Automatic Compliance of Privacy Policies in Federated Digital Identity Management

Anna Squicciarini, Marco Casassa Mont, Abhilasha Bhargav-Spantzel, Elisa Bertino
HP Laboratories Bristol
HPL-2008-8
Feburary 1, 2008*

Privacy, Privacy policy, Federated Identity Management

Privacy in the digital world is an important problem which is becoming even more pressing as new collaborative applications are developed. The lack of privacy preserving mechanisms is particularly problematic in federated identity management contexts. In such a context, users can seamlessly interact with a variety of federated web services, through the use of single-sign-on mechanisms and the capability of sharing personal data among these web services. Because of the latter feature, user's privacy is at a stake, if the sharing of such data among federated service providers is not properly controlled to ensure that privacy is preserved and user's privacy preferences are complied with. Current federated identity managed solutions adopt simplistic approaches to privacy management, based on contractual/legal approaches and/or limited simple checks on users' privacy preferences. We argue that more comprehensive privacy policies (consisting of access control and obligation constraints, along with privacy preferences) should be stated by federated service providers and proactively checked by these providers, before disclosing users' data to federated partners. To address such requirements, we introduce mechanisms and algorithms for policy compliance checking between federated service providers, based on an innovative policy subsumption approach. We formally introduce and analyze our approach. We also show how our approach is suitable for deployment and application in existing federated identity management solutions, such as Liberty Alliance, WS-* and Shibboleth.

Internal Accession Date Only Approved for External Publication

Submitted to IEEE Policy 2008, 2-4 June 2008, Palisades, NY, USA

Automatic Compliance of Privacy Policies in Federated Digital Identity Management

Anna Squicciarini[1], Marco Casassa Mont[2], Abhilasha Bhargav-Spantzel[3], Elisa Bertino[3]

[1]Information Sciences Technology, The Pennsylvania State University
[2]Hewlett-Packard Laboratory, Trusted System Lab
[3]Computer Science Department, Purdue University

Abstract

Privacy [13] in the digital world is an important problem which is becoming even more pressing as new collaborative applications are developed. The lack of privacy preserving mechanisms is particularly problematic in federated identity management contexts. In such a context, users can seamlessly interact with a variety of federated web services, through the use of single-sign-on mechanisms and the capability of sharing personal data among these web services. Because of the latter feature, user's privacy is at a stake, if the sharing of such data among federated service providers is not properly controlled to ensure that privacy is preserved and user's privacy preferences are complied with. Current federated identity managed solutions adopt simplistic approaches to privacy management, based on contractual/legal approaches and/or limited simple checks on users' privacy preferences. We argue that more comprehensive privacy policies (consisting of access control and obligation constraints, along with privacy preferences) should be stated by federated service providers and proactively checked by these providers, before disclosing users' data to federated partners. To address such requirements, we introduce mechanisms and algorithms for policy compliance checking between federated service providers, based on an innovative policy subsumption approach. We formally introduce and analyze our approach. We also show how our approach is suitable for deployment and application in existing federated identity management solutions, such as Liberty Alliance, WS- and Shibboleth.*

1 Introduction

Privacy in the digital world is an important problem that needs to be properly addressed. The lack of privacy preserving mechanisms is particularly problematic in federated identity management contexts [12], in which different parties manage, share and use personally identifiable information (PII) of individuals. The availability of such data is crucial to provide individuals with convenient and personalized services. However, if sharing of such data among the federation parties is not properly controlled, privacy breaches could easily occur.

Research to date has focused on developing privacy languages for open distributed systems, to enable the specification of privacy preferences by individuals and of privacy practices by companies and organizations that collect PII. However, little work has been carried out to address privacy issues in environments like federations, social networks, and virtual communities. Here, parties are known to each other and believe, often based on a wrong understanding, that data will be shared in a privacy conscious manner. Interactions are based on trust and/or on contractual agreements. No automated tool is typically in place to check if the preferences that a user has expressed are actually consistent with the policies by other parties in the federation that eventually receive this data.

In this paper we propose an approach for compliance checking of agreed privacy policies and preferences in a federated identity management context. Such context is an important example for other scenarios, such as social networks, which require PII sharing among parties. Our goal is to achieve a privacy preserving methodology, by which PII can be shared across parties by fully complying with the users' personal preferences.

More precisely, our targeted scenario involves a federation for digital identity management (FIM, from now on), composed by organizations or enterprises, each identified by a federated service provider (shortened as FSP).

FSPs choose to be part of a FIM and thus it is in their own interest to comply with privacy policies in order to take advantage of the benefits offered by the membership. FSPs in fact gain in credibility by keeping what is promised by their privacy policies and by cooperating with the federation. They also obtain accurate users' information in a timely manner which can, for instance, help them in refining business strategies. Users also have several advantages in joining a privacy-complying FIM. They can safely disclose personal information and specify their own privacy preferences without having

to resubmit them multiple times and to check FSPs' privacy policies at each PII disclosure.

We specifically focus on the challenging case of a FIM system that purely relies on the FSPs and and does not require a centralized identity provider. Such type of federation is the de-facto approach used by many service providers to share PII, based on user's interactions and business needs. However, our suggested approach can also apply to more conventional FIM scenarios involving both FSPs and identity providers.

Users are affiliated with one -or more- FSP of the federation to which they submit identity attributes and personal information at the time of registration. Users also disclose other identity attributes and credentials while interacting with FSP's to gain access to specific services or other resources. As a consequence, identity information is distributed among the various FSP's the user has visited. Additionally, users specify their preferences of how they want their data to be handled-from whom, for how long, and so forth.

FSP's besides interacting with users to provide them with services, interact among each other in order to exchange users' identity attributes and credentials. Federation of identities enables FSPs to automatically authorize users to access services (via single-sign-on, SSO [16]) and resources. By obtaining users' data from other members of the FIM, FSPs avoid requiring multiple submissions of attributes and credentials from users that could lead to inconsistency and data integrity violations-if different versions of the same data were to be submitted-.

The key problem addressed by our work is how to ensure that PII is exchanged between two or more parties only if agreed privacy policies and preferences are satisfied. Privacy policies and user's preferences thus need to be cross-checked between parties that exchange PII. To this extent, we introduce the concept of *policy subsumption*. Subsumption is the core of automated exchange of federated identities, in that it allows each FSP to automate the process of checking (privacy) policy compliance when transferring data to other FSPs. Such compliance checking assures that data is disclosed and managed consistently with the privacy policies, obligations and preferences initially agreed by the user. In fact, we propose an approach that allows specific users privacy preferences to be directly matched against service providers' policies, without requiring direct intervention by the users. Subsumption also improves FSPs autonomy, in that they are able to autonomously specify privacy preferences and practices, and verify their compliance with other FSPs on-the-fly, as needed.

The specific contributions of this paper are summarized as follows:

- An integrated notion of privacy policy that includes parametric obligations [11] and users' preferences.
- An approach to verify privacy policies and obligations subsumption.
- Methods that allows users to control how and according to which privacy policies PII is used and transferred among FSPs.

The remainder of the paper is organized as follows. We first present a summary of the main concepts related to ontologies and semantic representation of data. Next, we introduce our extended syntax for the logical representation of privacy policies and obligations. In Section 4 we discuss our approach to privacy policy subsumption. We then present the architectural components for integration of policy subsumption in existing systems for digital identity management. We discuss related work in Section 6. We conclude the paper in Section 7.

2 Background on Ontologies and Semantic Relationships among Data

To enforce privacy policies within a federation, parties typically share a common vocabulary. The approach we adopt in our work for expressing such vocabulary is based on ontologies [4, 3, 18]. We assume that a domain independent ontology exists that includes concepts for obligations, actions, privacy terminology, and so forth. In addition, we assume that domain dependent ontologies are also shared among FSPs, to define domain-dependent classes and properties. In what follows, we thus refer to a single ontology that includes both domain-dependent and domain-independent ontologies. We assume that such reference ontology consists of a pair $\{\mathcal{D}, P\}$, where $\mathcal{D}$ is a set of data variable names, atomic or compound, modelling users' PII, whereas P is a set of relationships connecting data items in $\mathcal{D}$. Example of variable names are "social-security number, last name " etc.. An atomic data element is an element which does not have any sub-element in the considered domain. In contrast, a compound data element is composed of two or more data elements, atomic or compound. Examples of atomic data elements are social security number and last name. Compound data elements are, for instance, medical report or X-ray report. As a convention, given a set $\{d\}$ subset of $\mathcal{D}$, we add a caption to denote specific instances of $\{d\}$. For example, we denote with $\{d\}_{val}$ an instance of data values for $\{d\}$.

To evaluate and compare data elements, we rely on the well-known relation is_part_of. This relation is modeled as the predicate *is_part_of(d, d')*, and reads as *elements in d' are part of d*. We represent this relation by the operator $\prec$ and write $\prec^* d$ to denote d's transitive closure.

We now clarify such notion with an example.

Example 1 *Consider Figure 1 reporting a graph representation of Medical Report, X-Ray, Diagnosis etc. In the graph representation -discussed in details in section 3.1- nodes are data elements and edges denote is_part_of relations. The fact*

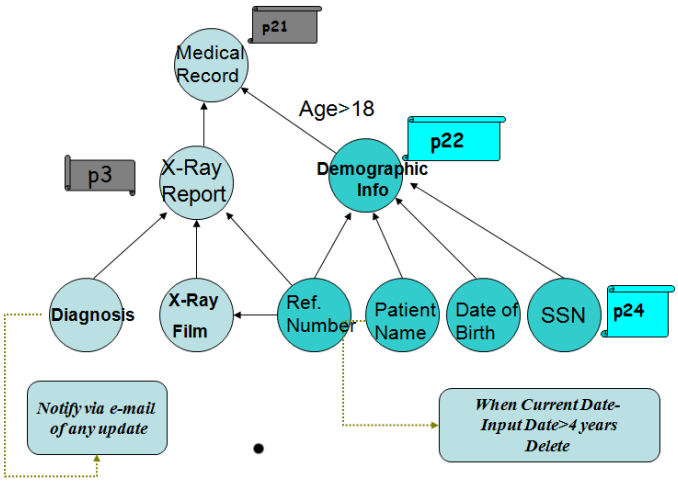

Figure 1. Example of a simplified data graph.

that the X-Ray film is part of the X-Ray Medical Report is represented by an edge from the node representing the former to the node representing the latter.

The following relations hold: Patient Demographic Info $\prec$ *Date of Birth and* $\prec *$*Patient Demographic Info*=$\{$*Ref. Number, Patient Name, Date of Birth, SSN*$\}$.

According to the above introduced notation, given $\{d\} = \{d_1, \ldots, d_k\}$, we denote as $\{d\}_{exp}$ the expansion of $\{d\}$, obtained by the set union of the transitive closure of all data items in $\{d\}$. Precisely, $\{d\}_{exp} = \prec^* d_1 \bigcup \ldots \bigcup \prec^* d_k$.

3 An Extended Specification Language for Privacy Privacy Policies in Federated Environments

In this section we provide our extended notions of privacy policies and obligations. We also introduce the notion of *data graph*, that is, a data structure representing relationships on data items, along with associated privacy policies and obligations.

3.1 Privacy policy components

Privacy policies define rules for accessing and using PII related to individuals, disclosed by these individuals. In order to reason about privacy policies - without having to deal with various specific syntaxes, we use a logical representation of privacy policies which is abstract and independent from any existing lower level language. Our aim is to explicitly represent privacy concepts -such as data handling constraints and obligations- along with more traditional access control constraints, at a logical level.

A privacy policy is composed of three main components, discussed in this section: user's *privacy preferences*, *data handling* rules and *obligations*.

Privacy preferences, denoted as *PPref*, express individuals' privacy preferences (preferences, for short) on how their data must be handled. For our purposes, we abstract from the actual encoding, and express preferences using terms from a predefined set, denoted as *PT*, of privacy terms. Examples of such terms are "preferred deletion date" and "notification preference". Each term is associated with a set of possible values. Each preference is defined as a pair $< pt, val >$ where *pt* is an element of *PT* and *val* is a value from the domain of pt. Each PII item has associated with one or more preferences. As discussed later on in the paper, such values will be used to customize policies conditions and obligation expressions. In case a user has no preferences, default preferences will apply based on common practices, organisations guidelines and/or common sense (by keeping into account common privacy practices).

Examples of privacy preferences are related to e-mail notifications to users (e.g. when a FSP accesses and/or discloses their PII), deletion dates of PII, lists of entities PII data should/should not be disclosed to, etc. For example a user, when disclosing his/her data (or afterwards, when self-managing it, at a service provider's web site) can define their preferences in terms of notifications (e.g. by opting-in) and deletion of his/her PII.

Data handling rules specify how users' PII will be handled by the FSP, for what reason and under which conditions. Specifically, we represent a data handling rule (or DataHand, for short) as a tuple of the following form:

$$DataHand = \langle purpose; recipient; access; cond \rangle \quad (1)$$

The *purpose* component specifies the allowed usage of the data and it can convey a number of different purposes, while the *recipient* constraints *who* is allowed to access the data; the *access* component specifies which are the access modes. *Purpose*, *recipient* and *access* components are specified as list of values. We do not mandate a specific set of acceptable values, as these depend on the specific language used to which the final policies will be encoded to and of the domain and business processes which the policies refer. The *cond* component specifies a set of conditions for the application of the policy. The conditions can refer to users' data values, recipient's attributes or contextual conditions and may include parameters from the *PPref* expression. Specifically, we support *recipient conditions* and *user conditions*. Recipient conditions dictate conditions that a recipient may have to satisfy. Such conditions may refer to technical capabilities of the receiving entity (for instance, the data storage modes, or the cryptographic protocols used for encode the data) or they could specify conditions on the recipient properties (e.g. "only American doctors can be recipients of the X-Ray taken in the US-Health clinic"). Users' conditions instead refer to conditions against data. For example, one may specify that

the rule applies only if the user lives in a certain country or has a given age.

For clarity, we assume conditions to be in a normalized form, that is, that they are expressed in conjunctive form. We also assume that all conditions in the set must be taken into account when enforcing the data handling rule.

We omit from our analysis other typical components of a privacy policy, such as the remedies and disputes (that are for instance mandatory in P3P[14]), since they are not significant in our framework. We in fact focus on upfront policy compliance/checking, and not on how to handle operational and violation matters.

Example 2 *The data handling rule ⟨{marketing, statistical-analysis}; HP; Read; Time < 13/12/2007 ⟩ says that data can be accessed by HP for marketing and for statistical analysis. The condition is an example of user condition, and it says that the data cannot be modified before the 13th of December 2007.*

As we mentioned, an important feature of the introduced representation is that data handling rules might include additional user's privacy preferences. For example, users' data may include user's consent to be notified when new marketing material is available. In this case the condition component of DataHand is parametric and is used to check if the action in DataHand is allowed, based on a specific choice (preference) made by the user. Both recipient and users conditions can be parametric. An example of parametric condition is provided in Example 4.

Obligations are the other main component of privacy policies. Their aim is to define expectations and duties on how to handle PII, also based on privacy preferences of users. Here, we discuss obligations from a logical point of view, under the assumption that obligations be mappable to meaningful operational obligations. Obligations' components are modelled by the following tuple:

$$Obl = \langle events, actions, corrective - actions \rangle \quad (2)$$

The *events* component defines a logical (AND) conjunction of events that need to happen in order to trigger the actions part[1]. For example, a simple event could be a time-based condition (e.g .CurrentTime $>=$ timex). The *actions* component defines which tasks must be executed in order to fulfil the obligation. This might include sending a notification to the data subject, deleting (part of) data, executing applications. Actions as well as events must keep into account privacy preferences *PPref*, as illustrated in Example 4. The *corrective-actions* component defines which tasks must be executed when an obligation is violated. There might be no corrective actions or these could be the same actions executed for the obligation enforcement or the corrective action might just consist of sending a notification to the administrator. Also corrective actions must keep into account privacy preferences.

The semantics of such tuple is given by a reactive rule of the form: WHEN *events* THEN *actions:* ON-VIOLATION: *corrective-actions.* Notice that obligations are parametric [11], in that they can be instantiated using users' preferred values (indicated in the privacy preferences).

Example 3 *The following is a simple example of obligation, parametric with respect to the event. The target is PII items referred by the policy which the obligation belongs. It states that if the FSP accesses user's data and the user opted in for being notified, the user should be notified: ⟨ {ACCESS(UserData); Notify-User(opted-in)}; Notify_User; notify_Admin⟩. The trigger-like representation of such rule is as follows:*

WHEN user (ACCESS(UserData) AND Notify-User(opted-in)) THEN Notify_User ON-VIOLATION (Notify_Admin).

3.2 Privacy policies

Having introduced the building blocks of privacy policies we can now present our integrated representation of privacy policies. As described in the following definition, privacy policies can be instantiated or generic, depending on whether they refer to a specific set of data values and users' preferences or if they are defined in terms of a set of data variable names.

Definition 3.1 [Privacy Policy tuple] A privacy policy **PPol** is defined as a tuple of the form $\langle \{d\}_{spec}, \{$ *DataHand; Obl; PPref* $\} \rangle$, where:

- $\{d\}_{spec}$ denotes the data specification of PII items, and it can be either defined in terms of a set of data variable names[2] $\{d\}$ or a set of data values, $\{d\}_{val}$.
- *DataHand* specifies how to handle $\{d\}_{spec}$; it is specified according to expression 1 (Section 3.1), possibly including preference values from *PPref*.
- *Obl* represents the unique identifier of the obligation which the FSP is subjected to with target data $\{d\}_{spec}$.
- *PPref* denotes user's privacy preferences; it is a not mandatory element.

From now on, we will refer to single policy components using the dot notation.

Instantiated privacy policies are the result of a successful policy matching against users' preferences. To denote privacy policy matching we employ a matching function that takes as input a policy, a user preference and returns false is there is no match or returns the instantiated policy if there is a match.

[1] Disjunction could also be supported. For simplicity we here consider a minimal form of obligations.

[2] We assume the variable domains to be clear from the context.

We denote **PPol** $\mathcal{M}$ *PPref* the matching between a privacy policy **PPol** and user's preference *PPref*. Instantiated privacy policies are denoted as $\mathbf{PPol}_{PPref}$.

By definition, the same policy[3] can refer to a set of data items/variable names. The policy applies as it is to each item/variable in $\{d\}_{spec}$. More precisely, in case the policy is specified by compound data items, it applies also to the items in their transitive closure, $\{d\}_{exp}$, (see Section 2), unless more specific policies for the smaller data items exist.

An example of a privacy policy is as follows.

Example 4 *Doctors are allowed to store X-Ray medical reports for statistical analysis under the condition that patient's age is greater than 18 yielding an obligation to delete PII after the number of years specified by the patient.*

$\langle\{X - RayReport, PatientLastName\}$
$\{DH; Ob1; \{DeleteTimePref\}\}\rangle$
$DH = \langle\{statysticalanalysis, marketing\}; Doctors; store;$
$\{BirthDate > 1980\}\rangle$
$Ob1 = \langle(CurrentTime >= (X - RayReport.PPREF.$
$DeleteTimePref)); Delete(PatientLastName);$
$ON - VIOLATION(Notify\_Admin)\rangle$

PPREF is a set containing just an attribute $\{$*DeleteTimePref*$\}$ *and is associated with the type of data "X-Ray Report". The actual value of this preference depends on the specific instance of the data.*

Possible alternative policies for a same data item could be expressed and combined into a unique disjunction form for a more efficient representation. However, we employ this simple yet effective representation (also referred to as canonic privacy policy) to reason about policy harmonization and subsumption with no ambiguity.

3.3 Data graph

We provide a graphical representation of the FSP's privacy policies and data, to be instantiated before actual data disclosure or upon disclosing it. Such data structure, referred to as *data graph*, is a key structure describing how the various data items are related -according to the domain ontology used by FSPs- and the the corresponding privacy policies and obligations the FSP is committed to.

Definition 3.2 [Data Graph]. Let *Ont*=$\{\mathcal{D}, P\}$ be the ontology used by the FIM system. Let $\{d\}$, $\{d\} \in \mathcal{D}$, be a set of data variable names and $\{d\}_{exp}$ be its expansion. Let **PPol**-*Set* be the set of privacy policies associated with $\{d\}$. Let *Cond-Set* be the set of conditions appearing in the condition components of DataHand of **PPol**s in **PPol**-*Set*. A graph PG_u for $\{d\}_{exp}$, is a directed graph $\{\mathcal{N}, E, \Phi\}$ where:

- $\mathcal{N}$ is the set of nodes; each node $n \in \mathcal{N}$ has one of the following forms:
  - *policy-node* - n is a pair $\langle b, DataHand\rangle$, where $b \in \{d\}_{exp}$, *DataHand* is a data handling rule of a policy **PPol** $\in$ **PPol**-*Set*, such that $\mathbf{PPol}.\{d\}_{spec}$ includes b. *DataHand* can be empty, if n has only one incoming edge;
  - *obligation-node* - n conveys Obl, where Obl denotes an obligation identifier of a policy **PPol** $\in$ **PPol**-*Set*.
- E is the set of edges; $E = \{e = (n, n')|$
  $(n = \langle b, DataHand\rangle$ is a policy-node $\wedge$ $n' = \langle b', DataHand'\rangle$ is a policy-node$\wedge n.b \prec n'.b')$
  $\vee$ $(n = \langle b, DataHand\rangle$ is a policy node $\wedge$ $n' = \langle Obl\rangle$is an obligation-node $\wedge$ $\exists$ **PPol** $\in$ **PPol**-*Set*$|$ **PPol** $= \langle\{d\}_{spec}, DataHand, Obl, PPref\rangle)\}$
- $\Phi : E \rightarrow Cond\text{-}Set \bigcup Cond\text{-}Set*$ is the edge labelling function. An edge label expresses conditions that apply to the data in the node pointed by the edge.

Example 5 *An example of data graph is shown in Figure 1. The circular nodes, such as the "Patient Demographic Info" correspond to the policy nodes, whereas the rectangular nodes such as "Delete after 3 months from the starting date" correspond to the obligation nodes. The edges are represented as directed arrows from one policy node to the other. Finally the value of edge labeling function, such as "Age>18" labeling the edge from "Medical Record" to "Patient Demographic Info" denotes the condition that applies to the "Patient Demographic Info" node.*

By definition data graphs represent the policies that apply to the expanded data set of an initial set $\{d\}$. This condition is required to ensure that also implicit data items are considered and their corresponding policies evaluated. As we will see, this design enables to check for policy subsumption without incurring in any inconsistency.

Notice also that edges connect either nodes of the same type or policy nodes with obligation nodes; nodes connected in the latter case bind components of a same policy. Such representation has the advantage of directly supporting links from a same obligation to potentially multiple policy nodes to which the obligation refers to. Edges are possibly labelled with conditions, which are assigned by the edge labelling function. Such function specifies the conditions that apply to the data of the node entering the edge. See for example Figure 1 and the condition Age>18.

If a policy element in a node is empty, it means that no specific privacy policy for the data in the node has been specified. Then, the closest non-empty policy appearing in an ancestor node is applied. This case can happen for example for a policy associated with contact information data, that thus applies also to the telephone number and postal address. Operationally, when compound data is to be shared, the privacy policy that applies to the compound data is used only if it is

[3] We drop the term privacy to refer to policies when obvious.

not superseded by a more specific policy that applies to the data subcomponent.

Example 6 *With reference to Figure 1, there is no specific policy specified for the patient's Date of Birth. In case such data is released, privacy policy identified as p22 is applied. By contrast, if SSN is released then p24 applies.*

Obligation nodes are attached to the node conveying the data referred in the obligation. For instance, in Example 4, the obligation node is attached to the patient last name node (see Figure 1), since the obligation refers to such data item.

4 Privacy Policy Subsumption

In order to provide a systematic approach for compliance checking of privacy policies we rely on the introduced notion of data graph, and on the notion of subsumption. The former is useful to identify, given a data set, the relevant set of policies. The latter is useful for understanding the relationships among policies.

Policies' subsumption is verified by comparing each policy component. In particular, to compare conditions it is necessary to identify the set of legal values satisfying a given condition. We compute such values by using a simple function, referred to as *LegalValues(X)*, which returns the set of values for a condition *X* (also used in Lines 4,5 of Algorithm 2). The function may return an unbounded set of values or variable names. Without loss of generality, in this discussion we focus on finite sets.

We define policy subsumption in what follows.

Definition 4.1 [Policy subsumption] Let **PPol**$_a$ and **PPol**$_b$ be privacy policies, specified according to Definition 3.1. **PPol**$_b$ *subsumes* **PPol**$_a$ if the following conditions hold:
• The data handling rule of **PPol**$_b$ is implied by the rule in **PPol**$_a$, that is:

- **PPol**$_a$*.DataHand.Access* $\subset$ **PPol**$_b$*.DataHand. Access*;
- **PPol**$_b$*.DataHand.Purpose* $\subset$ **PPol**$_a$*.DataHand.Purpose*
- For all conditions c appearing in **PPol**$_a$*.DataHand.cond*, the output of $LegalValues(c)$ is included in the corresponding output of $LegalValues(c')$, where c corresponds to the respective condition in the **PPol**$_b$.

• The obligation in **PPol**$_b$ matches the actions and conditions imposed by **PPol**$_a$. Precisely:

- **PPol**$_a$*.Obl.actions*=**PPol**$_b$*.Obl.actions* and **PPol**$_a$*.Obl. corrective-actions*= **PPol**$_b$*.Obl.corrective-actions*
- For all conditions c appearing in **PPol**$_b$*.Obl.Events*, the output of $LegalValues(c)$ is included in the corresponding output of $LegalValues(c')$, where c corresponds to the respective condition in the **PPol**$_b$.

Algorithm 1 Subsumption(**PPol**-*Set*$_a$, **PPol**-*Set*$_b$, $\{d\}_a$, PG^a, PG^b)

1: {Recall graph's structurePG^a=$\{\mathcal{N}, E, \Phi\}$}
2: { Initialize set Subsumed}
3: Subsumed $\leftarrow \{d\}$
4: **for all** $d \in \{d\}_a$ such that $\exists d_b$ and $d_a = \theta(d_b)$ **do**
5: {θ denotes the semantic equivalent function}
6: **if** $n.d \in \mathcal{N} \wedge n.PPol \neq \perp$ **then**
7: {Find the right policy in a's graph}
8: $DataHand_a = n.PPol_a.DataHand$
9: **else**
10: Let $e = (n', n) \in E$ be the entering edge for n
11: $DataHand_a = n'.PPol_a.DataHand$
12: **end if**
13: **if** $n.d \in N \wedge n.PPol \neq \perp$ **then**
14: {Find the associated policy b's graph}
15: $DataHand_b = n.PPol_b.DataHand_b$
16: **else**
17: Let $e = (n', n) \in E$ be the entering edge for n
18: $DataHand_b = n'.PPol_b.DataHande$
19: **end if**
20: **for all** $DataHand_a.X_a$ **do**
21: **CASE** $X_a = purpose$; $X_a = Access$
22: **if** $DataHand_a.X_a \subset DataHand_b.X_b$ **then**
23: Subsumed $\leftarrow$ Subsumed -d
24: EXIT For Loop
25: **end if**
26: **CASE** $X_a = cond$
27: SubCond =CheckCondition($DataHand_a.cond_a$, $DataHand_b.cond_b$)
28: **if** SubCond=false **then**
29: Subsumed $\leftarrow$ Subsumed -d
30: EXIT For Loop
31: **end if**
32: **ENDCASE**
33: **end for**
34: $Obl_a = PPol_a.Obl$
35: $Obl_b = PPol_b.Obl$
36: **for all** $Obl_a.X_a$ **do**
37: **CASE:** $X_a = actions$, $X_a = corrective\text{-}actions$
38: **if** $Obl_a.X_a \neq Obl_b.X_b$ **then**
39: Subsumed $\leftarrow$ Subsumed -d
40: EXIT for loop
41: **end if**
42: **CASE:** $X_a = events$
43: **if** $\exists c \in Obl_a.events_a$ **then**
44: SubCond_1 =CheckCondition($Obl_a.event_a, Obl_a.event_b$)
45: **if** SubCond_1=false **then**
46: Subsumed $\leftarrow$ Subsumed -d
47: EXIT For Loop
48: **end if**
49: **end if**
50: **end for**
51: **end for**
52: {Check that the two SPs support the same type of preferences}
53: $PPref_a = PPol_a.PPref$
54: $PPref_b = PPol_b.PPref$
55: **if** $PPref_a \not\subseteq PPref_b$ **then**
56: Subsumed $\leftarrow$ Subsumed -d
57: **end if**
58: **return** Subsumed, SubCond $\bigcup$ SubCond_1

• **PPRef** types supported by **PPol**$_a$ and **PPol**$_b$ are compatible, in terms both of privacy terms (pt) and of supported prefer-

Algorithm 2 Function CheckCondition(cond$_a$, cond$_b$)

```
1: SubCond ← ∅
2: if ∃cond_a ∧ cond_b st. cond_a.d = θ(cond_b.d') then
3:     if cond_a, cond_b of the form d op value then
4:         Lv_1=LegalValue(cond_a)
5:         Lv_2 =LegalValue(cond_b)
6:         if Lv_1 ⊄ Lv_2 then
7:             return SubCond ← false
8:         end if
9:     else
10:        if cond_a, cond_b of the form d op [val1, ..., valk] then
11:            if [val1, ..., valk]_b ⊈ [val1, ..., valm]_b then
12:                return SubCond → false
13:            end if
14:        end if
15:    end if
16:    {var denotes a variable}
17:    if cond_a, cond_b of the form d op var then
18:        SubCond ← cond_a
19:    else
20:        if cond_a, cond_b of the form d op [var1, ..., vark] then
21:            if [var1,..., vark]_a ⊈ [var1,..., vark]_b then
22:                return SubCond ← false
23:            else
24:                SubCond ← cond_a {Conditionally subsumed}
25:            end if
26:        end if
27:    end if
28: else
29:    SubCond← FALSE
30:    return SubCond
31: end if
```

ences.

Example 7 *Consider the following privacy policy, applied by hospital Global-Health. Nurses and doctors are allowed to store on X-Ray medical reports for statistical analysis under the condition that patient's age is greater or equal to 21 yielding an obligation to delete identifiable information after 4 year. Users will be notified in case of violations.*

$\langle\{X - RayReport, PatientLastName\}, \{DH1; Obl; PPref\}\rangle$
$DH1 = \langle statysticalanalysis; \{Doctors, Nurses\}; store;$
$BirthDate \leq 1988\rangle$
$Obl = \langle(CurrentTime - DisclosureTime >= 4);$
$Delete(PatientLastName); ON - VIOLATION(Notify\_Admin)\rangle$

The above policy is subsumed by policy of example 4. Subsumption holds because the two policies are specified against the same set of data variable names ({X-Ray Report, Patient LastName}), and the data is accessible to a set of entities which is a superset of the set of entities specified in the policy of example 4. The condition specified in Pol1 is included in Pol2 in that LegalValues(BirthDate<1990)= [1900, 1990] while LegalValues(BirthDate≤1988)= [1900, 1988]. Concerning the obligations, the same conditions are applied by both hospitals. Notice that subsumption final decision is subjected to the user privacy preferences, in that only if the years specified by the user's preferences are less than PPref.DeleteTimePref the condition will be satisfied.

A subsuming privacy policy is thus a policy which guarantees that the actions/conditions mandated in the subsumed policy are preserved when enforcing it.

Subsumption policy checking is carried out when two FSPs share some PII, for example to enable a transaction or a business interaction involving multiple parties. We refer to the FSP storing users' PII and knowing the associated *PPref* of users as the *Provider* FSP. We refer to the FSP receiving the users' PII as *Receiver* FSP. The outcome of a subsumption policy checking process will highlight (if it does not fail) if the conditions and constraints in the two sets of policies (of Provider FSP and Receiver FSP) are compatible - in particular if the Receiver FSP's policies satisfies all policies mandated by Provider FSP's ones, inclusive of user's *PPref*.

Subsumption can be verified according to different strategies, depending on the frequency of the checking and on whether subsumption is performed on policies specified in terms of data values -and instantiated privacy preferences- or not. Thus, we support two different versions of algorithms for subsumption checking:

1. Subsumption checking algorithm for policies on data variable names of specified types between Provider FSP and Receiver FSP;
2. Subsumption checking algorithm for "contextualized" policies (i.e subject to preferences and other metadata), given a specific instance of data, again between Provider FSP and Receiver FSP.

The first subsumption checking algorithm (and related processes) allows a Provider FSP to perform a general checking about the *compatibility* of its own policies with policies of other FSPs. Such check can be part of an ongoing, periodic verification process among service providers that belong to the same "circle of trust" [2] i.e. "a group of service providers that know each other and are used to engage in business interactions". To avoid not required executions, subsumption checking can be driven by explicit 'policy-change" or "data-type-change" events communicated to the FSP by other federated FSPs.

Notice that this kind of policy checking cannot keep into account specific values of privacy preferences at the granularity of a specific user. As such, some conditions cannot be fully checked in terms of subsumption (for example, obligation conditions might involve privacy preferences and constraints on value ranges) because of their dependency on instantiated privacy preference values. In this case, by assuming a high level compatibility of these policies, the algorithm specifies the conditions to be verified upon privacy preferences' instantiation (set *SubCond* returned by Algorithm 1), thus determining a conditional success.

The second subsumption checking algorithm (and related processes) performs a fine grained checking at the time of disclosing a specific piece of user's data, by keeping into account specific values of attributes in conditions and obligations along with specific, instantiated user's privacy preferences. It is always to be performed in case some conditions

need to be verified based upon users' actual preferences and values. Thus, it includes checking attribute values or comparing actual conditional ranges and may turn a conditional Success into a *Success* or a *Failure*.

Example 8 *Consider a case when there are two clinics US-Health and Global-Health. Both Clinics allow US doctors to view the results of enrolled patients which have the patients permission to do so. Global-Health also allows international doctors to view their patient records given an international license certificate. US-Health is particular to the USA and therefore does not allow international access. Algorithm 1 can be used by the clinics which can check for policy subsumption for all cases relating to access by US doctors. However Algorithm 2 would need to be carried out based on the request of a given doctor.*

The $Subsumption()$ algorithm in Algorithm 1 reports the first version of subsumption checking described above.

$Subsumption()$ first finds the appropriate policies to match by navigating the two FSPs' data graphs (line 2-17). Then, for each privacy policy to be checked, and in turn for each **PPol** component the algorithm evaluates whether the corresponding items of the two policies are included in one another. Equivalence of sets is instead mandated for the obligation components, to ensure that the two parties commit on the same type of obligations. This process requires verifying the single obligations and data handling components (lines 19-55), and running a specific function $CheckCondition()$, reported in Algorithm 2, whenever conditions are to be evaluated. Privacy preferences are matched (lines 57-59) to verify that at least the types of preferences supported by the two FSPs are compatible and based on the same privacy terms.

We provide a more generic definition of subsumption with respect to a specific data set $\{d\}$, in case $\{d\}$ denotes data variable names and thus privacy preferences are not instantiated.

Definition 4.2 [Data Set's Policies Subsumption] Let $\{d\}$ be a set of user's data variable names. Furthermore, let $PG_u = \{\mathcal{N}, E, \Phi\}$ be the data graph at the FSP Provider end associated to $\{d\}$ and $PG'_u = \{\mathcal{N}', E', \Phi'\}$ FSP Receiver's data graph. FSP Receiver's policies subsume FSP Providers' policies for data $\{d\}$ iff, the following conditions are met:

- $\forall d \in \{d\} \exists n' \in N' \wedge n \in Ns.t.\ n.d = \theta(n'.d) \wedge n.PPol$ subsumes $n'.PPol$ according to Definition 4.1;
- $E' \subset E$;
- The labeling function $\Phi : E \leftarrow Cond\text{-}Set \bigcup Cond\text{-}Set*$ is equivalent to Φ' in that each condition c appearing in $Cond\text{-}Set$ is mapped by ϕ' into a condition c' such that function $CheckCondition(c, c')$ (Algorithm 2) either returns *true* or condition c[4].

[4]We recall that c implies that the condition is to be evaluated upon user's instantiation of privacy preference

4.1 Analysis of subsumption

We now informally discuss the subsumption algorithm by analyzing its correctness, soundness and complexity.

Correctness In our context correctness refers to the ability of the $Subsumption()$ algorithm to determine subsumption for the data items that satisfy the subsumption conditions.

Theorem 4.1 [Subsumption Correctness] *Let $\{d\}$ be a set of data variable names. Let FSP_r and FSP_p be the FSPs maintaining respectively* ***PPol***-Set-*p and* ***PPol***-Set-*r privacy policy sets. Algorithm 1 returns in the $Subsumed$ set all and only the data items in $\{d\}$ for which subsumption holds, and it identifies the conditions under which subsumption holds for data items for which subsumption is conditional.*

The above result follows from the mechanism underlying subsumption checking. If subsumption is possible then it means that the privacy policies components are defined in such a way that all the components of the subsuming policy are included in the subsumed one, except for the special case of the obligations (line 20-21 of Algorithm 1). In the case of obligations, in fact, to avoid ambiguities and possible inconsistencies, we require actions and corrective actions to be identical. Although this approach may seem restrictive, from a practical standpoint it can in fact be easily addressed, if the Receiver FSP is willing to simply use the sender's obligations as they are.

Correctness for subsumption of conditions is guaranteed by Algorithm 2, which illustrates how all conditions are evaluated and compared based on their domain.

Soundness Soundness of Algorithm 1 is guaranteed if the process of subsumption is equivalent to the successful matching of users' privacy preferences with the FSP Receiver's policies, for *any* possible preference expressed by the user. The proof is reported in Appendix B.

Theorem 4.2 [Soundness of Policy Subsumption] *Let FSP_p be a FSP having user u's data according to privacy policies* $\mathbf{PPol}_a$. *Furthermore, let FSP_r be a second FSP applying privacy policies* $\mathbf{PPol}_b$ *on data $\{d\}$. Let* $\mathbf{PPol}_a$ $\mathcal{M}$ **PPref** *be verified. If Algorithm 1 returns subsumed then* $\mathbf{PPol}_b$ $\mathcal{M}$ **PPref**.

By definition, soundness differs from correctness since it shows that if the algorithm returns subsumed then it is based on a correct input that satisfies subsumption.

Decidability An additional important problem to consider within our context is that of *decidability*. In general, subsumption may be intractable [9]. Because of the restricted vocabulary adopted by our approach, subsumption is always

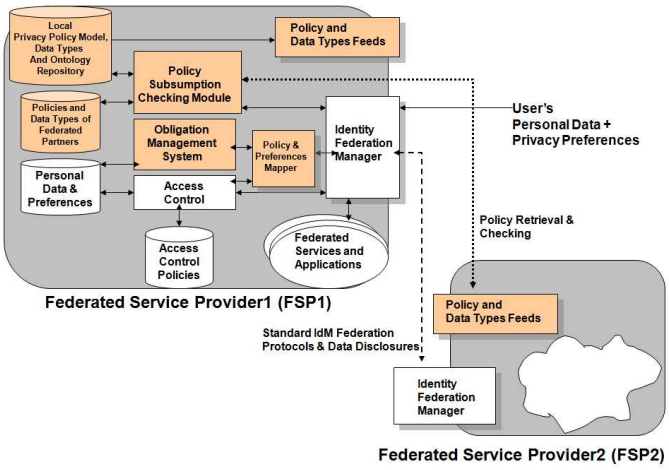

Figure 2. High-level System Architecture

decidable for the policies that can be expressed in our specification language.

The sets corresponding to the DataHand and Obl components are in fact enumerable. That is, both Algorithm 1 and 2 enumerate the sets which ensure the termination of the algorithms. For each component the subsumption relation is evaluated. After evaluating each element of the defined sets, the algorithms always terminates with a decision on whether the subsumption relation holds. The only assumption for ensuring that subsumption is in fact decidable is that policies components are all well specified, using elements of the reference ontology.

Complexity Algorithm 1 is polynomial of second order $n * m + 2m$ (given the two nested loops in Algorithm 1), where n is the maximum size of policy sets and m is the maximum size of data sets. 2 is the number of times Algorithm 2 is invoked. The algorithm is executed at most two times for each data for which corresponding policies are checked.

5 System Architecture

Current federated identity management solutions (e.g. based on Liberty Alliance, Shibboleth, OpenId, WS-*, etc. specifications) include provisioning, authentication, access control and authorization solutions. We designed our policy subsumption checking solution as a plug-in that can be deployed in such architectures and leverage some of their existing functionalities. Figure 2 provides a high-level view of the architecture of our solution. This architecture consists of the following components:

• *Policy and Data Types Feeds*: it explicitly advertises - at a FSP site - privacy policies (and related privacy preferences) that apply to PII, along with an indication of the used ontology. This information can be checked both by humans, in a readable format, and by our solution, once retrieved in a programmatic way (for example via accessible XML feeds);

• *Policy Subsumption Checking Component*: it implements policy subsumption checking between two FSPs, based on the algorithms described in Section 4. Such component is in charge of: (1) retrieving privacy policy information and data type definitions from other FSPs (via the *Policy and Data Types Feeds*); (2) checking for subsumption of the FSP privacy policies against the ones defined by the FSP partners, accordingly to the algorithms and steps illustrated in this paper.

• *Policy and Preferences Mapper*: it is in charge of manipulating users' PII and their privacy preferences and mapping them onto operational access control policies and privacy obligations. This module ensures that users' preferences can be managed and enforced over time [11];

• *Obligation Management System*: it is in charge of (operationally) managing and enforcing privacy obligations (e.g. on data deletion, data transformation, notifications, etc.) on locally stored PII. These obligations, inclusive of parametric obligations i.e. subject to privacy preferences defined by users, are consistent with the obligation policies advertised by the FSP, for a given data type [10];

• *Access Control*: it is a privacy-aware access control component, in charge of intercepting attempt to access, modify and store personal data and ensuring that this happens consistently to predefined access control policies and users' preferences. This component enforces fine-grained access to data and is configured by the Policy & Preferences Mapper, which operates like the mapper for obligations [10];

• *Identity Federation Manager*: it denotes a standard identity federation component (e.g. as specified by Liberty Alliance, WS-Federation, etc.), extended with a mechanism to intercept the disclosure of a user's PII from a FSP to another FSP and subordinates the actual disclosure based on a positive policy compliance check (in particular in case of conditional success), that keeps also into account any related user's privacy preferences. The *Federated Identity Manager* is activated in case a single-sign on session is required (due to users' activities, e.g. trying to access remote services) or in case data has to be transferred from one FSP to another FSP.

Notice that our approach differs from current solutions where the disclosure of data is subordinated by simple access control checks, primarily based on local information (e.g. simple access control preferences set by users) and/or contractual assumptions on the other federated party.

6 Related Work

The problem of privacy policy compliance has been investigated by both researchers and practitioners.

W3C P3P[14] addresses such problem in the context of User-Service Provider interactions; in this context, sets of users'

preferences are matched against service providers' privacy promises. Despite the fact that such type of compliance checking is outside the scope of our work, it is nevertheless important to highlight that policy checking in such context is pretty much simple. Matching is primarily based on comparing sets of labels and thus it does not keep into account the complexity of actual privacy policies including conditional data handling and privacy obligations.

In the context of multi-party interactions, involving two or more service providers, an approach has been developed in the context of the PRIME Project [15]. Such an approach is based on checking upfront the requests of credentials by a data requestor - in our context, this would be the FSP provider - against the policies of the sender service provider. Such matching is executed during a negotiation process between two service providers (and similarly, between a user and a service provider), in the context of an existing interaction, driven by an access control decision framework. This approach requires that all involved parties use the PRIME Toolbox/Middleware and engage in specific interactions.
In our work we aim at "detaching" the policy checking and subsumption phase from the operational/enforcement phase. Such phase is the phase where policies are actually deployed into service providers' systems and solutions that are in charge of enforcing them. It includes, for example, access control systems and obligation management systems.

A FSP can check, upfront, if the policies of a remote FSP can be subsumed - without having to engage in interactions or negotiations. Of course, the checking process can be carried out in a fine grained way, to the point that specific privacy preferences can be factored in, at the actual disclosure time of data, should subsumption of policies be conditional to this specific information.

Current commercial federated identity management solutions, such as the ones compliant to Liberty Alliance standards [2] and WS-* standards [8] do not provide policy subsumption/checking capabilities i.e. they do not provide automated ways to check for compliance of policy sets; they primarily rely on contractual agreements between an Identity Provider and Service Providers and on simple mechanisms to check for user's privacy preference, like P3P, at the time of PII disclosure. Within the Liberty Alliance[6] framework, for example, a multi-level policy based approach has been proposed to facilitate attribute exchange in federations. The aim of the approach is to give users, i.e. the data providers, full control on the release and usage of their information stored at the attribute providers while simplifying the matching of service providers' privacy policies with the users' privacy preferences. However, the proposed approach restricts the set of privacy policies and preferences to a small number of standardized privacy policies to which all federation entities can refer. Moreover, it does not provide a mechanism for the enforcement of these policies at the SP's site. Another interesting approach is by Ahn et al. [1], who propose a privacy preference expression language called PREP for recording the user's privacy preferences with Liberty enabled attribute providers. The proposed language, a restricted version of APPEL, supports the multi-level policy approach suggested in Liberty Alliance specifications. Our approach can be leveraged to provide additional "policy compliance checking" capabilities, complementary to these existing approaches. Similar observations apply to other federated identity management frameworks, such as OpenId[17], Higgins [5] and Shibboleth[7]. Our work can be exploited as an "add-on" capability to provide an additional level of trust, by enabling upfront checking of policies.
The problem we address in this paper has some similarities with policy reconciliation protocols [19, 9]. Policy reconciliation in general has the goal of determining policy rules that two or more parties have in common in the context of heterogeneous access control structure. Although related, such problem differs from ours since reconciliation focuses on harmonizing access control policies and on taking consistent authorization decisions. We instead focus on the specific problem of reconciling privacy rules, obligations and users preferences. Due to the different nature and purposes of such policies -one aims to establish common access control decisions while the other deals with privacy practices of PII holders-, policy reconciliation process is complementary to privacy policy subsumption. Privacy has been considered in [19]. However, they propose an approach based on cryptographic functions to protect the privacy of the policies themselves, and not for the data which the policies are specified for.

7 Conclusion

In this paper we address the problem of privacy in a federated environment. Specifically, we introduced mechanisms and algorithms for policy compliance checking between federated service providers, based on an innovative policy subsumption approach. We formally introduce and analyze our approach.We also introduce and describe a high-level architecture of our solutions that is suitable for deployment and application in existing federated identity management solutions, such as Liberty Alliance, WS-* and Shibboleth. The prototype of the modules illustrated are currently under development. We are investigating how to translate the logic formalism presented in the paper in a machine readable format, by leveraging existing privacy languages. Future work will include deploying our solution in the context of Shibboleth and Liberty Alliance and providing a quantitative analysis of its impact in terms of performance and efficiency.

References

[1] Gail-Joon Ahn and John Lam. Managing Privacy preferences for federated identity management. In *Digital Identity Management*, pages 28–36, 2005.

[2] Liberty Alliance Project. http://www.projectliberty.org.

[3] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology Matching: A Machine Learning Approach, 2003.

[4] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

[5] Higgings. Open Source Initiative, http://www.eclipse.org/higgins, 2007.

[6] http://www.projectliberty.org. Liberty architecture framework for supporting privacy preference expression languages (ppel's), 2003.

[7] Internet2. Shibboleth. http://shibboleth.internet2.edu.

[8] Web Services Standard Listings. http://en.wikipedia.org/wiki/list_of_web_service_specifications, 2007.

[9] Patrick McDaniel and Atul Prakash. Methods and Limitations of Security Policy Reconciliation. In *2002 IEEE Symposium on Security and Privacy*, pages 73–87. IEEE, MAY 2002. Oakland, CA.

[10] Marco Casassa Mont and Filipe Beato. On parametric obligation policies: Enabling privacy-aware information lifecycle management in enterprises. In *POLICY*, 2007.

[11] Marco Casassa Mont and Robert Thyne. A Systemic Approach to Automate Privacy Policy Enforcement in Enterprises. In *Privacy Enhancihng Technologies Workshop*, 2006.

[12] Eric Norlin and Andre Durand. Whitepaper on towards federated identity management. in ping identity corporation, 2002.

[13] Privacy Organization. http://www.privacy.org/.

[14] The Platform for Privacy Preferences 1.0 (P3P1.1) Specification. http://www.w3.org/tr/p3p/.

[15] Privacy for Identity Management in Europe. EU Project Framework VI PRIME. https://www.prime-project.eu, 2007.

[16] Vipin Samar. Single sign-on using cookies for web applications. In *WETICE*, pages 158–163, 1999.

[17] SourceID: Open Source Federated Identity Management. http://www.sourceid.org/resources/basics.html.

[18] M. Uschold and M. Gruninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review 11(2)*, 1996.

[19] Jonathan Voris, Sotiris Ioannidis, Susanne Wetzel, and Ulrike Meyer. Performance evaluation of privacy-preserving policy reconciliation protocols. In *POLICY*, 2007.

A Proof of Theorem 4.1

The proof for theorem 4.1 follows from the definition of subsumption. Given two policies $\mathbf{PPol}_a$ and $\mathbf{PPol}_b$, if Receivers **PPol** subsumes Provider **PPol** then all conditions of components of Definition 3.1 are satisfied.

The same conditions are checked for each policy component by Algorithm 1. Additionally, Algorithm 1 satisfies each condition of Definition 3.1 by making use of the CheckCondition() function to identify conditions compliance. The same set of checks is repeatedly executed for each data item identified; thus the result holds.

B Proof of Theorem 4.2

We prove this claim by contradiction. Assume that Algorithm 1 returns a data item d in the set $Subsumed$ that is in fact not subsumed. According to algorithm 1, d is in the subsumed set if the conditions $DataHand.X, DataHand_p.X_p \subset DataHand_r.X_r$ and $Obl.X, Obl.X_r,\ Obl.X_p = Obl.X_b$ with X ranging among the elements are satisfied (lines 46). Additionally the policy tuple and $PPref$ supported by by the two FSPs must be of the same structure and conditions need to be unconditionally inclusive one of another.

Since the above are the same conditions according to which subsumption is specified, if Algorithm 1 succeeds than the data input must be subjected to subsuming policies. This is however in contradiction with our initial assumption. We thus conclude that the thesis holds. □