A Formal Framework for Consent Management

Shukun Tokas and Olaf Owe

Department of Informatics, University of Oslo, Norway
{shukunt,olaf}@ifi.uio.no

Introduction

In response to the emerging privacy concerns, the European Union (EU) has approved the General Data Protection Regulation (GDPR) [1] to strengthen and impose data protection rules across the EU. This regulation requires controllers that process personal data of individuals within EU and EEA, to process personal information in a ”lawful, fair, and transparent manner”. Article 6 and Article 9 of the regulation [1] provide the criteria for lawful processing, such as consent, fulfillment of contractual obligation, compliance with a legal obligation etc. The regulation treats consent as one of the guiding principles for legitimate processing, and Article 7 [1] sets out the conditions for the processing personal data (when relying on consent).

Consent is defined as “any freely given, specific, informed and unambiguous indication of the data subject’s wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her” [1]. In particular, a data subject‘s consent reflects her choices/agreements in terms of the processing of personal data. These privacy requirements can be expressed through privacy policies, which are used to regulate the processing of personal data. The privacy requirements in the GDPR (as well as other privacy regulations) are defined informally, therefore, to avoid ambiguity the policy language equipped with a formal semantics is essential [2, 3]. We have previously studied static aspects of privacy policies and policy compliance from a formal point of view, a brief overview is given in [4]. Here, we look at a formal approach to address consent at the modeling level.

The aim of this work is to design a formal framework for consent management where a data subject can change her privacy settings through predefined interface definitions, which could be seen as part of a library system. The data subjects are seen as system users without knowledge of the underlying program. The framework consists of predefined language constructs for specifying privacy policies and consent, compliance and consent checking, and a semantics. We prove a notion of compliance regarding consent. To make a general solution, we consider a high-level modeling language for distributed service-oriented systems, building on the paradigm of *active objects* [5, 6] .

Central to the design of this framework are *(a)* a policy definition language that allows specification of privacy policies; *(b)* a formalization of policy compliance; *(c)* integration of privacy policies in a programming language; *(d)* a run-time system for dynamic checking of privacy compliance, with built-in consent management. The framework covers essential GDPR aspects, providing practical means to support for *privacy by design* (Article 25, Recital 78 [1]) and *data subject access request* (Article 7, Recital 63 [1]). It is essential that the policy terminology establishes precise link between the law and the program artifacts. For this, we let privacy policies and consent definitions be expressed in terms of several predefined names, reflecting standard terminology (names can be added as needed). Since the data subject is not always a legal scholar or program developer, it is necessary that the policy terminology used towards the data subject is simple but with a formal connection to the underlying programming elements. The rest of this abstract will provide motivation and basic notions related to privacy and consent specification, but we will omit details for the runtime system.

Language Setting

In the setting of active objects, the objects are autonomous and execute in parallel, communicating by so-called asynchronous method invocations. An Object-local data structure is defined by data types. We assume interface abstraction, i.e., remote field access is illegal and an object can only be accessed though an interface. This allows us to focus on major challenges of modern architectures, without the complications of low-level language constructs related to the shared-variable concurrency model. The programs we consider are defined by a sequence of declarations of interfaces (containing method declaration), classes (containing class parameters, fields, methods and an initialization) and data type definitions. Classes are defined by an imperative language and data types and associated functions by a functional language.

Interfaces may have a number of superinterfaces, letting the predefined interface *Any* be the most general interface (supported by any object). We let interface *Sensitive* be a subinterface of *Any*, corresponding to a system component (active object) with personal data. By static checking it is ensured that any object receiving personal data must support the interface *Sensitive* [4]. And we let interface *Principal* be a subinterface of *Any* corresponding to a system user, be it a person, an organization, or other identifiable actor. Interface *Subject* is a subinterface of both *Principal* and *Sensitive*, and corresponds to what GDPR refers to as “data subjects”. Interface *Sensitive* defines methods for accessing and resetting consented (and default) policies, by the data subject. Interface *Subject* offers methods for consent management including functionality for requesting and updating consent settings.

Runtime aspects

At runtime there will be a number of concurrent objects containing data values (of some type), and communicating by method calls. Data values with personal information will be tagged. The tags reflect associated consent information. This meta information is not directly accessible to the programmer or system user, but is understood by the runtime system, to restrict access to private data. Our framework includes a general solution for subjects to observe and change their privacy settings, and a way to delete private data (soft delete). The tags include privacy information such as identification of the *subject*, as well as *role*, *purpose* and *access rights*, specifying who (the roles that can use the data), why (purpose for which the data can be used), and how (kind of operation or access allowed on the data).

Formalization of Consent and Privacy Policies

We let privacy and consent definitions be expressed in terms of *role*, *purpose* and *access rights* for a given subject, where each of these range over a set of names, including predefined names reflecting standard terminology, names can be added as needed. A *role* is given by a name such as *Doctor*, *Nurse*, *Patient*, also arranged in a directed acyclic graph with a (transitive and anti-symmetric) less-than relation $<$. At the programming level, roles are reflected by interfaces or *Principals.* A Principal object may implement multiple interfaces to support several roles. A *purpose* is given by a name such as *health_care, advertising, treatment, billing, research.* The *purpose* names are arranged in a directed acyclic graph with a (transitive and anti-symmetric) less-than relation $<$, for instance $\textit{treatment} < \textit{health\_care}$ or $\textit{research} < \textit{health\_care}$. For *access-rights* we consider a fixed terminology for describing access rights, with *read*, *incr*, *self* and *write* access to the data. Access rights are given by a complete lattice, with $\sqcup$ and $\sqcap$ as lattice operators, with *full* (for full access) as top element and *no* (for no access) as bottom element. Furthermore, *read* gives read access, *write* gives over-write access, *incr* gives incremental access (adding an element to a list or set without reading the other

A	$::=$	$read \mid incr \mid write \mid self$	basic access rights
		$\mid no \mid full \mid rincr \mid wincr$	abbreviated access rights
		$\mid A \sqcap A \mid A \sqcup A$	combined access rights
$\mathcal{P}$	$::=$	(I, R, A)	policies
$\mathcal{P}s$	$::=$	$\{\mathcal{P}^*\} \mid \mathcal{P}s \sqcap \mathcal{P}s \mid \mathcal{P}s \sqcup \mathcal{P}s$	policy sets
$\mathcal{CP}$	$::=$	$[S, \mathcal{P}s] \mid [I, \mathcal{P}s]$	basic consent declarations
$\mathcal{CP}s$	$::=$	$\{\mathcal{CP}^*\} \mid \mathcal{CP}s \sqcap \mathcal{CP}s \mid \mathcal{CP}s \sqcup \mathcal{CP}s$	sets of consent declarations

Figure 1: BNF syntax definition of the policy language. I ranges over interface names, R over purpose names, and S over data subjects. Superscript $*$ denotes repetition.

purpose $treatm, health\_care$
    where $treatm < health\_care$
policy $\mathcal{P}_{MyDoc} = (\text{"}dr.Hansen\text{"}, treatm, full)$ `// specific policy`
policy $\mathcal{P}_{Doc} = (Doctor, treatm, rincr)$ `// general policy`
policy $\mathcal{P}_{Nurse} = (Nurse, treatm, read)$ `// general policy`
consent $\mathcal{CP}_{my} = [\text{"}Olaf\text{"}, \{\mathcal{P}_{MyDoc}, \mathcal{P}_{MyDoc}, \mathcal{P}_{Nurse}\}]$ `//consent specification`

Figure 2: Sample purpose, policy, and consent definitions. Subjects are identified by strings.

elements), and *self* gives the subject (full) access to data about itself. Thus $read \sqcup incr$ gives a principal read access and incremental access, but not general write access. This is quite useful in many connections and therefore we introduce *rincr*, as an abbreviation for it.

The language syntax for defining access rights, privacy policies, and consent is summarized in Figure 1 and some sample policies are found in Figure 2. A privacy policy $\mathcal{P}$ is given by a who-why-how triple (I, R, A), where I, R and A range over interfaces, purposes and access rights, respectively, each with their own hierarchy, following [4]. Policy sets form a lattice with $\sqcup$ and $\sqcap$ as lattice operators and with the empty set as the bottom element. A consent specification on data is given by the *subject* identity and a set of who-why-how policies and have the form $\{Subject, \{\mathcal{P}^*\}\}$. For example, we may specify consent for a patient p by $\{p, \{(Doctor, treatm, rincr), (Nurse, treatm, read)\}\}$. Private data in general is tagged with a single consent declaration, assuming the data concern the privacy of a single data subject.

References

[1] European Commission. The General Data Protection Regulation (GDPR) regulations of the European Parliament and of the Council. Accessed 28 Apr. 2019.

[2] R. Pardo and D. L. Métayer. Analysis of privacy policies to enhance informed consent. arXiv preprint arXiv:1903.06068, 2019.

[3] D. Le Métayer. Formal methods as a link between software code and legal rules. In G. Barthe, A. Pardo, G. Schneider, editors, *Proc. of 9th Int. Conf. on Software Engineering and Formal Methods, SEFM 2011*, volume 7041 of *Lecture notes in Computer Science*, pp. 3–18. Springer, 2011.

[4] S. Tokas, O. Owe, and T. Ramezanifarkhani. Language-based mechanisms for privacy by design. In *Revised Selected Papers from 14th IFIP Int. Summer School on Privacy and Identity Management, IFIP Advances in Inform. and Commun. Techn.*. Springer, to appear.

[5] O. Nierstrasz. A tour of Hybrid—a language for programming with active objects. In *Advances in Object-Oriented Software Engineering*, pp. 67–182. Prentice-Hall, 1992.

[6] E. B. Johnsen and O. Owe. An asynchronous communication model for distributed concurrent objects. *Softw. Syst. Modeling*, 6(1):39–58, 2007.