specifications. Their first jointly-developed specification, WS-Security, offers a mechanism for attaching security tokens to messages, including tokens related to identity.
Liberty Alliance [10]	The Liberty Alliance is a consortium of approximately 170 companies that develops specifications for federated identity management. It works on creating a single comprehensive federated identity specification. In March 2003, it released a new blueprint that described three separate specifications that can be used together or independently: First is the Identity Federation Framework (ID-FF) allows single sign-on and account linking between partners with established trust relationships. Second is Identity Web Services Framework (ID-WSF), allows groups of trusted partners to link to other groups, and gives users control over how their information is shared. Finally Identity Services Interface Specifications (ID-SIS) will build a set of interoperable services on top of the ID-WSF.