L2TAP+SCIP: An Audit-based Privacy Framework Leveraging Linked Data

Reza Samavi, Mariano P. Consens
Information Engineering, MIE, University of Toronto
{samavi,consens}@mie.utoronto.ca

Abstract — **We describe a framework designed to facilitate privacy auditing while accommodating a variety of privacy scenarios and policies that involve multiple participants. Our proposal is based on two ontologies, L2TAP and SCIP, designed for deployment in a Linked Data environment. L2TAP provides provenance enabled logging of events. SCIP synthesizes contextual integrity concepts and enables query based solutions for two important privacy processes (compliance and obligation derivation). We include an experimental validation of the scalability of our approach.**

Keywords-- Privacy, linked data, audit log, compliance queries, sparql

I. INTRODUCTION

As individuals are increasingly benefiting from the use of online services, there are growing concerns about the treatment of personal information (broadly referred to as privacy concerns). Society's ongoing response to these concerns gives rise to privacy policies expressed in legislation, regulation, and agreements. These policies steer organizations that collect, use, and share personal data towards respecting the privacy of the data subjects. Ensuring compliance with a variety of privacy policies is an increasingly complex task involving multiple participants including watchdogs and enforcement agencies that prosecute privacy violators. Therefore, technical solutions that facilitate the implementation of compliance offer value to all the organizations that deal with privacy concerns [1].

This paper introduces a framework designed to facilitate the auditing tasks of the multiple participants dealing with privacy concerns. While privacy auditing is commonly employed in industry, the infrastructure to support it is ad-hoc, and not supported by widely adopted technology standards. In contrast, there have been multiple proposals in the area of access control, whether to specify fine grained enforceable policies (e.g., XACML [2], EPAL [3], P-RBAC [4]), or coarse grained high-level privacy declarations (e.g., P3P [5]). Also, auditing has received less attention from the research community than areas such as access control or statistical privacy. However, there are solid theoretical foundations for policy auditing over logs [6], [7]. Furthermore, auditing is complimentary to other policy mechanisms and it can take place proactively or retroactively, helping with enforcement and compliance, and/or identifying participants that commit privacy violations (for whom the threat of prosecution is a deterrent).

Our goal is to develop a framework that can provide a technical foundation for a principled approach to privacy auditing, while maximizing the flexibility in three key areas; participants, policies, and processes. Our proposal is based on two ontologies designed for deployment in a Linked Data [8] environment. The paper describes query based solutions (with an experimental validation) for two important privacy processes. The solutions presented leverage our framework and highlight its applicability and practical benefits.

Our first proposed ontology, *L2TAP* (Linked Data Log to Transparency, Accountability and Privacy) allows participants to log (in RDF) privacy related events such as changes to the policies, as well as information access requests and activities. All the events in an L2TAP log are identified by web accessible URIs, which can be dereferenced by other participants (likely after authentication and over a secure https channel), therefore simplifying the support of *transparency*. The events logged include time stamped information obtained from the participants involved that is also linked data (potentially from multiple ontologies). The provenance information (who, when) in an L2TAP log facilitates supporting *accountability* among the participants involved. The mixing and matching of ontologies in the log is the basis for *flexibility*. Arbitrary processes can log privacy related events and/or analyze them. Participants with heterogeneous environments can be supported, since the linked data that they contribute to the log can be the result of a thin transcoding from legacy information formats. Analogously, multiple privacy policy languages can be supported (e.g., even without transcoding a string property can store the XML encoding of an XACML or an EPAL policy).

While L2TAP provides an arbitrarily flexible foundation for log-based privacy processing, our second proposed ontology, *SCIP* (Simple Contextual Integrity Privacy), enables query-based implementations of audit related processes, while preserving considerable amounts of flexibility. SCIP is a novel synthesis (in the form of a concrete ontology) of concepts inspired by the Contextual Integrity [9] perspective (where privacy is seen as the right to appropriate flows of personal information in a given social context). SCIP provides a simple (yet sufficiently rich) target for mapping the key concepts in L2TAP logs expressed using other privacy related ontologies or schemas. Our work shows how using SCIP (either directly or via mappings) to express event information in L2TAP logs enables scalable implementations of privacy processes using SPARQL queries plus limited RDFS reasoning support.

The paper structure and contributions are as follows. The next section presents an overview of our novel audit-based privacy framework (based on the L2TAP and SCIP ontologies available at http://l2tap.org) using a hypothetical scenario. In

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250607

Section 3 we show a query-based solution for deriving all the obligations associated with an access request. Section 4 describes a compliance checking process implemented via SPARQL queries: given an access request check the fulfillment of its associated obligations. Section 5 reports an experimental validation of the scalability of the compliance queries described earlier. Related work is discussed in Section 6, while Section 7 concludes the paper.

II. Overview of the Privacy Framework

We use a scenario to motivate our audit-based privacy framework and to introduce the L2TAP log ontology in Subsection A. We describe the SCIP ontology next; privacy preferences in Subsection B, and access requests and obligations in Subsection C.

A. L2TAP Logs

To picture a privacy-sensitive process consider Alice, who has recently been diagnosed with diabetes, and who joins a diabetes social network (dphr.org) to receive information and emotional support. Alice uploads the history of her daily blood glucose level at https://dphr.org/users/alice/A1c (using the personal health record functionality of the dphr.org network). Our scenario is concerned with respecting Alice's privacy preferences when her blood glucose level information is used by Bob, a physician and a researcher who is also a member of the dphr.org network.

The L2TAP ontology is used to log provenance assertions (using the recently proposed RDF provenance data model [10]) as shown in Fig. 1, where Alice logs her preferences encoded as triples in a named graph g1. We encode the graph g1 as an instance of prov:Entity (line 2) to show that this is what we want to provide provenance for. According to the RDF provenance model, provenance assertions for an entity, including *when* and by *whom* an entity was changed, are associated with the *activity* that introduces the change. For graph g1, the activity is captured by prov:Activity in line 4, while *who* logged the graph is captured in line 5 using prov:wasAssociatedWith. The time interval for *when* the graph is being logged is encoded in line 6 and 7 using prov:startedAtTime and prov:endedAtTime.

The triples that construct g1 include the triple to mint the URI of the privacy preference in the logger's domain (line 11) and the ontology describing the semantics of the privacy preferences triples (line 12). Other triples in g1 use the plugged ontology pointed in line 12 to express Alice privacy preferences (as described in Fig. 3). L2TAP is not aware of the semantics of this pluggable ontology but encode the triples described by the ontology. The SCIP ontology described in the next subsections is one instance of such pluggable ontologies.

```
01  @prefix prov: <http://www.w3.org/ns/prov#>.
02  <http://dphrLogger.org/g1> a prov:Entity;
03   prov:wasGeneretaedBy <http://dphrLogger.org/logEntry/idg1>.
04  <http://dphrLogger.org/logEntry/idg1>  a prov:Activity;
05  prov:wasAssociatedWith <http://dphrLogger.org/users/alice>;
06  prov:startedAtTime [time:inXSDDateTime "2011-11-05T18:00:00"];
07  prov:endedAtTime [time:inXSDDateTime "2011-11-05T18:00:00"].
08
09  <http://dphr.logger.org/g1> = {
10   <http://logger.org/userspp/alice_pp1>  a l2tap:PrivacyPreference;
11  owl:sameAs <http://dphr.org/memberspp/alice_pp1>;
12   l2tap:prefernceOntology < http://ontology.org/scip>; ...continued
```

Figure 1. Provenance assertions in L2TAP

B. SCIP Privacy Preferences

The SCIP ontology (as shown in Fig. 2) has concepts for the major privacy-related events that are logged, such as expressing privacy preferences, submitting access requests, performing obligations, and for access activities. In every privacy process multiple participants interact (Alice, Bob, as well as the social network dphr.org are the participants in our motivating scenario). The class scip:Participant captures the main actors in an information flow (i.e. data subject, data sender, and data requestor), as well as other actors specific to the privacy by audit log mechanism, such as the logger that is responsible to log, or obligation performers and witnesses. Obligations that Bob must fulfill when accessing Alice's data are captured by scip:Obligation while the actual event of Bob accessing Alice's data is captured by an scip:AccessActivity.

Going back to our motivating scenario, assume that Alice has expressed her privacy concerns as (i) a dphr.org user may access my daily glucose level for research purposes, provided the user is a university researcher, (ii) the user must obtain my consent prior to accessing this information, (iii) the user should send me a report about the on-going research after 180 days, (iv) I should be notified of any publications of the research results where my data is being used (unless my data is being used anonymously). SCIP captures these preferences using the scip:PrivacyPreference concept. Participants can use properties defined for this class to express applicable conditions and norms that must be respected when the information of data subjects are used. For example, scip:purpose expresses the intended purpose of usage while scip:obligation describes the obligations that must be performed when the data is used. SCIP uses other classes as well (see Fig. 2); the scip:DataItem class describes the attribute hierarchies for data items; the scip:Role class describes the roles of participants; scip:Purpose describes applicable purposes; scip:Privilege describes privacy privileges that can be granted; and scip:ObligationTemplate describes obligation templates.

Triples in Fig. 3 show how Alice uses the preceding properties and classes to encode her preferences (i) to (iii) (note that triples in this figure are the continuation of Fig. 1). Line 13 describes by whom the privacy preferences have been expressed. The participant who expresses the preference could be the data subject herself (as in our example), or it could be a service provider (dphr.org) defining its internal privacy policies(and they could originate from a legal body that set norms for information flows, such as HIPPA [11]). The triples in lines 14 and 15 describe the time interval for validity of the

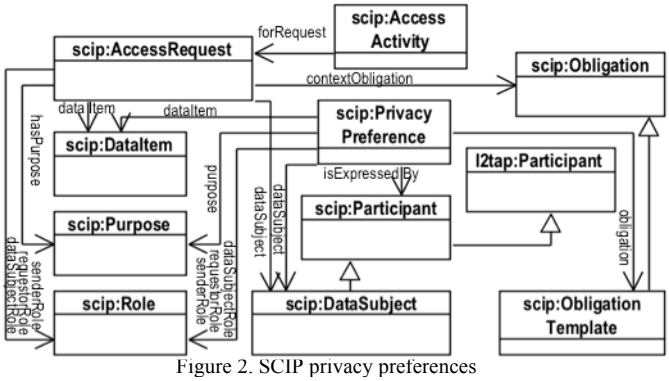

Figure 2. SCIP privacy preferences

```
13  scip:expressedBy <http://logger.org/users/alice>;
14  scip:hasValidity [time:hasBegining "2011-11-05T19:32:52Z";
15  time:hasEnd "2012-11-05T19:32:52Z"];
16  scip:dataItem <https://dphr.org/users/alice/A1c>;
17  scip:requestorRole <http://dphr.org/roles/university_researcher>;
18  scip:senderRole <http://dphr.org/roles/network_provider>;
19  scip:dataSubjectRole <http://dphr.org/roles/any>;
20  scip:purpose <http://dphr.org/purposes/med_research>;
21  scip:privacyPrivilege <http://dphr.org/privileges/use>;
22  scip:obligation <http://dphr.org/obs/ob1>;
23  scip:obligation <http://dphr.org/obs/ob2>
24  scip: propositionalExpression <http://dphr.org/exp/phy1> .
25  <http://dphr.org/obs/ob1>  a scip:Obligation;
26  scip:performAction <http://ontology.org/actions/obtain_consent>.
27  <http://dphr.org/obs/ob2> a scip:Obligation;
28  scip:occurrenceGap "180";
29  scip:performanceDuration "1";
30  scip:performAction <http://ontology.org/actions/notification >. }
```

Figure 3. Alice privacy preferences

privacy preferences, the data item is described in line 16, the acceptable roles for the data requestor, data sender, and data subject are described in lines 17-19,and the acceptable purpose is described in line 20. The privacy privilege that will be granted if the conditions in the privacy preferences are met is described in line 21.

In SCIP, roles are described by a lattice using rdfs:subClassOf. The superclass role http://dphr.org/roles/any is used to describe all possible data subject roles in line 19. Other subclasses can be used for specialized roles such as patients, researchers, and physicians. The researcher role can be further specialized to academic researcher, and so on. SCIP uses RDFS inheritance in a similar way to describe hierarchies for data items, purposes, and privacy privileges.

C. SCIP Access Requests and Obligations

When an access request is initiated by a data requestor (Bob in our scenario), SCIP describes the access request using properties of the scip:AccessRequest class as shown in Fig. 4. The property scip:dataRequestor captures the URIs of the data requestor, scip:dataSender captures who should send the data while scip:dataSubject captures whose data has been requested, scip:hasValidity (not shown in the figure) describes the time interval for which a request is valid. Similarly to the privacy preference schema, the access request schema describes the data item and the privacy privilege that have been requested,

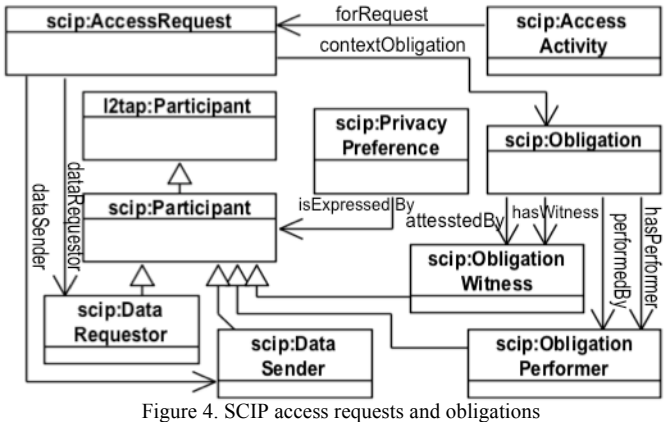

Figure 4. SCIP access requests and obligations

the purpose for accessing data, and the roles of the participants. These elements in access requests and privacy preferences are used to identify the context that a privacy preference applies to and the context that an access request belongs to, thus providing criteria to find matches and deriving obligations as described later on in Section 3.

Associated with each privacy preference are a set of obligations (line 22, 23). An obligation is an action that has to be performed either before or after the occurrence of an access activity [12]. When multiple obligations arise from an access request, a logical formula describes how the satisfaction of these obligations relates to the overall compliance of the access request against the privacy preference (line 24). A couple of properties of scip:Obligation describe temporal constraints for an obligation; scip:occurrenceGap (line 29) encodes the time interval for performing the obligation (before or after the access activity), while scip:performanceDuration (line 30) encodes the time interval required to perform the obligation. The actions to be performed are described using scip:perfromAction (line 18). The obligation class has other properties to describe state conditions for triggering an obligation, as well as who should perform the obligation and who can witness its performance and/or violation (Fig. 4). Note that obligations expressed in privacy preferences are templates for obligations that are instantiated when an access request is logged. Obligations and obligation templates will be further described in Section 3 and 4.

Finally, SCIP has a class scip:AccessActivity with properties that describe the access request for which an access activity occurs, when the activity occurs, and who performs the access activity.

III. OBLIGATION DERIVATION

In this section we describe how we can construct a SPARQL query that derives all applicable obligations arising from a data subject's privacy preferences. The process consists of the following three tasks: (i) finding matches between an access request and all the applicable privacy preferences, (ii) logging the set of obligations generated from the privacy preferences, and (iii) logging for each preference a logical expression, that describes how the individual satisfaction of each obligation contributes to the overall compliance of the originally matched access request.

A. Matching Access Requests and Privacy Preferences

Matching between privacy preferences and access requests is found by comparing the properties of the *contexts* for the instances of the privacy preferences and the access request. A context consists of the following properties, (i) the *roles* of the three main participants (*data requestor*, *data sender*, and *data subject*) in an information flow, (ii) the *purpose* for which the data subject's private information will be used, (iii) the type of *privacy privilege* will be granted (one of use, collect, or disclose), and (iv) the *data item* that is going to be accessed. In our running example, as shown in Fig. 3, Alice expresses applicable context to the privacy preferences using the acceptable *role* for the data sender (a network provider, line 18); the role for the data requestor (a university researcher, line 17); the acceptable *purpose*, (medical research, line 20); the applicable privacy privilege (use, line 21); the applicable *data*

item (diabetic data, line16). In SCIP, we are matching the context described in Fig. 3 using the SPARQL query provided in Fig. 5.

The properties of the context for an access request are included in lines 6-12 of the query in Fig. 5. Similarly, the properties of the contexts associated with privacy preferences appear in lines 14-20. Note that the query in Fig. 5 is parameterized with the URI of an access request. We used @ to distinguish between a parameter and a variable. The @request parameter will be substituted with the access request URI in the execution time. The matching in the query consists of checking the following items.

First the query checks if the data subject of an access request matches to the data subject to whom the privacy preferences is applicable, by using a common variable (?req_dsu) for the data subject of an access request (line 6) and the privacy preferences (line 14). The matching for the properties of the contexts will be based on the sub-classing. To match the *data items*, line 11 in the query includes the access request graph defining the requested data item, using scip:dataItem. Similarly, line 20 joins triples describing to which data items the privacy preferences applies. Using the data item hierarchy, we use rdfs:subClassOf property in the FILTER condition in line 23 to make sure that the requested data item is a subset of the data items the privacy preferences applies to. Since *purpose*, *privacy privilege*, and *roles* are also expressed in SCIP using partial order, the conjunctive FILTER conditions in lines 23-28 check that all these items expressed in the access request graph are in a lower order compared to the corresponding items in the privacy preferences graph.

When all conditions are met, the outcome of the joined graph patterns in the WHERE clause will be the applicable

```
01  CONSTRUCT {?ob a scip:Obligation .
02  @request   scip:contextObligation ?ob;
03              scip:propositionalExpression ?phi .}
04  WHERE {
05  @request  a scip:AccessRequest;
06              scip: dataSubject ?req_dsu;
07              scip:requestorRole ?req_drerole,
08              scip:senderRole ?req_dserole,
09              scip:dataSubjectRole ?req_dsurole,
10              scip:purpose ?req_pur;
11              scip:dataItem ?req_di;
12              scip:privacyPrivilage ?req_priv .
13  ?preference a scip:PrivacyPreference;
14              scip:isExpressedBy ?req_dsu;
15              scip:purpose ?pref_pur;
16              scip:requestorRole ?pref_drrole;
17              scip:senderRole ?pref_dserole;
18              scip:dataSubjectRole ?pref_dsurole;
19              scip:privacyPrivilege ?pref_priv;
20              scip:dataItem ?pref_di ;
21              scip:obligation ?ob ;
22              scip:propositionalExpression ?phi .
23  FILTER ((?req_di rdfs:subClassOf ?pref_di) &&
24  (?req_priv rdfs:subClassOf ?pref_priv) &&
25  (?req_pur rdfs:subClassOf ?pref_pur) &&
26  (?req_dsurole rdfs:subClassOf ?pref_dsurole) &&
27  (?req_drrole rdfs:subClassOf ?pref_drrole) &&
28  (?req_dserole rdfs:subClassOf ?pref_dserole))  }
```

Figure 5. Query for obligation derivation

privacy preferences for a given access request @request. It is possible that the outcome to be an empty set, meaning that no applicable privacy preferences are found for a given access request. We will discuss this in Subsection 4.C. If there is a non-empty solution for the match, the obligations will be extracted as discussed below.

B. Generating Obligations

Obligations are described as part of the data subject's privacy preferences using the SCIP ontology. When the match query finds the applicable privacy preferences, all the obligations associated with the found privacy preferences will be copied as obligations for the given access request. In other words, obligations linked to the privacy preferences are templates for the instances of the obligations for an access request. The query described in Fig. 5, joins together the matching privacy preference graph with the triple patterns of obligation templates using scip:obligation (line 21). All obligations (?ob) which are joined to this graph then will be used by the SPARQL CONSTRUCT (line 1-2) to inserted into the RDF graph, this time using scip:contextObligation property, linking a given access request (@request) to its obligations. Thus, the query structure in Fig. 5 not only provides solutions for the matching problem, but also generates all the obligations relevant to an access request.

Back to our running example, when the query in Fig. 5 is executed, four obligations will be generated and registered in the log. All these obligations (e.g. obtain_consent) are exact copy of the obligations described in the Alice's privacy preferences as described in Fig. 3. However, there could be obligations that cannot be simply copied due to some contextual parameters. For example, an obligation may require a data requestor sends a notification to the parents of a data subject if the data subject is under 16, but to her parents and herself if she is 16-18, and to herself if she is over 18. Therefore, generating the instances of obligations from the obligation template requires checking the date of the access request with the birth date of the data subject and then passing the parameter to the obligation. Constructing parameterized obligations has not been currently covered; nevertheless it is the future extension.

C. Propositional Combination of Obligations

For every access request there could be multiple obligations derived from the applicable privacy preferences. In the running example, Alice's preference (iv) introduces two obligations (either anonymize the data before using or inform the relevant publications). Bob can perform only one of them and still be in compliance. Other Alice preferences, however, do not have such a nature and Bob must perform all of them. The judgment on privacy compliance will be different depending on how multiple obligations are related to each other.

To address the problem of the combinatory effects of multiple obligations, we introduce φ, a propositional first order formula that describes how the individual satisfaction of each obligation contributes to the overall compliance of the access request. In the current SCIP proposal, we assume that the same participant who registers the privacy preferences expresses the formula φ. In our running example Alice, the data subject, expresses φ as part of her privacy preferences in line 24 in Fig. 3. In the process of obligation derivation, using the query in

Fig. 5, φ is also derived from the preferences along with the obligations and is linked to the access request using scip:propositionalExpression (line 3, Fig. 5). We discuss the logical structure of φ and its expressive power in more details in the Subsection 4.B.

IV. Compliance Checking

An important objective of the audit-log based privacy mechanism is to identify, in any given point in time, if an access request is in compliance with the applicable privacy preferences and policies. Since compliance of an access request is decided based on the status of its linked obligations, in this section we describe how to identify a *fulfilled* and a *pending* obligation using SPARQL queries. In Subsection B we describe how the logical formula φ can be evaluated in a series of queries that determine the states of *compliance* for an access request. The boundary situations for compliance is discussed in Subsection C. Due to the temporal properties of obligations, expressing the time intervals and reasoning about temporal constraints in SCIP, are discussed in Subsection D.

A. Fulfilled and Pending Obligations

The acceptable time interval for performing an obligation depends on the time that the information has been accessed (used) by the data requestor [12]. Therefore, the state of an obligation cannot be evaluated without the temporal reasoning about the occurrence time of the access activity and the obligation. In our running example, the data requestor is obliged to send a notification 180 days after the access to the information. The obligation may take 1 day to be performed. If we assume that the access activity occurs in day 0, the acceptable time interval for the obligation is (180, 181).

We state ob_i is a *fulfilled* obligation if it is performed within its acceptable time interval. In other words, the concept of obligation fulfillment is related not only to the binary evaluation of whether it is performed or not, but also to the temporal properties of its performance interval. Due to this time dependency, an obligation can be un-fulfilled but not violated, simply because it might be the case that the time for its performance has not been yet emerged. Thus, we state ob_i is a *pending* obligation, if in a point in time $t_{current}$ (the time that we observe the state of ob_i) it has not been performed and $t_{current} \leq t_{so}$, where t_{so} is the beginning of the acceptable time interval for ob_i. In SCIP all other states, which an obligation may go through and are not *fulfilled* or *pending* are considered *violated* (in some privacy literature (e.g. [13]) an invalid obligation is differentiated from a violated obligation, but in this proposal we assume all registered obligations are valid).

The process of identifying the state of an obligation (*fulfilled*, *pending*, *violated*) is implemented using multiple SPARQL ASK queries. The query shown in Fig. 6 returns *true* if an obligation (@obligation) is a *pending* obligation. This query has two parameters, the URI of the obligation (@obligation), and the current time (@currentTime). Line 4-6 of the query returns the triple patterns of a given obligation (@obligation). This pattern consists of the relative occurrence gap of the obligation from the occurrence time of an access activity, described by scip:occurrenceGap, and the time duration that requires for the obligation to be performed. If we limit the query to the triple patterns in the WHERE clause, the ASK query will return *true* for any registered obligation in the log. However, to find out if an obligation is *pending*, we have to find first if the obligation has been performed or not and second whether the access activity has been occurred and when. We use OPTIONAL graph patterns in lines 7-8 to left join the graph patterns of the occurrence of an access activity for the access request that the @obligation is associated with (line 3). Line 9 in the query optionally left joins the triple of performing the obligation. Similarly, line 10 adds to the graph the triple indicating if an obligation witness testifies the violation of the obligation. Note that an obligation can be attested by a *privacy witness* as a *violated* obligation regardless of whether it has been performed or not.

The query in Fig. 6 uses FILTER with multiple conditions (line 11-14). The first conjunctive condition states that for an obligation to be *pending*, there should not exist a triple stating its performer agent (!bound(?performerAgent)) meaning that the obligation has not been performed. The condition in lines 12-13 includes a nested disjunctive expression using the parameter @currentTime. This condition states that either an access activity has not been occurred (!bound (?accessTime)) or it has been occurred but the condition $t_{current} \leq t_{so}$ holds, meaning that there is still time for the obligation to be performed given the gap between the occurrence of the access activity and the time required for the obligation to be performed. So, in this case checking whether an obligation is *pending* or *violated* is a matter of temporal reasoning of the start and end of the access activity occurrence. We compute the acceptable time interval of an obligation based on a simple Z integer time model described in Subsection 4.D. The last conjunctive condition (line 14) states that for an obligation to be pending, it should not be the case that its violation is being attested.

The ASK query returning *true* or *false* for a *fulfilled* obligation will have a similar structure to the query shown in Fig.6 where the conditions in lines 11-14 will change to check the access activity occurred; the obligation has been performed; the time of performing the obligation is in the acceptable time interval; and the violation has not been attested.

B. Access Request Compliance

If the number of obligations arising from an access request is only one, the state of an access request can be directly determined form the state of an obligation (i.e. if the single obligation is *fulfilled* then the access request is also in compliance). However, if there is more than one obligation, the

```
01  ASK {
02  SELECT  DISTINCT @obligation
03  WHERE   { ?request scip:contextObligation @obligation.
04  @obligation rdf:type scip:Obligation.
05  @obligation scip:occurrenceGap ?occGap.
06  @obligation scip:performanceDuration ?pD.
07  OPTIONAL {?accessActivity scip:forRequest ?request} .
08  OPTIONAL {?accessActivity scip:accessedTime ?accessTime} .
09  OPTIONAL {?pendingObligation scip:performedBy ?performerAgent} .
10  OPTIONAL {?witness scip:attestsViolation ?pendingObligation} .
11  FILTER (!bound(?performerAgent) &&
12  ((!bound (?accessTime)) || (bound (?accessTime) &&
13  (xsd:integer(@currentTime) <= (xsd:integer(?accessTime) +
14  xsd:integer(?occGap) + xsd:integer (?pD))) && (!bound(?witness)) ) ) }
```

Figure 6. SPARQL query returning True or False for a pending obligation

process of determining the access request compliance is implemented in two steps, (i) assigning the truth-value to every individual obligation in a given point in time t, and (ii) evaluating the logical formula φ that describes the combinatory effect of multiple obligations. In the preceding subsection we demonstrated that we could determine the state of an individual obligation using SPARQL Ask queries. Therefore, the remaining task is substituting the propositional variables in φ with *true* or *false* values obtained from the firs step.

While determining the state of an obligation requires reasoning about some temporal constraints, the formula φ is free of time and refers to an access request in a given point in time t. φ as a template is built using any combination of (ob_1, ..., ob_n, ∧, ∨, ¬, (,)), where {ob_1, ..., ob_n} is a set of propositional variables representing obligations, and can be arbitrarily combined with logical *and*, *or*, and *negation* operators with any level of nested brackets. Thus, φ has a flexible and generic structure that can support any possible combination of obligations. φ_t is an instantiation of φ in a given point in time t. Since φ is a first order formula, it will be always evaluated to *true* or *false*, as long as the truth-values of its propositional variables (i.e. obligations) are determined.

In truth-value assignment to the φ's variables, a *fulfilled* obligation always receives *true* assignment and a *violated* obligation always *false*. However, the assignment of the truth-value to a *pending* obligation is not as straightforward as other two states and depends on the definition of the access request compliance. If we define the concept of compliance from the point of view of a privacy auditor (who checks whether an access request can be considered to be discharged), a pending obligation will be substituted with *false*, implying that the access request still needs to be inspected. We call this treatment of φ as φ_{tf} to indicate that only *fulfilled* obligations receive *true* assignment, everything else receives *false* assignment. However, if we define the concept of compliance in terms of compliance up to the point t in time, then a *pending* obligation will be assigned *true*. As a result φ will be evaluated to *false* only due to some violated obligations. We call this as φ_{tp} to indicate that *pending* obligations are substituted with *true*, just like *fulfilled* obligations.

Due to the dependency of *pending* obligations to the time of evaluating the state of an obligation ($t_{current}$), the method described above for the truth-value assignment allows us to ask queries about the past and future compliance by assigning different values to $t_{current}$. Furthermore, by having φ_{tf} formula where *pending* obligations are substituted with *false*, we can conclude that if φ_{tf} is evaluated to *true* (in compliance), it will remain *true* forever. However, if φ_{tp} is evaluated to *true*, it means that the access request is in compliance up to the point t, but we do not know if it will stay like this in the point $t + t'$.

When the truth-values are assigned, we need the SPARQL queries that determine the compliance of an access request by evaluating the corresponding φ_{tf} and φ_{tp}. Our goal is to provide answer to the questions such as: which access requests have been discharged? which access requests are in compliance in time t but are not discharged? which access requests are not in compliance in time t?

The query in Fig. 7 lists all access requests, which are not in compliance in time t. In line 4, we assume that formula φ_{tp} is substituted with the truth-values for every obligation. Note that we use the negation of φ_{tp} formula, meaning that it returns an instance of an access request only if the substituted formula φ_{tp} is evaluated to *false*. Using φ_{tp} means that for the condition becomes *true* even if all *pending* obligations are substituted with *true*, φ_{tp} must be still evaluated to *false*. In other words, the only possibility for φ_{tp} to be evaluated to *false* is due to one or more *violated* obligations. Note that before using the SPARQL query in Fig. 7, line 4 needs to be syntactically modified to include a combination of ASK queries (shown in Subsection 4.A) with the logical operators between them.

By changing the FILTER in line 4 we can provide answer to the other compliance queries. For example, the query with FILTER (!(φ_{tf}) && (φ_{tp})) (when substituted with the truth-value for every individual obligation) will return all compliant access requests up to the point in time t. The query with FILTER (φ_{tf}) returns all discharged access requests. The same queries in Fig. 6 and 7 can be used as building blocks to answer other queries such as what are the outstanding obligations of a particular data requestor in general or for a particular access request.

We recognize that there could be cases that the compliance of an access request depends on multiple φs derived from multiple applicable privacy preferences. Our current proposal supports evaluating the formula Φ as the conjunction of multiple formulas φ (i.e. the most restricted case for the combination of multiple sets of obligations). However, more complicated cases of combining multiple sets of obligations, such as when there are conflicts among obligations (as described in [14]), or a subset of obligations defined in the φ formula are linked to an access request, are out of the scope of this paper.

C. Boundary Situations in Access Request Compliance

During the process of obligation derivation and access request compliance checking, we may encounter a situation where no match can be found between the context of an access request and the contexts of privacy preferences. The question in this case is how we should interpret this situation in the access request compliance-checking step. In SCIP, when there is no match for the context of an access request, we add a constant "⊥" conjunctively to the formula φ meaning that the access request can never be in compliance. This approach follows the principles of Privacy By Design [15] to make protection of privacy as the default in the model. In other words, if no privacy preferences have been found, information cannot flow to any data requestor, providing the lower bound for the access request compliance (i.e. never be in compliance). In our running example, if Alice does not define any privacy preferences, her information stays *private* by default.

What if a data subject actually wants to provide access to her information without imposing any obligations or conditions? To express this situation, we revisit Subsection 3.A

```
01  SELECT  DISTINCT ?request
02  WHERE {
03  ?request scip:contextObligation ?obligation.
04  FILTER (!(φtp /substituted with truth-value using ASK queries) }
```

Figure 7. SPARQL query returning non-compliant access requests

in using class inheritance to find a context match. We require all classes that are used to define a context (i.e. role, purpose, data item, and privilege) to have a top superclass any (e.g. http://dphr.org/roles/any for roles). Then, the participant(s) who define privacy preferences (Alice in our example) can express a privacy preference stating that for *any* purpose, *any* privacy privilege, data senders and requestors in *any* roles can communicate *any* data items of a data subject. This privacy preference makes Alice's data item publicly available without any conditions or obligations. When in the process of obligation derivation, we encounter to such a situation (i.e. when there *is* a context match, but without conditions or obligations), we add a constant "T" to the formula φ disjunctively, allowing access request in such a context to be always in compliance (the upper bound).

While in both situations above no obligations are defined, the difference is that the upper bound compliance needs to be *explicitly* expressed by some privacy preferences. Therefore, by supporting the lower and upper bounds compliance with the same formula φ designed for the obligation combination, we are able to express the dichotomy of *private*, *public* contexts (in the classical access control systems [16]), where the default context will always be considered *private*.

D. SCIP Temporal Constraint Model

In the preceding subsections we observed that the state of an obligation could not be determined without some temporal reasoning. The basic time component in the SCIP ontology is the *time interval*. Obligations and access activities occur in different time intervals. The time model used in the query shown in Fig. 6 uses the simple representation for time intervals proposed in [12] (a pair of numbers in Z totally ordered by $\leq$). In this model, time intervals for post-obligations have positive and for pre-obligations they have negative values.

While representing time as an Integer provides a convenient way to reason about temporal constraints, SCIP can be parameterized based on the properties of alternative time models. For example the temporal condition expressed in lines 11-15 can be specified using Allen's interval algebra [17]. The SCIP ontology has two time classes, scip:TimeInterval and scip:Duration that are subclasses of intervals and durations in specific time models (such as the Z time model used previously, or the time model described by the OWL time ontology [18]). While the structure of the compliance queries discussed above does not change, the queries have to be tailored to properly compare time intervals in the selected ontology. For example, if the axiomatized time model described in [19] is used, the property of being an obligation in an acceptable time interval has to be expressed by relations in $T_{owltime\_inside}$. Similarly, if the OWL time ontology [18] is used, intervals are expressed in xsd:dateTime and the xsd date and time comparison operators can be utilized.

V. Experimental validation

This section describes an experimental setup for evaluating SPARQL compliance queries against a synthetic audit log.

Dataset A DBPedia access log is used as the starting point to create a synthetic log dataset (this is the same log used to develop the SPARQL benchmark described in [20]). We extract a URI from each query in the DBPedia access log, and the resulting list of URIs is used as input by a custom develop Java application (SyntheticSCIP) that outputs triples that model a hypothetical audit log scenario as follows. Each input URI is randomly assigned to a pool of 100 data subjects representing owners of the corresponding DBPedia resource. Triples for access requests are then generated using a random date (and a random access request interval of a few days), using the URI as the scip:requestedURI value. Next, for each access request, SyntheticSCIP generates five pre-obligations and ten post-obligations. SyntheticSCIP uses a simple probabilistic model to compute the probability of fulfillment of each obligation associated with an access request. The number of fulfilled obligations of a request follows a binomial distribution $Bin(n, p)$ where n=10, p=0.98. Thus, 2% of the obligations are not fulfilled (due to the obligation not been performed, or due to performance after the obligation interval expires, each of these two option has equal probability). In summary, SyntheticSCIP generates approximately 165 triples for each access request, with approximately 18.3% of these access requests been non-compliant with regards to the privacy obligations in the hypothetical model (this represents a highly non-compliant scenario, appropriate for a stress test). SyntheticSCIP is executed repeatedly to generate audit log entries for the following number of access requests (where the corresponding number of triples generated is in parenthesis); 1000 (164,021), 2000 (327,589), 5000 (818,181), 10000 (1,635,905), 20000 (3,271,599), 50000 (8,142,209).

Queries. The experiment evaluation selects four representative compliance queries. The first query (Q1) is a simple SPARQL ASK audit query to retrieve whether an obligation is performed or not. The query checks the existence of the triple with the scip:performedBy property for the corresponding obligation. The second query (Q2) retrieves all pending obligations. The third query (Q3) retrieves all violated pre-obligations *ob* the agents who agreed to perform *ob* and the agents who were expected (but failed) to perform *ob*. The last query (Q4) is the implementation of the query described in Fig. 7 that returns all access request that are not in compliance with derived obligations. This query gives a full picture of the status of privacy non-compliance.

Test Environment. The experiment executed the SPARQL queries on a Virtouso 6.1 (http://lod.openlinksw.com/) installation (using default settings) on an Ubuntu 11.4 computer with an Intel Pentium 4 3GHz CPU and 1GB of memory.

Results. The four queries are executed in the test environment against the different dataset sizes and elapsed execution times are plotted in Fig. 8. For example, Q4 (the more complex SPARQL query returning comprehensive non-compliance information) can process 750,000 obligations arising from 50,000 access requests with an elapsed time of 104 seconds. The experiment validates the linear scalability of the four compliance queries selected.

VI. Related Work

There are several proposals dealing with privacy in the linked data context. A lightweight privacy preferences vocabulary built on top of web access control and access control lists is proposed in [21], allowing linked data publishers

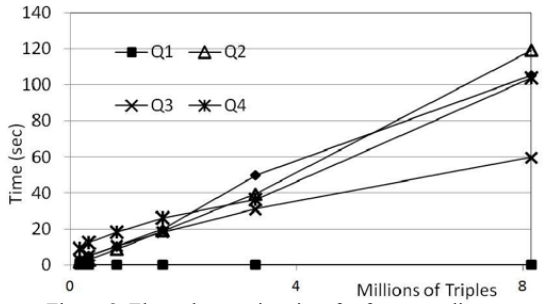

Figure 8. Elapsed execution time for four compliance queries

to express fine-grained access policies for their resources. An alternative access control framework in [22] for social web applications uses SWRL to express access rules, enabling linked data publishers to specify who can access which resources. The authors of [23] leverage the linked data architecture for providing authorizations (based on WebID [24]) and access restrictions at the document level. To address the privacy concerns in the emerging domains of linked data usage, the authors of [25] propose a privacy framework for policy specification and access control enforcement. The preceding five proposals support privacy by restricting access to information, without addressing privacy aspects such as usage and post-conditions on access. In contrast, the SCIP ontology proposed in this work captures a richer privacy model as described by the contextual integrity discipline.

There is body of research [6, 7, 26, 27] leveraging log mechanisms to support information accountability. The authors of [26] and [6] use Linear Temporal Logic to express privacy policies and to monitor their satisfaction. Similarly, [27] uses metric first order temporal logic for the same purpose, while providing an online monitoring algorithm. In our work, we limit the temporal reasoning aspect to the individual obligation satisfaction, while a propositional formula φ (discussed in Subsection 4.B) allows for arbitrary combinations of obligations. From the temporal reasoning perspective, SCIP is flexible, supporting the incorporation of different temporal models (as discussed in Subsection 4.D). There are additional aspects of log processing, such as reasoning over incomplete logs (as described in [6]) that are not addressed by the query-based solutions described in this paper (but these features could be incorporated into extensions of the SCIP ontology).

VII. Conclusions

This work describes query-based solutions for two important privacy processes (compliance and obligation derivation) implemented on top of a novel privacy framework that provides a technical foundation for privacy auditing. The framework maximizes the flexibility around participants, policies, and processes by combining provenance-enabled log ontology (L2TAP) with an ontology that synthesizes contextual integrity concepts (SCIP). The experimental validation of the scalability of the solutions highlights the framework's applicability and practical benefits.

Acknowledgment

Financial supports from the NSERC Canada and from IBM Privacy Award are greatly acknowledged.

References

[1] ENISA, *Privacy, Accountability and Trust; Challenges and Opportunities*. European Network and Info. Security Agency, 2011.

[2] OASIS, "OASIS eXtensible Access Control Markup Language v2.0 (XACML)," Feb. 2005.

[3] M. Backes, B. Pfitzmann, and M. Schunter, "A toolkit for managing enterprise privacy policies," in *Proc. ESORICS*, pp. 162–180, 2003.

[4] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo, "Privacy-aware role based access control," in *Proc. SACMAT*, pp. 41–50, 2007.

[5] L. Cranor, M. Langheinrich, M. Marchiori, and J. Reagle, "The platform for privacy preferences 1.0 specification." W3C Recomm., Apr. 2002.

[6] A. Datta, J. Blocki, N. Christin, H. DeYoung, D. Garg, L. Jia, D. Kaynar, and A. Sinha, "Understanding and protecting privacy: formal semantics and principled audit mechanisms," in *Proc. ICISS*, pp. 1–27, 2011.

[7] J. Cederquist, R. Corin, M. Dekker, S. Etalle, J. den Hartog, and G. Lenzini, "Audit-based compliance control," *Int. J. of Information Security*, vol. 6, pp. 133–151, Feb. 2007.

[8] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - the story so far," *Int. J. on Semantic Web and Info. Sys.*, vol. 5, pp. 1–22, Mar. 2009.

[9] H. Nissenbaum, *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford Law Books, 2009.

[10] L. Moreau and P. Missier, "PROV-DM: The PROV data model." W3C Working Draft, June 2012.

[11] US Congress, *Health Insurance Portability and Accountability Act of 1996, Privacy Rule. 45 CFR 164*, Aug. 2002.

[12] Q. Ni, E. Bertino, and J. Lobo, "An obligation model bridging access control policies and privacy policies," in *Proc. SACMAT*, pp. 133–142, 2008.

[13] K. Irwin, T. Yu, and W. Winsborough, "On the modeling and analysis of obligations," in *Proc. CCS*, pp. 134–143, 2006.

[14] H. Hu, G.-J. Ahn, and K. Kulkarni, "Ontology-based policy anomaly management for autonomic computing," in *Proc. TrustCol*, pp. 487–494, 2011.

[15] A. Cavoukian, *Privacy By Design, Take The Challeneg*. Office of Information and Privacy Commissioner of Ontario, 2009.

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, pp. 38–47, Feb. 1996.

[17] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, Nov. 1983.

[18] J. R. Hobbs and F. Pan, "An ontology of time for the semantic web," *ACM (TALIP)*, vol. 3, pp. 66–85, Mar. 2004.

[19] M. Grüninger, "Verification of the OWL-time ontology," in *Proc. ISWC*, pp. 225–240, 2011.

[20] M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo, "DBpedia SPARQL benchmark: performance assessment with real queries on real data," in *Proc. ISWC*, pp. 454–469, 2011.

[21] O. Sacco and A. Passant, "A privacy preference ontology (PPO) for Linked Data," in *Proc. LDOW Workshop at WWW*, 2011.

[22] H. Mühleisen, M. Kost, and J.-C. Freytag, "SWRL-based Access Policies for Linked Data," in *Proc. SPOT Workshop at SSW*, 2010.

[23] J. Hollenbach, J. Presbrey, and T. Berners-Lee, "Using RDF metadata to enable access control on the social Semantic Web," in *Proc. CCMLSK Workshop at CK*, 2009.

[24] H. Story, B. Harbulot, I. Jacobi, and M. Jones, "FOAF+SSL: RESTful Authentication for the Social Web," in *Proc. SPOT*, 2009.

[25] S. Speiser, "Policy of composition? composition of policies," in *Proc. POLICY*, pp. 121 –124, 2011.

[26] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum, "Privacy and contextual integrity: Framework and applications," in *Proc. SP*, pp. 184–198, 2006.

[27] D. Basin, F. Klaedtke, and S. Müller, "Policy monitoring in first-order temporal logic," in *Proc. CAV*, pp. 1–18, 2010.