By ALEXANDER PRETSCHNER,
MANUEL HILTY, *and* DAVID BASIN

DISTRIBUTED USAGE CONTROL

Using a server-side architecture to connect specialized enforcement mechanisms with usage control requirements and policies.

Computer systems play an increasingly prominent role in our daily lives. Interacting with these systems often involves disclosing personal data—data that can be traced back to particular individuals, collected in different contexts. For example, healthcare providers, insurance companies, and tax offices collect personal data explicitly. The use of credit or loyalty cards, as well as Internet shopping, leave implicitly created digital footprints. So does the use of mobile phones (traffic data) and the coming generation of motor vehicles (location data and sensed driving behavior). Moreover, public security concerns have led to increased monitoring of public spaces where personal data (images and contexts) is gathered without direct interaction with computerized services. The looming reality of ubiquitous computing will further increase the amount of personal data collected, and enhanced network capabilities give rise to potentially uncontrolled distribution.

These technologies improve, for the most part, the quality of our lives. Still, the question arises how all this potentially sensitive data can be protected. Two of the main technical challenges involve controlling data access and usage. While the fundamentals of access control appear to be well understood, this is not the case for usage control. Promising research has been carried out in the areas of both usage control specification [1, 8] and enforcement mechanisms [6, 9]. Missing, however, is a conceptual framework that encompasses both specification and enforcement. In this article, we close this gap.

To this end, we assume personal and other kinds of sensitive data are stored at trustworthy places called data providers. Third parties, called data consumers, request access to the data. Assuming that some form of access control is in place, our concern is what happens to the data once it has been released to the data consumer: how the data consumer may, must, and must not use it. Clearly, the scope of this problem extends beyond privacy concerns about personal data and is also related to the management of intellectual property rights.

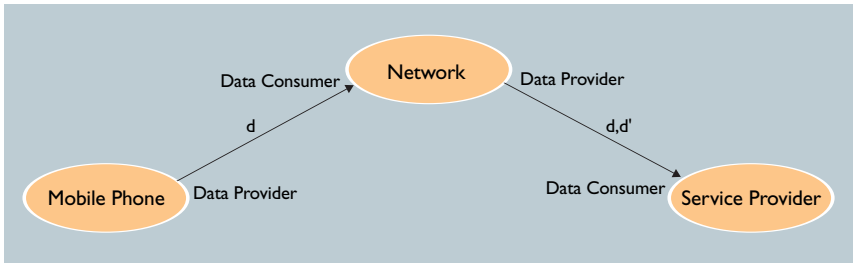


Figure 1. Roles of actors.

In this article, we describe the fundamentals of usage control, in particular, the notions of provisions, obligations, and compensations in the context of controllability and observability. This takes into account possible enforcement mechanisms like those provided by rights management mechanisms. However, many requirements on the consumer's behavior cannot be directly enforced. We therefore present a transformation-based approach to tackling this problem whereby non-enforceable requirements are transformed into requirements whose satisfaction can at least be observed. We describe a two-level policy language that is rich enough to express all these concepts and present a generic server-side architecture for implementing usage control. This architecture is compatible with different client-side enforcement mechanisms, such as dedicated client-side software architectures, trusted platform technologies, and other digital rights management (DRM) mechanisms. We view this server-side architecture as providing the missing link between specialized enforcement mechanisms on one side and usage control requirements and policies on the other.

Abstract System Model

We consider a distributed system consisting of a set of actors. An actor is an information system or information processing device, and each actor has an owner that is responsible for the actor's behavior. Actors can take actions including:

- Operations on data such as storage, distribution, different forms of read access (including playing music or videos), modification of payload and metadata, and processing such as the computation of statistics; and
- Communication, which is the sending and receiving of messages that are not subject to usage control, for example, requests for data or notifications of some kind.

Actors also have encapsulated states, that is, they cannot observe the states and operations of each other. Actors can assume different roles. One actor can send (a copy of) data to another actor. In this case, the former is the data provider and the latter the data consumer. These roles can change dynamically. Each data item has a data owner who possesses the rights to the data.

An example of dynamically changing roles is found in mobile computing and is depicted in Figure 1. Consider a location-based service with location information *d* coming from a GPS receiver in a mobile phone. To provide the service, the network infrastructure requests location data from the mobile phone. In this transaction, the mobile phone is the data provider and the network infrastructure is the data consumer. Then *d* is sent to a service provider, possibly with other data *d'*, for further processing. Now the network infrastructure is the data provider and the service provider is the data consumer.

The subscriber, who is the owner of the mobile phone, might want to restrict what happens to this data once it is given to the network infrastructure. The respective requirements can either be specified globally (via a subscriber agreement) or on a per-transaction basis. If the subscriber requires the service provider to delete the data after processing it, then the network infrastructure must stipulate this requirement when giving the location data to the service provider.

Usage Control Requirements

In order to control how data is used, the owner of a data provider must define a usage control policy that

specifies the requirements that must be satisfied by a data consumer who receives a copy of the provider's data. The requirements expressed in the policy can come from four different kinds of sources: the data provider's (owner's) own interests; the data owner's preferences; governing laws and regulations; or from an agreement with another actor that has previously sent the data.

Provisions and Obligations. We distinguish two basic classes of usage control requirements [2, 3]: provisions and obligations. Provisions are concerned with the past and present and, as such, represent access control requirements only. In contrast, obligations are concerned with requirements on the future that the data consumer must adhere to. The specification and enforcement of provisions is fairly well studied and understood in the access control community, and hence we focus on obligations.

Examples of obligations are "data *d* must not be stored for more than 30 days," "data *d* must not be further distributed," and "data *d* must not be processed for purposes other than *p*." Obligations impose constraints on operations on data, which can relate to:

- Time, for example, data may have to be stored at least, or at most, 30 days;
- Cardinality, limitations such as data may be copied at most three times;
- The occurrence of certain events, for example, data may be used until its owner explicitly states otherwise;
- Actions to be taken by the data consumer, such as notifying the data owner each time the data is used;
- The purpose for which the data may be used, for example, for scientific purposes only;
- Technical or governance restrictions, such as encrypted storage or adherence to governance standards; or
- The necessity of updates, because the freshness and correctness of personal data is often required by data protection regulations.

For both requirements and policies, we define notions of enforceability and violation. A requirement is enforceable if mechanisms can be employed such that all executions of the system satisfy the requirement. A requirement is enforced in a system when these mechanisms are actually employed. A policy is enforceable if all its requirements are enforceable, and enforced if all its requirements are enforced. A requirement is violated with respect to a system execution if the execution does not satisfy it. A policy is violated with respect to a system execution if at least one of its requirements is violated.

Controllability and Observability. Enforceability is tightly bound to the notions of controllability and observability. Controllable obligations are obligations for which the data provider can ensure that the data consumer executes respective operations only under the specified restrictions. Controllability only exists with respect to a given set of mechanisms. Trusted platform technology can be used as a mechanism to control certain obligations as, for example, within DRM, where such technologies are already in use. Alternatively, the data provider can use trusted systems in a more general sense, namely systems for which the data provider is certain they will behave in predefined ways. This is, for example, the case for dedicated software infrastructures in trustworthy environments. In such environments, the main concern is often to prevent unintentional, rather than deliberate, violations of obligations.

In many cases, full controllability is not achievable. Therefore, we also introduce the notion of observability, which is a weaker notion than controllability. In some cases, the data provider can observe whether obligations are adhered to. We call such obligations observable obligations. Recall that actors cannot observe each other's states and local actions. However, by receiving messages that describe parts of what is otherwise unobservable, they can acquire partial knowledge about the states and actions of other actors. Mechanisms for observing the fulfillment of obligations range from non-technical mechanisms like audits to technical mechanisms like the use of trusted systems that inform data providers about actions taken by data consumers (such as trusted logging mechanisms or the use of watermarks to identify the source of illegal copies). If an obligation is not observable, there may be an approximation of it that can be observed. For instance, it is difficult to see if data is actually deleted, but there may be technical means to show that the respective commands have been executed. Obviously, there also is a similar notion of approximation for controllability, but for the sake of simplicity, we do not address it here.

Observability can be exploited for enforcement purposes [2]: the data provider can observe whether an (approximation of an) obligation is violated and take a compensating action when this is the case. The compensating action can rectify the violation, it can be a penalty such as lowering a trust or credibility rating of the data consumer, or it can be some form of legal action. This is similar to enforcing a law that prohibits driving through a red traffic light. It is not possible to prevent car drivers from driving through red lights, but by installing cameras, the police can fine those who do so. We call a requirement of the form "if a violation of obligation *o* is detected, then the compensating action

a must be triggered" a compensation. Compensations are enforceable. Because full controllability is not achievable in general, we suggest a hybrid approach employing two mechanisms: one for controllability, and the other for observability, when controllability cannot be achieved.

Obligations that are neither controllable nor transformable into observable ones can only be trusted to be adhered to by the data consumer. The best the data provider can do here is to get a commitment to the obligation from the data consumer, and perhaps remind the data consumer of its duties later.

Policies

We have three types of enforceable requirements: provisions, controllable obligations, and compensations. To reflect the fact that some obligations are not enforceable, we define two policy levels. The intuition behind these two levels is as follows. A high-level policy is about what ideally should be enforced, and it directly reflects applicable laws, regulations, and agreements. It may thus contain requirements that are not enforceable. A low-level policy is a policy that can actually be enforced; it contains both references to what is stipulated by the high-level policy (including non-controllable obligations) and what will actually be enforced (including compensations). Note that for observable obligations, we enforce the compensation associated with the obligation and not the obligation itself.

High-Level Policies. A high-level policy specifies obligations as well as provisions, which encompass access control requirements and provisional actions. Provisional actions [5] are actions the requester is required to take in the time span between access request and data release, for example, obtain the data owner's consent or sign an agreement.

From High-Level to Low-Level Policies. We now describe the process of transforming a high-level policy into a low-level policy. In doing so, we derive the structure of a low-level policy language. As this structure is more complex than that of high-level policies, we also sketch a simplified metamodel of low-level policies. A prerequisite for this transformation is a description of the available enforcement and observation mechanisms and their capabilities. Such descriptions are provided in dedicated vocabularies. The transformation then consists of four steps.

1. Obligations are partitioned into controllable and non-controllable obligations. This is done with respect to a set of available control mechanisms such as the use of trusted systems as defined earlier. This requires that the available mechanisms are known and well understood. Controllable obligations are annotated with the applicable mechanisms. For obligations that are not fully controllable, a mechanism is specified as well, but the obligation is still considered in the next steps. This results in a combination of control and observation mechanisms.

2. Non-controllable obligations are partitioned into observable and non-observable obligations.

3. As many remaining obligations as possible are transformed into observable obligations by weakening them as much as necessary, as the previous example of data deletion shows. A minimum requirement here is that the violation of a newly created observable obligation implies the violation of the respective non-observable obligation.

4. Each observable obligation is annotated with an applicable observation mechanism capable of observing a violation of the obligation, and associated with compensating actions. In this way, compensations are specified.

The fulfillment of remaining non-observable and non-controllable obligations must be trusted. This trust must be established outside the policies and before access is granted.

Low-Level Policies. Here, we introduce the structure of low-level policies. Figure 2 shows the simplified metamodel of this policy language. A policy consists of a set of rules. In this article, we take a simplistic approach to combining rules: a request is permitted if at least one of the rules applies, and is denied otherwise. This restriction could be liberalized by employing different rule combination algorithms such as in XACML [7].

A rule has an access control part that defines its applicability, which essentially is a predicate over requester attributes, object attributes, and environment attributes. Further, a rule contains provisional actions and contracts. The applicability part and the provisional actions together cover all provisions as defined previously. Contracts reflect obligations in the high-level policy. They contain both the original obligation (what the policymaker wants) and what actually is enforced and how. In this sense, a controllable contract contains a controllable obligation (both its logical representation and a human-readable description) plus information about the enforcement mechanism (or a combination of mechanisms) and how it should be configured. It can also contain a compensation to back up the control mechanism. An observable contract contains an observable obligation and the compensation that will be enforced. To describe the compensation, we must specify the formula to observe (which can be the original obligation or an approximation thereof), an observation mechanism and a compensating action. A trusted contract contains a non-controllable and non-observable obligation.

The semantics of a rule is as follows. If the applicability part evaluates to true, then the entire set of provisional actions and contracts must be satisfied for the data to be delivered. How the provisional actions and contracts are processed is explained in the architecture description next.

For the sake of brevity, our definition of policies omits the description of attributes, actions (including provisional actions and compensating actions), control mechanisms, observation mechanisms, and purposes. Purposes are needed to specify obligations of the form "this data may only be used for purpose *p*." Similar to EPAL [1], such definitions are contained in a vocabulary, which we do not consider in more detail in this article.

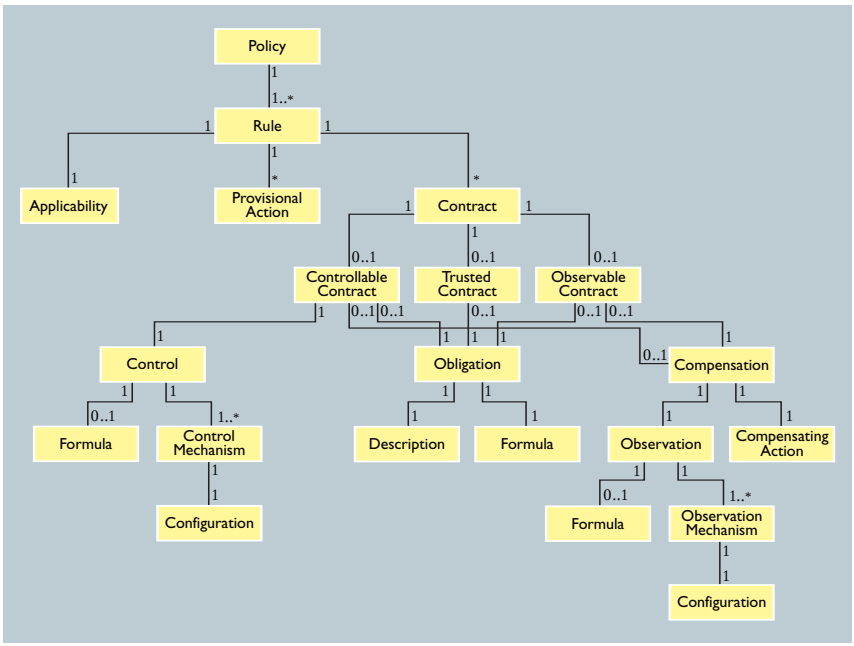


Figure 2. Policy language metamodel.

Architecture

We are now ready to sketch a generic architecture for data providers. It encompasses access control, contract negotiation, observation mechanisms, compensations, and the configuration of data with regard to trusted systems on the client side (for example, the issuing of rights objects for trusted platform technology, or attaching policies that are comprehensible to the trusted system). Since monitoring and enforcement are difficult in the context of open infrastructures such as the Internet, we envision first implementations in more controlled infrastructures including mobile phones or data servers of banks, supermarket chains, military organizations, and national administrations. On the one hand, this is because the information systems in these contexts are easier to control than systems in, say, public P2P networks. On the other hand, for some of the actors in these contexts, it can be assumed they unintentionally, rather than deliberately, violate obligations.

In order to understand the general layout of the architecture, we start by considering a generic process for obtaining data (see Figure 3a).

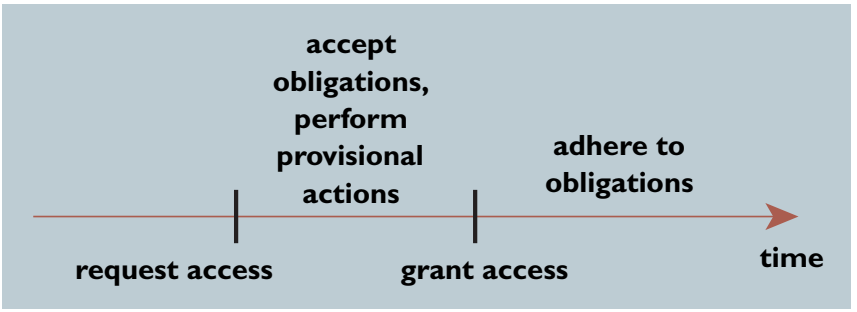
Process. Initially, a potential data consumer *C* requests data *d*. Upon receiving this request, the data provider *P* performs traditional access control. This involves evaluating the applicability part of each rule. The result of this step is a set of rules that associate *C* with *d* in the current state of the environment. If there is more than one applicable rule, *P* must choose one of them. *P* then sends the contracts and the descriptions of provisional actions contained in this rule to *C*. This guarantees access to *d* under the following conditions:

- The attributes relevant for the applicability part have not changed;
- Evidence of having taken the provisional actions is sent by *C*;
- *C* commits to the obligations; and
- *C* accepts possible compensations.

If *C* agrees with the provisional actions as well as the obligations and compensations as set forth in the contracts, *C* performs the provisional actions and gathers evidence for this. *C* sends this evidence to *P*, together with a statement that *C* accepts the other elements of the contract, and requests *d*. Finally, *P* checks whether the provisional actions have been performed, if all access control requirements are still satisfied, and if *C* has agreed with the contracts. If this is the case, *P* starts monitoring possible violations of the events that lead to compensating actions. For all controllable obligations, *P* wraps *d* into a DRM container or issues respective rights objects or policies for trusted systems, and releases a copy of *d*.

Structure. This procedure can be implemented using a generic architecture for data providers, as depicted in Figure 3b. Boxes represent functional units and arrows represent the main data flows. The request handler receives requests and forwards them to the rule filter. The rule filter retrieves all rules for which the respective access control conditions are met (which may include consulting external attribute databases or the

environment), and returns a set of provisional actions and contracts for each rule, of which one set is selected. This data is sent to the consumer who, in turn, provides the necessary information. In case the request can be granted, a compensation management component is triggered to monitor whether obligations are violated and possibly take actions when they are. In case controllable obligations are involved, the data object is modified or augmented so that trusted systems can handle the respective requirements.



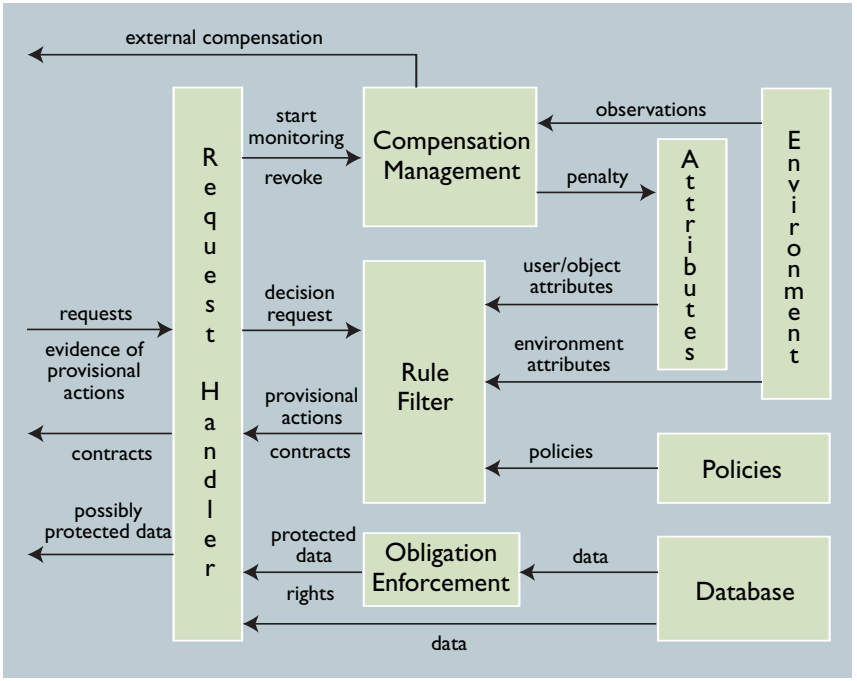


Figure 3a. Request, provisional actions, and access.
Figure 3b. Data provider architecture.

Perspective

With the ever-increasing availability of digital personal data, we firmly believe that usage control will be an enabler of future technologies, particularly in the context of mobile and ubiquitous computing. We have described here first steps toward a general solution, focusing on the fundamentals of usage control. Of course, any technical solution will likely come in conjunction with organizational, legal, and methodological support.

There are a number of challenges remaining. To date, it is unclear how to describe the general capabilities of existing control and observation mechanisms, and there certainly is potential here for new technologies. Because most privacy regulations incorporate the notion of "purpose," this must be allowed for in the policy language, possibly based on dedicated ontologies like those defined by ODRL [4] in the DRM context. Heterogeneous systems pose particular problems; it is unclear, for example, how an RFID tag can control the usage of the signals it emits.

While we believe there are no fundamental differences between usage control in the context of privacy and intellectual property management, this claim clearly needs to be substantiated. Other important problems to address concern usability, the propagation of rights, and controlling the ways that data can be combined and distilled. Although it is likely some time will pass before solutions are found, we believe solutions are achievable.

References

1. Backes, M., Pfitzmann, B., and Schunter, M. A toolkit for managing enterprise privacy policies. In *Proceedings of ESORICS* (2003), 162–180.
2. Bettini, C., Jajodia, S., Wang, X.S. and Wijesekera, D. Provisions and obligations in policy rule management. *J. Network and System Mgmt. 11*, 3 (2003), 351–372.
3. Hilty, M., Basin, D., and Pretschner, A. On obligations. In *Proceedings of ESORICS* (2005), 98–117.
4. Ianelli, R., Ed. *Open Digital Rights Language (ODRL)*; odrl.net.
5. Jajodia, S., Kudo, M. and Subrahmanian, V. Provisional authorizations. In *E-Commerce Security and Privacy*, Kluwer, 2001, 133–159.
6. Liu, Q., Safavi-Naini, R. and Sheppard, N. Digital rights management for content distribution. In *Proceedings of the Australasian Information Security Workshop* (2003), 49–58.
7. OASIS. eXtensible Access Control Markup Language (XACML), 2005. V 2.0.
8. Park, J. and Sandhu, R. The UCON ABC usage control model. *ACM Transactions on Information and Systems Security 7* (2004), 128–174. .
9. Smith, S.W. *Trusted Computing*. Springer, 2005.

Alexander Pretschner (pretscha@inf.ethz.ch) is a senior researcher at ETH Zurich in Switzerland.
Manuel Hilty (hiltym@inf.ethz.ch) is a research assistant ETH Zurich in Switzerland.
David Basin (basin@inf.ethz.ch) is a professor at ETH Zurich in Switzerland and head of the Information Security Group.