Semantic Cities
AAAI Technical Report WS-12-13

Data-Centric Privacy Policies for Smart Grids

Sebastian Speiser and **Andreas Harth**
Karlsruhe Institute of Technology (KIT), Germany

Abstract

Smart cities and smart grids heavily depend on data being exchanged between a large number of heterogeneous entities. Parts of the data which such systems depend on are relevant to the privacy of individuals, e.g., data about energy consumption or current location. We assume the use of semantic technologies for data representation and exchange, and express privacy requirements as formal policies. We take a data-centric view, that is, we attach policies that restrict isolated uses of data to the data directly. When systems exchange data, they also exchange the policies pertaining to the exchanged data. The main benefit of such an approach over a system-level view is that our data-centric approach works in scenarios without central control.

1 Introduction

The smart grid is the goal of current efforts to radically redesign the aged energy grid. The aim is to create a network of distributed energy suppliers implementing a demand-driven production. Like other smart systems, the smart grid builds on information exchange between large numbers of heterogeneous entities. In order to make energy grids more efficient and reduce environmental impact, the smart grid requires up-to-date and fine-granular data about energy consumption and production. Semantic technologies facilitate the integration of information from different entities, enabling technical interoperability. However, social aspects pose further restrictions on interoperability, especially the privacy requirements of individuals. There are scenarios how energy consumption and related data can be both used for value-added services and misused to the disadvantage of individuals. Collected data includes not only consumption data, but for example also geographical locations of electric vehicles, or floor plans obtained through a vacuum cleaning robot. The goal is to create an ecosystem of services creating value using information provided by smart appliances and at the same time protecting the privacy of individuals.

Traditional means, i.e., organisational control for ensuring compliance with usage restrictions are not sufficient in the smart grid, due to heterogeneity of available services and individuals' privacy requirements. Instead, we propose a method for formally expressing privacy policies of individuals in a way that data processors can automatically check the compliance of their actions. A special challenge is the lack of a central and complete view on the systems processing privacy-relevant information. The importance of the lacking central control and the heterogeneity of information providers and processors further increases when smart grids are coupled with other smart city systems such as smart factories. Without a central view, we cannot use existing policy approaches, which either are restricted to access control, or express usage restrictions on a system level, i.e., by describing the acceptable behaviour of the overall system. Instead, we propose the concept of data-centric policies, which employ a localised view on specific information uses. We attach policies to the information they protect and introduce a method to restrict admissible uses of information dependent on the protected information.

In Section 2, we describe a smart grid scenario for information sharing, motivating our contributions:

- We discuss the concept of data-centric privacy policies, which are particularly useful for applications without central control (Section 3).
- We present a language for expressing data-centric policies as a specialisation of the general language framework presented in (Krötzsch and Speiser 2011) (Section 3).
- We describe how policies can be attached to the data they protect (Section 4.1).
- We present patterns that formalise common aspects of privacy restrictions (Section 4.2).

In Section 5, we apply the approach to realise our scenario. Section 6 discusses related work. We conclude in Section 7. Parts of this paper are based on our technical report that introduced the notion of data-centric policies (Speiser 2012).

2 Scenario and Requirements

Our example scenario is about Carol, who lives in an apartment equipped with a smart meter and smart appliances, which track in detail her consumption of electrical energy. She has to share consumption data with her energy producer for billing purposes, and also allow the producer to store the data for one year, in order to protect himself in case of a dispute about the bills. Carol further wants to share her consumption data with an energy optimiser service, which consults her about potential ways to save energy. The op-

timiser service uses the collected data from different customers and historical energy pricing information to improve its service. Carol allows such further usage only after her data was anonymised, in which case she also allows usage for statistical purposes and sharing of her data for the same purposes. Carol's washing machine does not only provide consumption data, but also information about the washing behaviour, including selected washing programs, load of the machine, and used detergent. Carol gives the manufacturer of the washing machine access to this information for maintenance purposes, so that the manufacturer can monitor the relation between energy consumption and usage and can discover occurring problems.

The presented scenario shows characteristics that are typical in smart systems, and which a policy approach for such systems is **required** to address:

- Information is used in different contexts that cannot be anticipated when the information is created or collected, e.g., energy consumption data can be used for billing but also for servicing appliances.
- There is a wide range of types of information with different usage restrictions, e.g., publicly available weather data is integrated with confidential data about individuals.
- Restrictions can differ largely between information artefacts depending on their owner even though they belong to the same category, e.g., two persons having different privacy requirements regarding their home address.
- There is no central storage for information, e.g., it is impossible to centrally store detailed energy consumption information about every consumer in the world.
- There is no central entity controlling the usage of specific information artefacts, e.g., the manufacturer of Carol's washing machine and the energy producer are independent entities and both use Carol's consumption data.

A suitable information architecture for the smart grid can be realised based on the Linked Data[1] principles (Wagner et al. 2010). The open and royalty-free standards of the web architecture enable interoperability between heterogeneous entities, and impose low entry barriers for new players. Data is stored in a decentralised way, near to the owner (e.g., consumption data is stored on the smart meter, which is under control of the consumer), which enables the control of information disclosure respecting desired privacy policies. Furthermore, the scalability of the web architecture and the ease of integrating RDF data are inherited.

3 Data-centric Policies

Without a central and complete view on the systems in which information is used, we cannot gather a complete process description containing the past and future actions relevant to a data usage. Consider that we want to check the compliance of a usage action u to the policy of the used artefact a. Reasons for not having access to the history, i.e., the actions performed before u, can include: (i) the history is sensitive, e.g., a may be the anonymised version of an artefact a' whose identity must not be revealed; (ii) the producer of a does not want to reveal his processes; (iii) representing the history means too much data, which is amplified by the lack of clear constraints about what belongs to the relevant history: theoretically relevant are all actions that produced a or produced artefacts from which a was directly or indirectly derived. Reasons for not knowing about the future, i.e., the actions that will be performed after u, can include: (i) same lack of clear constraints as with the history: theoretically all future actions using products of u or their derivations are relevant; (ii) future is simply not yet known, because unanticipated usages of the information might become desirable; (iii) same as with the history, not all information users want to reveal their processes; (iv) the future execution is not yet known as decisions, which possible future processing branch to take, will depend on not yet known context information or other information artefacts.

Instead, we need to model a local view on an information usage whose compliance can be checked without having a global overview of the complete processes in which the usage is embedded. The local view does not necessarily mean that only an isolated usage action is modelled, e.g., for ensuring that a stored artefact is deleted within a time span, we have to model both the storage and the deletion action. Instead, we define locality as only restricting the usage action and the actions that must be performed in order to fulfill obligations, which are linked to the allowance to perform the usage. The locality is achieved by only regarding used and produced artefacts and their policies instead of the actions that produced the used artefacts and the actions that use the produced artefacts. Based on this notion of locality, we require that policies should be data-centric, as defined by the following three properties: (i) each artefact has its own policy that is attached to the artefact (a so-called sticky policy (Karjoth, Schunter, and Waidner 2003)); (ii) the policy of an artefact employs a local view on usages; (iii) the policy of an artefact can restrict the policies, which can be assigned to derived or copied artefacts. With data-centric policies the compliance of an isolated data artefact usage is verifiable without the need for a central instance knowing the complete process in which the usage is embedded and the policies can be transferred together with the artefacts between different providers, because they are attached to their protected artefacts. Two basic assumptions for data-centric policies, which have to be enforced by corresponding policy-aware systems, enable the localised view: (i) each used artefact has an attached policy, that was either set by the artefact's owner or was assigned in compliance with the policies of the artefacts used to derive the artefact; (ii) produced artefacts will be used in compliance with the assigned policy.

3.1 Vocabulary

In the following, we present a vocabulary for modelling a local view of information usages in first-order logic (FOL). For describing the vocabulary we use the terms *class* and *property* for FOL predicates of arity 1, respectively 2. The vocabulary is aligned to the Open Provenance Model (OPM) (Moreau et al. 2010), reusing the the terms *Artefact*, *Process*, *used*, *wasGeneratedBy*, and *wasTriggeredBy*. The

[1] http://www.w3.org/DesignIssues/LinkedData

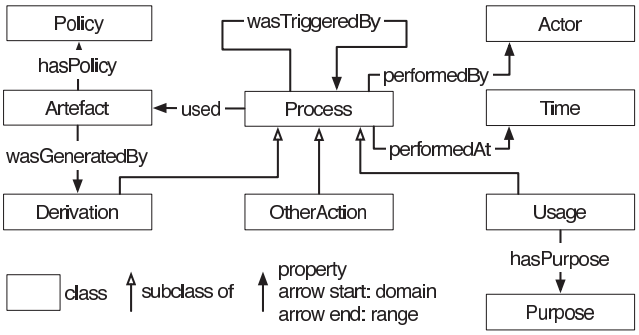

Figure 1: Vocabulary for modelling a local view on information usages

core of our model is illustrated in Figure 1.

An *Artefact* is an immutable physical manifestation of an abstract information object with a fixed *Policy*. A *Process* represents an action, which can be performed by a system. We further split processes into *Derivation*s (processes that generate a new artefact), and *Usage*s that only use artefacts without change. Usages are a general class characterised by their purpose. For derivations we define the following basic set of subclasses: *Storing* a new copy of an artefact, *Deletion* of an artefact, *Sharing* of an artefact with a *recipient*, i.e., giving a new copy to another entity, and a general class of *Modification*s, with the example of *Anonymisation*. The relations between processes and artefacts are modelled by the *used* and *wasGeneratedBy* properties. The *hasPolicy* property assigns to an artefact a *Policy*, which means that all processes using the artefact are required to comply to its policy. The *hasPurpose* property relates a usage to its *Purpose*, e.g., stating that a usage was non-commercial. A process p_1 *wasTriggeredBy* another process p_2, if p_1 can only have started after p_2 started. So, somewhat contrary to intuition, the "triggering" is rather a precondition but not a necessary cause of the triggered one. A usage restriction that requires attribution would thus be formalised as a policy requiring that the usage process *wasTriggeredBy* an attribution process, and not the other way around. Processes are *performedAt* at specific *Time* points and are *performedBy* an *Actor*.

3.2 Policies

In computer science, the notion of a policy refers to a formal description of the actions and behaviours that are allowed or required in a protected context. In the following, we restrict to the FOL fragment corresponding to Datalog with safe and connected rules. Datalog is a popular choice as formalism for policy languages, e.g., (Bonatti et al. 2010; Becker, Fournet, and Gordon 2010; Ringelstein and Staab 2010). We model a policy p as a union of conjunctive queries with the query predicate φ_p and one head variable; the individuals that are answers to the query of a policy form the set of compliant actions.

For example, a policy p that allows usage for statistical purposes is formalised as follows:

$$\mathsf{p}\colon\ \varphi_{\mathsf{p}}[x] \leftarrow \mathsf{Usage}(x) \wedge \mathsf{hasPurpose}(x,u) \wedge \mathsf{Statistical}(u).$$

A collection of such policy definitions $p\colon \varphi_p[x]$ will be called a *policy system*.

3.3 Content-based Restrictions

Restricting the policies that can be assigned to derived data artefacts lead to a localised view of data-centric policies. In the simplest case, derived artefacts inherit the policy of the used artefact (e.g., a copy of a confidential artefact is also confidential). However, in general the policy of the produced artefact can also allow broader usages (e.g., after anonymising an artefact) or allow narrower usages (e.g., a data recipient is not allowed to share it further). For restricting the allowed policies of produced artefacts there are two possibilities: (i) name-based policy restrictions, i.e., explicitly listing all admissible policies, and (ii) content-based policy restrictions, i.e., describing the set of admissible policies based on their definitions. Policies using name-based policy restrictions specify an exhaustive list of admissible policies for a produced artefact, which is problematic in scenarios with independent actors that employ policies with different names (i.e., not in the specified list) but a compatible set of allowed usages (i.e., should be in the list). In a scenario with a multitude of heterogeneous providers, it is not realistic to assume canonical names for policies with the same set of compliant usages. To overcome such incompatibilities, we require that the policy language supports content-based policy restrictions. Instead of explicitly listing admissible policies for produced artefacts, the content-based approach describes the admissible policies in terms of the minimal or maximal allowed usages. Content-based restrictions increase compatibility by making policies admissible based on the usages they allow independent of their names. Checking whether a policy fulfills at most or at least the same usages as a target policy corresponds to policy containment, which is well-studied for many policy languages as a tool for policy management, e.g., (Bonatti and Mogavero 2008). The approach of content-based policy restrictions goes one step further and uses policy containment as part of the policy definitions themselves, which requires meta-modelling (policies are both sets of allowed usages as well as part of usage descriptions) and self-referentiality (a policy can require that derived artefacts have a policy that allows at most the same uses as the original policy). Besides setting an upper bound on the actions allowed by a target policy, the containment condition can also set a lower bound. For example, an individual can decide to share energy consumption data with a statistics institute under the condition that derived artefacts are made available for public use and not only for paying clients.

Example 1 *Consider a policy* p *that allows a provider to store an artefact with policy* p_1 *allowing arbitrary usages:*

$$\varphi_{\mathsf{p}}[x] \leftarrow \mathsf{Storing}(x) \wedge \mathsf{wasGenBy}(a,x) \wedge \mathsf{hasPolicy}(a,\mathsf{p}_1)$$
$$\varphi_{\mathsf{p}_1}[x] \leftarrow \mathsf{Usage}(x).$$

The provider that wants to store the artefact, however specifies that he assigns policy p_2*, which only allows usages for*

statistical purposes. The storage action is classified as non-compliant, because the provider wants to use policy p_2 *instead of policy* p_1*. However, the intention of the artefact owner is most certainly that storage is also allowed under a policy that allows a restricted subset of usages. We capture this intention by reformulating* p *as* p' *using a content-based policy restriction for requiring that the stored artefact has a policy contained in* p_1*:*

$$\varphi_{\mathsf{p}'}[x] \leftarrow \mathsf{Storing}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p_a) \wedge \mathsf{containedIn}(p_a, \mathsf{p}_1).$$

Now, the storage of the artefact with policy p_2 *is compliant, as every usage for statistical purposes, i.e., every usage compliant to* p_2*, qualifies as an arbitrary usage, i.e., is also compliant to* p_1 *and thus* p_2 *is contained in* p_1*.*

3.4 Containment of Policies

This section outlines the semantics of enabling the use of policy containment as part of policy conditions for Datalog-based policies. A formal definition of the semantics for general FOL fragments and notes on implementation are given in (Krötzsch and Speiser 2011).

We introduce a new predicate containedIn that represents the containment relation between two policies based on their definitions:

$$\forall x, y.\mathsf{containedIn}(x, y) \leftrightarrow \forall z.(\varphi_x[z] \rightarrow \varphi_y[z]).$$

When allowing policy conditions to use containedIn, the question whether or not a process complies to a policy in turn depends on the evaluation of containedIn. The interpretation of the containedIn predicate goes beyond standard FOL, if it is used in a self-referential manner in policy conditions. Consider the definitions for policies r and s:

$$\mathsf{r}\colon \varphi_{\mathsf{r}}(x) \leftarrow \mathsf{Derivation}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{r}).$$

$$\mathsf{s}\colon \varphi_{\mathsf{s}}(x) \leftarrow \mathsf{Derivation}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{s}).$$

Whether r is contained in s depends on the condition $\forall p.\mathsf{containedIn}(p, \mathsf{r}) \rightarrow \mathsf{containedIn}(p, \mathsf{s})$, which holds if every policy p contained in r is also contained in s, i.e., $\mathsf{containedIn}(\mathsf{r}, \mathsf{s})$. In other words: whether r is contained in s depends on whether r is contained in s. Thus, we freely choose whether the containment holds or not. Content-based policy restrictions are useful to increase interoperability by making policies compatible that have the same intention but different names. The same intention of r and s is obvious from their definitions, which coincide except for the policy names. Therefore, the semantics of the containedIn predicate is that its extension is maximised: we assume that containment holds if there is no contradicting evidence, e.g., a policy allowing all usages is not contained in a policy allowing only usages for a specific purpose. Policies and background knowledge have to expressed in a restricted, connected fragment of FOL to ensure monotonicity (Krötzsch and Speiser 2011).

4 Practical Aspects of Policy Awareness

In this section, we describe a method to attach sticky policies to information artefacts, and present a number of patterns expressing common policy restrictions.

4.1 Sticky Policies

To attach policies to information artefacts, we have two possibilities: attaching by value, and attaching by reference. By value means, that we want to store and transport the policy together with an artefact. For this we need a serialisation that can be processed on a wide range of systems and particularly is compatible with web protocols that underly our information architecture. The W3C standard Rule Interchange Format (RIF) defines a core dialect corresponding to Datalog with a serialisation in XML. RIF allows to identify policies, i.e., rules, by IRIs. We adopt the Linked Data principle that IRIs of policies should be HTTP URIs and dereferencing leads to a description of the policy, i.e., a RIF document defining the policy. By having identifiers and a mechanism to get policies from identifiers, we can support attaching by reference: attaching the IRI of a policy to an artefact is sufficient to enable the user of the artefact to retrieve the policy and check compliance of his actions using the artefact.

HTTP as the information access protocol of our architecture supports the link header, which can be used to communicate a reference to related information about the served resource[2]. We can use the link header to specify the corresponding policy. The link header not only gives a reference as a URI, but also specifies the relation to the served resource, which we will set to *policy*. An example header when serving an artefact with a policy `P1` will look like:

```
Link: <http://ex.org/pols#P1>; rel=policy
```

When serving information represented in RDF via the Linked Data architecture, the returned graph is identified by an URI and so is a normal resource that we can further describe with RDF triples. Via the *hasPolicy* property we can assign a policy to an identified graph. Such identified graphs are similar to named graphs for RDF, whose development was motivated by a need for expressing metadata, including usage restriction (Carroll et al. 2005).

4.2 Patterns for Common Policy Restrictions

Attribute-based Usage Restrictions

The policy vocabulary can be extended with application-specific properties, that can be used for modelling compliant actions. For example we can model the isCustomerOf property between agents owning an appliance and its manufacturer. We can restrict usage to agents, of which Carol is a customer, as follows:

$$\varphi_{\mathsf{p}}[x] \leftarrow \mathsf{Usage}(x) \wedge \mathsf{performedBy}(x, m) \wedge \mathsf{isCustomerOf}(\mathsf{carol}, m).$$

Data Sharing / Rights Delegation

One key feature of our proposed approach is the ability to express restrictions on the policies of shared data. Rights delegation can be allowed by enabling the rights holder to share the data with further parties under the same or more restricted conditions. The following example policy p_1 allows arbitrary usage and the free delegation of this right:

$$\varphi_{\mathsf{p}_1}[x] \leftarrow \mathsf{Usage}(x) \vee \big(\mathsf{Sharing}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{p}_1)\big).$$

Of course, the delegation can be further constrained e.g. on the attributes of the actor or the artefact as shown before.

[2] `http://www.ietf.org/rfc/rfc5988.txt`

One common aspect of rights delegation is limiting the depth up to which rights receivers can further delegate the rights. This can be implemented by sharing an artefact with policies allowing a decreasing number of further sharings. A general set of policies, which allows n delegations is shown in the following, where p_1 is assigned to the original artefact (for $i \in [1, n-1]$):

$$\varphi_{\mathsf{p_i}}[x] \leftarrow \mathsf{Usage}(x) \vee (\mathsf{Sharing}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{p_{i+1}})).$$

$$\varphi_{\mathsf{p_n}}[x] \leftarrow \mathsf{Usage}(x).$$

Hierarchies and other Domain Knowledge are widespread when modelling privacy policies, e.g., the usage of a birthday implies usage of personal information (artefact hierarchy); marketing is a commercial purpose (purpose hierarchy); or a selling action has per definition a commercial purpose (domain knowledge). We can model such knowledge in the form of Datalog rules, e.g.,

$$\mathsf{Commercial}(u) \leftarrow \mathsf{MarketingPurpose}(u).$$
$$\mathsf{Commercial}(u) \leftarrow \mathsf{hasPurpose}(x, u) \wedge \mathsf{Selling}(x).$$

Anonymisation / Pseudonymisation
The extension of rights after an artefact is anonymised or pseudonymised is modelled in a similar way as rights delegation. A corresponding policy condition allows the transformation action and restricts the generated artefact's policy accordingly. Consider for example a policy that allows arbitrary usage (but no sharing) after anonymisation:

$$\varphi_{\mathsf{p_o}}[x] \leftarrow \mathsf{Anonymisation}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{p_a}).$$
$$\varphi_{\mathsf{p_a}}[x] \leftarrow \mathsf{Usage}(x).$$

Obligations can be specified using the wasTriggeredBy predicate. As example take a policy that has the obligation to notify an artefact's owner whenever it is used:

$$\varphi_{\mathsf{p_1}}[x] \leftarrow \mathsf{Usage}(x) \wedge \mathsf{wasTriggeredBy}(x, n) \wedge \mathsf{Notification}(n).$$

Time Spans are a common condition on obligations. Without a time restriction requiring a notification or deletion is not very valuable, because the obliged agent can postpone the fulfillment forever. The following policy requires that a stored artefact is deleted before the end of the year 2012:

$$\varphi_{\mathsf{p}}[x] \leftarrow \mathsf{Storing}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{wasTriggeredBy}(d, x) \wedge \mathsf{Deletion}(d) \wedge \mathsf{performedAt}(d, t) \wedge t \leq \text{"2012-12-31"}.$$

Absolute time restrictions are used, as the policies refer to concrete artefacts that are passed to concrete providers. To see the advantage of absolute times, consider a policy with a relative time restriction that would allow storage, if it triggers a deletion one year after the storage, and allows the same terms for the stored artefact. The deletion obligation could be circumvented by always storing a new copy of the artefact, which again can be kept for one further year.

5 Scenario Realisation

In the following, we present parts of the policies that Carol assigns to her data. First is the policy pa which Carol gives to energy consumption data, she gives to her energy producer PowerBob:

$$\varphi_{\mathsf{pa}}[x] \leftarrow \varphi_{\mathsf{pu}}[x] \vee \big(\mathsf{Storing}(x) \wedge \mathsf{performedBy}(x, \mathsf{PowerBob}) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{pu}) \wedge \mathsf{wasTriggeredBy}(d, x) \wedge \mathsf{Deletion}(d) \wedge \mathsf{performedAt}(d, t) \wedge t \leq \mathsf{now}() + 1\mathsf{y}\big)$$
$$\varphi_{\mathsf{pu}}[x] \leftarrow \mathsf{Usage}(x) \wedge \mathsf{performedBy}(x, \mathsf{PowerBob}) \wedge \mathsf{hasPurpose}(x, u) \wedge \mathsf{Billing}(u)$$

The policy either allows the use by PowerBob for billing purposes (reusing policy pu) or the storage for one year, where the stored artefact can be used under a policy contained in pu. The policy uses the *time span* and the *obligation* patterns. The following is a description of PowerBob storing the consumption data cda, which has the policy pa, including a scheduled deletion:

$$\mathsf{Storing(s)} \wedge \mathsf{used(s, cda)} \wedge \mathsf{performedBy(s, PowerBob)} \wedge \mathsf{wasGenBy(sa, s)} \wedge \mathsf{hasPolicy(sa, pb)} \wedge \mathsf{wasTriggeredBy(d, s)} \wedge \mathsf{Deletion(d)} \wedge \mathsf{performedAt(d, 31.12.2012)}.$$
$$\varphi_{\mathsf{pb}}[x] \leftarrow \mathsf{Usage}(x) \wedge \mathsf{performedBy}(x, \mathsf{PowerBob}) \wedge \mathsf{hasPurpose}(x, u) \wedge \mathsf{Billing}(u) \wedge \mathsf{performedAt}(x, t) \wedge t \leq 31.12.2012.$$

PowerBob assigns to the stored artefact sa the policy pb, which is more restrictive than (contained in) the required policy pu specified by Alice, because it additionally includes a time restriction. The other conditons of pa are also fulfilled, thus the storage s is classified as compliance. Now, if PowerBob wants to use the stored artefact sa, it is sufficient to check compliance to policy pb.

Carol assigns the policy po to the data she shares with the energy optimiser OptiPrime: We leave out the performedBy restrictions for space reasons:

$$\varphi_{\mathsf{po}}[x] \leftarrow \big(\mathsf{Usage}(x) \wedge \mathsf{hasPurpose}(x, u) \wedge \mathsf{Consulting}(u)\big) \vee \big(\mathsf{Anonymisation}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{ps})\big)$$
$$\varphi_{\mathsf{ps}}[x] \leftarrow \big(\mathsf{Usage}(x) \wedge \mathsf{hasPurpose}(x, u) \wedge \mathsf{Statistical}(u)\big) \vee \big(\mathsf{Sharing}(x) \wedge \mathsf{wasGenBy}(a, x) \wedge \mathsf{hasPolicy}(a, p) \wedge \mathsf{containedIn}(p, \mathsf{ps})\big).$$

The po policy uses the *anonymisation* pattern and the ps policy uses the *sharing* pattern.

Alice can employ the *attribute-based usage restrictions* and *domain knowledge* patterns to formulate a policy that allows for data artefacts containing information about a appliance that the manufacturer of the appliance can use the artefact for service purposes.

6 Related Work

A systematic treatment of usage control with their UCON model is given in (Park and Sandhu 2004). The model focuses on conditions on allowed usages and required obligations, when protected data is released into a system. The system is regarded as a closed environment in which all data usages take place (no sharing of the data with further parties is considered) and can thus be considered as a system-level

approach. P3P is a W3C standard for privacy policies in the web[3]. It is a XML language without formal semantics, which is supplied by various later publications, e.g. (Yu, Li, and Antón 2004). In (Speiser 2012), we discussed how P3P can be translated into our policy approach according to the semantics given in (Yu, Li, and Antón 2004).

The history-aware PAPEL policy language can be used for privacy policies (Ringelstein and Staab 2010). To illustrate the difference between history-awareness and our approach, consider a policy presented in (Ringelstein and Staab 2010): a patient record can only be shared after it is deidentified. PAPEL models the sharing action as compliant, if it was preceded by an deidentification. With our approach, the policy of the health record specifies that a compliant deidentification action can allow a policy for the produced artefact, which permits sharing; the policy engine evaluating the compliance of the sharing action does not have to know about the history of the artefact.

The SecPAL authorization language supports rights delegation (Becker, Fournet, and Gordon 2010). SecPAL policies are not attached to data artefacts. The S4P language for privacy policies presented in (Becker, Malkis, and Bussard 2010) also evaluates policies on a system level, which requires history-awareness: e.g. the compliance of an action, allowed by a rights delegation requires the system to evaluate the rightfulness of the delegation.

The data-purpose algebra allows the modelling of usage restrictions of data and the transformation of the restrictions when data is processed (Hanson et al. 2007). In their approach, a data item is associated with allowed usages and depending on the process performed on a data item a function is defined that transforms the allowed usages. We share the general idea of having a set of allowed usages for data artefacts, which can change depending on the process performed on the artefact. In our approach, transformation functions are not defined directly but restricted by containedIn conditions on policies of produced artefacts. The data-purpose algebra is particularly suitable for expressing transformations which hold for all data items processed by a specific system. In contrast, our approach integrates the transformation functions, i.e., target policy restrictions, into policies of data artefacts, which means that every artefact can define its own transformation functions.

7 Conclusions

The smart energy grid is a representative example of smart city technologies which have a common requirement: large-scale management, integration, and processing of information that is collected, controlled, and stored in a decentralised way by heterogeneous entities. The Linked Data principles embrace semantic technologies, which facilitate the information exchange and integration using open web standards. We extend this technology stack with a policy language for expressing privacy requirements of data owners. The language expresses policies as local restrictions on isolated information uses. The locality of policies allows compliance checking without the need for a complete view of all information-processing systems. Policies in our approach are sticky, i.e., attached to the protected information, enabling the exchange of policies together with data.

[3] http://www.w3.org/TR/P3P/

Acknowledgments

This work was supported by the EU FP7 (PlanetData, Grant 257641) and the Karlsruhe Service Research Institute.

References

Becker, M. Y.; Fournet, C.; and Gordon, A. D. 2010. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security* 18:619–665.

Becker, M. Y.; Malkis, A.; and Bussard, L. 2010. A Practical Generic Privacy Language. In *International Conference on Information Systems Security (ICISS)*.

Bonatti, P. A., and Mogavero, F. 2008. Comparing Rule-Based Policies. In *IEEE Workshop on Policies for Distributed Systems and Networks (POLICY08)*, 11–18.

Bonatti, P. A.; De Coi, J. L.; Olmedilla, D.; and Sauro, L. 2010. A Rule-based Trust Negotiation System. *IEEE Trans. on Knowledge and Data Engineering* 22:1507–1520.

Carroll, J. J.; Bizer, C.; Hayes, P.; and Stickler, P. 2005. Named graphs, provenance and trust. In *International Conference on World Wide Web (WWW)*, 613–622.

Hanson, C.; Berners-Lee, T.; Kagal, L.; Sussman, G. J.; and Weitzner, D. 2007. Data-Purpose Algebra: Modeling Data Usage Policies. In *Workshop on Policies for Distributed Systems and Networks (POLICY'07)*.

Karjoth, G.; Schunter, M.; and Waidner, M. 2003. Platform for enterprise privacy practices: privacy-enabled management of customer data. In *International Conference on Privacy Enhancing Technologies (PETS)*, 69–84.

Krötzsch, M., and Speiser, S. 2011. ShareAlike your data: Self-referential usage policies for the Semantic Web. In *International Semantic Web Conference (ISWC)*, 354–369.

Moreau, L.; Clifford, B.; Freire, J.; Futrelle, J.; Gil, Y.; Groth, P.; Kwasnikowska, N.; Miles, S.; Missier, P.; Myers, J.; Plale, B.; Simmhan, Y.; Stephan, E.; and den Bussche, J. V. 2010. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*.

Park, J., and Sandhu, R. S. 2004. The UCON_{ABC} usage control model. *ACM Trans. Inf. Syst. Secur.* 7(1):128–174.

Ringelstein, C., and Staab, S. 2010. PAPEL: A language and model for provenance-aware policy definition and execution. In *International Conference on Business Process Management (BPM)*, 195–210.

Speiser, S. 2012. A Data-centric View on Expressing Privacy Policies. Technical Report 3023, Institute AIFB, Karlsruhe Institute of Technology. Available online at http://www.aifb.kit.edu/web/Techreport3023.

Wagner, A.; Speiser, S.; Raabe, O.; and Harth, A. 2010. Linked data for a privacy-aware smart grid. In *GI Jahrestagung*, 449–454.

Yu, T.; Li, N.; and Antón, A. I. 2004. A formal semantics for P3P. In *Workshop on Secure Web Services (SWS)*.