LPL, Towards a GDPR-Compliant Privacy Language: Formal Definition and Usage

Armin Gerl[1](✉), Nadia Bennani[2], Harald Kosch[1], and Lionel Brunie[2]

[1] DIMIS, University of Passau, Passau, Germany
{Armin.Gerl,Harald.Kosch}@uni-passau.de
[2] LIRIS, University of Lyon, Lyon, France
{Nadia.Bennani,Lionel.Brunie}@insa-lyon.fr

Abstract. The upcoming *General Data Protection Regulation (GDPR)* imposes several new legal requirements for privacy management in information systems. In this paper, we introduce LPL, an extensible Layered Privacy Language that allows to express and enforce these new privacy properties such as personal privacy, user consent, data provenance, and retention management. We present a formal description of LPL. Based on a set of usage examples, we present how LPL expresses and enforces the main features of the GDPR and application of state-of-the-art anonymization techniques.

Keywords: Anonymization · GDPR · LPL · Personal privacy
Privacy language · Privacy model · Privacy-preservation · Provenance

1 Introduction

Privacy is a research field which is tackled by different disciplines including computer and legal sciences. Each discipline has its own point of view on this complex topic. In computer science, privacy languages, in addition to express privacy rules, have been proposed to solve individual problem statements of privacy like informing users of the privacy settings of a website [1] or sharing and trading with (personal) data [2]. Furthermore, a privacy language is a data model of formal description which is machine-readable for automatic processing.

The *General Data Protection Regulation (GDPR)*, which will enter into force on 25th May 2018 [3, Art. 99 No. 2], is designed to standardise data privacy laws across Europe, to protect and empower all EU citizens (*Data Subjects*) data privacy and to rework the way organizations (*Controllers*) approach data privacy. Hereby, it advises to take a set of technical and organisational measures that could be summarized by two main principles, which are *Privacy by Design* and *Privacy by Default*, especially to protect *Data Subject Rights*.

We interpret *Privacy by Design*, which is an already existing concept that becomes now a legal requirement in the *GDPR*, as the requirement for a cross-domain definition of privacy policies which can be integrated in current business processes [4]. Therefore, privacy should be made available in all technical systems. To reach the *Privacy by Default* principal, it should be ensured that data access is permitted solely to persons and organizations that have the rights to access it or to which the *Data Subject* gives an explicit consent [3, Art. 25]. Additional legal aspects are listed in Sect. 2.1.

Our objective is to design a privacy language which aims to facilitate expressing legal requirements under the usage of privacy-preserving methodologies. With a formal definition of the privacy language we want to fulfill the principle of *Privacy by Design* by creating a machine-readable privacy policy for integration in technical systems. Furthermore, to fulfill the principle of *Privacy by Default*, we aim to cover all crucial privacy processes including the *Data Subject* giving its *consent* to the data processing, storage of personal data, transfer of personal data between *Controllers*, and privacy-preserving querying. Hereby, our proposed privacy language will serve as the base for a privacy-preserving framework supporting all mentioned processes. The privacy language presented in this paper allows expressing a static status of an organization, which we plan to extend by dynamic scenarios in future works.

To illustrate the purpose of our language, let's take the example of a *Data Subject* who registers an account for a service of *Controller C1*. Based on the consent, the personal data as well as our privacy language representing the legal privacy policy will be stored. According to the agreed privacy policy, the data will be transferred to *Controller C2* for statistical processing, whereas an additional privacy policy is created between both *Controllers* represented by our privacy language. The original privacy policy will then be appended to allow provenance for the personal data. *Controller C2* is processing personal data from several sources with different privacy policies as a service. Based on the requesting entity, *Controller C2* anonymizes the data according to the different individual privacy policies. Therefore, it can preserve privacy according to the legal regulations while delivering the best possible data utility. Additionally, both *Controllers* have to fulfill the *Data Subject Rights* given by the *GDPR*, e.g. disclosure of personal data. The legally required responses will be generated automatically based on our privacy language, reducing the workload for a *Controller*. To the best of our knowledge, there is no language that lets express and enforce the illustrated privacy-preserving features.

The main contribution of this paper is to present our *Layered Privacy Language (LPL)*. A formal description of its components is given. Then a set of usage patterns illustrating how policies are enforced are presented. The main focus is hereby on the *Query-based Anonymization*. Our goal with LPL is to model and enforce privacy policies, so that in Large-Scale Data and Knowledge-Centered-Systems it is possible to handle different personal privacy settings and therefore comply with the GDPR.

The remaining of the paper is structured as follows. In Sect. 2, considered aspects of privacy are listed and objectives for our proposed privacy language are derived from them. Section 3 reviews related works and positions LPL to them. Section 4 presents the formal description of LPL. Section 5 presents several usage patterns to illustrate several privacy aspects. Finally, Sect. 6 concludes and outlooks for future works.

2 Requirements

A privacy language should be able to express both legal and privacy-preserving requirements. Those requirements will be derived from the law and regulations and current state of the art of privacy-preservation methodology.

2.1 Legal View

We put the focus on the legal situation of Europe. A privacy policy or a privacy form can be translated as a set of rules describing how data has to be *processed*. Hereby, *processing* is broadly defined as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction of data [3, Art. 4 No. 2]. *Data Minimisation* denotes that only the minimum amount of data, which is necessary for the *processing* of the *purpose*, should be inquired from the *Data Subject* [3, Art. 5 No. 1 c)]. A privacy policy consists of several *purposes of the processing* [3, Art. 4 No. 9], describing what data is used, how it is used, when it will be deleted, who will use the data and if the data is anonymized [3, Art. 13]. Therefore, a privacy language has to be at least capable of modelling a set of purposes that have a set of data, set of data recipients, retention and the possibility to describe anonymized data [5]. Additionally several aspects of the European laws on privacy should be considered:

- *Consent:* A user has to give his *consent* for the *processing* of its data [3, Art. 6]. Hereby, the GDPR specifies that a *consent* has to be given freely, specific, informed and unambiguous [3, Art. 4 No. 11].
- *Personal Data:* The *GDPR* specifies *personal data* as any information that is related to an identified or identifiable natural person. This is a broad definition including among others name, location data, (online-)identifier and factors of a natural person [3, Art. 4 No. 1].
- *Purpose of the Processing:* The *processing* of *personal data* is only allowed for the defined purpose for which the user gave its *consent*. The *GDPR* specifies that personal data can only be collected and *processed* for legitimate *purpose of the processing* [3, Art. 5 No. 1 (b)]. The *purpose of the processing* is determined by the *Controller* which is a natural or legal person, public authority, agency or other body [3, Art. 4 No. 7]. Exceptions to this are also possible but will not be further discussed [3, Art. 6].
- *Retention:* According to the *GDPR*, personal data has to be deleted when it is no longer necessary for the *purpose of the processing* for which it was collected, which is a part of the *'right to erasure'* or *'right to be forgotten'* of the data subject [3, Art. 14 No. 1]. Therefore, deletion of personal data is

strictly bound to the *purpose of the processing*. The policy is to delete data when it is no longer necessary for the *purpose of the processing* or the *purpose of the processing* is completed. For example if the *purpose of the processing* is solely to use the e-mail for the newsletter, then data is revoked, once the newsletter subscription is completed.
– *Data Subject Rights:* The *GDPR* defines several *Data Subject Rights* including among others the *'right of access by the data subject'* [3, Art. 15], *'right to rectification'* [3, Art. 16], *'right to erasure'* [3, Art. 17], and *'right to object'* [3, Art. 21] giving the *Data Subject* several rights that have to be considered. For example, if the *Data Subject* has given its *consent* for the *processing* of the *personal data* to a *Controller*, then the *Data Subject* has also the right to demand the deletion of the *personal data* if there is a valid reason for it [3, Art. 17].

This breakdown of the legal regulations omits further exceptions and special cases for each of the mentioned points in favour of the scope of this paper. We are further aware that the European law constraints are not compliant with other regulations like the Health Insurance Portability and Accountability Act (HIPAA), but similar basic concepts can also be found in these regulations [6].

2.2 Privacy-Preserving View

We will focus on *Anonymization and Privacy Model Requirements*, *Data Storage Requirements* and *Personal Privacy Requirements* to describe the considered requirements for our privacy language. We are aware that further requirements from other privacy research fields like database trackers [7] could be added, but those would be more relevant for a privacy framework, than a privacy language, and therefore are out of scope for this paper.

Anonymization and Privacy Model Requirements. There are several privacy models like *k-Anonymity* [8], *l-Diversity* [9] or *t-Closeness* [10] defining the properties a data-set must have to prevent re-identification. To explain the requirement regarding the privacy-preservation item, let's take in this section the example of the *k-Anonymity* model. The properties of the privacy models are usually adjusted by one or several parameters. Illustrated on *k-Anonymity*, the parameter k defines for a data-set, that for each QID-group, at least k records have to exist [8]. A QID (quasi-identifier) is hereby an attribute which can, in combination with other QID attributes, be used for identification, but can by itself not used for identification.

Based upon the chosen value, the properties of the anonymized data-set in terms of utility and privacy are influenced.

Utility describes the data quality of an anonymized data-set in relation to the original data-set. The quality of the data can hereby be highly dependent on the context in which the data-set is supposed to be used. But in general *utility* can be measured by *Accuracy*, *Completeness* and *Consistency*. *Accuracy* measures the similarity, e.g. loss of information, between the anonymized value and the original value. *Completeness* measures the missed data in the anonymized data-set. *Consistency* measures if the relationships between data items is preserved. Based upon those measurements several methods have been developed for *utility*, e.g. for *k-Anonymity* the *height* metric is used [11].

A trade-off between privacy and utility has to be found which is defined by the privacy model as well as the corresponding parameters. An open question is who decides on the privacy model and its parameters. The choice of the privacy-preserving parameters defines the privacy of a data-set. Should those settings exclusively be decided by a privacy expert, e.g. a privacy officer in a company, a national authority or can this even be influenced or set by a user, is the question.

A higher value for parameter k results in higher privacy for the data-set. With increasing privacy of the data-set, the more likely it is that anonymization has to be applied on the data which has a negative impact on the *utility*. It has to be considered that an overestimated value will result in an undesired loss of *utility*. An underestimated value will result in insufficient privacy. This leads to the requirement that the definition of diverse privacy models including their privacy-preserving parameters has to be supported. This includes that data attributes have to be able to be assigned to a specified privacy group (e.g. Quasi-identifier) to enable a correct application of the privacy model. Because it is an open question which entity (privacy officer, *Data Subject*, or both) should influence the definition of the privacy-preserving parameters, we consider that those parameters can be influenced by both.

Data Storage and Transfer Requirements. Thus far we considered privacy only for a homogeneous data-set, but privacy has also to be considered in data-warehouses, and other storage solutions, which implicates different data-sources and queries. Each query-result can be imagined as a data-set for which privacy has to be considered. Therefore, the data can be anonymized at different points in time of privacy-preserving data-warehousing. For example it is possible that the source data is already anonymized before it is integrated in the data-warehouse. Alternatively, it is also possible to anonymize the data for each query conducted on the data-warehouse. In general, the possibilities for the point of anonymization in a data-warehouse scenario are *anonymized sources*, *pre-materialization anonymization*, *post-materialization anonymization* and *query-based anonymization*. Each of the approaches has its own advantages and disadvantages. It is shown that *post-materialization anonymization* has significant advantages over *anonymized sources* and *pre-materialization anonymization* in terms of data quality. If an untrusted data publisher model is selected then anonymized data sources are a necessity and therefore the *post-materialization anonymization* approach cannot be chosen. Experiments for *query-based anonymization* have not been conducted and therefore cannot be compared [12]. Based upon these results, we assume that a *as-late-as-possible anonymization* is advantageous, which we consider as a requirement.

In (privacy-preserving) data warehousing the data is combined in a single warehouse system from one or more data-sources and queried by data-recipients.

Generally speaking data will be *transferred*, *materialized*, *anonymized* and *queried*. This process may be run through several times, thus the origin of the data may be lost if it is not explicitly tracked. Therefore, it is required to store for each data record the corresponding privacy policy. Assuming that different data-sources have inherent different privacy policies, this process will lead usually to a data warehouse with diverse privacy policies that have to be considered. For example, we assume data-sources *source1* and *source2*, whereas *source1* delivers data under privacy policy *policy1* and *source2* under *policy2*. Data from both data-sources is combined in data-warehouse *warehouse1* including their corresponding privacy policies *policy1* and *policy2*. Therefore, it is a requirement that privacy policies, related to a specific data record, can be stored and transferred with the data. Hereby, the data should be stored as long as possible in its raw form to support a *as-late-as-possible anonymization*.

Furthermore, we are aware that sequential queries of a database have to be considered for privacy preservation. Each release can hereby contain a different set of attributes. The combination of those attributes, which are retrieved over time, may allow the identification of a *Data Subject* and therefore cause a privacy issue [7]. Although we are aware of this issue, we will not consider it for *Query-based Anonymization* within the scope of this paper.

Personal Privacy Requirements. The approach of allowing a user to set his *Personal Privacy Preferences* has been addressed in [13]. This approach gives the user the control over its privacy settings. To be more specific, it considers the minimum necessary anonymization of the data and therefore retains the maximum *utility* of the data. This approach considers *Personal Privacy Preferences* in the anonymization process of the data [13]. But we also consider the privacy model as part of the personal privacy settings. Therefore, an approach to find the minimum necessary privacy model and value can be derived for a data-set, which we denote as a requirement.

Additionally, we consider that it is possible that records from a data-source with personal privacy policies exist [13]. Therefore, the diversity of privacy policies that has to be considered rises. When the data is queried and transferred to a data recipient, it is possible that new privacy policies, representing additional privacy policies, of the queried data-warehouse are applied to the data-set and will be mixed up with the previous privacy policies, which can cause conflicts. This is not only restricted to a data-warehouse scenario, like mentioned before, but for every transfer of personal data. With every transfer of data it is possible, if not prevented, that the original *Data Subject* can no longer be identified explicitly, but its personal data is still processed. Therefore, a loss in provenance occurred. If the *Data Subject* wants to exercise his *Data Subject Rights* or the *Controller* has to prove the origin of the processed data it will no longer be possible. This has to be prevented. Consequently, we denote *Provenance* as a requirement.

This leads to the requirement of defining personal privacy within a privacy language on such a fine-grained level that each attribute may be influenced by both the privacy officer defining the maximum allowed anonymization and the *Data Subject* defining its own minimal privacy settings. Considering that several personalized privacy policies may be transferred and aggregated it is necessary to be able to track the origin of the data and therefore provenance has to be implemented within a privacy language.

2.3 Objectives

Based upon the legal and privacy-preserving view, we formulated the following objectives for a privacy language both representing legal privacy policies and ensuring privacy utilizing privacy-preserving methods.

It should be able to layer privacy policies to track the origin of the data and therefore enable *Provenance* for the personal data. Hereby, data-source and data-recipient should be differentiated to be able to grant fine-grained data processing rules. This has also to include the processing of *Data Subject Rights*, which are given by the law. The structure of the privacy language should match the structure of a privacy policy which is based on purposes, describing the circumstances of the data usage. Therefore, for each purpose it should be possible to describe the data that is processed, the data-recipient and retention. It is required that privacy for a data-set can be specified utilizing privacy models. But also privacy settings for single data fields are required to enable fine-grained personal privacy. Therefore, both a minimum level, defining privacy settings of the *Data Subject*, and maximum level, defining the upper limit for the *Controller*, are required. Finally, the privacy language should support the user consent on data access in a legal and human-readable way, whereas it has to considered that multiple languages are supported. Summarizing a privacy language should fulfill the following requirements:

- Differentiation between data-source and data-recipient to enable fine-grained access-control
- Modelling of purpose-based privacy policies with modelling of: data, retention and anonymization enabling personal privacy and privacy models
- Layering of privacy policies to ensure provenance
- Human-readability

In the following we will compare related works according to our requirements.

3 Related Works

We define a classification for privacy languages based on a broad literature research as well as on our previously defined requirements, which we then apply on a set of privacy languages to demonstrate a research gap.

Several privacy languages have been proposed in the literature, each with their own distinct purposes. Although they are classified as privacy languages by other works [14,15], we do not see a strong focus on *Privacy* (in a legal sense)

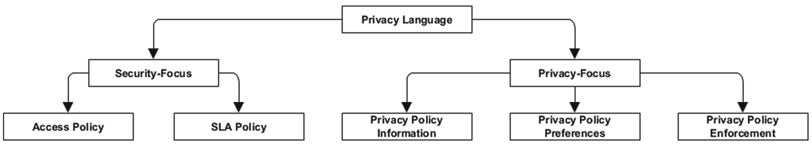
Privacy Language
Security-Focus
Privacy-Focus
Access Policy
SLA Policy
Privacy Policy Information
Privacy Policy Preferences
Privacy Policy Enforcement

Fig. 1. Categorizes for classification of privacy languages.

in every of them. Therefore, we developed a classification of privacy languages according to their intended purposes (see Fig. 1). Hereby, we use the *Privacy* for purposes in which the languages deal with legal aspects of privacy. We differentiate between five intended purposes, that we could identify by a broad literature research.

For *Access Policy*, policies for access control are implemented, such as *XACL* [16], *Ponder* [17], *Rei* [18], *Polymer* [19], *SecPAL* [20], *AIR* [21], *XACML* [22] and *ConSpec* [23]. For *Service Level Agreement (SLA) Policy*, agreements or contracts for B2B processes are implemented, such as *SLAng* [24,25] and *USDL* [26]. For *Privacy Policy Information*, policies are implemented to (only) inform about their contents, such as *P3P* [27] and *CPExchange* [28]. For *Privacy Policy Preferences*, personal privacy settings or preferences (of e.g. users) are modelled to be matched against policies, such as *APPEL* [29] and *XPref* [30]. For *Privacy Policy Enforcement*, policies are modelled and implemented to enforce privacy policies, such as *DORIS* [31], *E-P3P* [32], *EPAL* [33], *PPL* [34], *Jeeves* [35], *Geo-Priv* [36], *Blowfish Privacy* [37], *Appel* [38], *P2U* [2] and *A-PPL* [39].

Additionally, we analyse if the privacy languages consider several topics, that we derived from our requirements, which will be detailed as follows. For *Purpose-oriented*, the *purpose* of the *processing* of data is modelled as a high-level process and not only as low-level *CRUD* operations. For *Data-oriented*, each *data* can modelled uniquely and not only as (pre-defined) groups of data. For *Retention*, rules for automatic deletion of data or data-sets based on the retention have to be modelled. The possibility of an active deletion request, issued by the *Data Subject*, does not fulfill this criteria. For *Access-Control*, mechanisms for authentication and authorization have to be enabled by the model. For *Human-Readability*, the model should allow a human-readable presentation, so that the *Data Subject* is informed about the content. For *Privacy Model*, the minimal privacy properties of the data-set for a specific *purpose* have to be modelled. For *Personal Privacy*, the *Data Subject* should be able to dissent the use of data for a specific *purposes* or the *processing* of a specific *purpose*. Furthermore the anonymization of data for a specific *purpose* should be able to be influenced. For *Provenance*, after data has been transferred between (multiple) *Controllers* the original *Data Subject* should still be identifiable, so that this *Data Subject* can enforce his *Data Subject Rights*.

Based on the presented classification, we analysed a broad range of privacy languages. An broad and comprehensive overview is shown in Table 1. It can be observed that most privacy languages, which categorized as *Privacy Policy Enforcement*, are *Purpose-oriented* and *Data-oriented*. Furthermore, most

Table 1. Overview over fulfilled objectives for a privacy language combining both the legal and privacy-preserving views on privacy.

Category	Privacy language	Purpose-oriented	Data-oriented	Retention	Access-control	Human-readability	Privacy model	Personal privacy	Provenance
Access Policy	XACL	x	x		x				
	Ponder	x			x				
	Rei	x			x				
	Polymer	x							
	SecPAL		x		x				
	AIR	x	x		x	x			
	XACML	x	x		x				
	ConSpec	x			x				
SLA Policy	SLAng	x	x	x					
	USDL		x			x			
Privacy Policy Information	P3P	x	x	x	x				
	CPExchange	x	x	x	x				
Privacy Policy Preferences	APPEL	x	x						
	XPref	x	x						
Privacy Policy Enforcement	DORIS		x		x				
	E-P3P	x	x	x	x				
	EPAL	x	x		x				
	PPL	x	x	x	x				
	Jeeves	x	x		x				
	Geo-Priv	x	x	x	x		x		
	Blowfish Privacy	x	x				x		
	Appel	x			x				
	P2U	x	x	x	x				
	A-PPL	x	x	x	x				

consider the topics *Retention* and *Access-Control*. Both *Geo-Priv* and *Blowfish Privacy* specify the anonymization of personal data and therefore deal with the topic *Privacy Model*. The topics *Human-Readability*, *Personal Privacy* and *Provenance* are not dealt with by any of the privacy languages categorized as *Privacy Policy Enforcement*. The topic *Human-Readability* is only dealt with by *AIR* and *USDL*.

In summary, there is a lack of the legal and privacy-preserving requirements in the design of privacy languages, that we will consider. It is also mentionable that the representation of legal privacy policies (especially according to the *GDPR*) has not been the intention of any of the described privacy languages and has also not been done according to our knowledge. In the following, we give a formal definition for LPL implementing the described requirements.

4 Layered Privacy Language (LPL)

In this section, we present a formal description of our *Layered Privacy Language (LPL)* which satisfies the requirements presented in Sect. 2. The structure and the components of the language are depicted in Fig. 2, whereas attributes are omitted. All the elements presented in the diagram are described, including their attributes, in the following subsections. For clarity of the description, Table 2 gives for each element, notations that will be used for a single element, a subset of elements and the complete set.

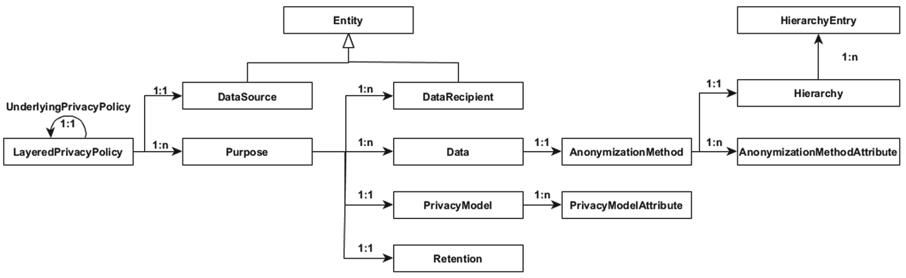

Fig. 2. Overview of the structure of LPL. Attributes are omitted for better readability.

4.1 Layered Privacy Policy

The root-element of our privacy language is the *LayeredPrivacyPolicy*-element *lpp*, which represents a privacy policy (legal contract), e.g. between a user and a company. Only a single *lpp* is supposed to be defined for a LPL compliant file, e.g. privacy policy. A *LayeredPrivacyPolicy*-element

$$lpp = (version, name, lang, ppURI, upp, ds, \widehat{P}) \tag{1}$$

is a tuple consisting of the following attributes:

- *version:* Version number for future version management of LPL.
- *name:* A textual representation of the privacy policy name.
- *lang:* Defining the language of the human-readable description in the LPL privacy policy.
- *ppURI:* A link to the legal privacy policy to assure compliance with the current law, which is implemented as a static human-readable description of the privacy policy.

Additionally, each *LayeredPrivacyPolicy lpp* can have a reference to an *UnderlyingPrivacyPolicy upp*. Let an *UnderlyingPrivacyPolicy*-element be

$$upp = (version, name, lang, ppURI, upp', ds, \widehat{P}) \tag{2}$$

where upp' is another *UnderlyingPrivacyPolicy*-element denoting a previously consented privacy policy. The set of all upp elements is denoted by UPP and $\widehat{UPP}$ denotes a subset of UPP.

Table 2. Overview over all elements, their formal definition and a reference to their definition. Bold styled sets are tuples, which inherit an order.

Element	Single element	Subset of elements	Set of elements	Definition reference
LayeredPrivacyPolicy	lpp	$\widehat{LPP}$	LPP	Section 4.1
UnderlyingPrivacyPolicy	upp	$\widehat{UPP}$	UPP	Section 4.1
Purpose	p	$\widehat{P}$	P	Section 4.2
Entity	e	$\widehat{E}$	E	Section 4.3
DataSource	ds	$\widehat{DS}$	DS	Section 4.3
DataRecipient	dr	$\widehat{DR}$	DR	Section 4.3
Retention	r	$\widehat{R}$	R	Section 4.4
PrivacyModel	pm	$\widehat{PM}$	PM	Section 4.5
PrivacyModelAttribute	pma	$\widehat{PMA}$	PMA	Section 4.5
Data	d	$\widehat{D}$	D	Section 4.6
AnonymizationMethod	am	$\widehat{AM}$	AM	Section 4.7
AnonymizationMethodAttribute	ama	$\widehat{AMA}$	AMA	Section 4.7
Hierarchy	h	$\widehat{H}$	H	Section 4.7
HierarchyEntry	he	$\widehat{\mathbf{HE}}$	$\mathbf{HE}$	Section 4.7

This allows to create layers of privacy policies to satisfy the objective of being able to track privacy policies over multiple entities.

Let the ('most underlying') *leaf-LayeredPrivacyPolicy*

$$lpp_{leaf} = (version, name, lang, ppURI, \emptyset, ds, \widehat{P}) \tag{3}$$

be the first privacy policy for which a consent is given for. In other words, the *LayeredPrivacyPolicy* with no *UnderlyingPrivacyPolicy* is the initial privacy policy, which is usually a consent between an user and a legal entity. If an additional privacy policy lpp_{new}, e.g. for a data-transfer to a third party, has to be added to an existing $lpp_{existing}$, then the $lpp_{existing}$ will be wrapped by lpp_{new}. This results in

$$lpp_{existing} = (version, name, lang, ppURI, \emptyset, ds, \widehat{P}) \tag{4}$$

$$lpp_{new} = (version, name, lang, ppURI, lpp_{existing}, ds', \widehat{P'}) \tag{5}$$

which is valid for each additional added privacy policy lpp'_{new}. Hereby, the data source (ds') will be the data recipient (dr) of $lpp_{existing}$ and $\widehat{P'}$ can be $\widehat{P}$ or a subset of it. The *DataSource*-element ds and a set $\widehat{P}$ of *Purpose*-elements will be described in the following. The set of all lpp elements is denoted by LPP and $\widehat{LPP}$ denotes a subset of LPP.

4.2 Purpose

The *Purpose*-element p, representing a legal *purpose of the processing*,

$$p = (name, optOut, required, descr, \widehat{DR}, r, pm, \widehat{D}) \tag{6}$$

is a tuple consisting of the following attributes:

- *name:* A textual representation of the identifying name, e.g. 'marketing'. In the set of purposes there should be no duplicate names.
- *optOut:* A boolean defining if the *Purpose* is opt-out for *true* or opt-in for *false*. Opt-out implies that the user has to actively deny this purpose. In the opposite, opt-in implies that the user has to actively accept this purpose.
- *required:* A boolean defining if the *Purpose* has to be accepted by the user. If the user does not accept a required *Purpose* then there cannot be a consent for the corresponding *lpp*.
- *descr:* A human-readable textual representation of the purpose expressed in the language defined by the language *lang* of *lpp*.

Moreover, each *Purpose* is linked to a set $\widehat{DR}$ of dr, one *Retention*-element r, optionally one *PrivacyModel*-element pm and a set $\widehat{D}$ of d. The set of all p elements is denoted by P and $\widehat{P}$ denotes a subset of P. It is important to note that the set $\widehat{P}$ may be empty or consist of contradictory purposes, which is valid for the structure but illogical for a privacy policy.

4.3 Entity

The *Entity*-element e, representing persons, companies or any other entity that has processing-right on the data,

$$e = (name, classification, authInfo, type) \tag{7}$$

is a tuple consisting of the following attributes:

- *name:* Used for authorization in access control.
- *classification:* Classifies the *Entity* in either *Person* or *Legal Entity.*
- *authInfo:* Used for authentication of the *Entity*, e.g. a hashed password.
- *type:* Either *DataSource* or *DataRecipient.*

The set of all e elements is denoted by E and $\widehat{E}$ denotes a subset of E. The *Entity*-element inherits the following 2 elements, which do not add additional attributes, but are used for better readability of LPL.

DataSource. The *DataSource*-element ds inherits from *Entity*, whereas the *type* is set to the corresponding value.

$$ds = (name, classification, authInfo, \text{`DataSource'}) \tag{8}$$

The *DataSource*-element describes the current authority granting *DataRecipients* the *processing* of data, based upon its own processing-rights. For example this can be the user (person) for whom the personal data is dedicated to or a company (legal entity) that has collected the personal data for a specific purpose. The set of all ds elements is denoted by DS and $\widehat{DS}$ denotes a subset of DS.

DataRecipient. The *DataRecipient*-element dr inherits from *Entity*, whereas the *type* will be set to *DataRecipient.*

$$dr = (name, classification, authInfo, \text{`DataRecipient'}) \tag{9}$$

The *DataRecipient*-element represents the authority that gets specific processing-rights (defined by the *Purpose*) granted. This can be a person or a legal entity. For example given the *DataSource*-element representing the user (person) which the personal data is referring to, then this authority can grant the *DataRecipient* all processing-rights via $\widehat{P}$. Assuming ds_C represents a *Controller* C that has collected the data from a user ds_U under specific processing-rights $\widehat{P}_C$ and wants to grant a third party dr_T processing-rights $\widehat{P}_T$, then ds_C can only grant dr_T the usage within the limits of its own processing-rights $\widehat{P}_T \subseteq \widehat{P}_C$. It has to be noted that the processing-rights of ds_C are a subset of the processing-rights of the user, who has all the processing-rights $\widehat{P}_T \subseteq \widehat{P}_C \subseteq \widehat{P}_U$ The set of all dr elements is denoted by DR and $\widehat{DR}$ denotes a subset of DR.

4.4 Retention

The *Retention*-element r defines when the described data has to be deleted.

$$r = (type, pointInTime) \tag{10}$$

The element consists of the following attributes:

- *type:* Describing the general condition of the retention. Possible values are *Indefinite*, *AfterPurpose* and *FixedDate.*

– *pointInTime:* Textual representation describing the exact conditions for the retention.

Depending on the *type*, the $pointInTime$ has diverse meanings. The *type Indefinite* without a value for $pointInTime$ defines that there is no time constrained for the deletion of the data. The *type AfterPurpose* defines that after the completion of the corresponding purpose p the data has to be deleted within the time-frame specified by *pointInTime*. Lastly the *type FixedDate* in combination with *pointInTime* explicitly defines the date for the deletion of the data within the corresponding p. The set of all r elements is denoted by R and $\widehat{R}$ denotes a subset of R.

4.5 Privacy Model

The *PrivacyModel*-element *pm* specifies the privacy conditions that have to be fulfilled for the data in a data-set. This element can be given but it is not mandatory. Alternatively, privacy can also be defined by *AnonymizationMethod*-element, defining personal privacy, or even omitted if not necessary, e.g. when the p does not describe any personal data.

$$pm = (name, \widehat{PMA}) \tag{11}$$

The applied privacy model is defined by the *name*, e.g. *k-Anonymity* [8] or *l-Diversity* [9]. Each privacy model can have a set of *PrivacyModelAttribute*-elements $\widehat{PMA}$. Currently, we limit to one privacy model *pm* for each purpose p. It may be a requirement that more than one privacy model is applied to a data-set [40], which would be a possible future extension. The set of all *pm* elements is denoted by PM and $\widehat{PM}$ denotes a subset of PM.

Privacy Model Attribute. A *PrivacyModelAttribute*-element *pma*, represents the configuration of a privacy model,

$$pma = (key, value) \tag{12}$$

is a tuple of the following attributes:

– *key:* Definition of a variable that is required by the correlating *pm*, e.g. k for *k-Anonymity.*
– *value:* Definition of the actual variable content, e.g. for k the value '2', which describes that there have to be at least two records within the same *QID*-set values to preserve the required k-anonymity property [8]

The set of all *pma* elements is denoted by PMA and $\widehat{PMA}$ denotes a subset of PMA. The decision for utilizing $\widehat{PMA}$ can be explained by the existence of privacy model (e.g. i *X, Y-Privacy* [41]) that support more than one variable.

4.6 Data

The *Data*-element d, representing a data field that is concerned by a purpose p,

$$d = (name, dGroup, dType, required, descr, pGroup, am) \tag{13}$$

is a tuple of the following attributes:

- *name:* Distinct name for the stored data field. A duplicate *name* within a $\widehat{D}$ of a *Purpose* is not allowed. Because this could lead to discrepancies in the processing, like it is possible with P3P *DATA-Elements*. P3P allows to define contrary rules for the same data element within one purpose. This made the determination of the valid rule unfeasible [42].
- *dGroup:* A textual representation of a logical data group. No predefined values are given. The logical data group can be used to specify data in e.g. a procedure directory, which usually does not refer to each data field but groups of it [3, Art. 30]. This enables to validate procedures directories [3] with a privacy policy automatically or even create one beforehand. E.g. data elements representing title, prename and surname of a person could have the *dGroup* '*name*'.
- *dType:* Defines the type of data that the attribute has. Possible types are *Text*, *Number*, *Date*, *Boolean*, *Value Set* for a set of predefined values and *Other* for any data type that doesn't fit the aforementioned types.
- *required:* A boolean defining if the data d to be accepted or could be neglected by the user. If the user does not accept a required d then the corresponding p will not be accepted. If the p is required, then the whole privacy policy lpp is not accepted.
- *descr:* A human-readable description of the data field. Possible notes on the anonymization, that is applied, can be added for better understanding. For example, the *age* will be only analysed in ranges from *'0–50'* and *'50–100'*.
- *pGroup:* This is the classification of the data field in *Explicit*, *QID*, *Sensitive* and *Non-Sensitive*. The processing of the data field by the privacy models is based upon this classification. E.g. for *k-Anonymity* the value of a data field which is classified *Explicit*, has to be deleted [8].

The *AnonymizationMethod*-element am defines the minimum anonymization for the data enabling personal privacy. The set of all d elements is denoted by D and $\widehat{D}$ denotes a subset of D.

4.7 Anonymization Method

The *AnonymizationMethod*-element am, represents the anonymization that is applied on a data,

$$am = (name, \widehat{AMA}, h) \tag{14}$$

is a tuple of the following attributes. The *name* represents the chosen anonymization method. There are several methods available, for example *Deletion*, *Suppression* or *Generalization*. Additionally each *AnonymizationMethod* has a set of *AnonymizationMethodAttributes*-elements $\widehat{AMA}$ and optionally a *Hierarchy*-element h. The set of all am elements is denoted by AM and $\widehat{AM}$ denotes a subset of AM.

Anonymization Method Attribute. An *AnonymizationMethodAttribute*-element ama, represents the configuration of a anonymization method,

$$ama = (key, value) \tag{15}$$

is a tuple of the following attributes:

– *key:* Definition of a variable that is required by the correlating am. Additionally, defining a maximum and minimum *Anonymization Level* is possible.
– *value:* Definition of the actual variable content.

The *key 'Minimum Level'* defines the minimum *Anonymization Level* that is applied to the actual data when used for a specific purpose. This allows to anonymize *as-late-as-possible.* The *key* with the value *'Maximum Level'* defines the maximum level of anonymization that is applied and therefore gives the creator of the privacy policy the possibility to define requirements for the anonymization. The set of all ama elements is denoted by AMA and $\widehat{AMA}$ denotes a subset of AMA.

Hierarchy. The *Hierarchy*-element h, saving all possible pre-calculated values for one data field and the correlating anonymization method. The hierarchy h will be used during the anonymization both for the *Minimum Anonymization*, enabling personal privacy, and the *Application of the Privacy Model.*

$$h = (\widehat{\mathbf{HE}}) \tag{16}$$

Each h consists of a tuple of *HierarchyEntry*-elements $\widehat{\mathbf{HE}}$, representing each entry in the hierarchy. We denote $h.length$ as the amount of he elements in $\widehat{\mathbf{HE}}$.

$$h.length = |\widehat{\mathbf{HE}}| \tag{17}$$

The tuple of all he elements is denoted by $\mathbf{HE}$ and $\widehat{\mathbf{HE}}$ denotes a sub-tuple of $\mathbf{HE}$. We decided for the calculation and storage of the possible anonymized values within the privacy language over the calculation of the anonymized value per query. The hierarchy is optional as not every *AnonymizationMethod*, e.g. *Deletion*, will have several possible values. Additionally it may also not be suitable for all use cases to store pre-calculated values. The set of all h elements is denoted by H and $\widehat{H}$ denotes a subset of H. Next section presents several usage patterns of LPL privacy policies.

5 Usage of LPL

In this section we present the life-cycle of LPL. Furthermore, we present *Query-based Anonymization*, *Provenance* and *Retention* based on LPL.

For the sake of clarity and to avoid redundancy, for each usage pattern we will define progressively the elements examples of LPL that are mandatory for the illustration. Each previously defined element could be referenced in further usage patterns if required. Only the relevant attributes will be instantiated for better readability.

5.1 Life-Cycle

For LPL, we present the life-cycle steps through the following scenario: A company e_{C1} wants to create a new web-service which collects and uses personal information. Therefore, e_{C1} creates a legal privacy policy that a user e_{U1}, using the service, has to accept. In this case ds_{U1} is the *'DataSource'* and dr_{C1} is the *'DataRecipient'*. Optionally, the data that is collected by e_{C1} could be transferred to a third party e_{C2} for a specific usage. Therefore, a contract between e_{C1} and e_{C2} has to be concluded for the data transmission, whereas ds_{C1} is the *'DataSource'* and dr_{C2} is the *'DataRecipient'*.

$$ds_{U1} = (\text{'U1'}, \text{'Person'}, publicKey_{U1}, \text{'DataSource'}) \tag{18}$$
$$dr_{C1} = (\text{'C1'}, \text{'Legal Entity'}, publicKey_{C1}, \text{'DataRecipient'}) \tag{19}$$
$$ds_{C1} = (\text{'C1'}, \text{'Legal Entity'}, publicKey_{C1}, \text{'DataSource'}) \tag{20}$$
$$dr_{C2} = (\text{'C2'}, \text{'Legal Entity'}, publicKey_{C2}, \text{'DataRecipient'}) \tag{21}$$

The usage of LPL in this scenario can be separated into the following steps (see Fig. 3):

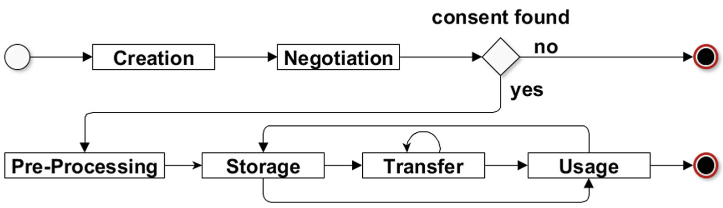

Fig. 3. Life-cycle of LPL.

1. *Creation:* Company e_{C1} converts the legal privacy policy to an LPL privacy policy lpp_{raw}. Hereby e_{C1} defines which *Purpose*- and *Data*- elements are necessary for the usage of the web-services as well as all other elements. In this case, a privacy policy will be transferred describing personal data to

be used by dr_{C1} and dr_{C2} for the purpose '*Marketing*'. This includes the anonymization for '*postal-code*', but not for '*salary*'.

$$lpp_{raw} = (version, \text{'LPP1'}, lang, ppURI, \emptyset, \emptyset, \{p_{U1}\}) \tag{22}$$

$$p_{U1} = (\text{'Marketing'}, optOut, required, descr, \{dr_{C1}, dr_{C2}\}, r, pm, \widehat{D}_1) \tag{23}$$

$$\widehat{D}_1 = \{d_{postal}, d_{salary}\} \tag{24}$$

$$d_{postal} = (\text{'postal-code'}, dGroup, dType, required, descr, \text{'QID'}, am_1) \tag{25}$$

$$am_1 = (\text{'Suppression'}, \{ama_1, ama_2, ama_3, ama_4\}, \emptyset) \tag{26}$$

$$ama_1 = (\text{'Suppression Replacement'}, \text{'*'}) \tag{27}$$

$$ama_2 = (\text{'Suppression Direction'}, \text{'backward'}) \tag{28}$$

$$ama_3 = (\text{'Minimum Level'}, \text{'2'}), ama_4 = (\text{'Maximum Level'}, \text{'4'}) \tag{29}$$

$$d_{salary} = (\text{'salary'}, dGroup, dType, required, descr, \text{'Sensitive'}, \emptyset) \tag{30}$$

2. *Negotiation:* The privacy policy lpp_{raw} is presented to the user e_{U1} via a user-interface, enabling an informed and voluntary consent. The user-interface should also give e_{U1} the possibility to dissent with defined parts of lpp_{raw} and still be able to form a contract with e_{C1}, whereas a personalized privacy policy $lpp_{ds_{U1}\text{-}dr_{C1}}$ is created. This leads to the insertion of ds_{U1}. If no consent is found the user cannot use the web-service and no data nor privacy policy of e_{U1} will be stored.

$$lpp_{ds_{U1}\text{-}dr_{C1}} = (version, \text{'LPP1'}, lang, ppURI, \emptyset, ds_{U1}, \{p_{U1}\}) \tag{31}$$

A user-interface that allows to personalize the LPL privacy policy has been developed and evaluated. It focuses on the consent or dissent to purposes. Further features, like the personalization of minimum anonymization for specific data or the consent or dissent to specific data, are planned for future work.

3. *Pre-Processing:* In this step, the $lpp_{ds_{U1}\text{-}dr_{C1}}$ is processed and validated. This step is conducted before the data values or the privacy policy is stored. For example if the privacy policy is modified by the user and stored, then it has to be re-validated to prevent malicious alterations.
4. *Storage:* Assuming consent is given by e_{U1} and therefore a contract between e_{C1} and e_{U1} is formed, the (personalized) privacy policy $lpp_{ds_{U1}\text{-}dr_{C1}}$ will be saved along with the data of e_{U1}. Therefore, $lpp_{ds_{U1}\text{-}dr_{C1}}$ is not intended for storing the actual data but to reference it.
5. *Transfer:* If e_{C1} transfers the data to e_{C2} the contract formed between those two entities is also converted into a LPL privacy policy $lpp_{ds_{C1}\text{-}dr_{C2}}$. And the existing personalized privacy policy will be added as an underlying privacy policy *upp*.

$$lpp_{ds_{C1}\text{-}dr_{C2}} = (version, \text{'LPP2'}, lang, ppURI, \{lpp_{ds_{U1}\text{-}dr_{C1}}\}, ds_{C1}, \{p_{U1}\}) \tag{32}$$

This allows a tracking of the data to its origin privacy policy. A (legal) privacy-aware usage is also possible, because each legal usage of data is defined by LPL and can be traced to the first consent between e_{U1} and e_{C1}. This step may be repeated several times if needed.

6. *Usage:* Whichever entity (in this scenario e_{U1}, e_{C1} or e_{C2}) wants to use the data it has to be verified that the entity is authenticated and authorized to query the data. If this is successful the data can be anonymized according to the corresponding purposes.

Summarizing a LPL privacy policy *lpp* should represent and ensure legal privacy policies utilizing the steps *Creation*, *Negotiation*, *Pre-Processing*, *Storage*, *Transfer* and *Usage*.

5.2 Query-Based Anonymization

In this step, LPL enables a query-based anonymization of the data of the *Data Subject* on purposes which have been consented to and expressed by the LPL privacy policy. Therefore, a query

$$q = \{e_{req}, p_{req}, \widehat{D}_{req}\} \tag{33}$$

is assumed as a request consisting of the following elements:

- e_{req}: The requesting entity. A e_{req} should be first authenticated and authorized to have access to the data.
- p_{req}: The purpose for which the data is requested. Data is only allowed to be used for designated purposes which are either consented to or given by the law.
- D_{req}: The requested data attributes are given to prevent undesirable access.

Hereby, the processes denoted as *Entity-Authentication*, *Purpose-Authorization*, *Entity-Authorization*, *Data-Authorization*, *Minimum Anonymization* and *Application of Privacy Model* will be conducted in the given order to allow a query-based anonymization (see Fig. 4).

Before the denoted process will be described in detail in the following, supportive data structures will be introduced.

Supportive Data Structures. For processing the LPL privacy policies, additional data structures are assumed available for a *Controller*. These are *Entity-Hierarchy*, *Entity-Lookup Table* $\widehat{E}_{lookup}$ and *Purpose-Hierarchy* including *Regulated Purposes*, which will be presented in the following. The usage of both *Entity-Hierarchy* and $\widehat{E}_{lookup}$ will be shown in the following sections for *Entity-Authentication* and *Entity-Authorization*. The usage of *Purpose-Hierarchy* and *Regulated Purposes* is shown during *Purpose-Authorization* process.

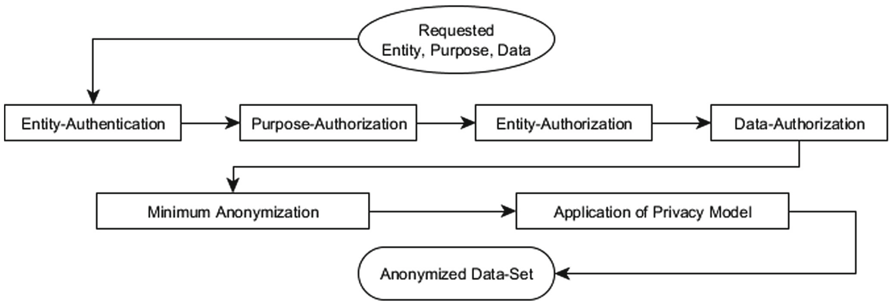

Fig. 4. Processes of the query-based anonymization.

Entity-Hierarchy. The *Entity-Hierarchy* allows to define *Child-Entities* that inherit the rights of the *Parent-Entity* (see Fig. 5). We assume that a privacy policy defines general data recipient roles e.g. a company, but for further control of the processing, which is defined in method descriptions, more fine-grained roles, e.g. a marketing department or even individual employees, have to be defined. This will be enabled by *Entity-Hierarchy*. Only the unique *name*, which specifies roles of e, is needed. This enables the *lpp* creator to define ds or dr, e.g. 'The user ds_{U1} accepts that the data is used by company dr_{C1} and dr_{C2}.'. If e_{U1} accepts this then dr_{C1} is granted the right to process the data according to the defined purpose. To limit the usage within dr_{C1} to a sub-set of employees, additional entities have to be defined to inherit the rights. It is also possible that a *Child-Entity* inherits from two *Parent-Entities*. This represents the use case when several users allow a company to use their data.

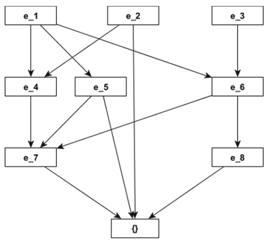

Fig. 5. Possible structure of *Entity-Hierarchy*.

$\widehat{E_{lookup}}$	
name	authorityChain
$name_1$	$authorityChain_1$
$name_2$	$authorityChain_2$
...	...
$name_n$	$authorityChain_n$

Fig. 6. General structure of $\widehat{E}_{lookup}$.

Entity-Lookup Table. The $\widehat{E}_{lookup}$ is the set of all entities (ds and dr) that exist within all stored *lpp* and additionally all entities that are defined in *Entity-Hierarchy*. Each entry of $\widehat{E}_{lookup}$ has to define the *name* and $authInfo$ (see Fig. 6). The $authInfo$ resembles the value that is authenticated against, e.g. the

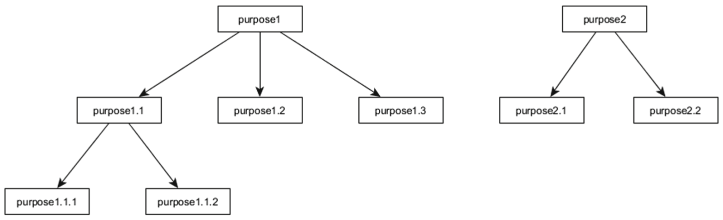

Fig. 7. *Purpose-Hierarchy* showing a possible inheritance hierarchy.

public key if a public/private key authentication is used or the hashed password if a username/password authentication is used. We introduce it to have the possibility to look up entities during the authentication process and potentially other processes without traversing the *Entity-Hierarchy*.

Purpose-Hierarchy and Regulated Purposes. We introduce the *Purpose-Hierarchy* for the *Purpose-Authorization*. The *Purpose-Hierarchy* is a data structure that consists of several trees of purposes. For each purpose it is possible to define *Child-Purposes* that inherit the rights of the *Parent-Purpose* (see Fig. 7). Therefore, only the unique *name* of p is needed which specifies purposes. This enables the *lpp* creator (e.g. dr_{C1}) to define a p in general in the first step, e.g. 'The user accepts that the d_{postal} and d_{salary} is used for '$Marketing$'.', and in the second step the privacy officer can define fine-grained inherent processes matching the requirements of a method description.

We do not assume that it is possible that a *Child-Purpose* inherits from two *Parent-Purposes*.

Additionally, we introduce *Regulated Purposes* that are given by law and regulations and don't have to be described by *lpp*. For example for the basic right for disclosure of confidential information [3, Art. 15] we denote the purpose '[*disclosure*]'. Those purposes allow entities to have access to data based upon the laws and regulations. For example a government agency might have the right to access specific data for a *Regulated Purpose* or a user is allowed to access its own data based upon the introduced *Regulated Purpose* '[*disclosure*]'.

Entity-Authentication. *LPL* will not be restricted to an access control solution by itself but builds upon existing access control methodology that we split into *Entity-Authentication* and *Entity-Authorization*.

Entity-Authentication is necessary to identify an entity e_{req} that requests the *usage* of data. In the following we will focus on the *Authentication* of an entity e against a privacy policy *lpp*. We will show how the previous structures will be used during the LPP life-cycle.

Creation. In the *Creation* step of a privacy policy *lpp* the *dr* entities will be defined. Assuming we have a privacy policy lpp_{raw} from Eq. (22). The purpose p_1 from Eq. (23) allows dr_{C1}, representing the company e_{C1}, to use the data $\widehat{D}_1$. For each dr, the corresponding *publicKey* will be added as the value for $authInfo$. For the *Creation*, we assume that the key pair, consisting of $publicKey_{C1}$ and $privateKey_{C1}$ for dr_{C1}, has been created beforehand.

Negotiation. During the *Negotiation* step it would be possible to add the ds_{U1} to lpp_{raw}, but the user still could deny the privacy policy which would make the generation of the public/private key pair obsolete. Therefore, the generation of $publicKey_{U1}$ and $privateKey_{U1}$ for ds_{U1} has to be conducted during the *Pre-Processing* step after consent is found.

Pre-Processing. During the *Pre-Processing* step, the set-up for the ds is conducted resulting in $lpp_{ds_{U1}\text{-}dr_{C1}}$, whereas the ds_{U1} from Eq. (18) is added to lpp_{raw}. Additionally, for each available e of $lpp_{ds_{U1}\text{-}dr_{C1}}$ including the corresponding *publicKey* will be saved in the $\widehat{E}_{look\text{-}up}$, if not available already. This cannot be done during the *Creation* because the user ds_{U1} has not been specified yet. Therefore, the $\widehat{E}_{look\text{-}up}$ will consist of ds_{U1}, dr_{C1} and dr_{C2}.

$$\widehat{E}_{look-up} = \{ds_{U1}, dr_{C1}, dr_{C2}\} \tag{34}$$

It is important to note that $\widehat{E}_{look\text{-}up}$ may be extended by additional entities, e.g. departments or employees, by the corresponding privacy officer.

Usage. During the *Usage* step an entity e_{req} requests data protected by the privacy policy $lpp_{ds_{U1}\text{-}dr_{C1}}$. To ensure that e_{req} is allowed to use the data, it has to be authenticated first. We assume the following scenarii:

Scenario 1: The employee e_{req1} of company C1 requests data concerned by $lpp_{ds_{U1}\text{-}dr_{C1}}$.
Scenario 2: The user e_{req2} requests data concerned by $lpp_{ds_{U1}\text{-}dr_{C1}}$.

The requesting entities have the following configuration

$$e_{req1} = (\text{‘C1’}, classification, privateKey_{C1}, type) \tag{35}$$
$$e_{req2} = (\text{‘U1’}, classification, privateKey_{U1}, type) \tag{36}$$

where e_{req1} is an employee using the authentication credentials from company e_{C1} and where e_{req2} represents the user e_{U1} from which the data protected by $lpp_{ds_{U1}\text{-}dr_{C1}}$ originates.

In general, we assume the following authentication process for our scenario. The $\widehat{E}_{look\text{-}up}$ is traversed to identify matching entities to the requesting entity e_{req}. We define that two entities match for our scenario if the following is valid.

$$e_{requesting}.name == e.name \tag{37}$$

If a matching entity e_{match} is found then the $publicKey_{match}$ is used to encrypt a *nonce* and sent to e_{req}.

$$encryptedMessage_{match} = encrypt(nonce, publicKey_{match}) \tag{38}$$

To successfully authenticate e_{req} the computed $encryptedMessage_{match}$ has to be decrypted with the $privateKey_{requesting}$ and sent back.

$$decryptedMessage = decrypt(encryptedMessage_{match}, privateKey_{requesting}) \tag{39}$$

The requesting entity will be authenticated if the $decryptedMessage$ equals the $nonce$.

$$message = decrypt(encrypt(message, publicKey), privateKey) \tag{40}$$

With this authentication mechanism we will describe the given scenarii. For scenario 1, the requesting entity is e_{req1}. In $\widehat{E}_{look\text{-}up}$ the matching entity is dr_{C1} with both $name$ values are '$C1$'. In the next step it will be evaluated if e_{req1} can authenticate as the dr_{C1} utilizing the public/private key authentication shown before

$$nonce = decrypt(encrypt(nonce, publicKey_{C1}), privateKey_{C1}) \tag{41}$$

which results in the authentication success for scenario 1.

In scenario 2, the requesting entity is e_{req2}. In $\widehat{E}_{look\text{-}up}$ the matching entity is ds_{U1}. Therefore, the authentication will be executed with $publicKey_{U1}$ from ds_{U1} and $privateKey_{U1}$ from e_{req2}.

$$nonce = decrypt(encrypt(nonce, publicKey_{U1}), privateKey_{U1}) \tag{42}$$

This successful authentication will result in the authorization of e_{U1} as ds of $lpp_{ds_{U1}\text{-}dr_{C1}}$, which allows the requesting entity to have all processing-rights authorized for the described data in $lpp_{ds_{U1}\text{-}dr_{C1}}$ as shown in the following.

In general, the authentication process for an entity requires the name of the requesting entity ($entityName$) as well as a $secret$ to verify the identity. The calculation of the $secret$ depends on the individual implementation of the authentication process which is based upon the $authInfo$. If a private/public key pair is used like in the aforementioned scenarii then the secret will be the $decryptedMessage$ that has to be verified against the $nonce$. Additional steps for setting up the $encryptedMessage$ and transferring it to the e_{req} would be necessary.

Purpose-Authorization. In the scenarii for *Purpose-Authorization*, we assume that the requesting entity e_{req} is authenticated and therefore only focus on the verification of the purpose p_{req} given in the query. The query is rejected if the purpose is invalid.

We assume that a purpose can have *Child-Purposes*, which are stored in the *Purpose-Hierarchy*. We assume the set-up of $lpp_{ds_{U1}\text{-}dr_{C1}}$ from Eq. (31). Moreover, we assume that the purpose '$Marketing$', as well as its child-purpose '$Newsletter$' and the purpose '*Development*' are available in *Purpose-Hierarchy*, next to the *Regulated Purposes* given by law and regulations, which we restrict to *[disclosure]* in the *Purpose-Hierarchy* (see Fig. 8). Therefore, we will describe the following scenarii for our set-up:

Scenario 1: Entity e_{req1} requesting the $\widehat{D}_1$ protected by $lpp_{ds_{U1}\text{-}dr_{C1}}$ for the purpose '*Newsletter*'.
Scenario 2: Entity e_{req2} requesting the $\widehat{D}_1$ protected by $lpp_{ds_{U1}\text{-}dr_{C1}}$ for the purpose '[*disclosure*]'.

In scenario 1, the entity e_{req1} requests data for the purpose of '*Newsletter*'. The data is protected by $lpp_{ds_{U1}\text{-}dr_{C1}}$ containing only purpose p_{U1} which *name* is '*Marketing*' defining the authorized purpose. Because we consider a *Purpose-Hierarchy*, where every *Child-Purpose* of the authorized purpose is also authorized, a set of *Authorized Purposes* has to be generated. Therefore, the requested purpose '*Newsletter*' will be identified in the *Purpose-Hierarchy* (see Fig. 8) and all its *Parent-Purposes* as well as the purpose itself will be returned {'Marketing', 'Newsletter'}. Then, the purpose is considered as an authorized purpose as its name behaves to the calculated purpose set.

In scenario 2, the entity e_{req2} requests data for the '[*disclosure*]' purpose. The corresponding set of *Authorized Purposes* consists therefore of {[disclosure]}. The purpose cannot be found in $lpp_{ds_{U1}\text{-}dr_{C1}}$, because this is a *Regulated Purpose* defining a special case and a new purpose has to be crafted for it during runtime automatically. For '[*disclosure*]' a new purpose $p_{disclosure}$ will be computed

$$\begin{aligned} p_{disclosure} = (&\text{`[disclosure]'}, optOut, required, descr, \{dr_{U1}\}, \\ & r, pm, \{d'_{postal}, d'_{salary}\}) \end{aligned} \tag{43}$$

$$dr_{U1} = (\text{`U1'}, \text{`Person'}, publicKey_{U1}, \text{`DataRecipient'}) \tag{44}$$

containing all d of the corresponding $lpp_{ds_{U1}\text{-}dr_{C1}}$ and the ds_{U1} is transformed to the dr of $p_{disclosure}$. Hereby d'_{postal} and d'_{salary} will be created by removing the respective am if existent. The authorization of e_{req2} will be executed according to the *Entity-Authorization.* Therefore, e_{req2} is requesting an authorized purpose.

In general, the authorization process for a purpose requires the *name* of the purpose and the corresponding *lpp* (see Listing 1). The authorization is successful if the *name* of the requested purpose or any of the corresponding *Authorized Entities* matches any of the p of the *lpp* or if a *Regulated Purpose* is requested.

Entity-Authorization. In the following, we will focus on the *Entity-Authorization.* An entity e_{req} is authorized to use the data if the *dr* or *ds name* of the *lpp* matches the *name* of e_{req}, whereas *name* can either specify a role or a specific user. We assume the employees e_{req3} and e_{req4} respectively from the marketing departments M1 and M2 of company C1 and C2.

$$e_{req3} = (\text{`M1'}, \text{`Person'}, authInfo, \text{`DataRecipient'}) \tag{45}$$

$$e_{req4} = (\text{`M2'}, \text{`Person'}, authInfo, \text{`DataRecipient'}) \tag{46}$$

We assume the set-up of $lpp_{ds_{U1}\text{-}dr_{C1}}$ from Eq. (31). User e_{U1} gave its consent on the usage of its personal information for the purpose of '*Marketing*' for

```
P authorizePurpose(purposeName, lpp):

    //initialize authorizedPurposes
    authorizedPurposes = {};

    //receive set of possible purposes
    possiblePurposes = purposeHierarchy.getParentPurposes(purposeName);

    if possiblePurposes != null
        //verify if purpose matches at least one p of lpp
        for possibleP : possiblePurposes

            switch (possibleP)
                //for each regulated purpose a individual case
                case '[disclosure]'
                    authorizedPurposes.add(createDisclosureP(lpp));

                //add purpose if name match
                default
                    for p : lpp.P
                        if match(possibleP, p.name)
                            authorizedPurposes.add(p);

    return authorizedPurposes;
```

Listing 1. Pseudocode describing the authorization of purposes of an *lpp* utilizing *Purpose-Hierarchy*. The special cases of *Regulated Purposes* are shown exemplary for '*disclosure*'. The *Entity-Hierarchy* is assumed to be accessible within the method.

company e_{C1}, which is encoded in $lpp_{ds_{U1}\text{-}dr_{C1}}$. No further agreements exist between user e_{U1} and company e_{C1}. Company e_{C1} has a marketing department '$M1$'. Company e_{C2} has a marketing department '$M2$'. Company e_{C1} grants e_{M2} from company e_{C2} access to the e_{U1} data for the purpose of '$Marketing$' encoded in $lpp_{ds_{C1}\text{-}dr_{M2}}$.

$$e_{M2} = (\text{'M2'}, \text{'Legal Entity'}, authInfo, \text{'DataRecipient'}) \tag{47}$$

$$p_{C1} = (\text{'Marketing'}, optOut, required, descr, \{e_{M2}\}, r, pm, \widehat{D}) \tag{48}$$

$$lpp'_{ds_{C1}\text{-}dr_{M2}} = (version, name, lang, ppURI, lpp_{ds_{U1}\text{-}dr_{C1}}, ds_{C1}, \{p_{C1}\}) \tag{49}$$

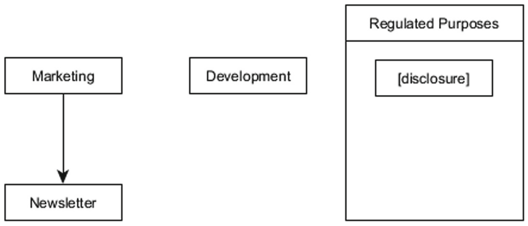

Fig. 8. *Purpose-Hierarchy* for the *Purpose-Authorization* scenarii. The *Regulated Purposes* are separated for better understanding.

Company e_{C1} and company e_{C2} exchange their data in a privacy conform way. We will present the following scenarii:

Scenario 1: The employee e_{req3} of the marketing department '$M1$' requests personal information of user e_{U1} for the purpose of '$Marketing$'.

Scenario 2: The employee e_{req4} of the marketing department '$M2$' requests personal information of user e_{U1} for the purpose of '$Marketing$'.

Scenario 3: The user e_{U1} makes use of him being entitled to the disclosure of confidential information [3, Art. 15], which is the basis for several additional interests, towards company e_{C1}. Therefore, e_{req2} requests its personal information for the purpose of '[*disclosure*]'.

For authorization we assume a role-based access control (RBAC) system [43] with the *roles* in Fig. 9. The *permission* is provided by the LPL privacy policy.

In scenario 1, the employee e_{req3} has the role '$M1$'. Based upon the (underlying) privacy policy $lpp_{ds_{U1}\text{-}dr_{C1}}$ the role '$C1$' is granted to use the data for '$Marketing$'. '$M1$' is a child-role of '$C1$' inheriting the permission to use the personal data of e_{U1}. Furthermore, the stated purpose in the LPL privacy policy '$Marketing$' matches the purpose of the requester. This concludes that e_{req3} is authorized to access the personal information of e_{U1} in scenario 1.

In scenario 2, an employee e_{req4} of company e_{C2} requests to access the personal data of e_{U1}. The value '$M2$' e_{req4} matches the data recipient *dr* of p_{C1}. The purpose '$Marketing$' matches the purpose defined in p_{C1} and therefore e_{req4} is authorized to access the data.

In scenario 3, the requesting entity e_{req2} matches the *DataSource*-element and therefore e_{U1} is authorized to request the personal data of itself.

In general, the authorization process for an entity requires the *name* of the e_{req} as well as the purpose for which it should be verified against (see Listing 2). The authorization is successful if the *name* of e_{req} or any of the corresponding *Parent-Entities* matches any of the *dr* of the purpose. This process has to be conducted after the *Purpose-Authorization* has been executed to consider special cases that are defined by the law and regulations as shown in scenario 3.

Data-Authorization. We assume that e_{req} is authenticated, authorized and uses an authorized purpose for the following scenarii. Hereby, the requested data $\widehat{D}_{req}$ has to be verified against the described data within the authorized purposes. If the verification is not successful, then the query will be rejected.

For the following scenarii we assume that an entity e_{req1} is querying different sets of data $\widehat{D}_{q1}$ and $\widehat{D}_{q2}$. Those requests are validated against p_{U1} of $lpp_{ds_{U1}\text{-}dr_{C1}}$. It is important to notice that p_1 only allows access to d_{postal}, d_{salary} and d_{age}.

$$\widehat{D}_{q1} = \{d_{postal}, d_{salary}\} \tag{50}$$

$$\widehat{D}_{q2} = \{d_{postal}, d_{salary}, d_{age}\} \tag{51}$$

$$d_{age} = (\text{`age'}, dGroup, dType, required, descr, pGroup, \emptyset) \tag{52}$$

```
boolean authorizeEntity(entityName, p):

    //receive set of authorized entities
    authorizedEntities = entityHierarchy.getParentEntities(entityName);

    if authorizedEntitites != null
        //verify if entity matches at least one dr of p
        for dr : p.DR
            if match(entityName, dr.name)
                return true;

    return false;
```

Listing 2. Pseudocode describing the authorization of an requesting entity e against a single *lpp* utilizing *Entity-Hierarchy*. The *Entity-Hierarchy* is assumed to be accessible within the method.

Additionally, we assume that the value of the *name* identifies a data-field and no additional matching between the *name* and the stored data-fields is necessary. This may change in a real world scenario but will not change basic behaviour that will be described in the following scenarii for this set-up:

Scenario 1: Entity e_{req1} requesting the $\widehat{D}_{q1}$ from p_{U1} of $lpp_{ds_{U1}\text{-}dr_{C1}}$.
Scenario 2: Entity e_{req1} requesting the $\widehat{D}_{q2}$ from p_{U1} of $lpp_{ds_{U1}\text{-}dr_{C1}}$.

In scenario 1, the entity e_{req1} requests $\widehat{D}_{q1}$ for the purpose of '$Marketing$'. In this scenario the requested set $\widehat{D}_{q1}$ is a sub-set of the data-set $\widehat{D}_1$ defined in p_{U1}. This means that all requested data-fields are defined in the authorized purpose and therefore the usage of the data is authorized. In scenario 2, the entity e_{req1} requests $\widehat{D}_{q2}$ for the purpose of '$Marketing$'. In this scenario, the requested set $\widehat{D}_{q2}$ is evaluated against the data-set $\widehat{D}_1$ defined in p_1. Each in $\widehat{D}_{q2}$ defined entry has to be also defined in $\widehat{D}_1$ and return an invalid result as the requested data '*age*' is not a member of $\widehat{D}_1$. Therefore, the usage of the data '*age*' is not authorized and the whole query will be rejected.

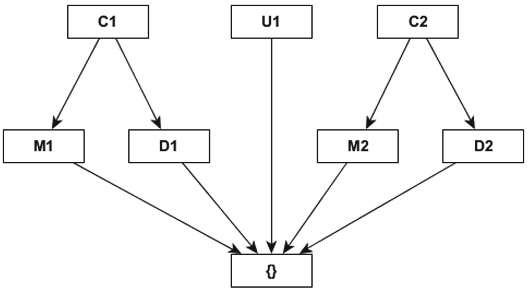

Fig. 9. Example roles for the *Entity-Authorization* scenarii. The {} denotes a default role without any processing-rights.

```
D authorizeData(requestedD, P):

    //initialize authorizedData
    authorizedData = {};

    //check for each requested data if it is authorized
    for d : requestedD
        //several authorized purposes are possible
        for p : P
            for dAuthorized : p.D
                if match(d.name, dAuthorized.name)
                    authorizedData.add(d);

    return authorizedData;
```

Listing 3. Pseudocode describing the authorization of data from *Authorized Purposes*. The set of requested is assumed to be computed from a parsed query.

In general, the authorization process for data requires the *name* of the data of the corresponding purpose p (see Listing 3). The authorization is successful if the *name* of the requested data matches any of d from any *AuthorizedPurpose*. This process has to be conducted after the *Purpose-Authorization* has been executed.

Minimum Anonymization. The *AnonymizationMethod*-element is introduced to specify (personalized) privacy settings for each *Data*-element. Only the relevant elements and attributes will be described for better understanding in the following three steps - *Negotiation*, *Pre-Processing* and *Usage*. We denote the anonymization of the data during the *Usage* step as *Minimum Anonymization* which is conducted after the *Data-Authorization*. We assume the following scenario (see Fig. 10).

The personal data $\widehat{D}$ of an user e_{U1} is requested by a company e_{C1}. The data is requested for the purpose '*Marketing*'. The corresponding purpose protecting the data is p_{U1} which is described by $lpp_{ds_{U1}\text{-}dr_{C1}}$ from Eq. (31). We focus only on the personal data for *postal-code* d_{postal} with the value of '94032' (for Passau in Germany). The configuration of am_1 from Eq. (26) describes the anonymization method *'Suppression'* with the replacement character *'*'* starting from the last character *'backwards'*. The minimum and maximum suppression levels are given.

Negotiation. In the *Negotiation* step it has to be verified if the, possibly from the user personalized, privacy policy is valid. The privacy policy is valid if, among to other conditions, the value of *Minimum Level* is not greater than the value of *Maximum Level*. Initially the value of *Maximum Level* and *Minimum Level* will be defined by e_{U1}, whereas *Maximum Level* defines the maximal usable anonymization for e_{U1} and the value of *Minimum Level* is an initial recommended proposal for the privacy requirements. This asserts the validity of the privacy policy before it is stored. We assume that the integrity of the value of *Maximum Level* is preserved.

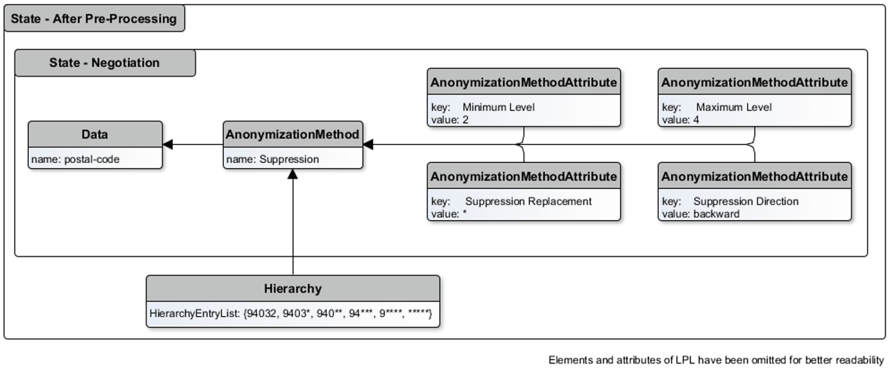
State - After Pre-Processing
State - Negotiation
Data
name: postal-code
AnonymizationMethod
name: Suppression
AnonymizationMethodAttribute
key: Minimum Level
value: 2
AnonymizationMethodAttribute
key: Maximum Level
value: 4
AnonymizationMethodAttribute
key: Suppression Replacement
value: *
AnonymizationMethodAttribute
key: Suppression Direction
value: backward
Hierarchy
HierarchyEntryList: {94032, 9403*, 940**, 94***, 9****, *****}
Elements and attributes of LPL have been omitted for better readability

Fig. 10. Relevant elements and attributes for the anonymization scenario in the state during the *Negotiation* and after the *Pre-Processing* step.

Pre-Processing. In the *Pre-Processing* step a *Hierarchy*-element h will be created for each anonymization method am. This step is conducted after the *consent* of the user for the privacy policy is given and before the data and the privacy policy are stored. In the *AnonymizationMethod*-element the *name* specifies the method applied on the data value. In our scenario, we assume that the anonymization method chosen by the user is *Suppression*. The set of *AnonymizationMethodAttributes ama* is utilized for this process. The ama_1 describes the character that is used for the replacement during the suppression, which is '$*$' in this scenario. The ama_2 describes the variation of the anonymization method. Thus, current scenario suppresses the value with '$*$' starting with the end of the *postal-code*. According to this configuration the *Hierarchy*-element will be created, which contains an ordered list of values, and added to the am. In our scenario the hierarchy h_1 for the postal-code '94032' will contain $\widehat{\mathbf{HE_1}}$.

$$\widehat{\mathbf{HE_1}} = \{\text{`94032'}, \text{`9403*'}, \text{`940**'}, \text{`94***'}, \text{`9****'}, \text{`*****'}\} \tag{53}$$

We denote that the first value is at *Level* '0' and the last element, in this case, is at *Level* '5'. This *Level* will be referred to by the *ama* with *Minimum Level* and *Maximum Level*. The hierarchy h_1 will be added to am'_1 replacing am_1.

$$am'_1 = (\text{`Suppression'}, \{ama_1, ama_2, ama_3, ama_4\}, h_1) \tag{54}$$

$$h_1 = (\widehat{\mathbf{HE_1}}) \tag{55}$$

Therefore, the h_1 holds all possible anonymized values for the data value. The values are ordered in an hierarchical way from the least to most anonymized value.

Usage. In the *Usage* step we assume e_{req1} is requesting the data of e_{U1} for the purpose '$Marketing$'. The corresponding p_1 for the request will be processed and therefore the defined anonymization method am'_1 has to be applied

```
value computeAnonymizedValue(d):

    //h(n) returns the hierarchy entry he at position 'n'
    return d.am.h(selectMinLevel(d.am.AMA).value);
```

Listing 4. Pseudocode describing the computation for the anonymized value utilizing hierarchy h.

to achieve the *Minimum Anonymization*. In this step only the *ama* with the *key* '*MinimumLevel*' and h are required to determine the anonymized value (see Listing 4). In our scenario ama_3 specifies the *Minimum Level* of '2', which means that the value on *Level* '2' of h_1 has to be selected. This results in the anonymized value of '$940**$' for the *postal-code*.

Application of Privacy Model. The functionality of the *PrivacyModel*-element and its corresponding *PrivacyModelAttribute*-element will be explained in the following.

In general, a privacy model describes the probability of a record in a data-set to be identified or de-anonymized [41]. For each record in a data-set, a privacy model can be defined utilizing *pm* and *pma* of LPL. The *pm* will be processed after the *Minimal Anonymization* has been conducted during the *Usage* step. To avoid computational overhead, we decided to compute the minimum required privacy model pm_{min} and apply it on the data-set. We specify the minimum required privacy model pm_{min} as the privacy model with the highest privacy requirements to guarantee that no initially given privacy constraints are violated. Therefore, the set of all defined privacy models has to be substituted. We decided to consider the attacks which are mitigated by the privacy models for a classification. Table 3 represents such a classification. Based upon this classification, a rule-set for minimizing the used privacy models can be created beforehand which we denote as *Privacy Model Substitution Table*. According to the classification, l-Diversity, which is the direct successor to k-Anonymity [9], covers all attacks of k-Anonymity, namely *Record Linkage* and *Attribute Linkage* [41]. Therefore, the set of privacy models 'k_{i1}-*Anonymity*' and 'l_{i2}-*Diversity*' will be substituted to '$l_{r1} - Diversity$'. The value for the parameter 'l_{r1}' of the resulting 'l_{r1}-*Diversity*' has to be calculated that it fulfills the privacy requirements of all prior privacy models. Both values can be treated equivalent because l-Diversity and k-Anonymity use similar definitions for the privacy. Therefore, 'l_{r1}' will have the maximum value of both 'k_{i1}' and 'l_{i2}' resulting in'$(l_{r1}, max(k_{i1}, l_{i2}))$'. According to the properties of t-Closeness, the minimum of the parameters will be used for the substitution of two t-Closeness privacy models [10]. The substitution with k-Anonymity or l-Diversity results in no reduction, because t-Closeness does not cover all the attack models of the other privacy models. Additional rules will be created by this scheme resulting in the *Privacy Model Substitution Table*.

Table 3. Excerpt of mitigated attack models by privacy models [41].

Privacy model	Attack model			
	Record linkage	Attribute linkage	Table linkage	Probabilistic attack
k-Anonymity	x			
l-Diversity	x	x		
t-Closeness		x		x

Table 4. *Privacy Model Substitution Table* for the scenarii used in Sect. 5.2.

Privacy model set	Substitution privacy model	Substitution privacy model attribute
$\{k_{i1}\text{-}Anonymity, k_{i2}\text{-}Anonymity\}$	$\{k_{r1}\text{-}Anonymity\}$	$\{(k_{r1}, max(k_{i1}, k_{i2}))\}$
$\{l_{i1}\text{-}Diversity, l_{2}\text{-}Diversity\}$	$\{l_{r1}\text{-}Diversity\}$	$\{(l_{r1}, max(l_{i1}, l_{i2}))\}$
$\{t_{i1}\text{-}Closeness, t_{i2}\text{-}Closeness\}$	$\{t_{r1}\text{-}Closeness\}$	$\{(t_{r1}, min(t_{i1}, t_{i2}))\}$
$\{k_{i1}\text{-}Anonymity, l_{i2}\text{-}Diversity\}$	$\{l_{r1}\text{-}Diversity\}$	$\{(l_{r1}, max(k_{i1}, l_{i2}))\}$
$\{k_{i1}\text{-}Anonymity, t_{i2}\text{-}Closeness\}$	$\{k_{r1}\text{-}Anonymity\}$, $\{t_{r2}\text{-}Closeness\}$	$\{(k_{r1}, k_{i1}), (t_{r2}, t_{i2})\}$
$\{l_{i1}\text{-}Diversity, t_{i2}\text{-}Closeness\}$	$\{l_{r1}\text{-}Diversity\}$ $\{t_{r2}\text{-}Closeness\}$	$\{(l_{r1}, l_{i1}), (t_{r2}, t_{i2})\}$

Note that it is possible that another privacy model exists which covers all attack models which would be more suitable for a substitution, but we limit our example only on k-Anonymity, l-Diversity and t-Closeness.

The computation of pm_{min} and the application on the data-set will be described in the following. For the scenarii we use the *Privacy Model Substitution Table* shown in Table 4. We assume records for e_{E1}, e_{E2}, e_{E3} and e_{E4} representing entries in a database table. For each record we assume that the record contains the postal-code d_{postal} and the salary per year d_{salary}. The values for each record are summarized in Table 5 for each e.

Table 5. Values for the d_{postal} and d_{salary} for each record.

Entity	Postal-code	Salary
e_{E1}	94032	36.000
e_{E2}	94032	45.000
e_{E3}	94034	38.000
e_{E4}	94032	45.000

We assume the privacy models 2-Anonymity pm_1, 3-Anonymity pm_2 and 2-Diversity pm_3.

$$pm_1 = (\text{‘k-Anonymity’}, \{pma_1\}) \tag{56}$$
$$pma_1 = (\text{‘k’}, \text{‘2’}) \tag{57}$$
$$pm_2 = (\text{‘k-Anonymity’}, \{pma_2\}) \tag{58}$$
$$pma_2 = (\text{‘k’}, \text{‘3’}) \tag{59}$$
$$pm_3 = (\text{‘l-Diversity’}, \{pma_3\}) \tag{60}$$
$$pma_3 = (\text{‘l’}, \text{‘2’}) \tag{61}$$

Only d_{postal}, which is classified as ‘QID’, will be considered in the anonymization process of k-Anonymity [8]. The specified d and pm are combined in the corresponding p individually for each record.

$$p_{E1} = (\text{‘p’}, optOut, required, descr, \widehat{DR}, r, pm_1, \{d_{postal}, d_{salary}\}) \tag{62}$$
$$p_{E2} = (\text{‘p’}, optOut, required, descr, \widehat{DR}, r, pm_1, \{d_{postal}, d_{salary}\}) \tag{63}$$
$$p_{E3} = (\text{‘p’}, optOut, required, descr, \widehat{DR}, r, pm_2, \{d_{postal}, d_{salary}\}) \tag{64}$$
$$p_{E4} = (\text{‘p’}, optOut, required, descr, \widehat{DR}, r, pm_3, \{d_{postal}, d_{salary}\}) \tag{65}$$

Explicit and non-sensitive attributes have been omitted for the following scenarii. For e_{E1} and e_{E2} the privacy model 2-Anonymity pm_1 is defined, for e_{E3} 3-Anonymity pm_2 is defined and for e_{E4} 2-Diversity pm_3 is defined.

Scenario 1: Data of e_{E1}, e_{E2} and e_{E3} are queried, the corresponding purposes are p_{E1}, p_{E2} and p_{E3}.
Scenario 2: Data of e_{E1}, e_{E3} and e_{E4} are queried, the corresponding purposes are p_{E1}, p_{E3} and p_{E4}.

We assume the same purpose ‘p’ for each query and will describe the computation of pm_{min} and the outcome for each scenario in the following.

In scenario 1, the data of the entities e_{E1}, e_{E2} and e_{E3} is queried. The *name* of the corresponding privacy models in pm_1, pm_1 and pm_2 are all the same. Therefore, the *value* of the corresponding attributes pma_1, pma_1 and pma_2 have to be compared, which show different configurations. The computation of the pm_{min1} results in the value ‘3’ for k, because no additional conflicts occur. The process will be executed in two steps. First pm_1 and pm_1 will be substituted the resulting pm will then be substituted with pm_2 to compute pm_{min1} finally. Therefore, for scenario 1 the valid privacy model for the data-set is 3-Anonymity.

$$pm_{min1} = pm_2 = (\text{‘k-Anonymity’}, \{pma_2\}) \tag{66}$$

Considering the corresponding values of Table 5 the data-set will be anonymized. The initial table T will be anonymized to table T’, whereas the postal-code will by suppressed to ‘9403∗’ for all records to achieve 3-Anonymity (see Table 6).

```
PM calculateMinimumPM(PM):

    resultPM = {};
    iteratorPM = PM.iterator();

    //initialize
    if resultPM.isEmpty()
        resultPM.add(iteratorPM.next());

    while iteratorPM.hasNext()
        tempResultPM = resultPM;
        pm = iterator.next();

        for pm' : resultPM
            //subsitutePM utilizes PrivacyModelSubstitutionTable
            minPM = substitutePM(pm',pm);

            //replace pm' in resultPM if necessary
            if not match(pm', minPM)
                tempResultPM.remove(pm');
                tempResultPM.add(minPM);

        resultPM = tempResultPM;

    return resultPM;
```

Listing 5. Pseudocode describing possible algorithm to select privacy models.

In scenario 2, the entities e_{E1}, e_{E3} and e_{E4} are queried. The *name* of the corresponding privacy models pm_1, pm_2 and pm_3 differ in this scenario. There exists a conflict between the privacy models '*k-Anonymity*' and '*l-Diversity*'. For the computation of the pm_{min2} both the correct privacy model and the corresponding value has to be determined by substituting first pm_1 and pm_2 and then substitute the result with pm_3. According to the *Privacy Model Substitution Table* the *name* for pm_{min2} will be '*l-Diversity*' with the *value* '3' for '*l*'.

$$pm_{min2} = (\text{`l-Diversity'}, \{pma_{min2}\}) \tag{67}$$
$$pma_{min2} = (\text{`l'}, \text{`3'}) \tag{68}$$

Considering the corresponding values of Table 5 the data-set will be anonymized. The initial table T will be anonymized to table T' (see Table 7). We assume for this scenario that the salary per year will be generalized to '50.000' and the postal-code will be suppressed to '9403∗' to match the conditions of 3-Diversity. In the scenarii we only showed the examples resulting in one privacy model. But it is also possible that the result contains several privacy models after the substitution has been conducted (see Listing 5).

5.3 Provenance

The *UnderlyingPrivacyPolicy* is introduced in LPL for *Privacy Policy Provenance*. This means that LPL enables to distinguish between different privacy policies and their origin. In this example we will show how the provenance is

Table 6. Transformation of table T to table T' for the given k-Anonymity privacy model. Column *Postal-code* is a *QID* and *Salary* is a *Sensitive Attribute.*

User	Table T		Privacy model	3-anonymous table T'	
	Postal-code	Salary		Postal-code	Salary
U1	94032	36.000	2-Anonymity	9403*	36.000
U2	94032	45.000	2-Anonymity	9403*	45.000
U3	94034	38.000	3-Anonymity	9403*	38.000

Table 7. Transformation of table T to table T' for the given k-Anonymity and l-Diversity privacy model. Column *Postal-code* is a *QID* and *Salary* is a *Sensitive Attribute.*

User	Table T		Privacy model	3-diverse table T'	
	Postal-code	Salary		Postal-code	Salary
U1	94032	36.000	2-Anonymity	9403*	<50.000
U3	94034	38.000	3-Anonymity	9403*	<50.000
U4	94032	45.000	2-Diversity	9403*	<50.000

preserved during the *Transfer* and *Usage* step (see Fig. 11). Assuming we have the scenario based on $lpp_{ds_{U1}\text{-}dr_{C1}}$ and $lpp_{ds_{C1}\text{-}dr_{C2}}$. Hereby, ds_{U1} agrees that the d_{postal} will be used by dr_{C1} and dr_{C2} for the purpose of '$Marketing$'.

Transfer. After the user e_{U1} has agreed on providing the *postal-code* to e_{C1} under $lpp_{ds_{U1}\text{-}dr_{C1}}$, e_{C1} can form a contract with e_{C2} for outsourcing the '$Marketing$' task creating $lpp_{ds_{C1}\text{-}dr_{C2}}$. Therefore, e_{C1} ensures the correct usage of the personal data by transferring the corresponding privacy policy with the personal data to e_{C2}. To ensure the provenance of the personal data, the privacy policy of the data-source will be added to the privacy policy of the dr. This represents the *Transfer* step, which can be repeated several times.

Usage. We will demonstrate following scenarii after the transfer of the data from e_{U1} to e_{C1} and from e_{C1} to e_{C2} has been executed. Hereby e_{req5} representing e_{C2} and e_{req2}, the original data source, will be compared as requesting entities as follows, demonstrating *Provenance.*

$$e_{req5} = (\text{`C2'}, classification, privateKey_{C2}, type) \tag{69}$$

Scenario 1: Request of d_{postal} from data-warehouse e_{E3} by e_{req5} for '$Marketing$'.
Scenario 2: Request of d_{postal} from data-warehouse e_{E3} by e_{req2} for '$[disclosure]$'.

We show for each scenario if the request is successful and which source for the data can be identified. Therefore, we assume the *Entity-Authentication*,

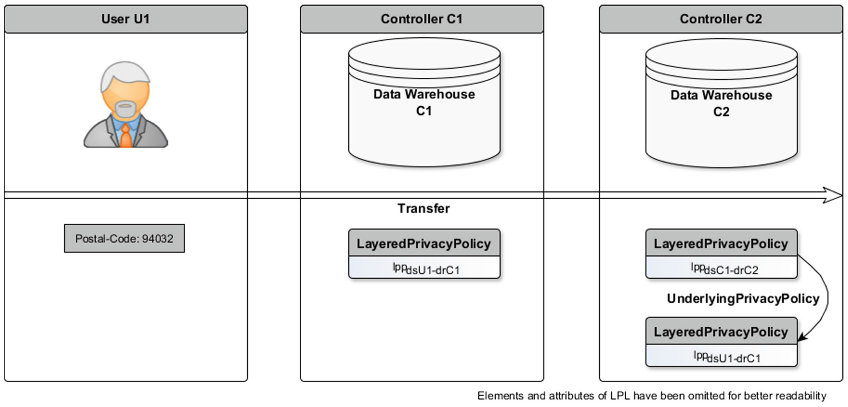

Fig. 11. Relevant elements and attributes for the layered privacy policy scenario visualizing the different privacy policy layers when the data is transferred.

Purpose-Authorization, *Entity-Authorization* and *Data-Authorization* have been conducted successfully.

In scenario 1, the $lpp_{ds_{C1}\text{-}dr_{C2}}$ has to be considered. *UnderlyingPrivacyPolicies* are considered successively. For *lpp* the purpose '*Marketing*' will be authorized successfully according to p_{U1}. The request will be successful and the data will be anonymized to the value '940 $**$' according to am_1. For $lpp_{ds_{C1}\text{-}dr_{C2}}$, it is possible to identify e_{U1} as source when the *UnderlyingPrivacyPolicies* are traversed utilizing the algorithm from Listing 6.

For scenario 2, the purpose '[*disclosure*]' will be authorized successfully as a *Regulated Purpose* and $p_{disclosure}$ will be created. Due to the missing *am* of d_{postal} the value '94032' will be returned. The source for the data is identified as the same like in scenario 1. For scenario 2, the *ds* could be identified successfully despite the original data has been transferred several times already and the original value '94032' was returned for e_{req2}. The *UnderlyingPrivacyPolicies* of LPL therefore enable the *Provenance* for the data source.

In general, to determine the source of a data in LPP it is necessary to firstly identify the data by the *name* within a purpose (see Listing 6). If the *lpp* has no *upp* then the *ds* of *lpp* is the source. If *upp* is available it has to be checked for all *p* of it, if the *name* is contained. If so the process is repeated till no *p* with *name* is found or no *upp* is available.

5.4 Retention

The *Retention*-element *r* provides the information when the data for a specific purpose *p* has to be deleted. With *Retention* a planned deletion of data is denoted, which has to be differentiated from an action-based deletion, e.g. like it is denoted by the right to erasure [3, Art. 17 No. 1 (b)] [3, Art. 17 No. 1 (c)] in which the user actively withdraws or objects. The retention process is executed during the *Usage* step.

```
e determineSource(lpp, data):

    dataSource = null;

    //if data can be found in any purpose
    for p : lpp.P
        for d : p.D
            //match data according to name
            if match(data, d)
                dataSource = lpp.ds;

    //recursivly iterate over all upp
    if lpp.upp != null
        temp = determineSource(lpp.upp, data);

        //if dataSource is found
        if temp != null
            dataSource = temp;


    return dataSource;
```

Listing 6. Pseudocode describing possible algorithm to determine the source of data.

There are three basic options that are used to define when the data has to be deleted for a specific purpose.

For the *type Indefinite* there is no designation point in time for the deletion of the data, so the data will not be deleted after a specific time.

$$r_1 = (\text{'Indefinite'}, \emptyset) \tag{70}$$

For the *type AfterPurpose* the deletion of the data depends on the completion of the purpose p itself, which will have to be managed separately within an encapsulating framework.

$$r_2 = (\text{'AfterPurpose'}, \text{'3 months'}) \tag{71}$$

After the purpose the data for this purpose has to be deleted within "3 *months*".

The last *type* is *FixedDate* which defines exactly the deletion.

$$r_3 = (\text{'FixedDate'}, \text{'01.01.2018'}) \tag{72}$$

Defining that on the *01.01.2018* the data of the corresponding purpose has to be deleted. The way the processing of data deletion based upon a rule r differs. One possibility is an automatic data deletion system. Within the system the data will be deleted exactly at the point in time when the r of the privacy policy defines it. The deletion checking has hereby to be done regularly. Further research on the requirements of an automatic data deletion system based on LPL is part of future work.

6 Conclusion and Future Works

This paper presents LPL, a privacy language that takes into account both legal and privacy-preserving requirements. After deriving the main objectives, we have given a formal definition of our privacy language. Later on, we describe the life-cycle of LPL as well as a usage pattern for *Query-based Anonymization* utilizing *Entity-Authentication*, *Purpose-Authorization*, *Entity-Authorization*, *Data-Authorization*, *Minimum Anonymization* and *Application of Privacy Model.* Additionally, we outline how *Provenance*, *Retention* is enabled. LPL does not cover all privacy aspects, which are partially already discussed by other works. Furthermore, LPL is an extensible language and can easily let new security and privacy concepts be integrated.

We consider LPL as a work in progress which we will extend in future works, hereby we will cover additional aspects of the GDPR by LPL.

First of all, there is an ongoing implementation work of LPL behind which we aim to validate experimentally a privacy-preserving framework based on LPL to allow an automatic query-based anonymization for data-storages, like data-warehouses. Not only anonymization, but also pseudonymization will be covered in future work and the impact of personal privacy and privacy models on privacy, utility and the performance will be evaluated.

To support privacy-aware applications (e.g. web-applications), which are based on such data-storages, it is necessary to optimize the response time for a query, by minimizing the computational overhead of LPL, so that common usage of applications is not hindered by privacy. We project a set of optimizations along the process steps. As examples we could cite the calculation of the minimum required privacy model or the authorization against every purpose for a requesting entity, because in both cases the execution time depends on the amount of processed privacy policies. Additionally we will extend our *Query-based Anonymization* to consider sequential queries and releases to avoid a privacy breach, were we will also consider logical interference.

Furthermore, we project to implement an user-friendly interface that enables the *Data Subject* to express simply its consent and have a fast overview over the privacy policy. Also personal privacy should be facilitated, allowing the user to refuse predefined purposes or adjust the privacy settings.

We assume that especially the combination of personal privacy and privacy models will influence the utility of the resulting data-set. Hereby, different factors, like the amount of personalized privacy policies or the properties of the data-set, will be investigated with the aim to identify all factors influencing the trade-off between privacy and utility.

Additionally, diverse *Regulated Purposes* have to be identified and implemented by our privacy-preserving framework to support the *Data Subject Rights* of the *GDPR*. On one hand this may relieve the *Controller* from burdensome manual responses and on the other hand this allows the *Data Subjects* to rely on a lawful execution of their requests.

Another future work concerns conflict detection between different privacy policies expressed in different LPL files. Those might occur during the transfer and aggregation of distinct data sources. Assuming the data sources origin from conflicting legal spaces, further investigations for the resolution of conflicts in terms of both legal and technical requirements and capabilities have to be executed. It is imaginable that *Provenance* has to be omitted in such a scenario for the sake of privacy. Therefore, scenarii with trusted and untrusted *Controllers* have to be considered in future works to assess under which circumstances *Provenance* can be provided.

Concluding we want to state that there is a various amount of open challenges in the field of privacy that arise from the GDPR from which LPL focuses on enforceable privacy policies.

References

1. Cranor, L.F., Arjula, M., Guduru, P.: Use of a P3P user agent by early adopters. In: Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, WPES 2002, pp. 1–10. ACM, New York (2002)
2. Iyilade, J., Vassileva, J.: P2U: a privacy policy specification language for secondary data sharing and usage. In: Proceedings of IEEE Security and Privacy Workshops, pp. 18–22, May 2014
3. Council of the European Union: General data protection regulation, April 2016. Regulation (EU) 2016 of the European Parliament and of the Council of on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC
4. Tsormpatzoudi, P., Berendt, B., Coudert, F.: Privacy by design: from research and policy to practice – the challenge of multi-disciplinarity. In: Berendt, B., Engel, T., Ikonomou, D., Le Métayer, D., Schiffner, S. (eds.) APF 2015. LNCS, vol. 9484, pp. 199–212. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31456-3_12
5. von Lewinski, K., Pohl, D.: Kommunikation von Datenschutz - Recht und (gute) Praxis. Stiftung Datenschutz, June 2017
6. Chowdhury, O., et al.: Privacy promises that can be kept: a policy analysis method with application to the HIPAA privacy rule. In: Proceedings of the 18th ACM Symposium on Access Control Models and Technologies, SACMAT 2013, pp. 3–14. ACM, New York (2013)
7. Shmueli, E., Tassa, T.: Privacy by diversity in sequential releases of databases. Inf. Sci. **298**, 344–372 (2015)
8. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertaint. Fuzziness Knowl. Based Syst. **10**(05), 557–570 (2002)
9. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data **1**(1) (2007). https://doi.org/10.1145/1217299.1217302. Article no. 3
10. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: Proceedings IEEE 23rd International Conference on Data Engineering, pp. 106–115, April 2007
11. Bertino, E., Lin, D., Jiang, W.: A survey of quantification of privacy preserving data mining algorithms. In: Aggarwal, C.C., Yu, P.S. (eds.) Privacy-Preserving Data Mining. Advances in Database Systems, vol. 34, pp. 183–205. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-70992-5_8
12. Fabian, B., Göthling, T.: Privacy-preserving data warehousing. Int. J. Bus. Intell. Data Min. **10**(4), 297–336 (2015)

13. Xiao, X., Tao, Y.: Personalized privacy preservation. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD 2006, pp. 229–240. ACM, New York (2006)
14. Kumaraguru, P., Cranor, L., Lobo, J., Calo, S.: A survey of privacy policy languages. In: Workshop on Usable IT Security Management (USM 2007) at Symposium On Usable Privacy and Security 2007 (2007)
15. Kasem-Madani, S., Meier, M.: Security and privacy policy languages: a survey, categorization and gap identification. CoRR, abs/1512.00201 (2015)
16. Hada, S., Kudo, M.: XML access control language: provisional authorization for XML documents, October 2000
17. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: Sloman, M., Lupu, E.C., Lobo, J. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44569-2_2
18. Kagal, L.: Rei: a policy language for the me-centric project. Technical report, HP Labs (2002)
19. Bauer, L., Ligatti, J., Walker, D.: Composing security policies with polymer. SIGPLAN Not. **40**(6), 305–314 (2005)
20. Becker, M.Y., Fournet, C., Gordon, A.D.: SecPAL: design and semantics of a decentralized authorization language. J. Comput. Secur. **18**(4), 619–665 (2010)
21. Khandelwal, A., Bao, J., Kagal, L., Jacobi, I., Ding, L., Hendler, J.: Analyzing the AIR language: a semantic web (production) rule language. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 58–72. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15918-3_6
22. Lockhart, H., Rissanen, E., Parducci, B.: eXtensible access control markup language (XACML) version 3.0. Technical report, OASIS (2013)
23. Aktug, I., Naliuka, K.: ConSpec - a formal language for policy specification. Electron. Notes Theoret. Comput. Sci. **197**(1), 45–58 (2008)
24. Lamanna, D.D., Skene, J., Emmerich, W.: Slang: a language for defining service level agreements. In: Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS 2003, pp. 100–106, May 2003
25. Meland, P.H., Bernsmed, K., Jaatun, M.G., Castejón, H.N., Undheim, A.: Expressing cloud security requirements for SLAs in deontic contract languages for cloud brokers. Int. J. Cloud Comput. **3**(1), 69–93 (2014). PMID: 58831
26. Oberle, D., Barros, A., Kylau, U., Heinzl, S.: A unified description language for human to automated services. Inf. Syst. **38**(1), 155–181 (2013)
27. Cranor, L., et al.: The platform for privacy preferences 1.1 (P3P1.1) specification. Technical report, W3C (2006)
28. Bohrer, K., Holland, B.: Customer profile exchange (CPExchange) specification, Version 1.0, October 2000
29. Cranor, L., Langheinrich, M., Marchiori, M.: A P3P preference exchange language 1.0 (APPEL1.0). Technical report, W3C (2002)
30. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: XPref: a preference language for P3P. Comput. Netw. **48**(5), 809–827 (2005). Web Security
31. Biskup, J., Brüggeman, H.H.: The personal model of data: towards a privacy-oriented information system. Comput. Secur. **7**(6), 575–597 (1988)
32. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, WPES 2002, pp. 103–109. ACM, New York (2002)

33. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise privacy authorization language (EPAL 1.2). Technical report, IBM (2003). https://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/
34. Ardagna, C., et al.: PrimeLife policy language. In: W3C Workshop on Access Control Application Scenarios. W3C (2009)
35. Yang, J., Yessenov, K., Solar-Lezama, A.: A language for automatically enforcing privacy policies. In: Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, pp. 85–96. ACM, New York (2012)
36. Schulzrinne, H., Tschofenig, H., Cuellar, J.R., Polk, J., Morris, J.B., Thomson, M.: Geolocation policy: a document format for expressing privacy preferences for location information. RFC 6772, January 2013
37. He, X., Machanavajjhala, A., Ding, B.: Blowfish privacy: tuning privacy-utility trade-offs using policies. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD 2014, pp. 1447–1458. ACM, New York (2014)
38. Turner, K.J., Reiff-Marganiec, S., Blair, L., Campbell, G.A., Wang, F.: APPEL: an adaptable and programmable policy environment and language. Technical report, Computing Science and Mathematics, University of Stirling, April 2014
39. Azraoui, M., Elkhiyaoui, K., Önen, M., Bernsmed, K., De Oliveira, A.S., Sendor, J.: A-PPL: an accountability policy language. In: Garcia-Alfaro, J., et al. (eds.) DPM/QASA/SETOP-2014. LNCS, vol. 8872, pp. 319–326. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17016-9_21
40. Prasser, F., Kohlmayer, F., Kuhn, K.A.: A benchmark of globally-optimal anonymization methods for biomedical data. In: 2014 IEEE 27th International Symposium on Computer-Based Medical Systems, pp. 66–71, May 2014
41. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: a survey of recent developments. ACM Comput. Surv. **42**(4), 14:1–14:53 (2010)
42. Yu, T., Li, N., Antón, A.I., A formal semantics for P3P. In: Proceedings of the 2004 Workshop on Secure Web Service, SWS 2004, pp. 1–8. ACM, New York (2004)
43. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996)