P2BAC: Privacy Policy Based Access Control Using P-LPL

Jens Leicht and Maritta Heisel
Paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Duisburg, Germany

Keywords: Access Control, Privacy Policy, P-LPL, Policy Enforcement.

Abstract: Privacy policies are used to inform end-users about the processing of their personal data by service providers on the Internet. These policies are, however, not systematically enforced. There could be discrepancies between the policy provided to the end-users and the actual access control policies applied by the service provider. We propose the Privacy Policy Based Access Control (P2BAC) system to tackle this issue. P2BAC uses computer-processable privacy policies expressed in the Prolog-Layered Privacy Language (P-LPL) to make decisions on whether some data may be accessed for a specific purpose. With P2BAC we extend the Privacy Policy Compliance Guidance (PriPoCoG) framework. Since P-LPL privacy policies can be customized by the end-user, we can consider end-users' privacy preferences during access control. P2BAC uses query rewriting to perform the access control. The decision point is implemented in Prolog and directly operates on the P-LPL privacy policy.

1 INTRODUCTION

Privacy policies are an important instrument for service providers to express their compliance with data protection legislation, especially since the General Data Protection Regulation of the European Union (GDPR) (European Parliament and Council of the European Union, 2016) came into effect. The enforcement of such privacy policies, however, is an open research topic that we approach by proposing an access control system working directly on privacy policies. In contrast, established access control systems require the definition of special access control policies.

We name our approach Privacy Policy Based Access Control (P2BAC) and suggest P2BAC as an extension to the Privacy Compliance Guidance (PriPoCoG) framework (Leicht et al., 2022). Using PriPoCoG ensures that the privacy policies to be enforced are GDPR-compliant. PriPoCoG is based on the Prolog-Layered Privacy Language (P-LPL), which enables data subjects (end-users) to personalize their privacy policies. This makes it possible in P2BAC to consider data subjects' privacy preferences when deciding whether access to some data should be granted or denied.

P2BAC provides efficient access control decisions based on query rewriting. For access requests concerning a single data subject we base the access decision on a Prolog implementation operating directly on the P-LPL privacy policy. Queries concerning larger data sets from multiple data subjects are handled separately, based on the concept of Purpose Based Access Control (PBAC) (Byun and Li, 2008).

P2BAC uses the concept of roles and role hierarchies as introduced in the hierarchical Role Based Access Control (RBAC) (INCITS, 2012). RBAC specifies three types of role hierarchy. While other RBAC-based access control systems are confined to one type of hierarchy, P2BAC supports all three types of role hierarchy.

We evaluate our approach by using a real-world privacy policy from the online marketplace ebay.com (eBay GmbH, 2021). Using ebay.com's privacy policy also extends the validation of the PriPoCoG framework.

In Section 2 we introduce necessary background information. Section 3 discusses related work and its relation to P2BAC. In Section 4 we present P2BAC and explain the P-LPL-based decision point, as well as query rewriting. We conclude our work in Section 5 and provide directions for future research.

2 BACKGROUND

In this section we give short introductions to the General Data Protection Regulation (GDPR), the privacy policy definition language P-LPL, the Privacy Policy

Leicht, J. and Heisel, M.
P2BAC: Privacy Policy Based Access Control Using P-LPL.
DOI: 10.5220/0011788500003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 686-697
ISBN: 978-989-758-624-8; ISSN: 2184-4356

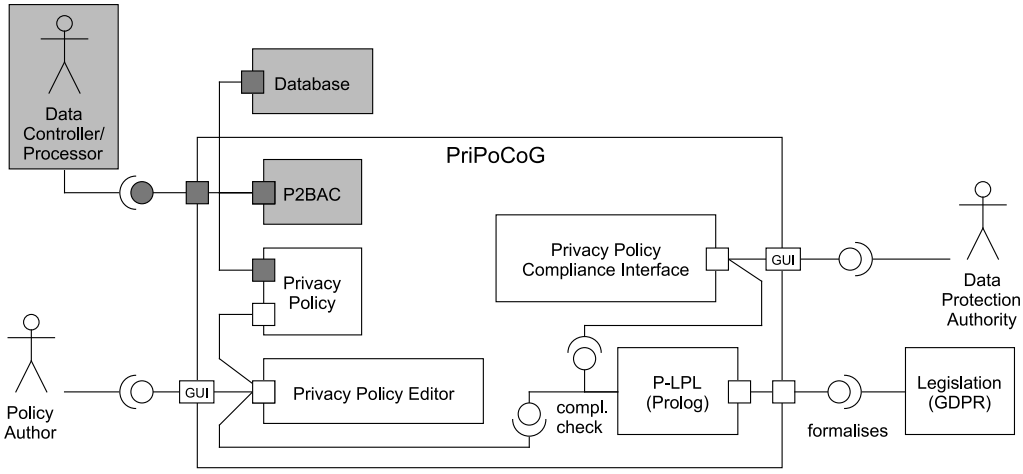

Figure 1: PriPoCoG framework with P2BAC extension highlighted in grey, based on (Leicht et al., 2022).

Compliance Guidance (PriPoCoG) framework, Purpose Based Access Control (PBAC), and role hierarchies.

2.1 GDPR Terminology

The General Data Protection Regulation of the European Union (GDPR) introduces some specific terminology (European Parliament and Council of the European Union, 2016).

The service provider accountable for the privacy policy and its enforcement on the service side is called *Data Controller*. Third parties processing some data on behalf of the data controller are called *Data Processors*. The European Union and its member states establish different *Data Protection Authorities*, which manage the enforcement of the GDPR. The person using a service and providing her or his personal data is called *Data Subject*.

2.2 P-LPL

The Prolog-Layered Privacy Language (P-LPL) is an extension and implementation of the Layered Privacy Language (LPL) by Gerl (Gerl, 2020). P-LPL uses Prolog[1], a declarative programming language, to formalize the language constructs of LPL and requirements from the GDPR. This formalization makes it possible to perform compliance checks on privacy policies expressed in P-LPL.

P-LPL provides the ability to store access control information inside the privacy policy. This could for example be access rules written in any access control language used by the data controller. Usually, that information would have to be given explicitly when defining the privacy policy, resulting in an organizational overhead for data controllers. With P2BAC, however, no explicit access control information needs to be stated in the privacy policy, because the privacy policy itself can be used to determine whether access is granted or denied.

2.3 PriPoCoG

The Privacy Policy Compliance Guidance (PriPoCoG) framework (Leicht et al., 2022) uses computer-interpretable privacy policies and checks them for compliance with the GDPR. Figure 1 shows an overview of PriPoCoG, with our extension highlighted in grey in the top left corner.

The base PriPoCoG framework uses P-LPL to formalise legislation (the GDPR). This formalisation is then used to check privacy policies for compliance with legislation. *Policy Author*s use the *Privacy Policy Editor* to define a *Privacy Policy* and check it for compliance with the GDPR. A *Data Protection Authority* can use the *Privacy Policy Compliance Interface* to check an existing *Privacy Policy*, expressed in P-LPL, for compliance with the GDPR.

With our proposed Privacy Policy Based Access Control, we extend the PriPoCoG framework by providing access control based on the GDPR-compliant privacy policies, thus, enforcing the privacy policies. The database is considered external here, as P2BAC could be used for any database, since the query rewriting is performed independently of the database, working directly with the privacy policy.

P2BAC provides an interface for data controllers and processors. They request access to some data inside a database, and P2BAC decides whether access is granted or not.

[1]https://www.swi-prolog.org/

2.3.1 Purpose Hierarchy

P-LPL organizes purposes of a privacy policy in a purpose hierarchy. This hierarchy contains purpose categories pre-defined by P-LPL, e.g., *serviceProvision* or *marketing*. Data controllers can define their own custom purposes. These purposes have to be assigned to the pre-defined categories. Such a purpose hierarchy also enables the use of very fine-grained purposes, while not overwhelming the data subjects, since they can get an overview of the purposes by looking at the main purposes higher up in the hierarchy.

2.3.2 P-LPL Privacy Policy

A P-LPL privacy policy is a computer-processable policy, which can be checked for GDPR-compliance. It consists of data elements defining the data that is handled by the data controller and data processor. Purposes describe the reason for which data is handled. An excerpt of a purpose from a P-LPL policy is presented in Listing 1 in Section 4.4.

P-LPL policies can be adjusted by the data subjects, providing their consent only for specific purposes. This customizability results in data subject-specific privacy policies.

Currently, P-LPL policies are defined manually using Prolog. However, an interactive editor is under development. P-LPL has been evaluated by translating two privacy policies from major e-commerce websites into P-LPL (Leicht et al., 2022; Ekundayo, 2022).

2.4 Purpose Based Access Control

Byun and Li's Purpose Based Access Control (PBAC) (Byun and Li, 2008) is based on *hierarchical* Role Based Access Control (RBAC) (INCITS, 2012). PBAC uses query rewriting for the Structured Query Language (SQL) (ISO, 2016). This rewriting allows for efficient access control on SQL databases.

2.4.1 Role Hierarchies

Hierarchical RBAC allows several types of hierarchies in which roles can be organized: (i) tree, (ii) inverted tree, and (iii) lattice. These hierarchies differ in the way access right are propagated through the structure. A role hierarchy allows fine-grained access control while keeping role management simple.

Purpose Based Access Control is limited to inverted trees. However, we introduce all three types of hierarchy, as P2BAC can operate with all of them.

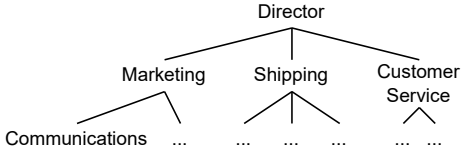

Figure 2: Role hierarchy (tree).

Tree. A tree role hierarchy is a basic structure where the parent node accumulates the access rights of all of its children. An example of a tree hierarchy is depicted in Figure 2. The role *Director* is the root of the hierarchy, with roles *Marketing*, *Shipping*, and *Customer Service* in the layer below the *Director*. The role *Marketing* is further split into sub-roles, e.g., *Communications*. The *Director* role inherits all the access rights of its children.

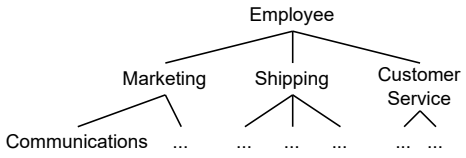

Figure 3: Role hierarchy (inverted tree).

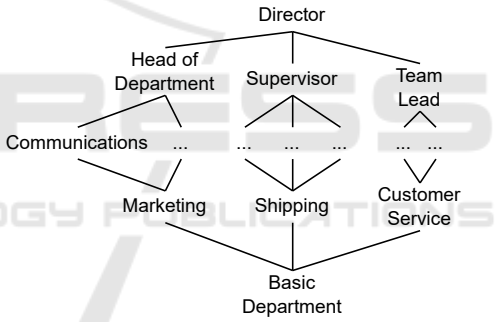

Figure 4: Role hierarchy (lattice).

Inverted Tree. An exemplary inverted tree hierarchy is depicted in Figure 3. Here, the root of the hierarchy is the most general role (*Employee*), which owns the least access rights. All roles below it inherit the rights of the more general role. The *Marketing* role, for example, inherits all access rights of the basic *Employee*, and can be assigned additional access rights.

Lattice. The lattice structure is a combination of tree and inverted tree, where access rights can be propagated top-to-bottom and bottom-to-top. Figure 4 shows an exemplary lattice, where the most basic role (*Basic Department*) is shown at the bottom, having the least access rights. *Marketing*, *Shipping* and *CustomerService* inherit all access rights from the *BasicDepartment*. The role *Communications* has the most specific access rights inherited bottom-up. The

upper half of the lattice operates like a tree hierarchy, and the *Director* inherits all access rights from its children: *Head of Department*, *Supervisor*, *Team Lead*.

2.4.2 Access Purpose

Access requests require an access purpose, which states for what purpose the data is requested. In Purpose Based Access Control the access purpose is determined from the role of the person requesting the access. Additionally, role attributes and environmental attributes are considered when identifying the access purpose.

2.4.3 Privacy Metadata: Access Codes

Purpose Based Access Control uses a binary encoding for representing and evaluating allowed/prohibited intended purposes and access purposes. Byun and Li evaluated the application of these access codes to be very efficient.

A bitwise "and" operation between the encoded *allowed intended purpose* (aip) and the *access purpose* (ap) is used to decide on the access request for a single data element. If the bitwise "and" operation returns a non-zero number, as shown in equation (1), access is granted, because the processing for the requested purpose is encoded in the *allowed intended purpose*. If the bitwise "and" results in zero (cf. equation (2), the access to this data element for the requested purpose is denied.

$$aip \,\&\, ap \neq 0 \Rightarrow \textbf{"Access granted"} \quad (1)$$

$$aip \,\&\, ap = 0 \Rightarrow \textbf{"Access denied"} \quad (2)$$

2.4.4 SQL Query Rewriting

Purpose Based Access Control uses an SQL extension, introducing the keyword **FOR**. This keyword is followed by the access purpose.

SQL rewriting is used to transform the queries containing the access purpose into queries that can be handled by the database itself. This transformation replaces the **FOR** section of the query with a **WHERE** condition containing *aip* and *ap* codes, which can be evaluated as described in Section 2.4.3.

3 RELATED WORK

Access control is a well-researched topic, with many different models for many diverse kinds of systems.

The well-established and standardized Role Based Access Control (RBAC) model lays the foundation for our access control system, as it introduces roles and role hierarchies, which we use to identify access purposes for database queries (INCITS, 2012).

Purpose Based Access Control (PBAC) by Byun and Li is based on RBAC and considers the purpose for which data is accessed when deciding on an access request (Byun and Li, 2008). We adapt PBAC to modern computer-processable privacy policies using P-LPL. Instead of separate access control policies, we use the privacy policy itself to decide whether access to personal data is granted or denied. PBAC has previously been implemented based on query rewriting (Mehta et al., 2017). Our P2BAC differs from this implementation in that we base our access decisions directly on the privacy policy and provide two different ways of evaluating access requests.

Hippocratic databases are also purpose-aware access control systems, implemented in prototypes of many different database systems (Agrawal et al., 2002; Laura-Silva and Aref, 2007). However, these never extended beyond the prototype state. Pallas et al. identified a lock-in issue with these prototypes and implemented an application-layer approach to PBAC, which resolves the lock-in issue as it can be used with any database system (Pallas et al., 2020). We preserve this database independence for access requests concerning single data subjects. P2BAC is however database-dependent for access requests concerning multiple data subjects. This reveals potential for future research.

Privacy-Aware Role Based Access Control (P-RBAC) by Ni et al. is another purpose- and privacy-aware access control model based on RBAC (Ni et al., 2010). It provides support for highly complex policies, which must be defined separately from the already used privacy policies. P-RBAC provides a conflict-checking algorithm, checking for conflicting rules inside the policy. This is necessary because P-RBAC has positive and negative permission rules, which could be defined in a conflicting way, permitting and prohibiting data access at the same time. Due to the permitting nature of P-LPL privacy policies, such conflicts cannot occur in P2BAC. P-RBAC extends access control to consider privacy; however, it still requires the definition of separate access control policies. Our P2BAC works directly on the privacy policy, thus avoiding the organizational overhead of defining separate access control policies.

Blockchain technology is also used in modern access control approaches. AuthPrivacyChain by Yang et al. is one approach using a blockchain (Yang et al., 2020). However, these approaches do not consider the context of privacy policies. In contrast, we focus on the efficient use of privacy policies to incorporate

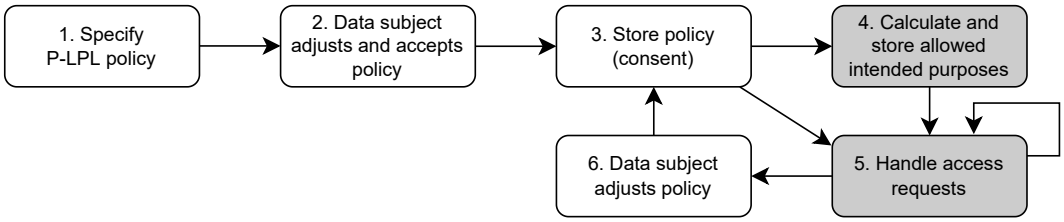

Figure 5: Process of using P2BAC: external behaviour/input in white, access control in grey.

data subjects' privacy preferences when performing the access control.

Betgé-Brezetz et al. developed an end-to-end privacy policy enforcement system for cloud services, which combines different policies (e.g., data controller, data subject, legislation) and enforces them (Betgé-Brezetz et al., 2013). This approach uses XACML to encode the policies that should be enforced. Compared to our P2BAC, directly working with the privacy policy that is negotiated between data subject and data controller, their approach introduces organizational overhead. Additionally, the separation of the privacy policy known by the data subject and the XACML policy applied at the system level can introduce discrepancies between the policies. The privacy policy provided to the data subject may change, while the XACML version does not and thus becomes out-dated.

Fatema et al. propose an enforcement system that can combine policies written in different policy languages (Fatema et al., 2010). Their enforcement system could be extended with our P2BAC decision point, allowing P2BAC to be used in combination with other policy languages.

Accountability is another important aspect in the context of privacy policies. Data controllers are obliged to collect data subjects' consent regarding the collection and processing of data. Data controllers need to be able to prove that data subjects provided consent before the collection and processing of data is performed. Jesus presents a concept that supports data controllers as well as data subjects with regard to accountability (Jesus, 2020). The proposed concept creates receipts every time a data subject provides consent. These receipts contain all purposes for which the data subject provided consent. Data controllers as well as data subjects can use the receipts to identify purposes for which consent was provided. P-LPL policies can be used as receipts, too. They clearly state for which purposes the data subject provided consent, ensuring that only accepted purposes can be used to access data subject's data.

4 CONTRIBUTION: P2BAC

We first give an overview of our proposed access control system P2BAC, followed by an example-driven introduction to each of its components. The example is based on a P-LPL version of ebay.com's privacy policy (Ekundayo, 2022). The P-LPL version of the privacy policy is available online[2]. We also explain P2BAC decision points, access codes and SQL query rewriting in the context of P2BAC.

4.1 Overview

Our privacy policy-based access control approach is based on Byun and Li's Purpose Based Access Control (Byun and Li, 2008). Their approach of using a purpose hierarchy to decide whether access is granted or not fits P-LPL very well. P-LPL privacy policies organize purposes in a hierarchy, which can be queried directly to make access control decisions. P2BAC extends the PriPoCoG framework (Leicht et al., 2022) as depicted in Figure 1, supporting the enforcement of P-LPL privacy policies.

Highly detailed system level access control policies are commonly quite complex and are, thus, not suitable for presentation to the data subjects. The hierarchical organization of purposes in P-LPL enables us to create different levels of detail when presenting the policy to the data subject. In combination with a suitable user interface, these levels of detail provide comprehensible and transparent privacy policies towards the data subject and fine-grained access control on the data controller side. A user interface for the representation of the privacy policies also improves the readability of P-LPL policies in general. Data subjects cannot be expected to understand the Prolog code behind a P-LPL policy, but a good visualisation will facilitate their understanding.

In Figure 5 we present the necessary steps that are required prior to using P2BAC in white and steps performed by P-LPL/P2BAC in grey. First, the data controller specifies a privacy policy using P-LPL. In the

[2]https://github.com/jensLeicht/PriPoCoG

second step the data subject needs to accept the privacy policy, at least partially, for access control requests to be evaluated positively. All access requests will be denied if the data subject has not given consent, yet. The customized policy is stored in step 3.

We distinguish two different types of access requests. Either a query tries to access data from multiple data subjects, for example when aggregating data for statistical purposes, or a single data subject's data is queried, for example when a customer service officer needs to access a single customer's address information.

For efficient handling of large queries concerning the data of multiple data subjects, P2BAC calculates access codes that are stored in the database together with the data itself. The access codes function like the *allowed intended purposes* (aip) in (Byun and Li, 2008) and allow very efficient evaluation of many purpose checks at once. The generation of allowed intended purposes is performed in step 4.

Smaller queries covering data concerning a single data subject can be handled directly in P-LPL, which is represented by the arrow from step 3 directly to step 5. Handling of access requests in step 5 is done by performing SQL query rewriting, either based on the access decision by P2BAC or using *allowed intended purposes* inside the database itself. Step 5 can be repeated indefinitely. If at any point in time, the data subject further adjusts his or her privacy policy, potentially withdrawing consent for some purposes, the adjusted privacy policy will be stored and the *aip*s will be re-calculated and updated in the database.

4.2 Concept

P2BAC is an adaptation of the Purpose Based Access Control by Byun and Li (Byun and Li, 2008). We make use of the computer-interpretable privacy policies in P-LPL and implement a decision point in Prolog, directly working with the privacy policy. This eliminates the need for separate access control policies, reducing the management overhead for the data controllers. Since data subjects are able to personalize the privacy policy, P-LPL and P2BAC assure that data subject's privacy preferences are considered when an SQL query tries to access their data.

As privacy policies only contain explicit consent for some processing of data for given purposes, it is unnecessary to consider *prohibited intended purposes* (pip) in our adaptation. Prohibitive rules induced by legislation can be checked directly in the P-LPL privacy policies during the compliance checks performed by the PriPoCoG framework (Leicht et al., 2022).

The P2BAC decision point is efficient in handling requests concerning data of a single data subject. If data for multiple data subjects is requested, we make use of access codes, as described in Sections 2.4.3 and 4.6. Data controllers using P2BAC should consider what data will be accessed in a single data subject context and what data may be used in a larger aggregating way. Based on these considerations, access codes should be generated for cases where larger data sets are requested regularly. For the single data subject context it is advantageous to use the P2BAC decision point:

- it reduces the overhead inside the database
- it does not require an adaptation of the database to accommodate the access codes
- it can be used as a proxy, processing the queries and forwarding the results to any (external) database
- it operates directly on the privacy policy and does not require an access code update after the data subject adjusted her or his privacy policy

The privacy policy, which is needed for a decision on an access request, is stored in a separate database and is linked to a data subject by database-internal IDs. The storage could be handled by any type of database that allows to store binary large objects (BLOBs) or character large objects (CLOBs). Even resource description framework (RDF) storage may be a viable solution.

In the following, we take an example-based look at the various parts of P2BAC and the different ways of making an access decision.

4.3 Roles and Purposes

In P2BAC the access decision is based on two important concepts: (i) roles, as defined in hierarchical Role Based Access Control (RBAC) (INCITS, 2012); and (ii) purposes, describing why some data may be processed. In the following we take a more detailed look at these concepts and how they interact in P2BAC.

4.3.1 Purpose Hierarchy

Ebay.com's purpose hierarchy with its 40 purposes is quite extensive; hence, we only use small excerpt in our running example. Figure 6 shows three of the purpose categories that P-LPL uses to manage the purpose hierarchy: *serviceProvision*, *marketing*, and *legalCompliance*. Ebay-specific purposes are the purposes *MailAdvertisements* (p24) and *MarketingCommunications* (p36). Both are sub-purposes of the *marketing* purpose category.

It is important to note that requesting access using one of the top purposes (e.g., *marketing*) reduces the chance of successfully accessing the data. Access to the requested data must be accepted by the data subject for all sub-purposes. This is necessary, because the person or software requesting access would be allowed to process the data for all sub-purposes.

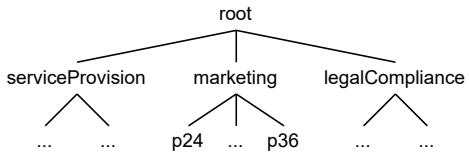

Figure 6: Purpose hierarchy containing purposes p24:"Mail Advertisements" and p36:"Marketing Communications".

4.3.2 Access Purposes

Access requests require an access purpose, which states for what purpose the data is requested. To determine this access purpose, P2BAC makes use of the role of the person requesting the data. This method of selecting the purpose uses a mapping of roles and purposes. In the following, we describe access purpose mappings for the different types of role hierarchy:

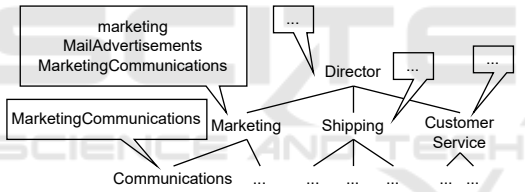

Figure 7: Role hierarchy (tree) with annotated purposes.

Tree. In a tree hierarchy, the parent nodes inherit all the access rights and all the mapped purposes from their children. Figure 7 shows an annotated version of the tree role hierarchy from Figure 2. The role *Communications* is mapped to the purpose *MarketingCommunications*, and the super-role *Marketing* is mapped to the purpose category *marketing* and the specific purpose *MailAdvertisements*. The *Marketing* role also inherits the *MarketingCommunications* purpose from its child *Communications*.

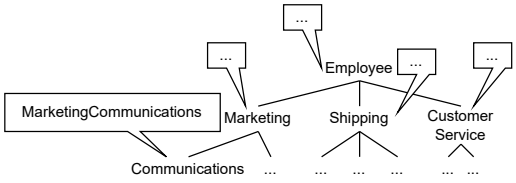

Figure 8: Role hierarchy (inverted tree) with annotated purposes.

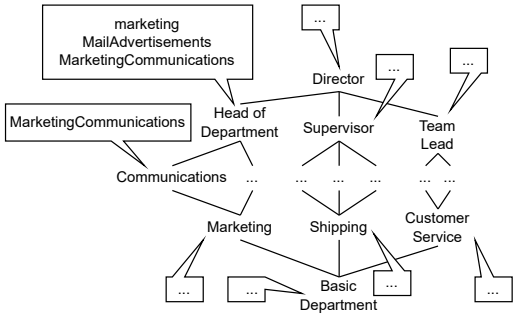

Figure 9: Role hierarchy (lattice) with annotated purposes.

Inverted Tree. In an inverted tree the children inherit all the access rights and purposes from their parents. Figure 8 shows the exemplary inverted tree hierarchy from Figure 3 with annotated purposes. In this example, the basic roles are not mapped to specific purposes, only the *MarketingCommunications* purpose is assigned to the *Communications* role.

Lattice. In the lattice hierarchy, both of the inheritance modes are combined. Figure 9 shows an exemplary purpose mapping for the lattice hierarchy in Figure 4. Basic purposes from the more general roles like the *Basic Department* are inherited upwards to roles such as *Communications*. In the upper part of the hierarchy the inheritance is continued as described for the regular tree: the role *Head of Department* inherits the purpose *MarketingCommunications* from the role *Communications*. The role *Director* inherits all the purposes of its children.

The examples show that a role can be mapped to multiple purposes. This especially applies to roles higher up in the role hierarchy. When accessing data, the purpose selection based on the role becomes difficult as the role accumulates many purposes from the hierarchy.

Since P2BAC access requests are only evaluated for a single specific purpose, a single purpose needs to be selected for the access request. This can be achieved by considering the software environment that is used to perform the access request. When that software is implemented, its internal operations can be associated with a specific access purpose.

If this approach is not possible because off-the-shelf software is used, plug-ins and middleware could be used to identify the purpose of a request. For example, when the data is requested from an email program and the person requesting the data has the role *Marketing*, then the requested purpose could be determined as *MailAdvertisements*.

Selecting the access purpose based on the software accessing the data also allows P2BAC to be used

```
purpose(p24 ,(‘‘ MailAdvertisements ’’, true , false ,
  1668495600,[...],[...],[    dt01 ,dt04 ,dt34 ,dt48 ,dt58 ,
  dt74 ],[    ],[    ],[ dr2 ],[ lb4 ],[    ], r )).
```

Listing 1: Excerpt of the P-LPL representation of the privacy policy sentence from Listing 2 (Ekundayo, 2022).

for software that operates automatically. A software regularly aggregating statistics about some data concerning customers, for example, could be assigned with a specific purpose. Data subjects could then deny consent for this purpose to prohibit the software from performing this statistical analysis on their personal data.

In contrast to our access purpose identification based on role and software, Byun and Li base their access purpose identification on roles and attributes (Byun and Li, 2008).

4.4 P-LPL Privacy Policy

The decision process for granting or denying access to data is based on the content of P-LPL privacy policies. Listing 1 shows the purpose *MailAdvertisements*, which is part of a translation of ebay.com's privacy policy. Listing 2 shows an excerpt of the original textual privacy policy, which was translated to the *MailAdvertisements* purpose.

The purpose shown in Listing 1 has the internal ID *p24* and is externally known by the unique name *MailAdvertisements*. The two Boolean values *true, false* state that this purpose can be opted out of, and that the data subject is not required to accept this purpose for a successful interaction with the data controller. The number *1668495600* is the Unix timestamp of the point in time when the data subject accepted this purpose. Due to space constraints, we removed the verbose textual headers and descriptions (*[...],[...]*) from the purpose. The list *[dt01,dt04,dt34,dt48,dt58,dt74]* references six data elements, which are defined in the P-LPL policy, stating what data may be processed for this purpose. The two empty lists (*[ ], [ ]*) state that no privacy models (e.g., *k*-anonymity) or pseudonymization methods are applied to the data. A data processor is assigned to this purpose, expressed by *dr2*, the ID of the processor receiving the data for this purpose. The legal basis for this purpose is referenced by *[lb4]*. This information is followed by another empty list *[ ]* which states that the purpose is not using automated decision making. Finally, *r* is a reference to the part of the policy describing for how long the data for this purpose may be stored (retention).

```
‘‘ Advertisements  by mail  (according  to  your
communication preferences  in  your eBay account ).’’
```

Listing 2: Excerpt from ebay.com's privacy policy (eBay GmbH, 2021): Purpose Mail Advertisements.

```
accessibleData (D,LPP,P,R) :-                                  1
        LPP = (_,_,_,_,_,_,_,_,_,PP,PH,_,_,_,_,_,_),           2
        getDataAndConsentPH(P,PP,PH,DT),                       3
        DT \= [],                                              4
        mapDataNames(DT,DTN),                                  5
        accessibleDataPurpose (D,DTN,R).                       6
```

Listing 3: Main rule of the P2BAC decision point (Prolog).

4.5 P2BAC Decision Point

We implement a P2BAC decision point in Prolog, working directly on the P-LPL privacy policy. This decision point can be used efficiently for access requests concerning a single data subject's data. For requests covering the data of multiple data subjects we propose to adapt the *allowed intended purpose* concept of Byun and Li (Byun and Li, 2008), which we describe in further detail in Section 4.6.

Listing 3 shows the main Prolog rule, which is used to evaluate an access request. The head of the rule in line 1 contains four variables. The first three variables are used as inputs, where *D* is the list of data elements that the decision point needs to consider.

In Listing 4 we show an exemplary access request encoded in Prolog. The list *[“address”,“name”]* corresponds to the variable D in Listing 3. The variable *LPP*, in Listing 3, needs to be initialized with the root tuple of the privacy policy, from which we then extract the list of purposes *PP* and the purpose hierarchy *PH* in line 2 of Listing 3. The third variable *P* takes the purpose for which the data is requested as input; in our example in Listing 4 we use the purpose *MailAdvertisements*.

Line 3 of Listing 3 extracts all allowed data elements for the given purpose *P* from either the purposes *PP* or the purpose hierarchy *PH* and returns the list of data elements in *DT*. If *P* is one of the purpose categories, e.g., *marketing*, *getDataAndConsentPH* calculates the intersection of the sets of data elements of each sub-purpose, returning a list of data elements that may be processed for any of the sub-purposes of the requested purpose category. Because of the use of the intersection, which is a restrictive way of deciding the access request, we suggest formulating precise access requests, providing the most specific purpose possible.

```
accessibleData ([‘‘ address’’ ,‘‘ name’’ ],LPP,
                                  ‘‘MailAdvertisements’’ ,R).
```

Listing 4: Access request in Prolog for the purpose *MailAdvertisements* with data *address* and *name*.

Line 4 checks whether the returned list of data *DT* is empty. The rest of the rule only applies to non-empty data lists. If the list is empty, the *accessibleData*-rule fails, stating that access is denied. A non-empty data list contains the internal IDs of the allowed data elements. For a comparison with the requested data elements, we extract the external unique identifiers of the data elements in line 5. Finally, the call in line 6 intersects the set of requested data with the set of allowed data and returns the resulting list of accessible data elements in *R*, which is forwarded to the caller of the *accessibleData*-rule.

The resulting list of accessible data is then used for query rewriting, giving only access to those data elements that are allowed to be processed for the requested purpose. We present more details on query rewriting in Section 4.7.

4.6 Access Codes for Aggregating Queries

If there are database tables that need to be accessed by aggregating data or retrieving data for a multitude of data subjects, requests to the P2BAC decision points might be inefficient, switching back and forth between database and P-LPL. To address this issue, we adapt the use of binary codes from Purpose Based Access Control, removing the *prohibited intended purpose*s, which do not exist in P-LPL privacy policies. In the context of P-LPL, all processing not explicitly expressed and consented to in the policy is prohibited. Data subjects can only provide consent for those purposes they are informed about, thus, only allowed purposes can be expressed.

4.6.1 Access Code Generation

The length of the binary access codes depends on the number of purposes inside the privacy policy. A realistic privacy policy (e.g., ebay.com) contains about 40 purposes; thus, the codes are 40 bits long. Each bit represents one of the purposes. Codes are generated for each data element that is accessed in large-scale requests. All bits corresponding to purposes that contain a data element for which the data subject provided consent are encoded with a "1" bit. All purposes not containing the data element or not having the data subject's consent are encoded as a "0" bit.

4.6.2 Example

Table 1 shows an exemplary database table *postal*, containing name and address of two data subjects, with *id* being a database-internal identifier. In addition to the main data (name, address, and id) the table contains the *aip* codes for each data element, representing each data subject's personal privacy policy. In this case the accepted policies are very similar, where the data subject *Gerald Gadget* accepted one more purpose compared to data subject *Margret Marple*. This difference is visible in a single byte of the hexadecimal *aip*, which is highlighted in bold.

The access codes are based on a P-LPL version of ebay.com's privacy policy (Ekundayo, 2022), which contains 40 purposes that are encoded into 40 bits. The *aip_name 838181D75F* allows data processing for all purposes that are required for service provision, including the opt-out purpose *MailAdvertisements*. The second *aip_name 8B8181D75F* allows one additional purpose: *MarketingCommunications*.

When a request for the purpose *MailAdvertisements* is prepared, the textual identifier of the purpose must be transformed into an *access purpose* (*ap*) binary code. Considering the 40 purposes and the fact that *MailAdvertisements* is the 24th purpose in the privacy policy, the *ap* code in hexadecimal notation is: *0000800000*.

As an example, we calculate an access request for the postal database shown in Table 1. Equation (3) visualizes the bitwise "and" operation for a request on the data *address* of the data subject *Margret Marple* for the purpose *MailAdvertisements*. We focus on the byte *81* of the *aip*, since this is the most relevant part of the request. The corresponding byte in the *ap* is *80*. All other bytes of the *ap* are zeros. The bitwise "and" of *aip* (*81*) and *ap* (*80*) results in binary sequence *10000000*. Since the result is non-zero, access for this request is granted.

$$\begin{array}{ll} \text{aip:} & 10000001 \text{ byte 81 (Margret Marple)} \\ \text{ap:} & 10000000 \text{ byte 80 (request } \textit{MailAdvertisements}\text{)} \\ \hline & 10000000 \neq 0 \Rightarrow \text{“Access granted”} \end{array} \tag{3}$$

These binary operations have to be performed per line of the database table. Computers are very efficient in the execution of binary operations, and Byun and Li evaluated access requests to be efficient (Byun and Li, 2008).

4.7 SQL Query Rewriting

P2BAC's access control is based on SQL query rewriting. In this section we take a look at some queries and how they are processed in P2BAC.

4.7.1 Multiple Data Subjects

Listing 5 shows a query trying to read data from multiple data subjects. The query requests all columns

Table 1: Exemplary structure of table “postal”.

name	address	id	aip_name	aip_address
Margret Marple	Mainroad 2, 44121 Ferrara, Italia	12345	8**3**8181D75F	1100**8**1D75F
Gerald Gadget	North 3, Diest 3290, Belgium	12346	8**B**8181D75F	1100**0**1D75F

SELECT ∗ **FROM** postal **FOR** MailAdvertisements

Listing 5: SQL query before rewriting, for the purpose *MailAdvertisements*.

from Table 1 for the purpose *MailAdvertisements*. Listing 6 shows the resulting query after rewriting. The * is used to identify the data elements that are requested: *name, address*. This information is used to initialize the *aipCheck*-calls in the **WHERE** part of the resulting query (*aip_name, aip_address*). The first parameter *0000800000* is the access purpose encoded in hexadecimal form (cf. Section 4.6).

A pseudocode implementation of the *aipCheck* is shown in Listing 7. Line 1 contains the signature of the function. In line 2 the bitwise “and” operation is executed, and if the result equals zero the function returns *false*. Returning *false* removes the current line of the database table from the result set. If the result of the bitwise “and” is not equal to zero the function returns *true*, and the query condition that called the function is fulfilled. Hence, the line becomes a member of the result set.

If the query in Listing 5 was restricted to the data of a single data subject, e.g., by adding a condition *WHERE id=12345*, the request could also be handled by the P2BAC decision point, as described in Section 4.5. The result of using the decision point would be a regular SQL query selecting all data elements in the row fitting id *12345*.

4.7.2 Single Data Subject

Listing 8 shows a second query that would be handled by the P2BAC decision point, because it is limited to a single data subject by expressing a condition on the *id* column. Data subject *Margret Marple* did not provide consent for the processing of the data element *address* for the purpose *MarketingCommunications*. Hence, the decision point would not rewrite the query, but instead deny the access request.

Another example of a request that is handled by the P2BAC decision point is shown in Listing 9. This query requests both *name* and *address* from *Gerald Gadget* for the purpose *MarketingCommunications*.

SELECT ∗ **FROM** postal **WHERE**
aipCheck(0000800000, aip_name)
AND aipCheck(0000800000, aip_address)

Listing 6: SQL query for address data for Mail Advertisement purposes after rewriting.

```
1  aipCheck(ap, aip ): Boolean
2    if (ap & aip) == 0 then
3      return false ;
4    else
5      return true ;
6    end if ;
```

Listing 7: Pseudocode for the function *aipCheck*.

The data subject *Gerald Gadget* provided consent for the purpose *MarketingCommunications*, so the request should be rewritten to allow this access. However, the privacy policy does not allow access to the *address* for the requested purpose. Hence, P2BAC rewrites the request to the SQL query shown in Listing 10, where only the *name* is selected.

4.7.3 Further Queries

So far, we only considered *select* queries. Here, we give short explanations for other types of queries.

Insert/Update. *Insert* and *update* queries are handled similarly to the *select* queries described above. The only difference lies in access decisions with restrictions in the allowed data. When the decision point decides that only a subset of the requested data elements may be changed, access will be denied completely. This prevents incomplete database inserts and updates.

The task of restricting write access to selected persons on the data controller and processor side is handled by the assignment of roles to persons and purposes to roles.

Missing Purpose. If a query does not contain an access purpose, P2BAC provides two ways of handling the query. Either the query is denied, as no decision can be made if no access purpose is given, or the root purpose of the purpose hierarchy is used to make a decision. This is the most restrictive way of handling such a request. The requested data must be present in all purposes of the privacy policy, and all of the purposes must be accepted by the data subject. If one

SELECT address **FROM** postal
WHERE id=12345
FOR MarketingCommunications

Listing 8: SQL query for address data for Marketing Communication purposes.

```
SELECT name, address FROM postal
  WHERE id=12346
  FOR MarketingCommunications
```

Listing 9: SQL query for address and name for Marketing Communication purposes.

```
SELECT name FROM postal
  WHERE id=12346
```

Listing 10: SQL query for address and name for Marketing Communication purposes.

purpose is not accepted or the data element is not allowed for one of the purposes, then access will be denied.

5 CONCLUSIONS AND FUTURE WORK

Conclusion. We presented P2BAC, a privacy policy-based access control system, which is integrated into the PriPoCoG framework. Depending on the size of the request P2BAC operates between the source of the request and the database or in the database itself. P2BAC does not require separate access control policies, but instead, directly interprets the personalized privacy policy of the data subject to decide on whether to grant or deny access. We explained every step of the decision process using an example.

Our contributions are: (i) the implementation of a P-LPL decision point in Prolog, (ii) providing propagation rules for all three types of role hierarchies, (iii) access purpose identification considering the software requesting the data, and (iv) using privacy policies instead of separated access control policies.

Viewed in combination with the PriPoCoG framework, P2BAC contributes to a comprehensive treatment of privacy policies in a formally grounded way, where different actors are supported in a variety of tasks.

Future Work. P2BAC is still partially database dependent for requests concerning multiple data subjects. The application-layer based implementation of PBAC by Pallas et al. has the benefit of operating database independently (Pallas et al., 2020). In the future P2BAC could be adapted to be implemented in a similar fashion, getting rid of the database dependence.

Since P2BAC uses P-LPL privacy policies to perform access decisions, we envision further development around the PriPoCoG framework to be beneficial for the acceptance of P2BAC. The framework is missing the data subject perspective, which is particularly important when having customizable privacy policies used for access control.

The multi privacy policy enforcement system by Fatema et al. is a good approach to combine different policy languages in a single system (Fatema et al., 2010). In the future an integration of P2BAC into their system may be beneficial for the applicability of P2BAC.

REFERENCES

Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. (2002). Hippocratic databases. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 143–154. Elsevier.

Betgé-Brezetz, S., Kamga, G.-B., Dupont, M.-P., and Guesmi, A. (2013). End-to-end privacy policy enforcement in cloud infrastructure. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, pages 25–32. IEEE.

Byun, J.-W. and Li, N. (2008). Purpose based access control for privacy protection in relational database systems. *The VLDB Journal*, 17(4):603–619.

eBay GmbH (2021). ebay.com privacy policy. https://www.ebay.com/help/policies/member-behaviour-policies/user-privacy-notice?id=4260. Accessed 2022-11-21.

Ekundayo, B. (2022). Translation of ebay's privacy policy into P-LPL and performance evaluation of the formal P-LPL policy. Bachelor's thesis, University Duisburg-Essen.

European Parliament and Council of the European Union (2016). Regulation 2016/679 of the European Parliament and of the Council of 27 April 2016 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 (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88.

Fatema, K., Chadwick, D. W., and Lievens, S. (2010). A multi-privacy policy enforcement system. In *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life*, pages 297–310. Springer.

Gerl, A. (2020). *Modelling of a privacy language and efficient policy-based de-identification*. Thesis, Universität Passau.

INCITS, A. (2012). Incits 359-2012 information technology—role based access control. *International Committee for Information Technology Standards (INCITS)*.

ISO (2016). *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*. International Organization for Standardization, Vernier, Geneva, Switzerland, ISO 9075-1:2016 edition.

Jesus, V. (2020). Towards an accountable web of personal information: The web-of-receipts. *IEEE Access*, 8:25383–25394.

Laura-Silva, Y. and Aref, W. G. (2007). Realizing privacy-preserving features in hippocratic databases. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 198–206. IEEE.

Leicht, J., Heisel, M., and Gerl, A. (2022). PriPoCoG: Guiding policy authors to define GDPR-compliant privacy policies. In *International Conference on Trust and Privacy in Digital Business*, pages 1–16. Springer.

Mehta, A., Elnikety, E., Harvey, K., Garg, D., and Druschel, P. (2017). Qapla: Policy compliance for database-backed systems. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1463–1479.

Ni, Q., Bertino, E., Lobo, J., Brodie, C., Karat, C.-M., Karat, J., and Trombeta, A. (2010). Privacy-aware role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 13(3):1–31.

Pallas, F., Ulbricht, M.-R., Tai, S., Peikert, T., Reppenhagen, M., Wenzel, D., Wille, P., and Wolf, K. (2020). Towards application-layer purpose-based access control. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 1288–1296.

Yang, C., Tan, L., Shi, N., Xu, B., Cao, Y., and Yu, K. (2020). Authprivacychain: A blockchain-based access control framework with privacy protection in cloud. *IEEE Access*, 8:70604–70615.