Web Semantics: Science, Services and Agents on the World Wide Web 50 (2018) 1–20

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

# Publishing privacy logs to facilitate transparency and accountability

Reza Samavi [a,*], Mariano P. Consens [b]

[a] Department of Computing and Software, McMaster University, 1280 Main St. West, Hamilton, Ontario L8S 4K1, Canada
[b] Information Engineering, MIE, University of Toronto, 5 King's College Road, Toronto, Ontario M5S 3G8, Canada

ARTICLE INFO

*Article history:*
Received 22 December 2016
Received in revised form 23 November 2017
Accepted 15 February 2018
Available online 21 February 2018

*Keywords:*
Privacy
Policy
Audit log
Accountability
Linked data
Semantic web
Ontology

ABSTRACT

Compliance with privacy policies imposes requirements on organizations and their information systems. Maintaining auditable privacy logs is one of the key mechanisms employed to ensure compliance, but the logs and their auditing reports are designed and implemented on an application by application basis. This paper develops a Linked Data model and ontologies to facilitate the sharing of logs that support privacy auditing and information accountability among multiple applications and participants. The L2TAP modular ontologies accommodate a variety of privacy scenarios and policies. SCIP is the key module that synthesizes contextual integrity concepts and enables query based solutions that facilitate privacy auditing. Other L2TAP modules describe logs, participants, and log events, all identified by web accessible URIs and include relevant provenance information to support accountability. A health self-management scenario is used to illustrate how privacy preferences, accountability obligations, and access to personal information can be published and accessed as Linked Data by multiple participants, including the internal and external auditors. We contribute query based algorithmic solutions for two fundamental privacy auditing processes that analyse L2TAP logs: obligation derivation and compliance checking. The query based solutions that we develop require SPARQL implementations with limited RDFS reasoning power, and are therefore widely supported by commercial and open source systems. We also provide experimental validation of the scalability of our query based solution for compliance checking over L2TAP logs.



## 1. Introduction

The protection of individuals' privacy is becoming increasingly more challenging in the era of social computing and data driven science. An important aspect of privacy protection is *information accountability*, ensuring the policies that govern the lifecycle of personal information (i.e., collect, use, transform, and share users' data) are respected by all parties who are involved in the process [1,2]. Ensuring compliance is a complex task involving multiple participants including the data subjects whose data are at stake and have privacy preferences, data collectors who will be liable if the privacy policies are violated, and internal and external auditors who oversee collection and usage of personal information. Privacy auditing supports information accountability by *validation* (verifies a posteriori if a participant has performed the tasks as expected), *attribution* (finds the responsible participant in case of a deviation from policies), and *evidence* (produces evidence that can be used to convince an auditor if a fault has or has not occurred) [3,1]. This paper presents a Linked Data [4] oriented model that facilitates privacy auditing.

The need for privacy auditing is even more important in the era of the personal web [5] where users are empowered to mix and match a variety of web resources and services to achieve their personal goals [6]. For example, consider the *Sharing Data with Fitness Coach* use case described in [7] (Fig. 1). Mary is interested in self-managing her blood pressure using Personal Health Record (PHR) services. PHR systems (e.g., Microsoft HealthVault [8]) are open platforms with published application programming interfaces (API) that allow users to utilize an extensible ecosystem of personal health applications [9]. Mary adds a blood pressure collector service (BPC) to her PHR platform to collect the measurements. Using the platform, Mary then shares the data with a health clinic (HC) and allows HC to share the results with her fitness club (FC) to arrange a fitness plan. Mary, as the data owner, is concerned about how and for what purposes her personal data might be used by these services. Services are also concerned with knowing what their privacy responsibilities are and ensuring compliance with the policies (e.g., HIPAA [10]) and Mary's privacy preferences. Hence, each of these services needs to have a local audit log as a proof of privacy compliance while an external auditor (including Mary) will need to check compliance across all services independently from the local auditors. In this typical scenario all participants involved should be able to extract their privacy responsibilities, show their

* Corresponding author.
*E-mail address:* samavir@mcmaster.ca (R. Samavi).

https://doi.org/10.1016/j.websem.2018.02.001
1570-8268/

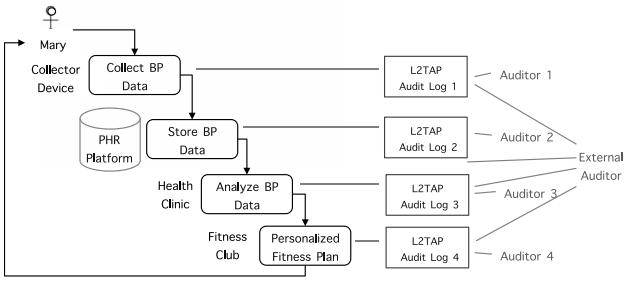


**Fig. 1.** Health self-management workflow.

commitment, and check if others also treat personal information in a lawful manner [11].

Privacy log maintenance is one of the key mechanisms employed in industry to ensure compliance. However, the logs and their auditing reports are designed and implemented on an application by application basis and not supported by widely adopted technology standards. There are solid efforts for standardizing system level audit logs (e.g., W3C Common Log Format [12] and Common Events [13]). These logs can be used for collecting information about the behaviour of programs for monitoring and debugging purposes, and for forensic analysis of security breaches (e.g., [14]). However, the privacy audit logs that we focus on have significantly different requirements from these system level logs (e.g., the privacy model has to capture policies that have deontic modalities; not just the event that occurred but actions that must occur in the future). Proposals on policy expression and evaluation systems such as XACML [15] and its privacy extensions [16–18] support real-time authorization decisions but do not provide specifications for a log that can be used for off-line privacy auditing.

In this paper we propose a Linked Data model for sharing privacy logs among multiple participants that supports SPARQL [19] query based privacy auditing. The model includes a family of ontologies with a modular structure, designed following the principles of Linked Data [20]. The principles support expressing all resources in the privacy log using URIs and making the log events publishable and de-referenceable by authorized participants. Our ontology model, L2TAP (Linked Data Log to Transparency, Accountability and Privacy), consists of four modules and allows participants to log privacy related events in RDF [21]. The privacy events include timestamped information obtained from the participants involved in privacy-sensitive information flows. Our first module, L2TAP-core defines specifications for capturing provenance of privacy events. L2TAP-initialize provides necessary concepts for initializing a privacy audit log and L2TAP-participant registers participants with the logger of a privacy log. The key module in our ontology is L2TAP–SCIP (Simple Contextual Integrity Privacy) that captures the basic privacy concepts inspired by Contextual Integrity (CI) [22] (defined by Nissenbaum as the appropriateness of information flows with regards to norms). Capturing provenance assertions in the log supports *accountability* among the participants involved. Leveraging Linked Data and the modular structure are the basis for the *flexibility* and *extensibility* of our approach, since the modules can be substituted with an ontology with a different expressive power without affecting the core audit log specifications.

The key contribution of our work is designing query-based solutions, grounded on the L2TAP ontology, that supports two important privacy processes: obligation derivation and compliance checking. For a given participant, we formalize a SPARQL query to derive the participant's obligations from the asserted privacy policies. For a given event of access, we devise an algorithm that uses multiple SPARQL queries to determine if the access is in compliance with privacy policies. The compliance query solutions support all three aspects of information accountability: validation, attribution and evidence. These SPARQL query solutions are supported with the limited RDFS [23] reasoning power and implementable using available Semantic Web technologies (e.g., [24]). We experimentally *validate* that compliance query answering using our audit log specifications can scale linearly. While in the general case the SPARQL query has arbitrarily high polynomial data complexity, we show that current SPARQL engines can evaluate typical compliance queries (occurring in realistic scenarios) efficiently.

The paper is structured as follows. Section 2 provides an extensive investigation of related work and provides the rationale for our research on privacy auditing. Section 3 outlines the ontology development methodology, the ontology requirements, the ontology design principles, an overview of the L2TAP modular ontology model, the core structure of each module, and the L2TAP extended structure. Section 4 describes how the L2TAP extended structure can be used for a typical scenario. This section also describes the relationship between this proposal and two closely related standards: (1) the PROV ontology [25] for provenance management as logging privacy events has many similarities with managing provenance of an object in general, and (2) the XACML policy language [15], which is a commonly accepted language for expressing and enforcing access policies. Section 5 describes SPARQL solutions for obligation derivation and compliance checking. In Section 6, we report the results of two experiments to validate the scalability of our approach. We conclude in Section 7.

## 2. Related work

In this section we investigate areas of related research: compliance auditing, system logging, provenance models, policy languages, privacy extensions of policy languages, and privacy and Linked Data.

**Compliance Auditing**. Feigenbaum et al. [26] provide a theoretical treatment for accountable systems and in [27] show that there is a strong link between accountability and deterrence. Compliance auditing allows systems to be checked a priori, against privacy policies [28,29] and supports attribution and evidence when policies are violated. Julisch et al. [30] identified the need to build information systems that meet audit requirements (compliance by design) across multiple organizations. Hence, having logs that support privacy auditing becomes a necessary component of achieving information accountability.

There is also solid theoretical research on logical frameworks for expressing and reasoning about privacy policies. The proposals rely on auditing of knowledge traces, which is close to the notion of our privacy log [31–34]. Barth et al. use Alternating-time Temporal Logic (ATL) to build a logical privacy model and design a privacy language called Logic of Privacy and Utility (LPU), to express norms [31]. Similar to our approach, the concept of norms in this work has been adapted from the contextual integrity perspective [22]. The LPU language allows all communications between agents and their actions to be recorded in logical traces and compliance is related to the logical concepts of satisfiability and entailment [31]. An auditing algorithm is designed to examine compliance with norms and identify irresponsible agents who have violated privacy norms. Datta et al. [32] extended the LPU language with reasoning about information accountability over incomplete logs. Basin et al. [33] use Metric First Order Temporal Logic (MFOTL) to express policies, which are then monitored to verify whether the trace of actions satisfies desired temporal properties. Cederquist et al. [34] describe a framework that uses audit logs to enforce compliance with discretionary access control policies. While the described logical models provide a highly expressive language for capturing the knowledge for privacy auditing, the expressivity is

coupled with high complexity that jeopardizes the scalability of the solutions. The proposals also do not provide the necessary vocabulary for the log specifications.

**System Logs**. In the classical sense, logs are designed and used for monitoring and debugging purposes or collecting evidence for security breaches. The UNIX Syslog extension [35], Windows MSDN security log [14], and Log4j [36] are examples of such log implementations. There are also proposals to standardize log specifications. The W3C Common Log format [12], the Common Events [13] and security models for logging [37–39] are examples of the log standardization efforts. The common characteristic of the logs specified in this body of literature is that the logs are designed to be used by the developers and administrators for diagnostic or forensic analysis. Therefore the queries that will be run against the log are not quite known at design time and also the log's schema is kept simple (a flat file) as the semantics necessary to answer forensic queries are incorporated into the program logic used for log analytics. The same log can be used and interpreted differently depending on the types of forensic questions in the analysis stage.

Privacy logs are different from classical logs in multiple aspects. First, the privacy requirements and the queries that a privacy log needs to answer are known at design time. A log for privacy needs to collect only necessary information to answer the known privacy queries. Second, in a privacy scenario, where diverse participants work with the same private data, different interpretations of a system log are not desirable. Compliance should be checked against privacy policies that are transparent to all participants, as opposed to interpreting system logs. Therefore in a privacy log, the log schema, rather than the local log analytics program, should bear the semantics of the log events shared by all participants. Finally, a privacy log should be able to capture not only the events asserting the actions occurred but also the events that assert privacy policies and preferences (deontic modalities). These unique characteristics motivate research studies focusing on semantically describing privacy audit log specifications.

**Provenance Models**. One class of potential candidates to be considered for privacy auditing is research proposals on provenance (e.g., the PROV model [25] and Open Provenance Model (OPM) [40]). A provenance model records assertions on what happened to an information object, when, and by whom. While a provenance model provides a generic way of describing extensive events that happened to a resource, it is not expressive enough to capture requirements for the deontic modality of a privacy log such as user's privacy preferences, purpose of usage, and data holder's obligations. We are aware of prior research on using provenance for auditing of data usage (e.g., [41,42]). The studies are mainly focused on architectural design for combining provenance assertions with usage policies but they do not provide design specifications for a practical and scalable privacy log. In fact, Grunwell et al. [42] identified the scalability of their approach as a challenge needing to be addressed. The L2TAP log described in this paper complements this body of standards by designing a scalable privacy log that supports query based solutions for privacy auditing. In Section 4.4, we describe a partial mapping of L2TAP to PROV-O [25] that allows the provenance assertions of log events to be extended by the concepts in PROV-O.

**Policy Languages**. There are several proposals on policy languages that can be used by organizations to express their internal policies and facilitate privacy policy enforcement. P3P [43] is a W3C standard that enables websites (as data receivers) to express privacy policies, including the purpose of collecting data, availability of consent, and retention period of collected data. Besides being known as complex and confusing [44], the focus of P3P is on expressing privacy policies and not on how the policies can be enforced. XACML [15] and EPAL [45] are two XML based policy languages that allow fine-grained access control policies to be expressed and enforced. Both XACML and EPAL are policy evaluation systems dealing with the real-time process of deciding and enforcing access policies.

Representing policies in machine readable format has received considerable attention from the digital rights management community. The W3C Open Digital Rights Language (ODRL) Community Group [46] works on developing a language for the semantics of the policy expressions and interoperability. The language specifies a set of RDF classes, predicates, and named entities to define the ODRL information model [47]. Since the limited natural language description is at the core of ODRL to capture policies, a number of extensions (e.g., [48]) and complementary work (e.g., [49]) are proposed to formally capture the semantics of complex policies and express dependencies between policies. Daga et al. [49] proposed an efficient framework for reasoning over policy propagation when data artefacts are used on web scale workflows. A recent initiative undertaken by the W3C Permissions & Obligations Expression Working Group [50] intends to extend ODRL to develop a standard for defining permissions and obligations.

The work presented in this paper complements the policy languages described above as for a semantic policy evaluation language to support off-line privacy auditing, a log format similar to our approach should be developed. For example, having the policies and the log in an XML type language would then allow developing an XQuery based [51] solution (similar to our SPARQL solution) to derive obligations and compliance queries. We are not aware of the availability of such a solution for privacy auditing on XML logs.

**Privacy Extensions**. XACML has been extended to evaluate, in real time, some aspects of privacy policies, such as role and purpose definition [16,17]; yet it does not supporting expressing and enforcing data user's obligations. Ni et al. [52] propose an obligation model for access control systems for capturing different types of obligations associated with an access activity. Li et al. [18] extend the XACML access control language (ExtXACML) with an implementation that allows the access module and the domain dependent obligation modules to interact with the privacy enforcement point in an XACML implementation via a number of events. These events are triggers to change the state of an obligation, thus capturing the temporal characteristics of an obligation and allowing pre- (must be performed prior to access) and post-obligations (must be performed after access) to be expressed. The extended language includes an algorithm to investigate the effects of the privacy obligations on each other. Similarly, Liu et al. [53] propose an obligation policy language, called Eagle, for actions required to be performed when using streams of heterogeneous data. Obligation policies are defined in terms of semantic event patterns, where at run-time a policy compliance checker (using Description Logic Programs) matches instances of events with the event patterns introduced in the obligation policies and identifies the actions that must be performed. Rao et al. [54] propose an extension of XACML architecture to support context intensive and sophisticated privacy policies. While these extensions improve the application of XACML, they still do not support a systematic way of generating XML based privacy logs that can be used for off-line privacy auditing. Furthermore, management of the attribute based policy languages, such as XACML and EPAL, is cumbersome in practice [55] and the implementations provided for the enforcement of privacy policies (e.g., XACML implementation [56]) do not support a query based solution for privacy auditing and follow an imperative paradigm that makes it hard to scale up when the number of policies increases [57].

**Privacy and Linked Data**. Addressing privacy through accountability received considerable attention from the Semantic Web

community [58–62]. Hendler and Berners-Lee et al. [58] identified information accountability (as defined in [1]) as a useful approach for managing data usage on the web. The authors pointed out that the current accountability mechanisms on the web are inadequate as (1) they have often been built for a specific Web application (not supported by an industry standard), and (2) they require complex auditing logic. Our proposal is an effort to address these shortcomings.

Kagal and Pato [59] proposed an architecture to support a set of semantic policy tools including a Semantic-Web-based policy language for compliance checking using SPARQL queries. Seneviratne and Kagal [60] proposed an end to end architecture for implementation of a web-scale transparent and accountable privacy system, but they do not provide a set of vocabularies that can be directly used for privacy auditing on the web.

There are also several proposals dealing with privacy in the Linked Data context. The study conducted by Corsar et al. [61] motivates our work as it demonstrates the impact of a Linked Data environment on user privacy and the responsibilities of key stakeholders, including researchers, ethic committees, and the Linked Data community when data are published. Bettencourt et al. [62] proposed a framework to publish Linked Data privately. Sacco and Passant proposed a lightweight privacy preferences vocabulary, built on top of the web access control and access control lists allowing Linked Data publishers to express fine-grained access policies for their resources [63]. An alternative access control framework is proposed by Mühleisen et al. for social web applications [64]. This framework uses SWRL [65] to express access rules, enabling Linked Data publishers to specify who can access which resources. Hollenbach et al. [66] leverage the Linked Data architecture for providing authorizations and access restrictions at the document level. The authorization in this proposal is based on WebID [67]. To address the privacy concerns in the emerging domains of Linked Data, Speiser et al. propose a privacy framework for policy composition and access control enforcement [68]. The body of work described here mainly focuses on policy evaluation for accessing published Linked Data, rather than addressing data usage privacy requirements. The focus of our research is not to design a privacy model just for Open Linked Data, but rather to utilize the flexibility and convenience of Linked Data infrastructure as an emerging standard for representing and publishing privacy logs that support query based privacy auditing.

## 3. L2TAP privacy model overview

Our model provides a flexible and extensible infrastructure to capture what participants must perform and comply with when accessing personal information (prescriptive) and what participants have in fact performed when given access to personal data (descriptive). Our model formalizes privacy log specifications that capture all relevant *privacy events* and provides solutions to the *privacy queries* of participants involved in the workflow. Note that the information model we are proposing in this paper does not cover enforcement of the tasks needed to be performed by the participants (e.g., obligation enforcement), nor guarantees the accuracy of what participants are reporting, rather, the model provides the framework for expressing the content of logs when obligation events are reported, such that they can be effectively audited.

In this section, we describe the ontology design for the privacy audit log in Section 3.1. The core structure of the L2TAP ontology and its key module, SCIP, are described in Section 3.2. In Section 3.3, we describe how the L2TAP structure can be extended for reuse and expressivity.

### 3.1. Ontology design

We express our privacy model as an ontology. The scope of an ontology defines the concepts that the ontology should express. The conditions under which the ontology will be used determines its expressive power. While higher expressivity enhances the ontology reasoning power, it makes the ontology more complex, jeopardizing its practicality. To design our ontology we mainly follow the design methodology described in [69] and [70]. Since the methodology does not cover the non-functional aspects of the ontology design, we follow the steps described in [71] to include non-functional requirements. The competency questions (formalized as queries) help us identify necessary concepts and relations.

**Intended Users**. The main users of our ontology are: (1) individuals who should be able to express their privacy preferences as their private information are at stake; (2) data collectors (services) in a workflow that should be able to express their privacy policies, access to personal data, and their abiding by the conditions and obligations derived from policies and preferences; and (3) privacy auditors who want to audit compliance with data collectors' privacy policies and individuals' privacy preferences. Auditors can be internal to each service, checking compliance at the service level, or external to the workflow, accessing an aggregated log and checking overall compliance with the privacy policies.

**Functional Requirements**. The design of the ontology stems from the contextual integrity (CI) privacy requirements [22]. Our ontology should support capturing and reasoning about:

1. the main *actors* of an information flow, i.e. data receiver, data sender and data subject;
2. the *roles* that actors play in an information flow and the relationship between different roles;
3. the *type of information* (attributes in CI);
4. *norms* (policies and regulations, e.g., HIPAA [10]) that are considered as the backdrop of an information flow to check its appropriateness;
5. the *contex* t that is the structured social setting characterized by the roles that actors play, the canonical activities and actions that people in roles perform (healthcare services, educational institutions and social networks are examples of different social contexts), the value (purposes) they offer, and the applicable norms for respecting individuals' privacy rights;
6. *temporal constraints* that are inherent in the history of activities performed by the actors such as access activities or activities performed to fulfil obligations;
7. and the accountability concepts of *validation*, *attribution*, and *evidence* [3].

**Non-Functional Requirements**. The main characteristics and qualities that our privacy audit ontology should support are:

1. *scalability*, large privacy logs (a graph with millions of triples);
2. *extensibility*, accessing privacy audit related information expressed in other ontologies;
3. and *flexibility*, in terms of: the infrastructure that the ontology needs to operate, mapping to other linkable ontologies for the users who may need ontologies with more or less expressive power, and integration with legacy systems.

**Design Principles**. We made a number of ontology design decisions to support the non-functional requirements listed above. First, we designed the ontology with a modular structure, which allows modules to be reused, extended, and substituted. Second,

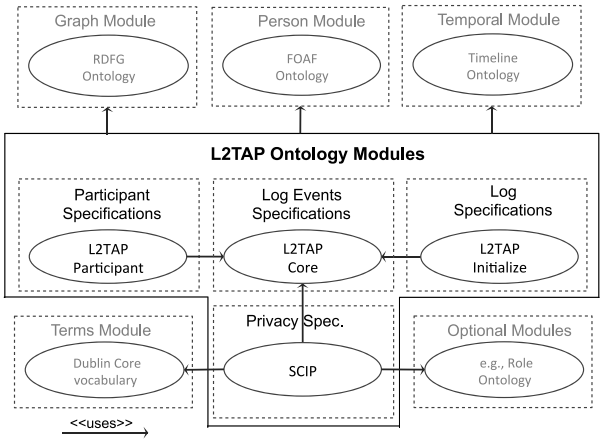


**Fig. 2.** Modular structure of the L2TAP ontology.

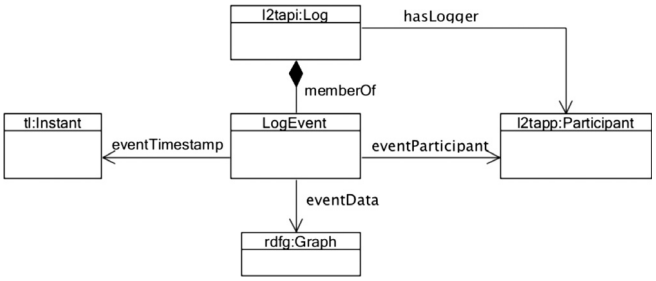


**Fig. 3.** L2TAP core structure.

the choice of data model is crucial for the practical value of the ontology in terms of supporting scalable logging and the availability of Semantic Web technologies to answer compliance queries efficiently. A relational SQL-based log is a scalable solution, but it is not flexible enough with changes to the log's schema. Alternatively, a first order privacy logic is highly expressive (e.g. Privacy MFTL [33]) but its implementation is not supported by a scalable technology. Using an RDF [21] data model and SPARQL [19], with the limited reasoning power of RDFS [23], we can support the flexibility and extensibility required for a web-scale privacy audit log implementation, supported by off-the-shelf semantic technologies (e.g., Open Link Virtuoso [72], Oracle Spatial and Graph Semantic [24] and IBM DB2 [73]).

Third, to support flexibility, all L2TAP ontology modules follow the principles of Linked Data [20] to publish privacy logs. All assertions are represented as RDF triples and quads. Web accessible URIs are used as names for all resources. A publishable audit log simplifies transparency and accountability, since contribution of a log event into the log can be traced back to the responsible participant in a workflow. Flexibility of a privacy log also depends on the level of granularity of personal information that it can support. An individual's personal information can span from a very specific data item (e.g., an entry in a lab report) to a very general data item (e.g., the entire health record). Leveraging the Linked Data infrastructure and expressing everything in the log (including data items) using dereferenceable URIs provides a flexible and generic way of representing resources involved in a privacy event.

### 3.2. L2TAP core structure

The L2TAP ontology model is depicted in Fig. 2 and consists of four L2TAP modules and five external modules. We use the following namespaces to define terms in each module:

L2TAP-Core: http://purl.org/l2tap#
L2TAP-Initialize: http://purl.org/l2tapi#
L2TAP-Participant: http://purl.org/l2tapp#
L2TAP–SCIP: http://purl.org/scip#

The active ontology and the archived versions are available on our ontology website: http://l2tap.org. The modular structure of the L2TAP ontology and the additional properties and concepts described in this paper are from version 2.0 of the L2TAP ontology, not the deprecated, non-modular version 1.0 [74,75]. The latest version of the ontology presented in this paper is designed with a modular structure after functional and non-functional requirements of the ontology were extensively investigated and revised.

**L2TAP-core**. This module provides a generic infrastructure for logging privacy events and reasoning about their provenance. Log events are temporal objects and L2TAP-core is centred around answering provenance queries for these temporal objects; for example, *when* has a log event been contributed to the log? *who* has contributed the log event to the log? and *what* is the content of the log event? The core types of L2TAP and their relationships are illustrated by the UML diagram of Fig. 3.

Two main classes in L2TAP are `l2tap:Log` and `l2tap:LogEvent`. Everything in the `l2tap:Log` is expressed in terms of some `l2tap:LogEvent`. A log may have multiple log events as its members but a log event belongs to exactly one `l2tap:Log`. Any L2TAP log event consists of a header of the event and a body. The L2TAP-core ontology is used to express the header of a log event. Other modules are used to encode the body of the log event.

The *who* of a log event is captured by the `l2tap:Participant` concept. The participant can be a person, a group of persons, an organization, a software agent or similar entities. The L2TAP participant is a super-type with subclasses of `l2tap:Logger` who initiates a log and `PrivacyParticipant` who contributes the content of privacy log events to the log. A logger registers all participants and when registered, participants can contribute their privacy events to the log and in turn the logger considers the registered participants accountable for their assertions in the log.

The *when* is associated with an *instant* defined in an external temporal ontology module as shown in Fig. 2. One instance of such a temporal ontology is the Timeline ontology [76]. To make temporal reasoning consistent across different L2TAP modules, all modules should use the same temporal ontology. A timeline in the Timeline ontology is a linear ordering over time points but can be associated with several coordinate systems for capturing a particular point in time. For example, in our scenario, each service (e.g., HC or FC) may use different relative timelines depending on the time they get involved in the workflow. The Timeline ontology uses XML-Schema datatypes [77] for such a coordinate system, thus we can link the instant occurrence of a log event to a literal in the corresponding XSD datatype (i.e., `xsd:dateTime`). The restrictions on using the same canonical coordinate system and the datatype in an L2TAP log is captured using the `tl:onTimeLine` property in the Timeline ontology. As described in bi-temporal databases [78,79], we make a distinction between the *application time* and *system time*. Thus, the `l2tap:eventTimestamp` has two sub properties: `l2tap:receivingTimestamp` that captures the time instant of receiving the last triple of a log event from a participant and `l2tap:publicationTimestamp` that captures the time that the log event is available for a URI request.

The *what* of a log event is the payload (body) of the event and is captured by the `l2tap:eventData` property. The L2TAP-core makes minimal commitment, with respect to the event data, limited to an assertion that there exists a graph associated with this log header. We use the RDFG ontology [80] to state that the body and the header of an event are subgraphs of the same graph. The body of log events captures additional assertions using the other

three L2TAP modules for log initialization, participant registration and privacy events.

We use URIs to identify collections of statements in the body of a log event to create a *quad* or a named graph [81]. Using named graphs allows us to express the important intuition that the assertions made in the header of a log event are applicable to all triples in the named graph (the body). In addition, we can support the modular structure of L2TAP by making the semantics of statements captured in the body of an event opaque to the semantics of the header. For example, the ontology module describing the semantics of privacy events (L2TAP–SCIP) can be substituted with another ontology without impacting the L2TAP-Core module that describes the header of the log.

**L2TAP-initialize**. This module supports assertions about the log itself. `l2tapi:Log` is the only class in this module. This module uses `l2tapi:hasLogger` to point to the logger in the L2TAP-participant internal module. The module also uses the `l2tapi:logTimeline` and `l2tapi:logDiscreteTimeline` to capture two different temporal descriptions in the Timeline external module [76]. The first temporal concept captures a point in time on the universal timeline (e.g., the event $e_1$ logged on the 8th of May, 2017 at 14:20 EDT) and the second concept captures an integer value in $\mathbb{Z}$ for the time, which is a duration relative to or an offset from the universal time point (e.g., the obligation $ob_1$ must occur 3 days after event $e_1$). The ontology provides answers to the queries about the log characteristics such as: who is the logger? how is the time expressed in the log? when has the log been initialized?

**L2TAP-participant**. This module supports assertions about participants. `l2tapp:Participant` is the main class in this module with two subclasses: `l2tapp:Logger` and `l2tapp:PrivacyParticipant`. The logger is the publisher of the log events and owns the domain of the log and can talk about the events and their assertions in the log. Only participants who are known to the logger can contribute log events into the log. Following the Linked Data principles, using http:// (or https:///) URIs as names for resources requires data publishers choosing part of an http:// namespace that they control, by virtue of owning the domain name, and minting URIs in this namespace to identify the resources in their datasets [20]. Therefore, if the logger wishes to identify a participant as the *who* for a log event, the logger must mint the URI of the participant with the namespace in its domain. The logger uses the `l2tapp:registersAgent` property in this module to register a FOAF [82] agent as a participant. A registered participant will be an instance of the `l2tapp:PrivacyParticipant` class.

This module is also used to express different aspects of a log's participant such as: the hierarchical structure of participant types (represented as a lattice using `rdfs:subClassOf`), the authentication method of a participant, and other domain-dependent characteristics of participants (e.g., using FOAF to add triples about the WebID [83] or method of transferring triples to the log [84]).

**L2TAP–SCIP**. This module is the most comprehensive ontology module and allows assertions about the privacy semantics of an event to be expressed. Some of the representative compliance queries that the SCIP ontology should be able to answer are: who accessed the information and when? who performed an obligation and when? is an obligation violated? what are the data requestor's pending obligations? There are also general access request management queries, of which the most complex one is the obligation derivation query. The generic form of access request queries are: given an access request what are the participants' outstanding obligations? what data can I access without obligations? how many access activities are associated with a specific data item in a specific period of time?

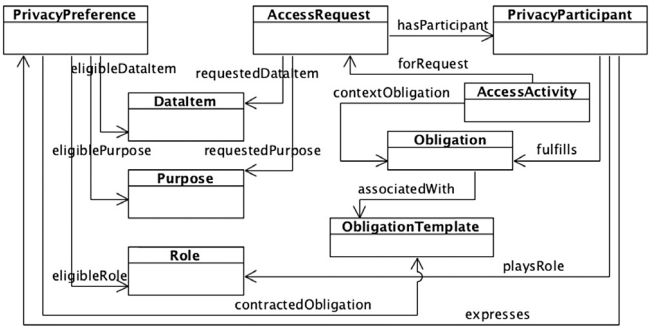


**Fig. 4.** SCIP core structure.

The core types of SCIP and their relationships are illustrated by the UML diagram of Fig. 4. The SCIP ontology provides mapping targets for basic concepts of privacy in CI. For example, the CI's notion of *actors* in an information flow is mapped to the concept of `PrivacyParticipant` in the SCIP ontology. But using the extended structure of SCIP, participants are further sub-typed as `DataRequestor` (the one who receives the personal information), `DataSender` (the one who sends the personal information), and `DataSubject` (whom the personal information belongs to). Therefore, all participants expressed using the SCIP module are `rdfs:subClassOf` the `Participant` class expressed in L2TAP-core.

The type of personal information that is at stake is captured using the `DataItem` class representing the notion of *attributes* in CI. The SCIP `Role` and `Purpose` classes are considered to capture the concepts of *roles* the actors play and *purposes* of sharing private data in a context, respectively. The SCIP `DataItem`, `Role`, and `Purpose` classes are used to capture the abstract and concrete data items, roles, and purposes, respectively. In SCIP, these three concepts are represented as a lattice using `rdfs:subClassOf` with a respective superclass *Any* on top (e.g., `scip:PurposeAny` and `scip:RoleAny` for purpose and role hierarchies, respectively). The lattice structure allows concrete eligible purposes (and similarly eligible roles and eligible data items) to subsume one or more classes of abstract purposes. Thus privacy preferences can be defined at any level of granularity for an abstract class of purposes or for a specific concrete purpose. For example, a PHR user may assert in her privacy preferences that her health data can be used for any secondary usage purpose. But a more cautious user may only allow her data to be used for an academic research purpose. Hence, an abstract class of secondary usage purpose can be defined with a number of concrete subclasses such as academic research, marketing research, etc. Our approach is analogous to proposals for using purpose hierarchies in purpose based access control models [85].

The notion of *norms* in CI is expressed by multiple classes and properties. The `PrivacyPreference` class captures privacy policies and preferences. The policies are defined in terms of type of data, roles, purposes, and obligations. The same classes of `Role` and `Purpose` are used to define eligible roles and purposes for data access. The `DataItem` class is used to specify the personal data to which the conditions and obligations apply. The `Obligation` class captures assertions on types of obligations that the data accessor must fulfil before (pre-obligation) or after (post-obligation) the data access [52]. As part of her privacy preferences, a data subject may express different sets of conditions and obligations depending on the context: for example, one set for accessing her personal health record by the emergency department and another for a personal trainer to offer an exercise plan.

The `AccessRequest` defines the context of access in terms of which data item is being requested for which purpose, and in which capacity (role) the data requestor requests access to personal information. Then at the time of an access, multiple instances of the `AccessActivity` class capture the response to an access request and the actual access that occurred. The `AccessResponse` class is a subclass of `AccessActivity` and captures the access decision and the applicable obligations. The `ActualAccess` is the second subclass and records the occurrence of an actual access.

### 3.3. L2TAP extended structures

While the core of L2TAP focuses on essential privacy auditing concepts, extended structures are necessary for its practical usage. The core types of L2TAP and SCIP, shown in Fig. 3 and Fig. 4 respectively, can be extended using subtyping. For example, a data requestor or a data sender are special kinds of `l2tapp:PrivacyParticipant`, which in turn is a special kind of `l2tap:Participant`. We define subtyping using `rdfs:subClassOf`.

Subtyping can also be applied to core properties of L2TAP and SCIP. For example, a publishing timestamp is a special kind of event timestamp. We use `rdfs:subPropertyOf` to capture property subtyping. Subtyping provides a flexible and simplified extended structure for capturing additional semantics necessary for privacy nuances. For example, the participant who fulfils an obligation could be the data requestor or another participant to whom the responsibility is being delegated. If this distinction is unimportant then the `scip:fulfils` property captures which participant has fulfilled the obligation but if this distinction needs to be captured, the assertion can be made using the `scip:delegateFulfils` property as a special kind of `scip:fulfils` indicating that the participant is the one to whom the responsibility is being delegated. In the next section, when we model the motivating scenario, we show in more detail how subtyping is useful to capture more advanced uses.

The L2TAP ontology can also be extended to support reuse, expressiveness, and interoperability. For example, in L2TAP, roles are expressed as a lattice without any time interval attached to a role. This limited expressiveness may not be sufficient for a specific application domain, where the Publishing Roles Ontology — PRO (http://purl.org/spar/pro) is used to describe the roles. To take advantage of PRO's expressiveness, the ontology can be plugged into L2TAP by a simple mapping of the `scip:Role` and `pro:Role` classes. For example, in the "collaborative health research" scenario described in [75], the `pro:editor` and `pro:distributor` roles from the PRO ontology can be used to express privacy obligations of different roles in publishing the research results. We can also use the inverse property of `pro:withRole` and the `pro:RoleInTime` class to attach a validity time interval to each of these two concrete roles. The same approach can be applied to `scip:Purpose` or `scip:DataItem` to extend the expressivity of L2TAP. In this example, plugging PRO into L2TAP might be motivated by reasons other than extending expressivity such as supporting a scenario where a legacy system is using PRO and reuse or interoperability is a requirement.

The modular structure also supports substitution of a module with an ontology with more or less expressive power without impacting other L2TAP modules. For example, L2TAP-Core provides minimal structure for expressing provenance assertions of log events. PROV-O [25] is an ontology with an extensive provenance semantics compared to L2TAP-core. If in an application domain using PROV-O is justified, given the volume of the ontology (30 classes and 50 properties compared to L2TAP-Core with 4 classes and 6 properties), then the L2TAP-core can be substituted with PROV-O to enjoy the extended semantics (described in Section 4.4).

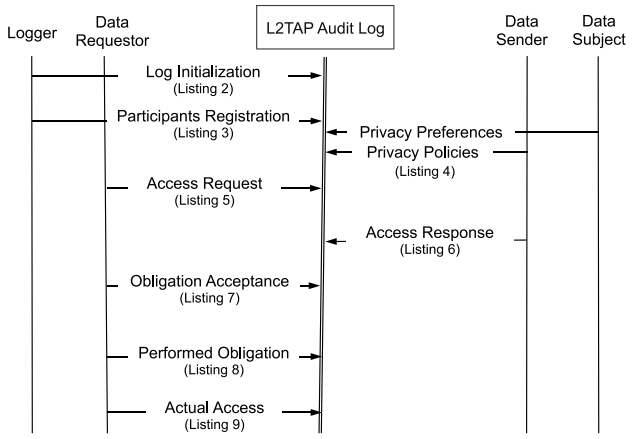


**Fig. 5.** A sequence diagram visualizing event occurrences in an L2TAP log.

## 4. Application of the L2TAP ontology

The purpose of this section is to show how the concepts and relations introduced in Section 3 can be put into practice to initiate a privacy audit log and populate it with privacy events necessary for answering competency queries. In this section we first describe how to publish three types of log events: log initialization events, participant registration events, and privacy events in Sections 4.1, 4.2, and 4.3, respectively. We then illustrate how the provenance support of L2TAP is related to the PROV ontology in Section 4.4 and how L2TAP–SCIP can be used to log real-time events generated by XACML in Section 4.5.

We use the simple "Sharing Data with Fitness Coach" scenario described in Section 1 for publishing the log. The sequence diagram in Fig. 5 helps visualize the steps for initiating the log for our example and logging the events for requesting and obtaining access to personal information by a service. The main participants in our example scenario are the L2TAP logger, services (data requestors) and Mary (data provider). Participants are depicted on the top of the diagram. We follow the steps in Fig. 5 and for each sample event (sequence) we include a reference to the serialization of the event in Turtle [86]. These events use the namespace prefixes of our ontology: **l2tap**, **l2tapi**, **l2tapp**, and **scip**, denoting terms from different modules of the L2TAP ontology, and prefixes **exphr**, **exhc**, and **exlog**, denoting terms specific to the example: PHR, Health Clinic, and the instance of the log initiated for this scenario, respectively. We also illustrate how the L2TAP ontology can be used in combination with other ontologies, such as Timeline [76], FOAF [82], and Dublin Core [87]. To avoid repeating prefixes for each listing, all prefixes used are listed in Listing 1.

```
1  @prefix l2tap:<http://purl.org/l2tap#>.
2  @prefix l2tapi:<http://purl.org/l2tapi#>.
3  @prefix l2tapp:<http://purl.org/l2tapp#>.
4  @prefix scip:<http://purl.org/scip#>.
5  @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
6  @prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
7  @prefix tl:<http://purl.org/NET/c4dm/timeline.owl#>.
8  @prefix time:<http://www.w3.org/2006/time#>.
9  @prefix rdfg:<http://www.w3.org/2004/03/trix/rdfg-1/>.
10 @prefix foaf:<http://xmlns.com/foaf/0.1/>.
11 @prefix owl:<http://www.w3.org/2002/07/owl#>.
12 @prefix dcterms: <http://purl.org/dc/terms/>.
13 @prefix prov: <http://www.w3.org/ns/prov#>.
14 @prefix sp: <http://spinrdf.org/sp#>.
15 @prefix exphr: <https://dphr.org/>.
16 @prefix exhc: <https://hc.org/>.
17 @prefix exlog: <https://log1.org/>.
```

Listing 1: List of all prefixes used.

### 4.1. L2TAP log initialization

The first event is Log Initialization. This type of event defines which instance of an L2TAP log is being initialized. The event is responsible for capturing the log characteristics such as who the logger is, which timeline the log is adapting and other characteristics that could be important for the logger. The L2TAP-core module only expresses that there exists an initialization event in an L2TAP log. It is the responsibility of the L2TAP-initialize module to capture the characteristics of the log.

```
1  exlog:logevent-e1 a l2tap:LogInitializationEvent;
2    l2tap:memberOf exlog:log1;
3    l2tap:eventParticipant exlog:logger1;
4    l2tap:receivingTimestamp exlog:e1-t1;
5    l2tap:publicationTimestamp exlog:e1-t2;
6    l2tap:eventData exlog:logng-ng1.
7  exlog:e1-t1 a tl:Instant;
8    tl:atDateTime "2016-01-26T12:00:00Z"^^xsd:dateTime;
9    tl:onTimeLine exlog:tlphysical.
10 exlog:e1-t2 a tl:Instant;
11   tl:atDateTime "2016-01-26T12:00:00Z"^^xsd:dateTime;
12   tl:onTimeLine exlog:tlphysical.
13 exlog:logng-ng1 a rdfg:Graph.
14 exlog:logng-ng1 {
15   exlog:log1 a l2tapi:Log;
16     l2tapi:hasLogger exlog:logger1;
17     l2tapi:logTimeline exlog:tlphysical;
18     l2tapi:logDiscreteTimeline exlog:tldiscrete.
19   exlog:tldiscrete a tl:DiscreteTimeLine;
20     time:unitType time:unitDay.
21   exlog:tlphysical a tl:PhysicalTimeLine;
22     owl:sameAs tl:universaltimeline.
23   exlog:logger1 a foaf:Agent;
24     l2tapi:initializesLog exlog:log1.}
```

Listing 2: Log initialization event.

Listing 2 provides an example of using L2TAP-core and L2TAP-initialize to encode the log event that initializes a log with the `exlog:log1` URI. The header of the log (lines 1–13) captures the provenance assertions of this log event. `exlog:logevent-e1` is the URI of the first log event in this log and is an instance of the `l2tap:LogInitializationEvent` class (line 1). This event is a member of the log of interest (line 2). The *who* is captured in line 3 indicating that a logger with the `exlog:logger1` URI has contributed this event to the log. The *when* is captured in lines 4–5 and 7–12, and the *what* is captured in line 6 and lines 13–21. The *what* includes the specifications of the log, the adapted timeline ontology and other characteristics of the log, which are all captured using terms in the L2TAP-initialize module as a named graph (lines 14–24). Notice that we used TriG [88] extended Turtle syntax, which supports named graphs.

The quad in line 15 asserts that `exlog:log1` is a `l2tapi:Log` and this log has `exlog:logger1` as its logger (line16). This new log has also adapted a physical timeline with the `exlog:tlphysical` URI (line 17) and a discrete timeline with the `exlog:tldiscrete` URI (line 18) to capture two instances of the `tl:PhysicalTimeLine` and `tl:DiscreteTimeLine` classes in the Timeline ontology [76], respectively. In line 20, we use the OWL time ontology [89] to express the time unit for this log on the discrete timeline. In line 22, we use one of the OWL [90] primitives, `owl:sameAs`, to state that for this log the URI of the physical timeline identifies the same resource as the universal timeline (UTC), which is a constant in the Timeline ontology. Note that we used the `tl:onTimeLine` property to assert that the time instants that this log event is being logged (triples in lines 4 and 5) make reference to the same timeline that the log is being initialized with (compare the URIs in lines 9, 12 and 17). Finally, in line 23 we encode that the logger is a `foaf:Agent` [82] and who has initialized this log (line 24).

### 4.2. Participant registration

The second event in the log is Participant Registration. In our scenario the blood pressure collector service (BPC), the PHR platform, the fitness club (FC) and Mary herself, are examples of participants in the log. This event can be used to assert participant class membership. For example, each service or a class of services may have specific consent conditions. In this case, a class of participants with one set of applicable privacy policies can be registered by the logger and then the class membership becomes asserted in the body of this event.

Listing 3 shows the RDF coding of registering the PHR platform and the health clinic (HC) with the initialized log. Analogous to the log initialization event, statements in lines 1–13 are the header of this log event. Note that the type of this event is `l2tap:ParticipantRegistrationEvent` (line 1) and the participant who contributes this event is the logger (line 3). The logger asserts that from this point in time forward this participant is known to the logger.

The logger registers the PHR platform known with the `exphr:phr1` URI and HC (`exhc:hc`) as `foaf:Agent` using quads in lines 15–16. The logger also mints both participants' URIs with a URI in the logger's domain using `owl:sameAs` (lines 20 and 23). The named graph can be used to capture additional information about the participant. Suppose in our motivating scenario BPC and HC can be instances of a class of services that should comply not only with their own privacy policies but with the privacy policies of the PHR platform. This can be expressed by creating two classes of participants as *platform* and *service* and then class membership can be asserted as in lines 24–26.

```
1  exlog:logevent-e2 a l2tap:ParticipantRegistrationEvent;
2    l2tap:memberOf exlog:log1;
3    l2tap:eventParticipant exlog:logger1;
4    l2tap:receivingTimestamp exlog:e2-t1;
5    l2tap:publicationTimestamp exlog:e2-t2;
6    l2tap:eventData exlog:logng-ng2.
7  exlog:e2-t1 a tl:Instant;
8    tl:atDateTime "2016-01-27T12:00:00Z"^^xsd:dateTime;
9    tl:OnTimeLine exlog:tlphysical.
10 exlog:e2-t2 a tl:Instant;
11   tl:atDateTime "2016-01-27T12:00:01Z"^^xsd:dateTime;
12   tl:OnTimeline exlog:tlphysical.
13 exlog:logng-ng2 a rdfg:Graph.
14 exlog:logng-ng2 {
15   exphr:phr a foaf:Agent.
16   exhc:hc a foaf:Agent.
17   exlog:logger1 l2tapp:registersAgent exphr:phr, exhc:hc.
18   exlog:participants-phr1 a l2tapp:Participant;
19     l2tapp:registeredAgent exphr:phr;
20     owl:sameAs exphr:phr.
21   exlog:participants-hc1 a l2tapp:Participant;
22     l2tapp:registeredAgent exhc:hc;
23     owl:sameAs exhc:hc.
24   exlog:participants-phr1 a exlog:Participants-platform.
25   exlog:participants-hc1 a exlog:Participants-platform,
26   exlog:Participants-service.}
```

Listing 3: Participant registration event.

### 4.3. Privacy events

The rest of the events (rows) in the diagram in Fig. 5 are privacy events, capturing the events of requesting and obtaining access to personal information. The first two of these are the data subject and data sender expressing the privacy preferences and privacy policies applicable to use of the personal information. Listing 4 shows assertions of privacy preferences by Mary. To save space, the header only shows the *who* of this log event, namely Mary, the data subject (line 2). The body describes privacy preferences asserted by Mary using the SCIP ontology module (lines 5–59).

#### 4.3.1. Privacy preferences and obligations

In the motivating scenario, Mary states that *her blood pressure data should be used only for preparing a training plan and by a registered practitioner*. Listing 4 shows how the SCIP ontology is used to encode the context of the applicable privacy preferences described above. Line 6 describes by whom the privacy preferences are expressed using the `scip:expressedBy` property. The quad in line 7 describes the validity time interval of the policies. Note that this time interval makes reference to the same timeline that the L2TAP log refers to (lines 17–20). The legitimate purpose for using the data is encoded in line 9. Mary expresses that the acceptable role for participants who request access is `Practitioner` encoded in line 10, which is an abstract role with two concrete roles as its members: `Clinician` and `Planner` (lines 15–16). The acceptable privilege that will be granted if the obligations are fulfilled is `Use` encoded in line 11. Analogously to the roles and purposes, we express privacy privileges as a lattice with a superclass `scip:PrivilegeAny` on top and `scip:Collect`, `scip:Use` and `scip:Disclosure` as the immediate subclasses [91].

```
1  exlog:logevents-e3 a l2tap:PrivacyEvent;
2    l2tap:eventParticipant exlog:participants-Mary;
3  . . .
4  exlog:logng-ng3 {
5    exlog:pprefs-pp1 a scip:PrivacyPreference;
6      scip:expressedBy exlog:participants-Mary;
7      scip:hasValidity exlog:pprefs-pp1-ti1;
8      scip:eligibleDataItem exphr:Mary-BP;
9      scip:eligiblePurpose exphr:Treatment;
10     scip:requestorRole exphr:Practitioner;
11     scip:eligiblePrivilege exphr:Use;
12     scip:contractedObligation exphr:obs-ob1, exphr:obs-ob2,
13     exphr:obs-ob3;
14     scip:propositionalExpression exphr:obs-exp-phy1 .
15   exphr:Planner rdfs:subClassOf exphr:Practitioner.
16   exphr:Clinician rdfs:subClassOf exphr:Practitioner.
17   exlog:pprefs-pp1-ti1 a tl:Interval;
18     tl:beginsAtDateTime "2016-01-01T00:00:00Z"^^xsd:dateTime;
19     tl:endsAtDateTime "2018-01-01T00:00:00Z"^^xsd:dateTime;
20     tl:onTimeLine exlog:tlphysical.
21   exphr:obs-ob1 a scip:ObligationTemplate;
22     dcterms:title "obtain consent";
23     scip:occurrenceGap exphr:obs-tasks-ob1-ti1;
24     scip:taskDuration exphr:obs-tasks-ob1-ti2;
25     scip:obligationVarName "ob_consent"^^xsd:string.
26   exphr:obs-tasks-ob1-ti1 a tl:Interval;
27     tl:durationXSD "-P2D"^^xsd:duration;
28     tl:onTimeLine exlog:tlDiscrete.
29   exphr:obs-tasks-ob1-ti2 a tl:Interval;
30     tl:durationXSD "P1D"^^xsd:duration;
31     tl:onTimeLine exlog:tlDiscrete.
32   exphr:obs-ob2 a scip:ObligationTemplate;
33     dcterms:title "anonymize data";
34     scip:occurrenceGap exphr:obs-tasks-ob2-ti1;
35     scip:taskDuration exphr:obs-tasks-ob2-ti2;
36     scip:obligationVarName "ob_anonymize"^^xsd:string.
37   exphr:obs-tasks-ob2-ti1 a tl:Interval;
38     tl:durationXSD "P60D"^^xsd:duration;
39     tl:onTimeLine exlog:tlDiscrete.
40   exphr:obs-tasks-ob2-ti2 a tl:Interval;
41     tl:durationXSD "P1D"^^xsd:duration;
42     tl:onTimeLine exlog:tlDiscrete.
43   exphr:obs-ob3 a scip:ObligationTemplate;
44     dcterms:title "delete data";
45     scip:occurrenceGap exphr:obs-tasks-ob3-ti1;
46     scip:taskDuration exphr:obs-tasks-ob3-ti2;
47     scip:obligationVarName "ob_delete"^^xsd:string.
48   exphr:obs-tasks-ob3-ti1 a tl:Interval;
49     tl:durationXSD "P60D"^^xsd:duration;
50     tl:onTimeLine exlog:tlDiscrete.
51   exphr:obs-tasks-ob3-ti2 a tl:Interval;
52     tl:durationXSD "P1D"^^xsd:duration;
53     tl:onTimeLine exlog:log1-tlDiscrete.
54   exphr:obs-exp-phy1 a sp:Filter ;
55      sp:expression [ a sp:and ;
56          sp:arg1 [ sp:varName "ob_consent"^^xsd:string ] ;
57          sp:arg2 [ a sp:or;
58             sp:arg1 [ sp:varName "ob_anonymize"^^xsd:string ] ;
59             sp:arg2 [ sp:varName "ob_delete"^^xsd:string ] ] ] . }
```

Listing 4: Privacy event for privacy preferences.

There are also norms associated with a context that describe obligations or actions that need to be performed. For example, Mary defines:

$ob_1$: *The practitioner at HC must obtain my consent at least two days prior to access*
AND 60 days after access, EITHER
$ob_2$: *HC should anonymize my data*
OR
$ob_3$: *delete my data from the HC server*.

Obligations, included in privacy preferences, are templates for future association of executable obligations that must be performed by the data receivers. The obligation template in the SCIP ontology has a limited expressive power to capture only simple temporal obligations, i.e. obligations that need to be fulfilled sometime before or after access to data is being permitted such as $ob_1$, $ob_2$ and $ob_3$ described above. These obligations are captured as instances of `scip:ObligationTemplate` in lines 21, 32, and 43 in Listing 4. Obligations in SCIP are promises (as speech acts [92]) made by data receivers. The description of action(s) associated with an obligation is expressed in natural language as a resource in Dublin Core Terms [87] with its title captured as an RDFS literal using `dcterm:title` (lines 22, 33, and 44).

The `scip:ObligationTemplate` class has two temporal properties. The `scip:occurrenceGap` property is used to capture the acceptable time of performing an obligation as an integer value on the discrete timeline and its value is represented in XSD (a positive integer indicates occurrence *after* the access activity, and a negative integer *before*). For example, in line 27 we encode the `scip:occurrenceGap` of $ob_1$ as a negative integer $-2$ with the lexical representation in XSD as '−P2D'. The `scip:taskDuration` property is used to encode the time required to perform the obligation and it can refer to a zero integer if the time is negligible. For $ob_1$ we consider the time required to process Mary's consent and to discharge $ob_1$ is 1 day (lines 24, 29 and 30). In line 28 we use `tl:onTimeline` to encode that the duration is interpreted using the discrete timeline that the log is being initialized with. Since the description of $ob_1$ is not clear about at most how many days prior to access the consent should be obtained, the time frame for fulfilment of the obligation is considered from the time the log is being initialized to two days prior to access. However, the subproperties of `scip:occurranceGap` in the SCIP ontology can be used to explicitly capture the start and the end of an acceptable period for performing an obligation. Further discussion on evaluating the occurrence of an obligation relative to the occurrence of an access activity is provided in Section 5.2.1.

The second and third obligations are post-obligations and must be fulfilled after access has been granted. Similar to $ob_1$, we encode these two obligations as instances of `scip:ObligationTemplate` in lines 32–53. Note that we use positive integers to capture the acceptable intervals for fulfilling these obligations. However, these two obligations are disjunctive and fulfilling either of them would be sufficient. To capture the combination effect of multiple obligations, we introduce $\phi$, a propositional expression, that describes how the individual satisfaction of each obligation contributes to the overall compliance of an access request.

The formula $\phi$ is a propositional expression encoded in the small subset of the SPARQL grammar[1] given in Table 1 using the same EBNF notation (we also maintain the rule numbers to facilitate cross referencing the original SPARQL grammar, although the rules have been simplified). The decision to use a subset of the SPARQL grammar to validate the propositional expression $\phi$

[1] https://www.w3.org/TR/sparql11-query/#grammar.

**Table 1**
Subset of SPARQL grammar for $\phi$.

| | | | |
|---|---|---|---|
| [110] | `Expression` | ::= | `ConditionalOrExpression` |
| [111] | `ConditionalOrExpression` | ::= | `ConditionalAndExpression ( '\|\|' ConditionalAndExpression )*` |
| [112] | `ConditionalAndExpression` | ::= | `ValueLogical ( '&&' ValueLogical )*` |
| [113] | `ValueLogical` | ::= | `UnaryExpression` |
| [118] | `UnaryExpression` | ::= | `'!' PrimaryExpression \| PrimaryExpression` |
| [119] | `PrimaryExpression` | ::= | `BrackettedExpression \| Var` |
| [120] | `BrackettedExpression` | ::= | `'(' Expression ')'` |

has two advantages. First, it allows us to use SPIN [93] to encode $\phi$ in the corresponding Privacy Preference. Second, we can use $\phi$ directly within a SPARQL `FILTER` expression that is part of our query-based compliance checking solution (as we describe in Section 5.2.2).

Returning to our example and Listing 4, the $\phi$ for the obligations associated with Mary's privacy preferences is the formula $ob_1 \wedge (ob_2 \vee ob_3)$. We assign an IRI to $\phi$ using `scip:propositionalExpression` (line 14), and describe the expression as an `sp:expression` of a SPARQL `sp:Filter` in lines 54–59. We encode the expression with multiple nested boolean functions. First we use the AND function in SPIN (line 55) with $ob_1$ as its first argument and the rest of $\phi$ as the second argument. In turn, the second argument represents an OR function (line 57) with $ob_2$ and $ob_3$ as its first and second arguments, respectively (lines 58–59). We also need to connect each of these variables to one of the obligations expressed in the privacy preferences. Therefore, we use the `scip:obligationVarName` property to point to a symbolic name equal to the name string we use for the variables in expressing $\phi$ (lines 56, 58 and 59).

The obligation template described above has multiple limitations. The task that needs to be performed for fulfilment of an obligation is expressed as a literal and the semantics of the task are not captured in our ontology. There are also other types of obligations such as obligations with multiple actions, obligations with specific action plans, conditional obligations, cascading obligations, and repeating obligations, described in the related literature (e.g., [52,94,95]), that SCIP cannot capture. However, given the extensible structure of the SCIP ontology if an obligation model with more or less expressive power is required, then the obligation class in the other ontology can be mapped to the obligation template class in the SCIP ontology to exploit the expressive power of the imported obligation model.

#### 4.3.2. Access requests

The `scip:AccessRequest` class is used to encode a request by a service (e.g., BPC or HC) to access Mary's data. SCIP provides multiple properties to allow a requestor to assert the requested data item(s), the purpose of access, the requested privacy privilege, and the requestor's role. An access request can be initiated by a class of participants or an individual participant. In our motivating scenario, we assume one of the services (HC) logs the access request (Listing 5).

```
1  exlog:logevents-e4 a l2tap:PrivacyEvent;
2    l2tap:eventParticipant exlog:participants-hc1;
3  . . .
4  exlog:logng-ng4 {
5    exlog:requests-req1 a scip:AccessRequest;
6      scip:dataRequestor exlog:participants-hc1;
7      scip:dataSender exlog:participants-phr1;
8      scip:dataSubject exlog:participants-Mary;
9      scip:requestedDataItem exphr:Mary-BP;
10     scip:requestedPurpose exphr:Training_plan;
11     scip:requestedPrivilege exphr:Use.
12   exlog:participants-hc1 scip:requestorRole exphr:Clinician.
13   exlog:participants-phr1 scip:senderRole exphr:Platform.}
```

Listing 5: Log event for access request.

Line 5 in the body encodes that the access request is an instance of `scip:AccessRequest`. The `scip:dataRequestor` property in line 6 captures the URI of the data requestor (HC); `scip:dataSender` (line 7) captures who should send the data (PHR), while `scip:dataSubject` (line 8) optionally captures whose data has been requested as the data items may belong to a group of data subjects. Line 9 encodes the URI of the requested data items, line 10 the purpose for accessing data, and line 11 the requested privacy privilege. Roles of the data requestor and sender are presented in lines 12 and 13, respectively.

#### 4.3.3. Access responses

For each access request event there should exist an access response event that encodes (1) access decision and (2) the applicable obligations. Depending on the domain of the application, there could be different access control mechanisms that a data provider uses to determine the boolean access decision (e.g., [96] is a recent proposal for fine grained access control of triple stores). However, deriving obligations from privacy preferences is more challenging as it requires matching the context of access that has been described in an instance of an access request and the context generally described in privacy preferences or policies. In Section 5.1 we describe a query-based solution that uses the transitive closure of `rdfs:subClassOf` to derive obligations from privacy preferences. Obligations can also be derived using a more or a less complex mechanism. Regardless of which mechanism is being used, L2TAP requires logging the access decisions and derived obligations.

```
1  exlog:logevents-e5 a l2tap:PrivacyEvent;
2    l2tap:eventParticipant exlog:participants-phr1_Auth;
3  . . .
4  exlog:logng-ng5 {
5    exlog:responses-res1 a scip:AccessResponse;
6      scip:responseTo exlog:requests-req1;
7      scip:accessDecision "true"^^xsd:boolean;
8      scip:contextObligation exlog:req1-obs-ob1, exlog:req1-obs-ob2,
9      exlog:req1-obs-ob3;
10     scip:contextExpression exlog:req1-exp-phy1.
11   exlog:req1-obs-ob1 a scip:Obligation;
12     scip:associatedWith exphr:obs-ob1.
13   exlog:req1-obs-ob2 a scip:Obligation;
14     scip:associatedWith exphr:obs-ob2.
15   exlog:req1-obs-ob3 a scip:Obligation;
16     scip:associatedWith exphr:obs-ob3.}
```

Listing 6: Log event for access response.

The RDF code of the access response privacy event is shown in Listing 6. The *who* in this log event (line 2) is `exlog:participants-phr1_Auth`, the access authorization agent of the PHR platform who is implementing the access control mechanism. The event's data is an instance of `scip:AccessResponse` class (line 5) and makes reference to the access request logged by the service (HC) using the `scip:responseTo` property (line 6). The event encodes the boolean access decision using the `scip:accessDecision` property (line 7). The quads in lines 8–16 encode the derived obligations that are applicable to the request. The realization of an obligation from the obligation template is captured using `scip:associatedWith` (lines 12, 14 and 16). The $\phi$ formula that expresses how satisfaction of these obligations relates to the overall compliance of the access request is encoded using the `scip:contextExpression` property in line 10.

#### 4.3.4. Accepting and performing obligations

After the access response has been logged, the ontology supports an optional event capturing the fact that the data requestor accepts to perform the obligations (Listing 7). This event refers to the URI of the access response (line 5). By the virtue of logging this event, the service not only acknowledges existence of the obligations but also, as a performative act, commits itself to perform the obligations. The event's participant can make additional assertions beyond the simple acceptance of obligations such as who is going to perform the obligation and who is witnessing the fulfilment of an obligation. These assertions are optional and can be captured by the extended SCIP structure. In this case, the type of derived obligations will be specialized as `scip:AgreedObligation` (line 6) and the `scip:obligationPerformer` and `scip:obligationWitness` properties in lines 7–8 will be used to capture the delegated responsibilities. In our example, Mark, an HC employee, and HCAuditor, one of the HC internal privacy auditors, are registered participants who will perform and witness the obligation, respectively.

```
1 exlog:logevents-e6 a l2tap:PrivacyEvent;
2   l2tap:eventParticipant exlog:participants-hc1;
3 . . .
4 exlog:logng-ng6 {
5   exlog:participants-hc1 scip:accepts exlog:req1-obs-ob2.
6   exlog:responses-req1-ob2 a scip:AgreedObligation;
7     scip:obligationPerformer exlog:participants-Mark;
8     scip:obligationWitness exlog:participants-HCAuditor.}
```

Listing 7: Log event for obligation acceptance.

Listing 8 shows a log event that captures the fact that HC has performed the first obligation. Line 5 asserts that the delegated employee (Mark) fulfils the obligation using the `scip:delegateFulfils` property, which is a subproperty of `scip:fulfils`. The time instant of performing the obligation (using the same timeline as for the log) is encoded using the `scip:obligationOccurredIn` property (lines 6–9).

```
1 exlog:logevents-e7 a l2tap:PrivacyEvent;
2   l2tap:eventParticipant exlog:participants-HC;
3 . . .
4 exlog:logng-ng7 {
5   exlog:participants-Mark scip:delegateFulfils exlog:req1-obs-ob2.
6   exlog:req1-obs-ob2 scip:obligationOccurredIn exlog:e7-t1.
7   exlog:e7-t1 a tl:Instant;
8     tl:atDateTime "2016-02-01T12:00:00Z"^^xsd:dateTime;
9     tl:onTimeLine exlog:tlphysical.}
```

Listing 8: Log event for performing an obligation.

#### 4.3.5. Access data

Finally, SCIP has a class named `scip:ActualAccess` which is a subclass of `scip:AccessActivity` and records the occurrence of the actual access. Listing 9 shows the log event of an access when HC (line 2) accessed Mary's health data. Line 6 refers to the URI of the access request event using `scip:accessFor`. Lines 7–10 capture the time instant that the access activity occurred.

```
1 exlog:logevents-e8 a l2tap:PrivacyEvent;
2   l2tap:eventParticipant exlog:participants-HC;
3 . . .
4 exlog:logng-ng8 {
5   exlog:access-req1-ac1 a scip:ActualAccess;
6     scip:accessFor exlog:requests-req1;
7     scip:accessOccurredIn exlog:e8-t1.
8   exlog:e8-t1 a tl:Instant;
9     tl:atDateTime "2016-02-03T12:00:00Z"^^xsd:dateTime;
10    tl:onTimeLine exlog:tlphysical.}
```

Listing 9: Log event for access activity.

### 4.4. Relationship with PROV-O

In this subsection, we provide a partial mapping from the L2TAP ontology to the classes and properties of PROV-O [25] to capture the provenance assertions of an L2TAP log event. We use the log event capturing Mary's privacy preferences to illustrate the mapping (as shown in Listing 10). In lines 1 and 3, we map `l2tap:Log` and `l2tap:LogEvent` to the `prov:Collection` and `prov:Entity` concepts, respectively. The `l2tap:Participant` can also be mapped to `prov:Agent` (line 8). Then the *who* of the log event is expressed using the `prov:wasAssociatedWith` property (line 10) and the *what* of the log event is captured using `prov:value` (line 6). The `prov:value` property can refer to an IRI as a constant, whose interpretation is outside the scope of PROV. This definition is consistent with our approach where L2TAP-core does not require to interpret the triples inside the named graph of a log event.

There are provenance aspects of a privacy log event that cannot be captured using the PROV ontology. In PROV, any change made by an agent on an entity should be through a `prov:Activity`. Therefore, capturing the participant of a log event (the *who*) requires first adding a triple to define the URI of the act of generating the log event as a `prov:Activity` (line 9) and then associating the activity with a `prov:Agent` (line 10). More importantly, the fact that the PROV ontology does not make a distinction between the concepts of *application time* versus *system time* [78,79] prevents us from capturing the *when* of a log event using PROV. Therefore, in lines 4–5, we use the `l2tap:receivingTimestamp` and `l2tap:publicationTimestamp` properties to capture the temporal properties of the event.

```
1  exlog:log1 a l2tap:Log, prov:Collection;
2    prov:hadMember exlog:logevents-e9.
3  exlog:logevents-e9 a l2tap:LogEvent, prov:Entity;
4    l2tap:receivingTimestamp exlog:e9-t1;
5    l2tap:publicationTimestamp exlog:e9-t2;
6    prov:value exlog:logng-ng9;
7    prov:wasGeneretaedBy exlog:logActivity-act3.
8  exlog:participants-phr1 a l2tap:Participant, prov:Agent.
9  exlog:logActivity-act3 a prov:Activity;
10   prov:wasAssociatedWith exlog:participants-phr1;
11   prov:qualifiedAssociation [
12       a prov:Association;
13       prov:agent :mark;
14       prov:hadRole :security-expert;
15       prov:hadPlan :security-guide;
16       rdfs:comment "mark followed the tamperEvident guide" ].
17       exlog:logng-ng9 {
18   exlog:pprefs-pp9 a scip:PrivacyPreference; //...continued }
```

Listing 10: Using PROV concepts in a log event.

Despite the shortcomings of PROV, the partial mapping allows a privacy log, published using the L2TAP ontology, to be extended using other properties in PROV. For example, we used the PROV activity association (line 12) and `prov:agent`, `prov:hadRole`, and `prov:hadPlan` properties (line 13–15) to assert that the logger had assigned the responsibility of making the log event tamper evident to Mark (a `prov:Agent`). Mark was a security expert (line 14) and he had followed the assigned guideline (line 15).

### 4.5. Logging XACML policies and events

XACML is an authorization language that supports the expression of complex access policies. The XACML implementation [56] realizes the XACML policy language. The implementation consists of multiple components such as Policy Information Point (to document the policies), Policy Enforcement Point (PEP) and Obligation Service [15] that deal with the real-time processes of deciding and

enforcing whether a subject has accessed a particular resource in particular circumstances. XACML does not support a privacy audit log in its implementation as everything is performed in real-time. Our model supports auditing, which is an off-line enforcement mechanism. In this subsection we describe how different documents generated by the XACML implementation can be expressed by L2TAP and logged. Supplementing with L2TAP extends the XACML capabilities as it allows real-time events generated by XACML to be logged by L2TAP for post auditing purposes. The logging feature is particularly important for organizations that are using XACML for authorization purposes but cannot enforce post-obligations [52]. In addition, expressing XACML documents using L2TAP supports the extensibility of our approach as we can leverage advantages of granular access policies expressed in XACML policies (e.g., the GeoXACML extension standardized by OGC [97]).

We use our motivating scenario described in Section 1 to illustrate a lightweight transcoding from the XACML documents to the log events generated using the L2TAP ontology but we limit the illustration to only the log event of expressing privacy preferences both as an XACML document (Listing 11) and as an L2TAP privacy event (Listing 12).

```
1 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:conformance-
  test:IIA1:rule" Effect="Permit">
2 <Description> A user may access my daily BP for the purpose of an
  exercise plan, provided the user is an instructor. </Description>
3 <Target> <Subjects> <Subject>
4 <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
  equal">
5 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  Instructor </AttributeValue>
6 <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2
  .0:subject:role" DataType="http://www.w3.org/2001/XMLSchema#string"/
  >
7 </SubjectMatch> </Subject> </Subjects>
8 <Resources> <Resource>
9 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI
  -equal">
10 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
  https://dphr.org/users/Mary/BP</AttributeValue>
11 <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1
  .0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#
  anyURI" />
12 </ResourceMatch> </Resource> <Resource>
13 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0
  :function::regexp-string-match">
14 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  Training-plan</AttributeValue>
15 <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1
  .0:resource:purpose" DataType="http://www.w3.org/2001/XMLSchema#
  string" />
16 </ResourceMatch> </Resource> </Resources>
17 <Actions> <Action>
18 <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
  equal">
19 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  Read</AttributeValue>
20 <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0
  :action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string
  " />
21 </ActionMatch> </Action> </Actions> </Target>
22 <Condition FunctionId="urn:oasis:names:tc:xacml:2.0:function:regexp-
  string-match">
23 <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2
  .0:resource:purpose" DataType="http://www.w3.org/2001/XMLSchema#
  string" />
24 <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0
  :action:purpose" DataType="http://www.w3.org/2001/XMLSchema#string"
  />
25 </Condition>
26 </Rule>
```

Listing 11: Privacy Preferences in XACML.

XACML defines a number of standard variables such as "subject-id", "resource-id", and "action-id", that can be extended to express complex access rules in which variables are used in predicates. The policy fragment of Mary's first privacy rule in XACML is shown in Listing 11. Line 5 in this serialization expresses the role of the data requestor (an instructor). Similarly, the purpose is defined in line 14. The constraint over these two variables are defined in lines 23–25. The constraint requires that the string values of the purpose and the requestor role must be equal to the string values of the counterpart variables in an XACML request document.

The same XACML rule is illustrated in Listing 12 using L2TAP. Note that the coding of Mary's privacy preferences in this listing is similar to what we observed in Listing 4. The `scip:sourcePreference` and `scip:sourcePreferenceLanguage` properties are added to encode the original XACML policy document as part of the event data. Similarly, other documents generated by PEP, using *Request*, *Response*, and *Obligation* elements of XACML, can be logged using L2TAP Access Request, Access Response, Obligation Acceptance, Performing Obligation, and Access Activity. Note that due to the real-time nature of the XACML policy evaluation, all L2TAP events related to an access including acceptance and fulfilment of pre-obligations will be logged at one timestamp.

```
1 exlog:logevent-e10 a l2tap:PrivacyEvent;
2   l2tap:memberOf exlog:log1;
3   l2tap:eventParticipant exlog:participants-Mary;
4   l2tap:eventTimestamp "2016-09-26T12:00:00Z"^^xsd:dateTime;
5   l2tap:publicationTimestamp "2016-09-26T12:01:00Z"^^xsd:dateTime;
6   l2tap:eventData exlog:logng-ng3.
7 exlog:logng-ng10 a rdfg:Graph.
8 exlog:logng-ng10 {
9   exlog:pprefs-pp10 a scip:PrivacyPreference;
10  scip:expressedBy exlog:participants-Mary;
11  scip:dataItem exphr:users-Mary-BP;
12  scip:purpose exphr:Training_plan;
13  scip:requestorRole exphr:Planner;
14  scip:privacyPrivilege exphr:Read;
15  scip:sourcePreference "...."^^ xsd:string;
16  scip:sourcePreferenceLanguage "oasis:names:tc:xacml:2.0:policy:
        schema:os". }
```

Listing 12: Privacy Preferences in L2TAP.

## 5. Mechanisms for querying L2TAP logs

In this section we describe SPARQL query based solutions for the competency questions we have put forward to design the L2TAP ontology. For the sake of brevity we will describe only the complex queries and leave the formulation of other queries on the ontology website (http://l2tap.org). In Sections 5.1 and 5.2, we describe SPARQL solutions for obligation derivation and compliance checking, respectively. In Section 5.3 we describe the complexity of the algorithm we devised for compliance checking. In Section 5.4 we discuss how the boundary situations in an access request compliance check can be evaluated. Finally, in Section 5.5 we discuss the validity and completeness conditions for a reliable L2TAP log.

### 5.1. Obligation derivation

Deriving obligations from a data subject's privacy preferences or from an organization's privacy policies requires finding all applicable policies and then extracting obligations from them. To find the applicable policies we match the context expressed in privacy preferences with the context described in an access request. As

we described before, in Contextual Integrity [22] the context is defined by the following properties, (1) the roles of the three main participants (data requestor, data sender, and data subject), (2) the data item that is going to be accessed, (3) the purpose for which the data item will be used, and (4) the type of privacy privilege that will be granted.

```
1  INSERT {GRAPH @g1 {
2      ?ob a scip:Obligation;
3          scip:associatedWith ?obTmp.
4      ?response scip:responseTo @request.
5      ?response scip:contextObligation ?ob;
6          scip:contextExpression ?phi.}}
7  WHERE {GRAPH @g2 {
8      @request a scip:AccessRequest;
9       scip:dataSubject ?req_dsubject;
10      scip:dataRequestor ?req_requestor;
11      scip:dataSender ?req_sender;
12      scip:requestedPurpose ?req_pur;
13        scip:requestedDataItem ?req_di;
14        scip:requestedPrivilege ?req_priv.
15     ?req_dsubject scip:dataSubjectRole ?req_dsurole.
16     ?req_requestor scip:requestorRole ?req_drrole.
17     ?req_sender scip:senderRole ?req_dserole.
18     ?preference a scip:PrivacyPreference;
19         scip:expressedBy ?preference_subject;
20         scip:eligiblePurpose ?pref_pur;
21         scip:requestorRole ?pref_drrole;
22         scip:senderRole ?pref_dserole;
23         scip:dataSubjectRole ?pref_dsurole;
24         scip:eligiblePrivilege ?pref_priv;
25         scip:eligibleDataItem ?pref_di;
26         scip:contractedObligation ?obTmp;
27         scip:propositionalExpression ?phi.
28     ?req_dsubject rdfs:subClassOf ?preference_subject.
29     ?req_di rdfs:subClassOf ?pref_di.
30     ?req_priv rdfs:subClassOf ?pref_priv.
31     ?req_pur rdfs:subClassOf ?pref_pur.
32     ?req_dsurole rdfs:subClassOf ?pref_dsurole.
33     ?req_drrole rdfs:subClassOf ?pref_drrole.
34     ?req_dserole rdfs:subClassOf ?pref_dserole.
35     BIND (?obTmp AS ?ob)}}
```

Listing 13: The SPARQL query template for obligation derivation.

The SPARQL `INSERT` query in Listing 13 is used for context matching and automatically generating the derived obligations as an RDF graph. The context of the access request is included in lines 8–17 and the context associated with the privacy preferences appear in lines 18–27. Note that the query is parameterized with the URIs of an access request (`@request`), the named graph that the obligations will be inserted to (`@g1`) and the named graph that the obligations will be derived from (`@g2`). The data subject's URI is expressed in line 9. Line 19 includes the URIs of all participants who expressed some privacy preferences. Using the class hierarchy, we can infer applicable privacy preferences either expressed by the data subject or by an instance of a superclass (e.g., the PHR platform) (line 28).

The matching for the other properties of the context are also based on subclassing. For example, using the data item hierarchy, the pattern in line 29 makes sure that the requested data item is a subclass of the data items for which the privacy preferences are applicable. Since purpose, privacy privilege, and roles are also expressed using a lattice, the graph patterns in lines 29–34 check that all the items expressed in the access request graph are subsumed (using the transitive closure of `rdfs:subClassOf`) by the corresponding items in the privacy preference graph. It is possible for the query result to be an empty set, meaning that no applicable privacy preferences are found for a given access request. We will discuss this case in Section 5.4.

We understand that the proposed query solution is limited in deriving more complex obligations such as conditional obligations or obligations that their fulfilment depends on the fulfilment of other obligations [52,95]. The extensible structure of the L2TAP ontology allows a more expressive obligation model to be adapted.

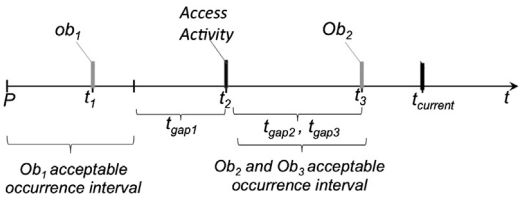


**Fig. 6.** Timeline for occurrences of obligations and an access activity.

### 5.2. Query-based compliance checking

The key to privacy auditing is to identify, at any given point in time, if an access request is in compliance with the applicable privacy preferences and policies. Compliance is determined by the status of the logged obligations.

#### 5.2.1. Evaluating individual obligations

We developed a SPARQL query for resolving the three states of an obligation: *pending*, *fulfilled*, and *violated*. The time model we use to determine the state of an obligation is a simple representation of numbers in $\mathbb{Z}$, totally ordered by $\leq$ [52]. As we have shown in encoding privacy preferences (Listing 4) we capture the occurrence time of post-obligations and pre-obligations using positive and negative values, respectively, relative to the time that the actual access activity occurred.

Fig. 6 shows the progress of time on the timeline of the log and occurrence of obligations and access activities during the life cycle of an access request for our motivating scenario. In this example, $P$ is the start time of the timeline, $t_1$ is the time that $ob_1$ occurred, $t_{gap1}$ is the occurrence gap for $ob_1$ (we encoded this value as a negative integer), $t_2$ is the occurrence time of the access activity, $t_{gap2}$ and $t_{gap3}$ are occurrence gaps for $ob_2$ and $ob_3$, respectively (we encoded these values as positive integers), $t_3$ is the occurrence time of $ob_2$, and $t_{current}$ is the time that we are evaluating the fulfilment of obligations. In this figure we also show the acceptable intervals for performing each obligation.

Formulas (1) and (2) are the temporal formalization that we use in the SPARQL obligation evaluation queries. In this formula the latest acceptable time interval for fulfilment of an obligation $ob$ is expressed as $(t_{start}, t_{end})$. This acceptable time interval is relative and is a function of $t_{access}$: the occurrence time of the corresponding access activity, $t_{gap}$: the acceptable time gap that the performance of the obligation can start before or after $t_{access}$, and $t_{ob}$: the time duration that $ob$ takes to complete.

$$t_{start} = t_{access} + t_{gap} - t_{ob} \quad (1)$$

$$t_{end} = t_{start} + t_{ob} \quad (2)$$

Given the information above, we can devise three different SPARQL `ASK` queries to find the state of an obligation: $ob_i$ is a *pending* obligation if, at the point in time $t_{current}$, $ob_i$ has not been performed and $t_{current} \leq t_{end}$; $ob_i$ is *fulfilled* if $ob_i$ has been performed and $t_{end} \leq t_{current}$; if $ob_i$ is not pending or fulfilled then it is considered *violated* (authors in [98] make a distinction between a violated and invalid obligation, but in this proposal we assume all logged obligations are valid). For example, in Fig. 6, if $P$ is zero, $t_{access}$ is 10 and $t_{current}$ is 30 then the state of $ob_2$ is acceptable to be *pending* since using the above functions the pair $(t_{start}, t_{end})$ will be substituted with (69, 70) given $t_{gap2} = 60$ (days) and $t_{ob_2} = 1$ (day). However, the state of $ob_1$ must be fulfilled, since $t_{gap1} = -2$ (days) and $t_{ob_1} = 1$ (day) and $(t_{current} = 30) \not\leq (t_{end} = 8)$ for $ob_1$.

```
1  ASK
2    WHERE {
3      ?response scip:responseTo ?request.
4      ?response scip:contextObligation @obligation.
5      @obligation rdf:type scip:Obligation;
6      scip:associatedWith ?obTemplate.
7      ?obTemplate scip:occurrenceGap ?ginterval.
8      ?obTemplate scip:taskDuration ?tinterval.
9      ?ginterval tl:durationXSD ?occGap.
10     ?tinterval tl:durationXSD ?pD.
11     OPTIONAL {?accessActivity scip:accessFor ?request} .
12     OPTIONAL {?accessActivity scip:accessOccurredIn ?acinstant} .
13     OPTIONAL {?acinstant tl:atDateTime ?accessTime} .
14     OPTIONAL {?performerAgent scip:delegateFulfils @obligation.}
15     OPTIONAL {@obligation scip:obligationOccurredIn ?obendinstant.}
16     OPTIONAL {?obendinstant tl:atDateTime ?obligationTime.}
17     FILTER ((!bound(?performerAgent) && !bound (?accessTime))||(!
           bound(?performerAgent) && bound(?accessTime) && (xsd:
           integer(@currentTime) <= fn:max((xsd:integer(?accessTime)
           + xsd:integer(?occGap) - xsd:integer(?pD)), (xsd:integer(?
           accessTime) + xsd:integer(?occGap))))))}
```

Listing 14: $Q_{ob}$: SPARQL query for evaluating whether an obligation is pending.

The query shown in Listing 14 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`). Formulas (1) and (2) are translated to the `FILTER` conditions in line 17 of this query. Note that representing time as an integer provides a convenient and simple way of comparing occurrences of obligations and access activities, however the modular structure of L2TAP allows using an alternative, more complex, time model. A necessary condition for the log to be valid is consistency of the temporal ontology adapted across all L2TAP modules. If, for instance, an application domain uses the axiomatized OWL time ontology [99,100] as its temporal model, then the logical expression in line 11 (Listing 14) will be replaced with a propositional variable (e.g., `@obligation_inside`) whose truth value will be determined by the adapted temporal model ($T_{owltime\_inside}$). With this approach, the general structure of the compliance queries will not change, rather it will be tailored to properly compare time intervals in the plugged time ontology. Note that the SPARQL queries to check if an obligation is *fulfilled* or *violated* have similar structure to Listing 14 with the `FILTER` fragment changed to capture the formalism described earlier in this section and in Formulas (1) and (2). The query formulation for *fulfilled* and *violated* obligations are available on the ontology website (http://l2tap.org).

### 5.2.2. Evaluating combination of obligations

If there is more than one obligation, the process of determining the access request compliance is implemented in two steps, first evaluating the state of every individual obligation in a given point in time $t$, and then evaluating the propositional formula $\phi$ that describes how fulfilment of each individual obligation contributes to the overall compliance of the access request. Formula (3) shows the $\phi$ formula associated with Mary's preferences. We evaluate $\phi$ by substituting each obligation in this formula with a propositional variable whose truth value is determined based on the answer of the $ASK_{fulfilled}(ob_i)$ query. An active request may have some pending obligations. Therefore, the access request might be currently in compliance, but it may fail to be in compliance in the future due to the violation of a pending obligation. In the timeline shown in Fig. 6, if we assume that the current time is 80 and the data holder has completed $ob_1$ in $t_1 = 5$, $ob_2$ in $t_3 = 70$, and has not performed $ob_3$ but has accessed Mary's data in $t_2 = 10$, then the evaluation of $\phi$ will be as shown in Formula (4).

$$\phi_t \equiv ob_1 \wedge (ob_2 \vee ob_3) \tag{3}$$

$$\phi_{80} \equiv T \wedge (T \vee F) \equiv T \tag{4}$$

We can now formalize in Algorithm 1 the steps of evaluating $\phi$ using a set of SPARQL queries to determine overall compliance of an access request. Input parameters for this algorithm are the access request $rq$ and current time $t$.

**Data**: Access request: $rq$, currentTime: $t$
**Result**: Boolean compliance value for $rq$

1 $OB \leftarrow deriveObligations(rq)$ ;
2 $OBV \leftarrow extractObligationVariableNames(rq)$ ;
3 **foreach** $ob_i \in OB$ **do**
4 $\quad o_i \leftarrow ASK_{fulfilled}(ob_i)$ ;
5 $\quad BIND \leftarrow BIND \| $ 'BIND' $\|$ '(xsd:boolean(' $\|$ $string(o_i)$ $\|$ 'AS ' $\|$ $string(obv_i)$ $\|$ ')' ;
6 **end**
7 $IRI \leftarrow extractPropositionalFormula(rq)$ (Listing 15) ;
8 $Triples \leftarrow extractSPINTriples(IRI)$ ;
9 $FILTER \leftarrow convertToSPARQLFilter(Triples)$ ;
10 Write *FILTER* and *BIND* to line 6 and 7 of Compliance ASK Query $Q_{rq}$ in Listing 16, respectively;
11 **return** ($answer\_of$ Query $Q_{rq}$)

**Algorithm 1:** Compliance checking algorithm

```
1  SELECT DISTINCT ?phi
2  WHERE {
3    ?response scip:responseTo @request.
4    ?response scip:contextExpression ?phi.}
```

Listing 15: $Q_{\phi}$: SPARQL query returning the propositional formula for a given request.

The first step in this algorithm (lines 1–2) is deriving all obligations and their variable names associated with the given access request ($rq$) from privacy policies as described in Section 5.1. Then inside a loop (lines 3–6) for each obligation $ob_i$, the truth value obtained from the SPARQL $ASK_{fulfilled}$ query will be assigned to the propositional variable $o_i$. Then the SPARQL `BIND` fragment is produced by concatenating the truth value of each $ob_i$ with the extracted variable name for each obligation ($obv_i$). After all obligations have been evaluated and the BIND fragment is generated, in line 7, we retrieve the IRI of the propositional expression $\phi$, using Listing 15 . In line 8, we use a path expression of the form (`sp:and | sp:or | sp:not | sp:arg1 | sp:arg2 | sp:varName`)* to return all the SPIN triples forming the tree of $\phi$, using the retrieved IRI as the root. In line 9 of this algorithm, we call a parser function (e.g., [101]) and convert SPIN triples, describing $\phi$, to the SPARQL `FILTER` fragment. In line 10, we use the SPARQL query template shown in Listing 16 ($Q_{rq}$). We add the FILTER query fragment (generated in line 9) and the BIND fragment (generated in lines 3–6) to this query template (line 6) and then we bind the truth value of each $ASK_{fulfilled}$ query to the boolean variable of the $\phi$ formula presented in the `FILTER` condition. Note that in lines 5 and 6 of this query we will have two `FILTER` statements: one for $\phi$ and one for `?accessDecision`. Therefore, if the access decision logged by the access response event is false, even if all obligations are fulfilled, the access request will be noncompliant.

```
1  ASK
2  WHERE {
3    ?response scip:responseTo @request .
4    ?response scip:accessDecision ?accessDecision .
5    FILTER (xsd:boolean(?accessDecision))
6    # query fragment from Line 6 Algorithm 1
7    # BIND(o_i (from ASK_fulfilled) AS ?ob_i)
8      }
```

Listing 16: $Q_{rq}$: SPARQL query for evaluating access request compliance.

Back to our example, we used the template in Listing 16 and generated the compliance query for Mary's privacy preferences in Listing 17. In line 6 of Listing 17 the equivalent `FILTER` fragment for the $\phi$ formula in Eq. (3) is generated. Then in lines 7–9, we bind the truth values obtained from the $ASK_{fulfilled}$ queries to the obligation variables ($ob_i$). Alternatively, instead of evaluating separate ASK queries and including the results using `BIND`, we could generate (and evaluate) a single SPARQL query in a modified

Listing 17 . To do this, Algorithm 1 is changed to generate one EXISTS for each $ob_i$ directly in the FILTER expression for $\phi$. We should then replace the obligation variable for the corresponding EXISTS expression with the contents of the WHERE in the $ASK_{fulfilled}$ query (similar to Listing 14 for $ASK_{pending}$). Note that if we take this alternative approach, we should also modify the ASK query in Listing 14 and substitute OPTIONAL and !bound with FILTER and NOT EXISTS.

```
1 ASK
2 WHERE {
3   ?response scip:responseTo exlog:requests-req1 .
4   ?response scip:accessDecision ?accessDecision .
5   FILTER (xsd:boolean(?accessDecision))
6   FILTER (xsd:boolean(?ob_consent) && (xsd:boolean(?ob_anonymize)
        || xsd:boolean(?ob_delete)))
7   BIND (xsd:boolean(true) AS ?ob_consent)
8   BIND (xsd:boolean(true) AS ?ob_anonymize)
9   BIND (xsd:boolean(false) AS ?ob_delete)}
```

Listing 17: $Q_{rq}$: Example SPARQL compliance query.

### 5.3. Compliance query growth analysis

The complexity of the compliance query we are generating using Algorithm 1 is a function of the number of derived obligations. The order of growth will determine if the algorithm can scale up when a large number of obligations are derived from the privacy policies. We start by presenting the compliance checking procedure with the time cost of each statement in the algorithm. When a statement returns query results, we decompose the corresponding query to determine the cost model of the query.

The input size (number of obligations) is $n$. The cost for the statements in lines 1 and 2 is a constant $c_1$. The statements in lines 3–6 include a *for loop* executed $n$ times (number of obligations). The cost of running the query in Listing 14 can be denoted as $\sum_{i=1}^{9} c_i^o$ as the cost is the sum of one constant time value of finding the pattern, seven constants for the cost of each operation in the FILTER condition and one constant for concatenating each truth value assignment to the BIND fragment [102]. Line 7 returns $\phi$ using the query in Listing 15 where the cost of running the query is considered a constant $c_2$, as the query only contains a simple graph pattern matching without any FILTER condition and independent to the number of obligations. The cost associated with extracting the SPIN triples (line 8) and converting to SPARQL FILTER (line 9) is $\sum_{i=1}^{n} c_i^s$ since the length of the $\phi$ formula will increase as the number of obligations increases. The operation in line 10 is a simple concatenation and the associated cost is a constant $c_3$. The cost associated with running Query $Q_{rq}$ in Listing 16 is $\sum_{i=1}^{n+1} c_i^q$ as the number of operations in the FILTER condition is the sum of the number of propositional variables in the $\phi$ formula (number of obligations) plus one operation checking the access condition. Therefore, the approximation of the algorithm's running time is:

$$T(n) = c_1 + (\sum_{i=1}^{8} c_i^o)n + c_2 + \sum_{i=1}^{n} c_i^s + c_3 + \sum_{i=1}^{n+1} c_i^q$$

We can observe that the running time of the algorithm is a linear function of the number of obligations. In Section 6 we experimentally validate the linear behaviour of the algorithm when the number of obligations is growing.

### 5.4. Boundary situations by obligation derivation

The $\phi$ formula and the SPARQL query-based solution for obligation derivation also support the second principle of Privacy by Design [91] to make protection of privacy as the default setting. According to this principle, waiving conditions and obligations should be explicitly expressed in privacy preferences. To avoid the interpretation of lack of derived obligations as an access without any condition, we add a constant $\bot$ conjunctively to the $\phi$ formula. For example, in our scenario, if Mary does not define any privacy preferences, her data will be considered *private* by default; the $\phi$ formula will always be evaluated as false and any access will be considered non-compliant (the upper bound).

In our ontology design, the lattice representation of roles, purposes, data items, and privacy privileges have a top superclass *Any* on top (e.g., scip:PurposeAny for purpose hierarchy). If an individual actually wants to provide access to her data without imposing any obligations, then the participant(s) who defines privacy preferences can state that for *any* purpose, *any* privacy privilege, data senders and requestors in *any* roles can communicate *any* data items of the data subject. In this case, we add a constant $\top$ to the formula $\phi$ disjunctively, allowing access request in such context to be always compliant (the lower bound). By supporting the upper and lower bounds of compliance we are able to express the dichotomy of *private* and *public* contexts, where the default context will always be considered *private*.

### 5.5. L2TAP log validity and completeness

While the L2TAP ontology provides the necessary vocabularies to encode and publish the log of privacy events as Linked Data, it does not impose any constraints on how these vocabularies should be used in order to generate a *valid* and *complete* log. One aspect of the log validity is its integrity, i.e., making a log tamper-evident. For this purpose, there are proposals applicable to logs in general (e.g., [39]) that can be adopted for the L2TAP privacy audit log. In [103], we proposed a blockchain based approach for maintaining the L2TAP log integrity. Another aspect of the log validity is the consistency of the log. Each L2TAP module and the union of modules must satisfy four groups of validity constraints: (1) *cardinality constraints* (e.g., there exists only one log initialization event per L2TAP log or a log event cannot have more than one event participant), (2) *uniqueness constraints* (e.g., each log event has a unique URI), (3) *ordering constraints* (e.g., the log initialization event has precedence to all other log events or a participant should be registered before contributing a log event), and (4) other constraints that we generally call *impossibility constraints* to enforce requirements for preventing existence of a certain pattern of graphs in a valid L2TAP log instance (e.g., URIs of concepts are disjoint from the URIs of properties).

An intuitive way of ensuring validity constraints is to see the privacy log as a relational database and the validity constraints as integrity constraints over the database schema. However, we must address multiple challenges when we represent validity constraints of an RDF store as a relational database integrity constraints [104]. For example, in a relational type log, the closed world assumption (CWA) is presumed while it is not assumed in the RDF type log. In addition, the OWL semantic, which does not make the unique name assumption (UNA), is problematic in an RDF log, as it violates the relational type primary key constraint required for the log entries. One workaround to address these types of problems is running a set of SPARQL queries to check uniqueness and other integrity constraints prior to adding a new instance into the log (as described in [105,106]). We support validity in L2TAP, by enforcing conformity of all modules with the OWL 2 RL profile [90]. In addition, we express the four groups of constraints as SWRL [65] rules that must be imposed prior to populating the log (the rules are available online at http://l2tap.org).

Another requirement of a trustable privacy log is having a way of guaranteeing that information in the log is complete, thus the log can be trusted. Analogous to relational databases, the quality of data are directly affected by its completeness. Halevy [107] demonstrated that in an incomplete database, when some assertions are

**Table 2**
Log datasets and elapsed execution times.

| Acc. Req. | Triples ($10^6$) | Execution time (s) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
| 10K | 1.69 | 3.7 | 30.3 | 35.0 | 27.4 | 39.7 | 239.5 |
| 20K | 3.37 | 4.8 | 127.7 | 99.7 | 79.8 | 50.1 | 315.2 |
| 50K | 8.43 | 5.7 | 136.7 | 153.1 | 128.3 | 85.7 | 322.6 |
| 100K | 16.87 | 5.8 | 326.6 | 340.5 | 308.1 | 140.7 | 333.0 |
| 400K | 67.46 | 6.3 | 865.5 | 569.7 | 852.8 | 506.6 | 384.7 |
| 1000K | 168.65 | 6.5 | 1933.3 | 603.2 | 2189.0 | 511.6 | 440.4 |

made about database relations, it can be inferred whether a query returns complete answers. In our approach, we are not formalizing the completeness statements as in relational databases [108], but we acknowledge that this is an important aspect of a trustable log and in an analogous way we want to give the logger the opportunity to assert certain completeness statements about the log by virtue of saying them in the log initialization event. The `l2tap:assertsCompleteness` captures this performative act by dereferencing a document (a resource) with its title described as a literal in the Dublin Core vocabulary [87].

## 6. Experimental validation

This section describes an extended scalability validation of our privacy audit model using two experimental evaluations. In the first experiment, we run six representative SPARQL compliance queries against an L2TAP audit log with different dataset sizes where the number of obligations per access request is constant. In the second experiment, we run one compliance query against a dataset with a constant number of access requests but with a growing number of obligations. In Section 6.1 we describe how the datasets are generated. In Section 6.2, the characteristics of the test environment are reported. In Section 6.3 the experimental results of the two experiments are explained. To support reproducibility of experiments, all generated datasets, representative compliance queries used for the experiments, and the specifications of the test environment are available online at http://l2tap.org.

### 6.1. Dataset

A DBpedia query 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 [109]). Each row in this log represents a `SELECT` query imposed on a specific DBpedia resource. We extracted the URIs of the DBpedia resources from the log. The resulting list of URIs is used as input by a custom developed Java application (SynthSCIP) that outputs statements that model a hypothetical audit log scenario. Each input URI is randomly assigned to a pool of 100 data subjects representing owners of the corresponding DBpedia resources. Statements for access requests are then generated using a random date (and a random access request interval of a few days), using the DBpedia URI as the `scip:requestedDataItem` value.

Next, for each access request, SynthSCIP generates five pre-obligations ($ob_1, \ldots, ob_5$) and ten post-obligations ($ob_6, \ldots, ob_{15}$). To construct the $\phi$ formula for each access request, SynthSCIP first picks a random pre-obligation index (a number between 1–5) and creates a disjunctive expression for all obligations with indices smaller than the selected index and then creates a conjunctive expression for the rest. The two logical expressions are then combined as one conjunctive expression. Likewise, for the post-obligations, SynthSCIP picks a random index between 6–15 and creates the disjunctive and conjunctive expression accordingly. For example, if the random index for pre-obligations is 3 and for the post-obligations is 9, then the constructed $\phi$ formula will be as follows:

$$\phi \equiv (ob_1 \vee ob_2 \vee ob_3) \wedge (ob_4 \wedge ob_5) \wedge (ob_6 \vee ob_7 \vee ob_8 \vee ob_9) \wedge (ob_{10} \wedge ob_{11} \wedge ob_{12} \wedge ob_{13} \wedge ob_{14} \wedge ob_{15})$$

SynthSCIP uses a simple probabilistic model to compute the probability of fulfilment of each obligation associated with an access request. The number of fulfilled obligations of a request follows a binomial distribution $Bin(n, p)$ where $n = 15$, $p = 0.98$. Thus, 2% of the obligations are not fulfilled (either due to the obligation not being performed, or due to performance after the obligation interval expires, with each of these two options having equal probability). In summary, SynthSCIP generates approximately 168 triples for each access request, with 18.3% of these access requests being non-compliant with regards to the privacy obligations (this represents a highly non-compliant scenario).

SynthSCIP is executed repeatedly to generate audit log entries for a growing number of access requests as described in Table 2. The first two columns in this table show the number of access requests generated, and the RDF file size respectively. The last six columns show elapsed execution times of SPARQL queries in our experiment described in Section 6.3.

### 6.2. Test environment

The experiment executed the SPARQL queries on a Virtuoso [72] 06.01.3127 installation on an Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-36-generic-pae i686) server with four CPUs (two Dual-Core AMD 2200 MHz, and two Dual-Core AMD 1000 MHZ) and 8.2 GB of memory. The RDF index scheme of our Virtuoso installation [72] consists of two full indices over RDF quads plus three partial indices. The indices are column-wise for: S:subject, P:predicate, O:object, and G:graph. The index scheme consists of: (1) a composite primary key of P, S, O, and G; (2) a bitmap index for lookups on object value of P, O, G, and S; (3) a partial index for cases where only S is specified; (4) a partial index for cases where only O is specified; and (5) a partial index for cases where only G is specified.

The Java `Date.getTime()` method is used to capture the time difference between sending a query to the RDF store on the server over `HTTP` and receiving the results. All the times reported are obtained as the average of the five independent executions to account for variability in the test environment.

### 6.3. Experimental results

We describe two types of experiments. The first experimental evaluation selects six representative compliance queries (using the templates shown in Listings 14 and 16 ). We execute each query against the population of access requests which starts from 10,000 requests and goes up to one million requests. We report the sum of the elapsed execution times of these queries. The first query (Q1) is a simple SPARQL ASK audit query that retrieves whether a random obligation $ob_i$ is performed or not. The second query (Q2) retrieves all pending obligations in the corpus. The third query (Q3) retrieves all fulfilled obligations and Q4 retrieves all violated obligations in

**Table 3**
Execution time with increasing obligations.

| # of obligations | Triples ($10^6$) | Q6 time (s) |
|---|---|---|
| 15 | 8.48 | 3.344 |
| 30 | 15.96 | 5.097 |
| 45 | 23.43 | 7.849 |
| 60 | 30.91 | 10.389 |

the corpus alongside with the URI of the participants who were expected (but failed) to perform $ob_i$. The fifth query (Q5) retrieves all access activities that occurred in the log in a random month. The last query (Q6) is an implementation of the query described in Listing 16 that returns false for a given access request if the request is not compliant with respect to its derived obligations. The implementation follows the procedure described in Algorithm 1. The reported times for Q6 are the sum of elapsed execution times for checking the compliance of randomly picked 100 access requests after their propositional formulas are constructed based on the answers to $ASK_{fulfilled}$ queries (Q3) for the corresponding obligations. This query (Q6) gives a full picture of the status of privacy non-compliance for the 100 access requests.

The queries (Q1–Q6) are executed in the test environment and elapsed execution times are plotted in Fig. 7. The experiment validates the linear time growth for two complex queries (i.e., Q2 and Q4). For the other four queries, the results validate the sub linear time growth as we observe that the marginal increase in the time for answering these queries declines as the number of access requests increases.

For the second experiment we pick the same query Q6, but this time we run the query in the test environment against a corpus of 50,000 access requests with different numbers of obligations per access request. As the number of obligations increases the propositional formula $\phi$ reflecting the combination of obligations gets more complex. We randomly construct the $\phi$ formula as a boolean combination (with $\wedge$ and $\vee$ operations) of derived obligations for each access request. In order to take into account the variability of generated $\phi$ formulas, we compute the elapsed execution time of Q6 as the average of 100 independent executions as the number of obligations increases from 15 up to 60. The results are shown in Table 3 and the elapsed execution time is plotted in Fig. 8. An access request with 60 associated obligations in this dataset requires evaluation of a logical expression with 60 propositional variables where the value of each variable will be substituted with the answers of $ASK_{fulfilled}$ queries. For example, Q6 for such an access request, in a dataset of 50,000 requests with 60 obligations, can be processed in 10.4 s. This experiment also validates the linear time growth for executing Q6 as the number of obligations increases. However, comparing the results of these two experiments shows that the rate of change for the second experiment is higher. This can be explained by the cost involved for the substitution operation. In the first experiment the number of substitutions is fixed, while in the second experiment it varies depending on the number of obligations.

The results of these two experiments validate the scalability of our solutions and highlight the practical benefits of our approach in supporting a variety of compliance queries. We achieve this desirable behaviour even though our selected compliance queries required the evaluation of multiple joins and complex propositional formulas. While our experiments validate the scalability, we acknowledge the limitation of our experiments as the absolute time required to process a query and the computational power needed to achieve an acceptable response time (as described in Human Computer Interaction guidelines [110]) requires further investigation.

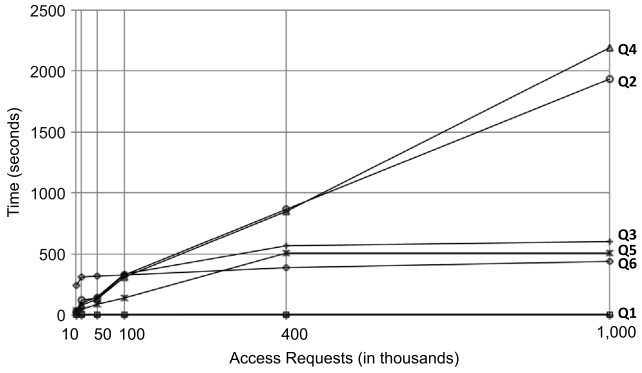


**Fig. 7.** Elapsed execution times for Q1–Q6.

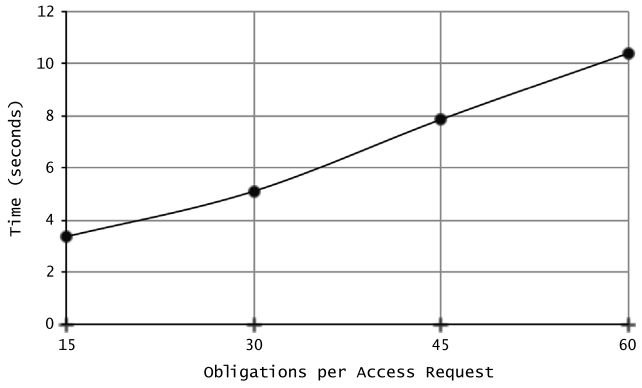


**Fig. 8.** Elapsed execution times for Q6 with increasing obligations.

## 7. Conclusions

This paper describes a privacy model that supports SPARQL query-based privacy auditing processes that execute on logs published as Linked Data using the L2TAP modular ontology. The first three modules, L2TAP-core, L2TAP-initialize and L2TAP-participants, provide provenance-enabled logging of privacy events. The L2TAP–SCIP module provides support for the basic concepts of contextual integrity.

The literature survey (Section 2) identifies a lack of widespread agreed-upon technological solutions to support the privacy audit policies mandated in legislations such as HIPAA [10] and EU Directives [111]. The privacy model proposed here leverages Linked Data to facilitate transparency among participants, relying on the modular ontology structure for extensibility. Publishing L2TAP logs enables the key contribution described in the paper, the implementation of two important privacy auditing tasks, obligation derivation and compliance checking, using an algorithm that generates SPARQL queries while processing L2TAP logs. The paper includes an experimental evaluation that gives evidence of the scalability of this approach.

There are a number of directions for extending the privacy model presented in this paper both from practical and theoretical perspectives. We believe that the L2TAP ontology has the potential to become a basis for the specifications of an industry standard for privacy audit logs. As the first step in this direction, the ontology needs to be extended to capture more complex privacy policies and obligations. Second, the application of the ontology needs to be carried out in a wide range of cross organizational and/or jurisdictional privacy auditing tasks. In fact, the problem of privacy auditing when privacy logs are originated from heterogeneous

environments has been reported in [112] as an important privacy auditing problem. Resolving conflicts between different privacy policies, compliance checking and deriving obligations from multiple privacy policies are challenges that need to be addressed. From a theoretical perspective, we currently reduced the log completeness problem to capturing the loggers' statements as performative acts. Instead of declaring the statements by an oracle, we could partially infer the conditions of a log's completeness. For example, whether the derived obligations are complete can be determined by the virtue of completeness of obligation derivation queries. Formalizing the concept of privacy log completeness is an interesting theoretical expansion of the current research.

## Acknowledgements

Financial support from the NSERC Canada CGS D2-379441-2009 & Discovery RGPIN-2016-06062 and Privacy Awards from IBM and the Information and Privacy Commissioner of Ontario are greatly acknowledged.

## References

[1] D.J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, G.J. Sussman, Information accountability, Commun. ACM 51 (6) (2008) 82–87. http://dx.doi.org/10.1145/1349026.1349043.
[2] C.J. Bennett, International privacy standards: Can accountability be adequate? Priv. Laws Bus. Int. 106 (2010) 21–23.
[3] ENISA, Privacy, Accountability and Trust; Challenges and Opportunities, European Network and Info. Security Agency, 2011.
[4] C. Bizer, T. Heath, T. Berners-Lee, Linked data - The story so far, Int. J. Semant. Web Inf. Syst. 5 (2009) 1–22.
[5] R. Samavi, M.P. Consens, T. Topaloglou, A privacy framework for the personal web, in: The Personal Web, Springer, 2013, pp. 87–112.
[6] M. Chignell, J. Cordy, J. Ng, Y. Yesha, The Smart Internet: Current Research and Future Applications, Vol. 6400, Springer, 2010.
[7] HL7, Consent directive use cases, Community-based collaborative care. http://wiki.hl7.org/index.php?title=Consent_Directive_Use_Cases. (Last accessed June 2015).
[8] Microsoft HealthVault, Microsoft personal health record system, Microsoft, http://www.healthvault.com/. (Last accessed June 2015), (Last accessed June 2016).
[9] K.D. Mandl, I.S. Kohane, No small change for the health information economy, New England J. Med. 360 (13) (2009) 1278–1281.
[10] US Congress, Health Insurance Portability and Accountability Act of 1996, Privacy Rule. 45 CFR 164, 2002.
[11] R. Gajanayake, B. Lane, R. Iannella, T. Sahama, Accountable-ehealth systems: The next step forward for privacy, Electron. J. Health Inform. 8 (2) (2013) 11.
[12] W3C, Extended Log File Format - W3C Working Draft WD-logfile-960323, W3C https://www.w3.org/TR/WD-logfile.html. (Last accessed September 2016).
[13] L. Shields, W. Heinbockel, CEE: Common event expression, in: NIST's 5th Annual IT Security Automation Conference, 2009.
[14] Microsoft, Microsoft Windows MSDN security log, Microsoft. https://msdn.microsoft.com/en-us/library/windows/desktop/aa363658(v=vs.85).aspx. (Last accessed September 2016).
[15] T. Moses, OASIS extensible access control markup language (XACML), OASIS. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. (Last accessed June 2015).
[16] T. Moses, Privacy policy profile of XACML v2.0, OASIS Standard, OASIS. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-privacy_profile-spec-os.pdf. (Last accessed September 2015).
[17] A. Anderson, Core and hierarchical role based access control (RBAC) profile of XACML v2.0, in: OASIS Access Control TC Committee Draft, Vol. 1, 2005, p. 13.
[18] N. Li, H. Chen, E. Bertino, On practical specification and enforcement of obligations, in: Proceedings of the Second ACM Conference on Data and Application Security and Privacy, CODASPY, ACM, New York, NY, USA, 2012, pp. 71–82. http://dx.doi.org/10.1145/2133601.2133611.
[19] W3C, SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/. (Last accessed June 2015).
[20] T. Heath, C. Bizer, Linked data: Evolving the web into a global data space, Synth. Lect. Semant. Web Theory Tech. 1 (1) (2011) 1–136.
[21] W3C, Semantic Web Standards - Resource Description Framework (RDF), Accessed June 2015, W3C, http://www.w3.org/RDF/. (Last accessed June 2015).
[22] H. Nissenbaum, Privacy in Context: Technology, Policy, and the Integrity of Social Life, Stanford University Press, Stanford, CA, USA, 2009.
[23] W3C, RDF Vocabulary Description Language 1.0: RDF Schema, W3C. http://www.w3.org/TR/rdf-schema/. (Last accessed June 2015).
[24] Oracle Database, Oracle Spatial and Graph RDF Semantic Graph, Oracle. http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/rdfsemantic-graph-1902016.html. (Last accessed June 2015).
[25] L. Moreau, P. Missier, PROV-DM: The PROV Data Model, W3C Recommendation, W3C, 2013.
[26] J. Feigenbaum, A.D. Jaggard, R.N. Wright, Towards a formal model of accountability, in: Proceedings of the 2011 Workshop on New Security Paradigms Workshop, ACM, 2011, pp. 45–56.
[27] J. Feigenbaum, J.A. Hendler, A.D. Jaggard, D.J. Weitzner, R.N. Wright, Accountability and deterrence in online life, in: Proceedings of the 3rd International Web Science Conference, WebSci, ACM, New York, NY, USA, 2011 7:1–7:7. http://dx.doi.org/10.1145/2527031.2527043.
[28] S. Pearson, A. Charlesworth, Accountability as a way forward for privacy protection in the cloud, in: M. Jaatun, G. Zhao, C. Rong (Eds.), Cloud Computing, in: LNCS, 5931, 2009, pp. 131–144.
[29] P. Golle, F. McSherry, I. Mironov, Data collection with self-enforcing privacy, in: Proceeding of Computer and Communications Security, 2006, pp. 69–78.
[30] K. Julisch, C. Suter, T. Woitalla, O. Zimmermann, Compliance by design–Bridging the chasm between auditors and IT architects, Comput. Secur. 30 (6) (2011) 410–426.
[31] A. Barth, A. Datta, J.C. Mitchell, H. Nissenbaum, Privacy and contextual integrity: Framework and applications, in: Proceedings of the IEEE Symposium on Security and Privacy, IEEE, 2006, pp. 184–198.
[32] A. Datta, J. Blocki, N. Christin, H. DeYoung, D. Garg, L. Jia, D. Kaynar, A. Sinha, Understanding and protecting privacy: Formal semantics and principled audit mechanisms, in: Proceedings of the International Conference on Information Systems Security, Springer, 2011, pp. 1–27.
[33] D. Basin, F. Klaedtke, S. Müller, Policy monitoring in first-order temporal logic, in: Proceedings of the International Conference on Computer Aided Verification, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 1–18. http://dx.doi.org/10.1007/978-3-642-14295-6_1.
[34] J.G. Cederquist, R. Corin, M.A. Dekker, S. Etalle, J.I. den Hartog, G. Lenzini, Audit-based compliance control, Int. J. Inf Security 6 (2–3) (2007) 133–151.
[35] J. Jeong, High performance logging system for embedded UNIX and GNU/Linux applications, in: Proceeding of the 19th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, IEEE, 2013, pp. 310–319.
[36] C. Gülcü, The Complete Log4j Manual, QOS, 2003.
[37] K. Kent, M. Souppaya, Guide to Computer Security Log Management, Vol. 92, NIST, 2006.
[38] R. Accorsi, Log data as digital evidence: What secure logging protocols have to offer?, in: Proceeding of 33rd Annual IEEE International Computer Software and Applications Conference, vol. 2, IEEE, 2009, pp. 398–403.
[39] V. Stathopoulos, P. Kotzanikolaou, E. Magkos, Secure log management for privacy assurance in electronic communications, Comput. Secur. 27 (7–8) (2008) 298–308. http://dx.doi.org/10.1016/j.cose.2008.07.010.
[40] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, et al., The open provenance model core specification (v1. 1), Future Gener. Comput. Syst. 27 (6) (2011) 743–756.
[41] R. Aldeco-Perez, L. Moreau, Information accountability supported by a provenance-based compliance framework, 2009.
[42] D. Grunwell, T. Sahama, Information accountability and health big data analytics: A consent-based model, in: Proceeding of 17th International Conference on E-Health Networking, Application & Services, HealthCom, IEEE, 2015, pp. 195–199.
[43] L. Cranor, M. Langheinrich, M. Marchiori, J. Reagle, The Platform for Privacy Preferences (P3P)1.0 Specification, W3C. http://www.w3c.org/TR/P3P/. (Last accessed November 2016).
[44] C. Perera, C. Liu, R. Ranjan, L. Wang, A.Y. Zomaya, Privacy-knowledge modeling for the internet of things: A look back, Computer 49 (12) (2016) 60–68.
[45] M. Backes, B. Pfitzmann, M. Schunter, A toolkit for managing enterprise privacy policies, in: Computer Security–ESORICS, Springer, 2003, pp. 162–180.
[46] W3C, ODRL Vocabulary & Expression, W3C Candidate Recommendation 26 September 2017, W3C. https://www.w3.org/TR/odrl-vocab/. (Last accessed November 2017).
[47] R. Iannella, S. Guth, D. Pähler, A. Kasten, ODRL: Open digital rights language 2.1. W3C ODRL Community Group, 2012.
[48] S. Steyskal, A. Polleres, Towards formal semantics for ODRL policies, in: International Symposium on Rules and Rule Markup Languages for the Semantic Web, Springer, 2015, pp. 360–375.
[49] E. Daga, A. Gangemi, E. Motta, Reasoning with data flows and policy propagation rules, Semantic Web, Preprint. (Accessed November 2017) pp. 1–21.
[50] W3C, Permissions & Obligations Expression Working Group, W3C. https://www.w3.org/2016/poe/charter. (Last accessed November 2017).
[51] S. Boag, D. Chamberlin, M.F. Fernández, D. Florescu, J. Robie, J. Siméon, M. Stefanescu, XQuery 1.0: An XML query language, 2002.

[52] Q. Ni, E. Bertino, J. Lobo, An obligation model bridging access control policies and privacy policies, in: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT, ACM, New York, NY, USA, 2008, pp. 133–142. http://dx.doi.org/10.1145/1377836.1377857.

[53] Z. Liu, A. Ranganathan, A. Riabov, Specifying and enforcing high-level semantic obligation policies, Web Semant.: Sci. Serv. Agents World Wide Web 7 (1) (2009) 28–39. http://dx.doi.org/10.1016/j.websem.2008.02.002.

[54] J. Rao, A. Sardinha, N. Sadeh, A meta-control architecture for orchestrating policy enforcement across heterogeneous information sources, Web Semant.: Sci. Serv. Agents World Wide Web 7 (1) (2009) 40–56. http://dx.doi.org/10.1016/j.websem.2007.10.001.

[55] K. Arkoudas, R. Chadha, J. Chiang, Sophisticated access control via smt and logical frameworks, ACM Trans. Inf. Syst. Secur. 16 (4) (2014) 17.

[56] Sun Microsystem, Sun's XACML Open Source Implementation. http://sunxacml.sourceforge.net/. (Last accessed June 2015).

[57] F. Turkmen, J. den Hartog, S. Ranise, N. Zannone, Analysis of xacml policies with smt, in: International Conference on Principles of Security and Trust, Springer, 2015, pp. 115–134.

[58] J. Hendler, T. Berners-Lee, From the Semantic Web to social machines: A research challenge for AI on the World Wide Web, Artificial Intelligence 174 (2) (2010) 156–161.

[59] L. Kagal, J. Pato, Preserving privacy based on semantic policy tools, IEEE Secur. Priv. 8 (4) (2010) 25–30. http://dx.doi.org/10.1109/MSP.2010.89.

[60] O. Seneviratne, L. Kagal, Enabling privacy through transparency, in: Proceeding of the Twelfth Annual International Conference on Privacy, Security and Trust, PST, 2014, pp. 121–128. http://dx.doi.org/10.1109/PST.2014.6890931.

[61] D. Corsar, P. Edwards, J. Nelson, Personal privacy and the web of linked data, in: Proceedings of the IEEE Workshop on Privacy Online, IEEE, 2013.

[62] N. Bettencourt, N. Silva, J. Barroso, How to publish privately, in: Proceedings of the 2014 IEEE Workshop on Privacy Online, IEEE, 2014.

[63] O. Sacco, A. Passant, A privacy preference ontology (PPO) for linked data, in: Proceedings of the Workshop on Linked Data on the Web Co-Located with the 20th International World Wide Web Conference, WWW, vol. 813, CEUR-WS, Hyderabad, India, 2011.

[64] H. Mühleisen, M. Kost, J.-C. Freytag, SWRL-based access policies for linked data, in: Proceeding of the SPOT Workshop on Trust and Privacy on the Social and Semantic Web, 2010.

[65] W3C, SWRL: A Semantic Web Rule Language, Combining OWL and RuleML, W3C Member Submission, W3C. http://www.w3.org/Submission/SWRL/. (Last accessed June 2015).

[66] J. Hollenbach, J. Presbrey, T. Berners-Lee, Using RDF metadata to enable access control on the social Semantic Web, in: Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge, vol. 514, 2009.

[67] H. Story, B. Harbulot, I. Jacobi, M. Jones, Foaf+ssl: Restful authentication for the social web, in: Proceedings of the First Workshop on Trust and Privacy on the Social and Semantic Web, SPOT, 2009.

[68] S. Speiser, Policy of composition? Composition of policies, in: Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks, POLICY, IEEE Computer Society, Washington, DC, USA, 2011, pp. 121–124. http://dx.doi.org/10.1109/POLICY.2011.21.

[69] M. Gruninger, M. Fox, Methodology for the design and evaluation of ontologies, in: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI, vol. 95, 1995.

[70] M. Uschold, M. Gruninger, Ontologies: Principles, methods and applications, Knowl. Eng. Rev. 11 (02) (1996) 93–136.

[71] M.C. Suárez-Figueroa, A. Gómez-Pérez, B. Villazón-Terrazas, How to write and use the ontology requirements specification document, in: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Springer, 2009, pp. 966–982.

[72] OpenLink Software, Virtuoso Universal Server, OpenLink Software. https://virtuoso.openlinksw.com/. (Last accessed November 2016).

[73] IBM Software, NoSQL Graph Store, IBM. http://www-01.ibm.com/FieldGuide/NoSQL. (Last accessed November 2016).

[74] R. Samavi, M.P. Consens, L2TAP+SCIP: An audit-based privacy framework leveraging Linked Data, in: 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, 2012, pp. 719–726. http://dx.doi.org/10.4108/icst.collaboratecom.2012.250607.

[75] R. Samavi, M.P. Consens, Publishing L2TAP logs to facilitate transparency and accountability, in: Proceedings of the Workshop on Linked Data on the Web Co-Located with the 23rd International World Wide Web Conference, WWW, vol. 1184, CEUR-WS, Seoul, Korea, 2014.

[76] Y. Raimond, S. Abdallah, Timeline Ontology. http://purl.org/NET/c4dm/timeline.owl. (Last accessed November 2016).

[77] W3C, XML Schema Definition Language (XSD) 1.1, W3C. https://www.w3.org/TR/xmlschema11-2/#ISO8601. (Last accessed April 2017).

[78] C.S. Jensen, R.T. Snodgrass, Semantics of time-varying information, Inf. Syst. 21 (4) (1996) 311–352.

[79] L. Anselma, P. Terenziani, R.T. Snodgrass, Valid-time indeterminacy in temporal relational databases: Semantics and representations, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2880–2894.

[80] W3C, RDFG: Named Graph Vocabulary, W3C. http://www.w3.org/2004/03/trix/rdfg-1/. (Last accessed June 2015).

[81] J.J. Carroll, C. Bizer, P. Hayes, P. Stickler, Named graphs, Web Semant.: Sci. Serv. Agents World Wide Web 3 (4) (2005) 247–267. http://dx.doi.org/10.1016/j.websem.2005.09.001.

[82] D. Brickley, L. Miller, FOAF Vocabulary Specification, Namespace Document. http://xmlns.com/foaf/spec/. (Last accessed November 2016).

[83] S. Tramp, H. Story, A. Sambra, P. Frischmuth, M. Martin, S. Auer, et al., Extending the WebID protocol with access delegation, in: Proceedings of the Third International Workshop on Consuming Linked Data, 2012.

[84] N. Popitsch, B. Haslhofer, Dsnotify - a solution for event detection and link maintenance in dynamic datasets, Web Semant.: Sci. Serv. Agents World Wide Web 9 (3) (2011) 266–283. http://dx.doi.org/10.1016/j.websem.2011.05.002.

[85] J.-W. Byun, E. Bertino, N. Li, Purpose based access control of complex data for privacy protection, in: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, 2005, pp. 102–110. http://dx.doi.org/10.1145/1063979.1063998.

[86] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, Turtle-terse RDF triple language, in: W3C Team Submission, vol. 14, 2008, p. 7.

[87] DCMI, Dublin core metadata element set, version 1.1, DCMI. http://dublincore.org/documents/dcmi-terms/. (Last accessed November 2016).

[88] W3C, RDF 1.1 TriG, RDF Dataset Language, W3C. https://www.w3.org/TR/trig/. (Last accessed April 2017).

[89] J. Hobbs, F. Pan, Time ontology in OWL, W3C Working Draft, Vol. 27. https://www.w3.org/TR/owl-time/. (Last accessed April 2017).

[90] W3C, OWL 2 Web Ontology Language Profiles - W3C Working Draft, W3C. http://www.w3.org/TR/2009/WD-owl2-profiles-20090421/. (Last accessed June 2015).

[91] A. Cavoukian, Privacy by Design, Take the Challenge, Office of Information and Privacy Commissioner of Ontario, 2009.

[92] J.R. Searle, Speech Acts: An Essay in the Philosophy of Language, Vol. 626, Cambridge University Press, 1969.

[93] H. Knublauch, SPIN, SPARQL Inferencing Notation, W3C Member Submission. http://spinrdf.org/. (Last accessed July 2017).

[94] M. Hilty, D. Basin, A. Pretschner, On obligations, in: European Symposium on Research in Computer Security, Springer, 2005, pp. 98–117.

[95] M. Hashmi, G. Governatori, M.T. Wynn, Modeling obligations with event-calculus, in: International Workshop on Rules and Rule Markup Languages for the Semantic Web, Springer, 2014, pp. 296–310.

[96] A. Padia, T. Finin, A. Joshi, Attribute-based fine grained access control for triple stores, in: 3rd Society, Privacy and the Semantic Web - Policy and Technology Workshop, 14th International Semantic Web Conference, 2015.

[97] A. Matheus, J. Herrmann, Geospatial extensible access control markup language (geoxacml), Open Geospatial Consortium Inc. OGC (2008).

[98] K. Irwin, T. Yu, W.H. Winsborough, On the modeling and analysis of obligations, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS, ACM, New York, NY, USA, 2006, pp. 134–143. http://dx.doi.org/10.1145/1180405.1180423.

[99] J.R. Hobbs, F. Pan, An ontology of time for the semantic web, TALIP Spec. Issue Temporal Inf. Process. 3 (1) (2004) 66–85. http://dx.doi.org/10.1145/1017068.1017073.

[100] M. Grüninger, Verification of the OWL-time ontology, in: Proceedings of the 10th International Conference on the Semantic Web - Volume Part I, ISWC, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 225–240.

[101] H. Knublauch, The TopBraid SPIN API. http://topbraid.org/spin/api/. (Last accessed July 2017).

[102] L.N.P. Obermeier, L. Nixon, A cost model for querying distributed RDF-repositories with SPARQL, in: Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense, Tenerife, Spain, 2008.

[103] A. Sutton, R. Samavi, Blockchain enabled privacy audit logs, in: International Semantic Web Conference, Springer, 2017, pp. 645–660.

[104] J.F. Sequeda, M. Arenas, D.P. Miranker, On directly mapping relational databases to RDF and OWL, in: Proceedings of the 21st International Conference on World Wide Web, WWW, ACM, New York, NY, USA, 2012, pp. 649–658. http://dx.doi.org/10.1145/2187836.2187924.

[105] J. Tao, E. Sirin, J. Bao, D. McGuinness, Integrity constraints in OWL, in: Proceeding of Association for the Advancement of Artificial Intelligence, AAAI, 2010.

[106] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semant.: Sci. Serv. Agents World Wide Web 5 (2) (2007) 51–53. http://dx.doi.org/10.1016/j.websem.2007.03.004.

[107] A.Y. Levy, Obtaining complete answers from incomplete databases, in: Proceedings of the 22th International Conference on Very Large Data Bases, VLDB, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996, pp. 402–412.

[108] W. Nutt, S. Razniewski, Completeness of queries over SQL databases, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, in: CIKM, ACM, New York, NY, USA, 2012, pp. 902–911. http://dx.doi.org/10.1145/2396761.2396875.

[109] M. Morsey, J. Lehmann, S. Auer, A.-C. Ngonga Ngomo, DBpedia SPARQL benchmark: Performance assessment with real queries on real data, in: Proceedings of the 10th International Conference on the Semantic Web - Volume Part I, ISWC, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 454–469.

[110] J. Nielsen, Response times: The 3 important limits, Nielsen Norman Group. https://www.nngroup.com/articles/response-times-3-important-limits/. (Last accessed November 2017).

[111] EU Directives, Eu directive 95/46/EC on the protection of individuals rights with regard to the processing of personal data, Official J. EC, 1995.

[112] University Health Network, Privacy Impact Assessment, ConnectingGTA Project, Toronto, Ontario, May 2012.