A Declarative Data Protection Approach: From Human-Readable Policies to Automatic Enforcement

Francesco Di Cerbo[1(✉)], Alessio Lunardelli[2], Ilaria Matteucci[2], Fabio Martinelli[2], and Paolo Mori[2]

[1] SAP Security Research, Sophia Antipolis, France
francesco.di.cerbo@sap.com
[2] IIT-CNR, Pisa, Italy
{alessio.lunardelli,ilaria.matteucci,fabio.martinelli,
paolo.mori}@iit.cnr.it

Abstract. In recent years, almost any object we use in our lives is connected and able to generate, collect and share data and information. This leads to the need of having, on the one hand, legal regulations, such as the new General Data Protection Regulation, able to guarantee that privacy of humans is preserved within the sharing process, and on the other hand, automatic mechanisms to guarantee that such regulations, in addition to user privacy preferences, are applied. The goal of this work is to propose an approach to manage data protection policy, from their specification in a controlled natural language to their translation into an automatically enforceable policy language, UPOL, for access and usage control of personal information, aiming at transparent and accountable data usage. UPOL extends and combines previous research results, U-XACML and PPL, and it is part of a more general proposal to regulate multi-party data sharing operations. A use case is proposed, considering challenges brought by the new EU's GDPR.

Keywords: Personal data protection · GDPR · Privacy · Security

1 Introduction

The increased adoption of cloud solutions to store and share data among entities, companies and objects leads to the necessity of having a clear legal regulation for personal data protection, in place in a transnational environment to regulate the current practice for data exchanges among data producers, consumers and cloud providers. This awareness has driven the advent of the new European General Data Privacy Regulation (GDPR Regulation (EU) 2016/679) [8], entered in force in May 2018. This regulation has a deep impact on the legal framework and the requirements for data processing of European citizens because it is applicable to all entities, all over the world, aiming at processing personal data belonging to citizens of an European country.

M. J. Escalona et al. (Eds.): WEBIST 2018, LNBIP 372, pp. 78–98, 2019.
https://doi.org/10.1007/978-3-030-35330-8_5

To be compliant with this regulation, all such entities need to perform a deep revision of their data management processes and, consequently, of the software deputed to manage them in order to be compliant with the new prescriptions. In particular, we focus on a number of key requirements of GDPR formalised in Articles 5 - Principles relating to processing of personal data - and 25 - Data protection by design and by default - dealing with *"lawfulness, fairness and transparency"*, *"purpose limitation"* and *"data minimisation"*.

The principle of *"lawfulness, fairness and transparency"* deals with requirements on how data must be processed from the owner of the personal data prospective. The *"purpose limitation"* principle aims at linking the request of data processing with the purpose of use of them that has to be explicitly accepted by the data subject (explicit consent) with a contract prepared under the fairness and transparency principles. *"Data minimisation"* imposes to *data controllers*, such as, companies collecting personal data of customers, to reduce the amount of collected data to the strict minimal necessary to carry out the requested service, also reducing "the extent of processing, the period of their storage and their accessibility".

To guarantee these requirements, in this paper we propose a Data Protection approach able to regulate both access and usage of data with an eye to the GDPR regulation, i.e., to fulfill the data minimisation requirement in its part referring to computation purpose verification, data retention and data access. Access and usage of data are regulated by the definition of rules. In our approach, we consider to have a unique document that collects both legal requirements and user rules about access and usage control of a piece of data, i.e., we refer to the concept of *Data Sharing Agreement* (DSA) associated to the target data. It is an agreement among contracting parties regulating how they share data. A DSA represents a flexible mean to assure privacy, in terms of GDPR directive, of data exchanged on the Cloud. Here, we refer to the work in [11], in which the authors introduced a Controlled Natural Language for DSA aiming at lowering the barrier to adoption of DSA, and, at the same time, ensuring mapping to a chosen enforceable language.

Our activity identified two solutions, whose combination would allow a data controller to achieve a transparent usage of personal data. They are PPL [7,15] and U-XACML [10]. The novelty of our proposal consists of an approach (policy language and reference architecture) that unifies both benefits and allows to achieve new results, namely:

R1. to meet GDPR's transparency requirement by fully controlling information processing operations. This is achieved through the full support of the Usage Control model proposed by Park and Sandhu [13,16], achieved through our contribution as simple extension to a well-known access control standard, XACML [12].

R2. the control and tracking of processing purpose(s), for the operations requesting an access to the protected pieces of information, both at the moment of the access request and during their consumption. This aims at meeting GDPR's purpose limitation.

R3. the support for GDPR's data minimisation, considering for example data retention conditions.
R4. to ease compliance audit of a solution integrating our approach, also considering the different stakeholders' background involved in the audit process: lawyers, software architects, public bodies but also citizens (data subjects).

The concepts presented here appeared in an earlier form in [6], however this work extends them by:

- recalling the notion of Data Sharing Agreements as way to express and group all security and data protection constraints expressed at high abstraction level (human language) by different parties to regulate data sharing operations.
- recalling the CNL4DSA high level controlled natural language introduced in [11], functional to the expression of Data Sharing Agreements.
- introducing a novel architectural component, named DSA Mapper, in charge of automatically translating the security and data protection constraints expressed in CNL4DSA to enforceable UPOL policies.
- combining all previously mentioned extensions to present an extended architecture and a complete data flow for regulated data sharing operations, covering data provisioning and consumption operations.
- achieving a new result, a simplification for conducting audit given by the simplified understandability and transparency of the extended approach.

The paper is structured as follows: next section describes the motivation of this work that is mainly part of a EU FP7 project named Coco Cloud about the sharing of sensitive data through the Cloud, and how parts of this effort are continued in project C3ISP. Section 3 briefly recalls the existing access and usage control policy languages while Sect. 4 presents the new one, named UPOL able to integrate and enhance in a unique language the expressiveness of both languages. Section 5 presents a use case for an UPOL policy and finally Sect. 6 draws the conclusion of the paper and discusses about ongoing and future work.

2 Usage Control in Coco Cloud and C3ISP

The Confidential and Compliant Cloud (Coco Cloud) EU FP7 funded project [4] proposes a data centric approach to enhance data security on the cloud. In particular, the project is aimed at enabling its users to regulate the usage of the data they share on the cloud, in order to increase their trust in, and consequently their adoption of, cloud services. In the Coco Cloud scenario, multiple subjects are involved in the data sharing, including companies, public bodies, and citizens. Consequently, the data sharing must be automatically regulated by digital contracts, called Data Sharing Agreements (DSA) [2], defined by the sharing parties, which must be paired with the data when they are shared on the Cloud and must be enforced every time such data are accessed and used by the Cloud users [3]. A peculiar goal of the Coco Cloud project is to embed in the DSA also a further set of constraints to allow a legally compliant data

sharing in the cloud. Hence, the project places an early emphasis on understanding and incorporating legal and regulatory requirements into the data sharing agreements. The European data protection legal framework has been one of the key legal focuses of our work.

To this aim, the first step consists in an automatic translation of both legal and sharing parties constraints in a policy format that can be directly enforced. In fact, DSA are automatically mapped into Usage Control Policies expressed through the *UPOL* language. The Usage Control model [13] is adopted because the factors that are taken into account to define constraints in the DSA are mutable, i.e., they can change over time. As a matter of fact, one of the main improvements introduced by the Usage Control model with respect to traditional access control ones is the management of those user and resource attributes which change their values over time, thus requiring the continuous evaluation of the usage control policy to promptly react to changes. In particular, attribute values could change in such a way that an access which was previously authorized according to the previous attribute values and is now in progress should instead be forbidden because of the new values of such attributes. Hence, the usage control policy is continuously evaluated during the data access time, and the access can be interrupted when the policy is not satisfied any more. For instance, the physical position of a person is one of these mutable attributes because, obviously, it changes when the person moves from one place to another. A DSA could state that a critical document produced by a company can be read only by the employees of such company when they are located within a given area (e.g., the building of a company). Hence, an employee could open the document on her mobile phone when she is located within the company building but, as soon as she exits the building, the usage control policy is violated and the countermeasure defined in the policy is taken. For instance the policy could require to close such document saving the unsaved changes in a temporary document copy).

Moreover, the Usage Control model also allows policy makers to express that some actions, called obligations, must be executed as a consequence of the execution of other actions or when certain events occur. For instance, the DSA of a piece of data could include an obligation which requires that the related file is deleted after a given date. This concept has clear application in expressing a personal data retention period, according to the GDPR. Another example of obligation recalling the previous DSA is the one that, when the user leaves her country, deletes the document from her mobile device.

The usage control model allows to define very expressive data sharing agreements which satisfies the requirements of many application scenarios. For example, we applied it to e-government, corporate and healthcare [2] solutions. Moreover, part of the described approach is used in C3ISP[1], an EU-funded H2020 project focussing on cyber security and in particular on the sharing of particularly critical pieces of information, that are necessary for organising an effective defense against online attacks but that may also be used, if in wrong hands, to conduct malicious activities. This is the case of information describing cyber

[1] Homepage: https://c3isp.eu/.

attacks trace logs: other defenders may tune up their countermeasures in order to identify the latest attacks (malware received per email, specially crafted web requests targeting a software's vulnerability etc.) but on the other hand, an attacker may get to know where and how a target system is vulnerable and may be successfully breached. Specific extensions are being studied and they will be discussed in the future.

This paper describes *UPOL*, the language we defined to express the enforceable version of the usage control policies representing the DSAs exploited in the Coco Cloud and C3ISP European projects. We also describe how we automatically obtain UPOL policies starting from DSAs in which the rules are expressed in a *Controlled Natural Language* [11]. Such language has been introduced to help the user to express in a natural language style yet controlled way not only her own constraints on data but also legal regulations [9] on them.

3 Background

In this section, we recall some background notions about languages used for expressing policies at both high level through a controlled natural language and low level trough XACML-based languages.

3.1 Controlled Natural Language

The core of Controlled Natural Language named CNL4DSA [11] (CNL4DSA) is the notion of *fragment*, a tuple $f = \langle s, a, o \rangle$ where s is the subject, a is the action, o is the object. The fragment expresses that "the subject s performs the action a on the object o", e.g., "the doctor reads the medical report". It is possible to express authorisations, obligations, and prohibitions by adding the *can/must/cannot* constructs to the basic fragment. Fragments are evaluated within a specific *context*. In CNL4DSA, a *context* is a predicate c that usually characterises factors such as users' roles, data categories, time, and geographical location. Contexts are predicates that evaluate either to *true* or *false*. To describe complex policies, contexts must be combined. Hence, we use the Boolean connectors *and*, *or*, and *not* for describing a *composite context* C which is defined inductively as follows (Eq. 1):

$$C \ := c \mid C \ and \ C \mid C \ or \ C \mid not \ c \tag{1}$$

The syntax of a *composite authorisation fragment*, F_A, is as follows:

$$F_A \ := nil \mid can \ f \mid F_A; F_A \mid if \ C \ then \ F_A \mid after \ f \ then \ F_A \mid (F_A) \tag{2}$$

with the following meaning:

- *nil* can do nothing.
- *can* f is the atomic authorisation fragment that expresses that f is allowed, where $f = \langle s, a, o \rangle$. Its informal meaning is *the subject s can perform the action a on the object o.*

- $F_A; F_A$ is a list of composite authorisation fragments.
- *if* C *then* F_A expresses the logical implication between a context C and a composite authorisation fragment: if C holds, then F_A is permitted.
- *after* f *then* F_A is a temporal sequence of fragments. Informally, after f has happened, then the composite authorisation fragment F_A is permitted.

The list of authorisations represents all the composite authorisation fragments that define the access rights on the data.

Also, CNL4DSA has a specific syntax expressing composite obligation and prohibition fragments. Similar to the authorisations, the obligation fragment indicates that *the subject s must perform the action a on the object o*, while, for the prohibition, *the subject s cannot perform the action a on the object o*.

of a composite obligation fragment, F_O, is inductively defined as follows:

$$F_O \ := \ nil \mid must \ f \mid F_O; F_O \mid if \ C \ then \ F_O \mid after \ f \ then \ F_O \mid (F_O) \qquad (3)$$

The intuition is the same as for F_A, except for *must* f that represents the atomic obligation: *the subject s must perform the action a on the object o*. The atomic obligation *must* f expresses that f is required.

Finally, the syntax of a composite prohibition fragment, F_P, is as follows:

$$F_P \ := \ nil \mid cannot \ f \mid F_P; F_P \mid if \ C \ then \ F_P \mid after \ f \ then \ F_P \mid (F_P) \qquad (4)$$

The atomic prohibition is represented by *cannot* f: *the subject s cannot perform the action a on the object o*. The atomic prohibition *cannot* f expresses that f is not permitted.

3.2 Enforceable Policy Languages

We identified the Usage Control model as the theoretical underpinnings for our objective but we observed a number of limitations in the current approaches. Essentially, we looked at a number of declarative solutions (i.e. controllable through a configuration policy) and we concentrated on three technologies that have available implementations:

- **XACML:** the eXtensible Access Control Markup Language [12] is a standard produced by the OASIS Consortium which defines an XML based language for expressing Attribute Based Access Control policies and a reference architecture for the enforcement of such policies. Several open source, academic and commercial implementations of the XACML standard are currently available on the market. The main limitation is that it only covers access control models and not usage control. XACML only partially fulfill R1, R2, and R3.
- **U-XACML:** the UCON XACML [10] is an extension of the XACML standard aimed at supporting usage control functionalities, most notably the continuous policy evaluation during the access to the requested resources. U-XACML extends both the XACML language, in order to introduce proper constructs to express which XACML conditions must be satisfied at access request time and which must be satisfied for the whole duration of the access,

and the reference architecture, in order to introduce further components devoted to the continuous policy evaluation and to the management of usage sessions. U-XACML fulfills R1, partially R2 but not R3.
– **PPL:** as well extends XACML with the possibility to verify resource processing purposes against a policy plus it caters the automatic execution of obligations defined by a resource owner. It can be used to implement R3 especially with respect to data retention, R2 but not completely R1.

In order to achieve the fulfillment of requirements **R1, R2, R3** starting from the previously listed technologies, we defined a new concept, UPOL. In other words, we extended the XACML standard by combining the advantages brought in by two other extensions, U-XACML and PPL.We also extended all previous approaches in order to fulfill **R4**, by designing a process for transforming a human-readable set of directives into an actionable policy; this allows an easier understandability of UPOL policies by non-technical stakeholders, considering for example legal experts and citizens.

From the combination of the mentioned technologies, UPOL achieves new capabilities as detailed in Sect. 4. It is the language we used to express DSAs terms and conditions in a machine-enforceable manner. UPOL policies therefore regulate the usage of the data they are paired with: following the sticky policy model [14], each policy (that regulates the access to a resource) get attached to a resource to form a *bundle*, normally protected by means of strong encryption. This imposes that data can be processed only by means of special mechanisms able to decrypt the bundle and to allow its access in accordance to the associated policy. Any attempt to consume arbitrarily a resource once it is protected by such bundle, is destined to fail. The UPOL language is based on the Usage Control model, which extends traditional access control model by dealing with attributes related to the subjects and of the objects which change their values over time. The Usage Control model allows to define policies which are continuously evaluated during the execution of an access, in order to revoke such ongoing access when the corresponding policy is not valid any longer. In particular, the usage control policies define authorization and condition rules which must be satisfied before and/or during the usage of such data (*pre-/ongoing-authorizations* and *pre-/ongoing-conditions*), along with obligations (similarly, *pre-/ongoing-obligations*). UPOL comprises all such categories, extending the XACML capabilities with two new contributions the *asynchronous* and *synchronous* obligations, normally implemented by a trusted third party. The asynchronous obligations are usage control obligations which have to be fulfilled when an event occurs. Events may used to model reactions to mutable attributes as in Park and Sandhu model but, extending it, they may not be connected to an access request, as such as when the retention period for a piece of data expires (as requested for GDPR's data minimization). The synchronous obligations are again usage control obligations. For instance banners that appears while one watches a streaming video, or logging of the exact consumption time of a resource, for accountability purposes. Such kind of obligations may also be considered as *session* obligations and can be used in order to pinpoint when an

user starts and terminates to use a resource as well as when the user's access right to the resource is revoked.

In UPOL, the violation of pre-authorization or pre-condition rules prevents the access to the protected data. Instead, when pre-authorization and pre-condition rules are satisfied, the access to the data is allowed, and the ongoing-authorization and ongoing-condition rules are enforced continuously while the access is in progress. In this case, the violation of ongoing rules causes the interruption of the usage of the data. Session obligations may be associated to passed or failed checks, in order to model a desired behavior.

3.3 U-XACML

The U-XACML language is an extension of the XACML language which includes additional constructs to express Usage Control features. XACML is a standard developed by the OASIS consortium for expressing and managing access control policies in a distributed environment [12]. Briefly, the XACML standard defines a policy meta-model, syntax, semantics, and the related enforcement architecture. The top-level element of a policy is <*PolicySet*>, which includes a set of <*Policy*> elements (or other distinct <*PolicySet*>), each of which, in turn, includes a <*Target*>, which denotes the target of the policy, and a set of <*Rule*> elements which represent the authorization rules. A rule is defined by three main components: the <*Target*>, the <*Condition*>, and the effect of the rule which can be either Permit or Deny. The <*Target*> denotes the target of the rule, i.e., to which authorization requests the rule can be applied. The <*Condition*> elements are predicates evaluating the attributes. A rule can include the <*ObligationExpressions*> element, which, in turn, includes a set of <*ObligationExpression*> elements. Each <*ObligationExpression*> element includes the ID of the obligation and which effect will trigger its execution. A rule is applicable to an access request if the target of the access request matches the target of the rule and if all the conditions included in the rule are satisfied. If a rule is applicable to an access request, the effect declared for the rule concurs to determine whether the access request is permitted or denied, and the related obligation must be executed. As a matter of fact, the effects of all the applicable rule are combined to produce the effect to be actually enforced according to the combining algorithm specified at the beginning of the policy. For instance, the PERMIT OVERRIDE combining algorithm causes the policy to be evaluated to permit if at least one applicable rule has been assigned permit as effect.

XACML allows to express traditional attribute based access control policies dealing with immutable attributes and it does not have specific constructs to deal with mutable attributes and to express the continuity of policy enforcement. In particular, adopting a standard XACML system, the policy is evaluated at request time only, and no further policy evaluation are executed while the access is in progress. Consequently, once an access has been granted, if the attributes values change in such a way that the policy is not satisfied any more, no countermeasures are taken, and the ongoing access is not affected.

The U-XACML language extends XACML with usage control constructs as follows. To express the continuity of policy enforcement, the U-XACML language introduces in the <*Condition*> element a clause, called *DecisionTime*, which defines when the evaluation of this condition must be executed. The admitted values are *pre* and *on* denoting, respectively, *pre-decisions* and *on-decisions*. In this way, the conditions whose decision time is set to *pre* are the same as usual XACML conditions, since they are evaluated at access request time only. On the other hand, the conditions whose decision time is set to *on* must be continuously evaluated while the access is in progress. These conditions typically involve mutable attributes, because their values change over time thus requiring the re-evaluation of the condition. We recall that in U-XACML, XACML conditions are exploited to represent both UCON authorizations and conditions. In the same way, U-XACML extends the <*ObligationExpression*> element with the *DecisionTime* clause to define when the obligation must be executed. In this case too, the admitted values for the *DecisionTime* clause are: *pre* (*pre-obligations*, i.e., usual XACML obligations) and *on* (*on-obligations*), and *post* (*post-obligations*).

To deal with mutable attributes, U-XACML introduces a new element, <*AttrUpdates*>, which represents the attribute updates in the policy. This element includes a number of <*AttrUpdate*> elements to specify each update action. Each <*AttrUpdate*> element also specifies when the update must be performed through the clause *UpdateTime* which can have one of the following values: *pre* (*pre-update*), *on* (*on-update*), and *post* (*post-update*). U-XACML language, please refer to [5].

3.4 PPL

PPL (Primelife but also Privacy Policy Language) is another XACML extension that allows to express policies for personal data processing, also including specific credential capabilities. PPL directives refers to access and usage control security properties. In particular it was designed for modelling personal data exchanges between data subjects and data controller, according to definitions provided by the European Data Protection Directive 95/46/EC, very similar to those stated in GDPR. PPL adopts the "sticky policy" approach: a piece of data gets associated, for example in a bundle, to its policy and they form a unit, an atomic entity. Such approach is applied to regulate the exchange between a data subject and controller: once personal data handling terms (the "terms of use") have been agreed between the two actors, such terms become a PPL policy that gets associated to the personal data given to the data controller. By definition, this policy cannot be detached from the data and regulates each usage of the piece of information. One notable aspect is the expression of usage control obligations. Normally in XACML, an obligation must be fulfilled by the actor that issues an access request. In PPL, they are defined as *"a promise made by a data controller to a data subject in relation to the handling of his/her personal data. The data controller is expected to fulfill the promise by executing and/or preventing a specific action after a particular event, e.g., time, and optionally*

under certain conditions" [1]. PPL obligations may also apply to data processors, i.e., entities authorized by data controller to carry out computations on the data, under the responsibility of the data controller.

PPL obligations are expressed as in the following Eq. 5:

$$\textbf{Obligation} = \textit{Do}\ \textbf{Action}\ \textit{when}\ \textbf{Trigger} \tag{5}$$

where

$$\textbf{Trigger} = \textbf{Event} \wedge \textbf{Condition} \tag{6}$$

Obligations modeled in this way may or may not be dependent on access requests and therefore, they differ from access control obligations. They can be used to express a data retention period, e.g., data must be deleted by the data controller after 30 days from their submission. As shown in [7], triggers may also depend on contextual conditions like geographic location.

4 Our Data Protection Approach

Our data protection approach can be described as follows:

- specification of data protection requirements using a controlled natural language, with CNL4DSA
- transformation of CNL4DSA directives in UPOL enforceable control policy
- association of the control policy to a piece of information
- enforcement of the policy by a dedicated UPOL-enabled mechanism

As part of our contribution, we propose a reference software architecture for the materialization of our approach. The present section describes our architecture.

Once a data subject and controller agree on usage terms, they can express them using CNL4DSA, for example by means of an authoring tool, such as, the one briefly described in [9]. CNL4DSA terms can also be defined with a negotiation-less model: the CNL4DSA terms may represent the requirements of a subject and a controller can only decide to accept them to process the personal data, or the terms with which a controller can process a subject's data. The enforcement of CNL4DSA directives (i.e., the DSA) relies on their transformation in a UPOL policy. Such transformation, for its complexity, is most effective if performed automatically: to this extent, we foresaw a mapping function implemented by a specific software component, the *DSA Mapper*, and this main functionality is described in Sect. 4.1. As already mentioned, the UPOL language originates from XAMCL but adding statefulness to its interaction model. To cater for that, a number of contributions are proposed, in terms of reference architecture and language syntax, explained respectively in Sect. 4.2 and by means of an example in Sect. 5.

As a first step, we describe the mapping function we implement to automatically transform rules expressed in CNL4DSA (that is, a simple controlled natural language) to the UPOL language. Then, we describe the UPOL reference architecture and a comparison with the existing UCON_ABC.

4.1 Mapping Function

As mentioned, the mapping function allows to convert DSA statements into UPOL directives. The software component in charge of its implementation, the DSA Mapper, takes as input am .xml file representing a set of DSA rules and translates them in the UPOL language. The outcome of this tool is an enforceable policy, that will be evaluated at each request to process the associated piece of information.

An initial formulation for a mapping function has been presented in [11]. In the current and newest version, presented in the following, we have updated and simplified the process. In order to describe the mapping function, it is useful to recall that CNL4DSA has been developed considering the design of XACML constructs, thus it is possible to identify in each CNL4DSA statement the main XACML elements:

- A subject element is the entity requesting the access. A subject has one or more attributes.
- The resource element is a data, service or system component. A resource has one or more attributes.
- An action element defines the type of access requested on the resource. Actions have one or more attributes.
- An environment element can optionally provide additional information.

The mapping function implemented by the DSA Mapper is the results of two sub-functions: the *mapping function* that considers all the rules in a DSA as policies of a policy set, as it is defined in the standard XACML, and translate it in UPOL language, and the *BuildUPOL* function that is able to build an actual enforceable UPOL policy starting from the single policies output of the mapper function.

The mapping function takes each basic fragment $\langle s, a, o \rangle$, where s identifies the subject, a the action, and o the object (mainly the data which the DSA is referred to), and puts each of this element into a UPOL policy by using the appropriate tag, i.e., $\langle subject \rangle \ldots \langle \backslash subject \rangle$, $\langle action \rangle \ldots \langle \backslash action \rangle$, and $\langle resources \rangle \ldots \langle \backslash resources \rangle$, respectively. These represent the elements of the UPOL target ($\langle Target \rangle$). All the contextual conditions expressed in CNL4DSA are mapped into the tag $\langle Condition \rangle$. It is worth noting that even the attributes related to both subject and resources are mapped into the tag $\langle Condition \rangle$, in such a way to put all the contextual conditions under the same tag. This choice was made because the executable policy structure reflects the one of the CNL4DSA statement in which the conditions on subject, object, and environment are specified all together into the context.

The BuildUPOL function takes in input the output of the mapping function, i.e., the UPOL specification of each rule composing the DSA, to build a valid UPOL policy. The *BuildUPOL* functionality is in charge of building the UPOL policy skeleton, that represents the constant part of the UPOL policy. It is made of:

- the header of the policy comprehensive of:
  - the namespace,
  - the XML schema (version 3)
  - name of the schema (XACML 3)
  - name of the schema according to which the UPOL has to be validated
  - the policy ID
  - the combining algorithm (fist-applicable)
  - version of the policy
- Three specific tags:
  - $\langle description \rangle$, e.g., $\langle xacml : Description \rangle$ UPOL Policy $\langle /xacml : Description \rangle$.
  - $\langle DSAID \rangle$, e.g., $\langle upol : DSA\_id \rangle$ DSA-d75b9cbf-5893-4779-baf9 $\langle /upol : DSA\_id \rangle$.
  - $\langle Target \rangle$, that is empty because each rule has its own target specification with also the conditions (see below), e.g., $\langle xacml : Target/ \rangle$.
- Default deny rule
- All the rules derived from the DSA. Note that rules are inserted into the UPOL policy. According to the choice of applying the "first-applicable" combining algorithm, the order in which rules are inserted into the UPOL policy can follow a strategy. For example business reasons, regulations, personal preferences.

4.2 UPOL Architecture

The UPOL reference architecture is depicted in Fig. 1. The figure shows how the UPOL architecture interacts with a business software that involves the processing of personal information, to offer its data protection functionalities. This business software has at two main stakeholders of interest:

- a *Data Owner* that submits her/his personal data together with a DSA in the controlled natural language CNL4DSA
- a *Requestor* that uses the functionalities of the business software and needs to process a piece of information from the Data Owner.

The business software is only represented in its essential parts for the interaction with the UPOL architecture:

- a *Storage* for storing personal information
- a component that is at the same time part of the business software and the UPOL architecture, the PEP (later described).

UPOL originates from the XACML standard thus it inherits the XACML main components for its reference architecture, however extending their functionalities:

PEP. Policy Enforcement Point is part of a software system that intercepts any requests to perform security relevant actions, thus triggering the authorization decision process and enforces the results. In our UPOL design, it is also in charge of retrieving the *sticky policy bundle* from the business software storage (or data base) and to submit it to the PAP for reading the contained policy.

CH. Context Handler coordinates the communication and information exchange among the components of the enforcement engine in order to execute the policy evaluation process triggered by the PEP. In order to deal with UPOL policies, this component has been enabled to manage also the policy re-evaluation process which, instead, is triggered by a PIP and involves also the SM component.

PDP. Policy Decision Point evaluates authorization requests according to the applicable UPOL policies.

PAP. Policy Administration Point is in charge of retrieving applicable policies for PDP evaluation. It is an important component especially when adopting the sticky policy approach, as it allows to read the policy included in a bundle to permit its evaluation against the PEP request. It does so by complementing the request, issued by the PEP, extracting the UPOL policy from the bundle, and sending it to the CH.

PIP. Policy Information Point(s) retrieves all the necessary attributes of any actor (subject, action, resource, environment) for enabling policy evaluation. In order to evaluate UPOL policies, PIPs have been enabled also to detect when the values of the attributes change for the purpose of calling the CH to trigger the policy re-evaluation process.

The UPOL architecture also requires specific components, that are:

SM. Session Manager is responsible for tracking the usage sessions: a session is established once an authorization request is permitted and the associated operation is started. The session captures information about the operation by storing all relevant meta-data. It allows the execution of the continuous authorization phase, because it determines which sessions need a re-evaluation of authorization policies as a consequence of an attribute mutation, thus enabling the PDP to operate continuously.

OE. Obligation Engine is responsible for keeping track of obligation triggers and executing the associated action(s).

MAP. The DSA Mapper is responsible for the transformation of a DSA expressed in CNL4DSA format into a UPOL policy that is then associated to a piece of information in the sticky policy bundle, and stored in the business software storage according to its logic. Such bundle is subsequently handled by the PAP when the piece of information is to be used in order to extract such UPOL policy.

In our UPOL architecture, CH also implements an information exchange between OE and SM, to enable the mentioned session obligations to be triggered. The UPOL architecture is instrumental to the implementation of its new obligations; they go beyond the capabilities of XACML/U-XACML and PPL.

Table 1 details the different types of obligations supported by our architecture: those defined by XACML/U-XACML, PPL and obviously, the newly defined UPOL obligations.

The first row describes U-XACML obligations: they derive directly from the standard XACML obligations and they are associated to an access request. They

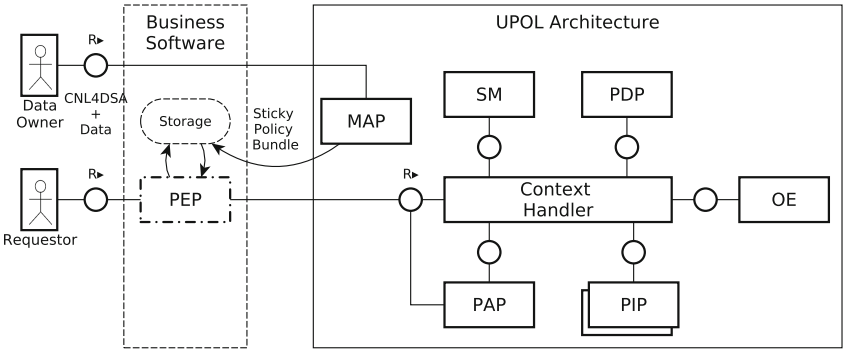

Fig. 1. The UPOL Reference Architecture, including the Mapper component to transform CNL4DSA documents in UPOL policies. Information can be provisioned by the Data Owner to the business software that uses the Mapper to obtain a sticky policy bundle to be persisted. When a Requestor asks for a resource, the PEP issues a request to the framework, in cooperation with the PAP that extracts the UPOL policy from that resource bundle.

Table 1. Obligations part of an UPOL policy, as presented in [6].

Obligation type	Reference event	Obligation action type	Obligation enforcement
U-XACML pre- or post-obligations	At the moment of the request	Punctual actions	PEP
PPL obligations	Dependent or independent from access request	Punctual actions	Trusted third party (through UPOL Mechanism)
UPOL session obligations	At the beginning, end or during data consumption	Punctual or continuous actions	Trusted third party (through UPOL Mechanism) or PEP

normally prescribe actions executed pre- or or post- an access request ("punctual actions" in the table) and for this reason, they are normally referred as pre- or post-obligations.

Second row is about PPL obligations. Differently from U-XACML, PPL obligations are not (necessarily) associated to an access request (see Formula 5). Their definition relies on triggers (see Formula 6) that may encompass a variety of situations. Expiration of a retention period but also proximity to a specific geographic location [7] are just initial examples of possible triggers. By definition, the enforcement of PPL obligations must take place reliably and certainly. Considering this requirement, PPL states that a trusted third party, different from Data Subject and Data Controller and trusted by both actors, is in charge of automatic obligation enforcement.

Lastly, the third row is about UPOL obligations. They can be defined using the notion of session: UPOL defines three new event types *StartAccess*, *EndAccess* and *RevokeAccess* that are part of UPOL triggers. Therefore, UPOL obligations consider the different stages of a usage session work-flow to prescribe the execution of specific actions. For example, a UPOL obligation can start at the beginning of a data consumption operation and terminate at its end. We recall that a (UPOL) session as managed by the Session Manager is created when an access request is approved and the requestor (or an agent) notifies the beginning of a resource consumption thus generating a session event *StartAccess*. *EndAccess* obligations are triggered when the requestor interrupts a resource consumption operation. *RevokeAccess* on the contrary occurs when a policy violation is detected and a session is interrupted by initiative of the UPOL mechanism (i.e., by initiative of the PDP).

As just presented, UPOL obligations differentiate from other obligations as they can be used to execute continuous actions; they result effective for example in streaming scenarios: showing banners during a video streaming, or to influence Big Data streaming analytics computation. One last consideration about the enforcement actor. XACML/U-XACML obligations are normally enforced at the requestor's end, by the PEP. PPL obligations, instead, are enforced by a third party (for example, a cloud provider) trusted by data subject and data controller. UPOL obligations are triggered by the Obligation Engine run by a trusted third party but they may be also executed by the PEP, according to the associated actions.

4.3 Comparison with the UCON$_{ABC}$ Model

The UCON model defined by Park and Sandhu in [13] presents a number of innovative features which enhances it with respect to traditional access control models, namely:

1. the factors taken into account to carry out the decision process, besides traditional authorizations (A), include also obligations (B) and conditions (C).
2. the continuity of the decision: the policy can state that the access control decision have to be made when the access request is received (like in traditional access control, *pre*), and/or have to be performed continuously during object consumption (typical trait of UCON, *ongoing*).
3. mutability of attributes: are subject or object attributes changing following to a decision? and when? This originates: *immutable*, $pre-update$, $ongoing-update$ or $post-update$ models respectively for models where no attribute changes are foreseen, updates takes place before, during or after an access takes place.

We claim that the UPOL language proposed in this paper is capable of implementing the typical features of the Usage Control model previously listed, by leveraging:

1. the native XACML constructs.
2. the constructs brought in by U-XAMCL.
3. the specific extensions offered by U-XACML combined with PPL: *pre* and *ongoing* conditions can originate specific events to trigger the newly defined UPOL obligations (*synchronous* and *asynchronous*).

Moreover, if used in conjunction with the sticky policy approach, UPOL spans its scope beyond access and usage control, as UPOL obligations can be triggered also without the reception of an access request. Such functionality is particularly helpful in cases like GDPR's data minimisation requirement, where a piece of data may reside on the Data Controller cloud only for a limited amount of time. Provided that a trusted third party runs a UPOL-aware enforcement mechanism to control the UPOL-regulated data on the cloud, such requirement may be fulfilled.

5 Use Case

Let us consider a simple e-commerce scenario involving three main actors: a company, *ACME*, that plays the role of the data controller, as it is defined in the GDRP, a customer of ACME, referred as *Customer*, that is the data subject, and a third-party, a *Marketing* service, that is able to produce profiles of customers and suggest marketing campaigns to ACME. The use case is depicted in Fig. 2. The customer submits a set of personal data, such as, address, credit card details, etc., needed by ACME to process her/his order. The treatment of such personal data is regulated by a DSA expressed in CNL4DSA, stating precisely data controller's rights and obligations as well as the customer's policies on her data. Data submission and consent recording take place through a specific service, called "Usage Control Service" (the proposed contribution, implementing the UPOL reference architecture) in the figure, that:

- associates to the personal data, using the sticky policy model, a UPOL policy, output of the DSA Mapper, that states in enforceable format the access and usage terms defined in the Data Sharing Agreement;
- enforces the UPOL policy upon incoming personal data access requests;
- monitors usage of personal data, through interactions with the ACME information systems, enforcing continuous authorizations as well as usage control obligations.

When the third party uses the ACME information systems, they interact with the enforcement system, creating requests to access the protected resources. Information systems cater for a number of attributes that allow their requests to be evaluated by the enforcement system, as well as providing indications about the beginning and the end of information processing operations. The interaction protocol between systems and enforcement also allows the interruption of an operation, in case of changes in the evaluation conditions.

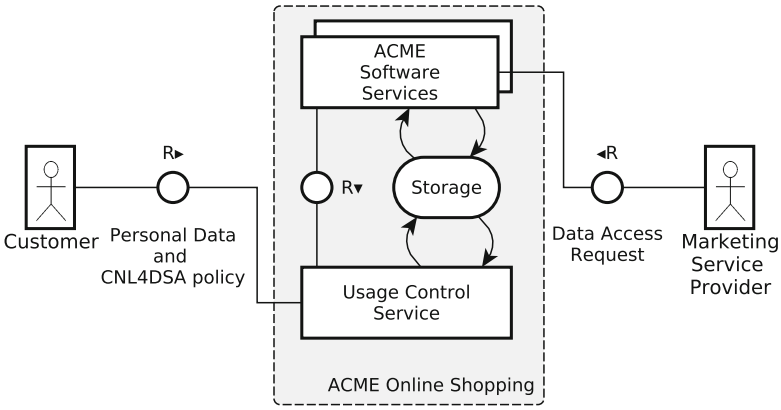

Fig. 2. An e-commerce use case, extended from [6].

It is out of the scope of the current work to include a detailed analysis of the interaction protocol and of the architecture of the enforcement system. Focusing instead on the policy expression, it can be modeled as a (sticky) UPOL *Policy* where a *Rule* with *Target*: *subject role = marketing-third-party, contractor = ACME* may access the data. The UPOL representation of (part of) such policy can be found at Listing 1.1. Two conditions (*pre* and *ongoing*) control the geographic location of the subject, provided by the information systems, before and during the access in order to protect against data export clauses, GDPR Article 49 (the UPOL policy in Listing 1.1 actually shows only the ongoing condition, since the other is identical except for the value of *DecisionTime* that would be *pre* instead of *ongoing*). In CNL4DSA, these two conditions are expressed through a CNL4DSA context `hasLocation` in case of *pre* condition and `hasContinuosLocation` in case the context condition needs to be checked *ongoing*. We can model the notification obligation by means of session obligations, triggered by *StartAccess*, *EndAccess* and *RevokeAccess*. In this way, the beginning and the end of sessions can be recorded for future use. Last but not least, each access of the third-party will trigger an email notification to the data subject, in order to fully meet the transparent processing requirements. In CNL4DSA, this rule is expressed as an obligation composed fragment in which the simple fragment has as subject `system` and as action `notifyByEmail`. It might be worth noting that other accesses performed by the data controller (as the trigger of this obligation is on the fulfillment of the Rule that applies only on requestors with role=marketing-third-party) will not trigger such notification. Data minimization (with respect to retention directives) is also enforced by means of a specific obligation to delete the data associated to the policy after 3 months from the moment when the personal data is received.

Listing 1.1. UPOL Use Case.

```
<!-- DETAILS OMITTED -->
<Rule RuleId="Permission:Marketing-Third-Party" Effect="Permit">
   <!-- XACML Target: ABAC check for attribute 'marketing-third-party'
     ↪ of requestor
     in the authentication system, i.e. role == 'marketing-third-party'
     ↪ -->
 <Target>
  <AnyOf>
   <AllOf>
    <MatchMatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
     ↪ string">
        marketing-third-party
     </AttributeValue>
     <AttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
      Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
     ↪ subject"
      DataType="http://www.w3.org/2001/XMLSchema#string"
      MustBePresent="false" />
    </Match>
   </AllOf>
  </AnyOf>
 </Target>

  <!-- Continuous Authorization: requestor must be in EU to process the
     ↪ information -->
  <upol:Condition DecisionTime="ongoing">
  <!-- Standard condition definition (with DecisionTime="pre") is identical
     ↪ and thus omitted -->
   <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
     ↪ equal">
    <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
        one-and-only">
            <xacml:AttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-
     ↪ location"
                Category="urn:oasis:names:tc:xacml:1.0:subject-category:
            access-subject"
                DataType="http://www.w3.org/2001/XMLSchema#string"
                MustBePresent="true">
            </xacml:AttributeDesignator>
    </xacml:Apply>
    <xacml:AttributeValue DataType="http://www.w3.org/2001/
     ↪ XMLSchema#string">
        EU</xacml:AttributeValue>
   </xacml:Apply>
  </upol:Condition>
  <ob:ObligationsSet xmlns:ob="http://www.primelife.eu/ppl/obligation">
    <ob:Obligation>
     <ob:TriggersSet>
     <!-- UPOL Session Obligations: trigger on each session event if
```

```
44      evaluation result is "Permit"-->
45          <upol:TriggerRuleEvaluated FulfillOn="StartAccess" Effect="Permit
        ↪ " />
46       <upol:TriggerRuleEvaluated FulfillOn="EndAccess" Effect="Permit" /
        ↪ >
47       <upol:TriggerRuleEvaluated FulfillOn="RevokeAccess" Effect="Permit"
        ↪ />
48       </ob:TriggersSet>
49
50       <!-- Action : notify data subject to record an access for future
        ↪ reference -->
51       <ob:ActionNotifyDataSubject>
52        <ob:Media>Mail</ob:Media>
53        <ob:Address>customer.email@email.provider</ob:Address>
54       </ob:ActionNotifyDataSubject>
55      </ob:Obligation>
56     <!-- Obligation: delete after 3 months from information received -->
57     <ob:Obligation>
58      <ob:TriggersSet>
59       <TriggerAtTime>
60        <!-- the trigger is set in 3 months time-->
61         <MaxDelay>
62          <Duration>P0Y3M0DT0H0M0S</Duration>
63         </MaxDelay>
64       </TriggerAtTime>
65      </ob:TriggersSet>
66       <ob:DenyAllAndDeleteNow/>
67    </ob:Obligation>
68   </ob:ObligationsSet>
69  </Rule>
70  <!-- DETAILS OMITTED -->
```

6 Conclusion

The attention given by online service operators to data protection, especially for personal information, is constantly growing. Such trend can be explained also by some recent changes in the legal framework that bring new requirements for achieving full compliance. For example, the EU General Data Privacy Regulation require significant changes in the way entities collect and process personal data of EU citizens, anywhere in the world, but similar requirements are also requested for operating in markets like Australia, China and Russia.

In this work we presented our proposal to address such new requirements. It consists of an approach to data protection that uses data protection policies to achieve such compliance, using as technical means, a combination of access and usage control measures. Our approach foresees to facilitate the production and human understandability of the data protection policies, by allowing stakeholders to express them using a controlled natural language, subsequently transformed into a new and automatically enforceable policy. To this extent, we developed a new policy language, UPOL, as part of our commitments in the EU

FP7 Coco Cloud project. Its main aim is to obtain a unique language that is powerful enough to express legal, security, and privacy constraints in automatically enforceable policies, focussed on the sharing and management of (personal or otherwise sensitive) data over the Cloud. Now, our development continues in the EU H2020 C3ISP project, extending data protection also to Big Data analytics scenario, especially considering cyber security information sharing.

Our initial results count the implementation of a UPOL mechanism, capable of enforcing automatically obligations as trusted element by data subjects, processors and controllers. We managed to define UPOL policies that go towards the fulfillment of some data controller obligations as stated by the GDPR. We are currently working towards structuring and extending more our language, in order to support more data protection use cases, looking at personal data but also at more in general, confidential information especially in the cyber security domain.

Acknowledgements. This work was partly supported by EC-funded projects Coco Cloud [grant no. 610853] and by C3ISP [grant no. 700294].

References

1. Ardagna, C.A., et al.: Primelife policy language. In: W3C Workshop on Access Control Application Scenarios. W3C (2009)
2. Caimi, C., Gambardella, C., Manea, M., Petrocchi, M., Stella, D.: Legal and technical perspectives in data sharing agreements definition. In: Berendt, B., Engel, T., Ikonomou, D., Le Métayer, D., Schiffner, S. (eds.) APF 2015. LNCS, vol. 9484, pp. 178–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31456-3_10
3. Carniani, E., D'Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage control on cloud systems. Fut. Gener. Comput. Syst. **63**, 37–55 (2016). https://doi.org/10.1016/j.future.2016.04.010
4. Coco Cloud Consortium: Coco Cloud website (2016). http://www.coco-cloud.eu
5. Colombo, M., Lazouski, A., Martinelli, F., Mori, P.: A proposal on enhancing XACML with continuous usage control features. In: Desprez, F., Getov, V., Priol, T., Yahyapour, R. (eds.) Grids, P2P and Services Computing, pp. 133–146. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-6794-7_11
6. Di Cerbo, F., Martinelli, F., Matteucci, I., Mori, P.: Towards a declarative approach to stateful and stateless usage control for data protection. In: Proceedings of the 14th International Conference on Web Information Systems and Technologies, WEBIST 2018, Seville, Spain, 18–20 September 2018, pp. 308–315 (2018). https://doi.org/10.5220/0006962503080315
7. Di Cerbo, F., Some, D.F., Gomez, L., Trabelsi, S.: PPL v2.0: uniform data access and usage control on cloud and mobile. In: Matteucci, I., Mori, P., Petrocchi, M. (eds.) 1st IEEE/ACM International Workshop on TEchnical and LEgal aspects of data pRIvacy and SEcurity, TELERISE 2015, Florence, Italy, 18 May 2015, pp. 2–7. IEEE Computer Society (2015). https://doi.org/10.1109/TELERISE.2015.9
8. European Parliament and Council: Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation) (2016). Accessed 27 Apr 2016. http://goo.gl/LfwxGe

9. Gambardella, C., Matteucci, I., Petrocchi, M.: Data sharing agreements: how to glue definition, analysis and mapping together. ERCIM News **106**, 28–29 (2016). http://ercim-news.ercim.eu/en106/special/data-sharing-agreements-how-to-glue-definition-analysis-and-mapping-together
10. Lazouski, A., Martinelli, F., Mori, P.: A prototype for enforcing usage control policies based on XACML. In: Fischer-Hübner, S., Katsikas, S., Quirchmayr, G. (eds.) TrustBus 2012. LNCS, vol. 7449, pp. 79–92. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32287-7_7
11. Matteucci, I., Petrocchi, M., Sbodio, M.L.: Cnl4dsa: a controlled natural language for data sharing agreements. In: Proceedings of the 2010 ACM Symposium on Applied Computing SAC 2010, pp. 616–620. ACM, New York (2010). https://doi.org/10.1145/1774088.1774218. http://doi.acm.org/10.1145/1774088.1774218
12. OASIS: eXtensible Access Control Markup Language (XACML) Version 3.0 (2010)
13. Park, J., Sandhu, R.: The UCON ABC usage control model. ACM Trans. Inf. Syst. Secur. (TISSEC) **7**(1), 128–174 (2004)
14. Pearson, S., Casassa Mont, M.: Sticky policies: an approach for managing privacy across multiple parties. Computer **44**(9), 60–68 (2011)
15. Trabelsi, S., Njeh, A., Bussard, L., Neven, G.: PPl engine: a symmetric architecture for privacy policy handling. In: W3C Workshop on Privacy and Data Usage Control, vol. 4 (2010)
16. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal model and policy specification of usage control. ACM Trans. Inf. Syst. Secur. **8**(4), 351–387 (2005). https://doi.org/10.1145/1108906.1108908. http://doi.acm.org/10.1145/1108906.1108908