Classifying Usage Control and Data Provenance Architectures

Paul Georg Wagner

Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
paul.wagner@kit.edu

Abstract

Given the ubiquity of data acquisition and processing in our everyday life, protecting data sovereignty in distributed systems is a significant topic of research. Usage control and provenance tracking systems are very promising steps towards a technical solution for the problem of data sovereignty. However, due to their complexity and diversity these systems are still not fully understood. In this work we investigate the functionality of usage control and provenance tracking systems. We classify them into three different categories based on their security goals and properties. Furthermore we identify generic use cases for these systems that help to understand what attack vectors system operators have to be mindful of.

1 Introduction

In the age of ubiquitous computing, data are quickly becoming the most important assets of many private enterprises and public IT infrastructures. Therefore securely managing databases and preventing cyber attacks as well as data theft have been crucial IT security tasks for quite some time. However, in recent years

the focus of these IT security goals somewhat changed. While in the past it was sufficient to protect local infrastructures such as computer systems and databases from unauthorized access, many modern business processes require extensive data exchange with remote stakeholders such as clients, business associates and customers. Examples for this can be found in the context of digital supply chains and collaborative predictive maintenance [5]. Current research projects such as the *International Data Space* [7] push for highly interconnected business ecosystems on a big scale and in many different areas. In such scenarios valuable business data are being disclosed into computer systems operated by external stakeholders, who might have conflicting interests. From an IT security perspective the data owner needs a way to control his information even when it is being processed in remote infrastructures. It needs to be ensured that the data recipients cannot inadvertently disclose the received information, or even deliberately misuse it for their own benefit.

Similar challenges also exist when considering the topic of data privacy protection. Unlike with business data, personally identifiable information of a single individual seldom holds great monetary value. Nevertheless the protection of shared personal information is still of great concern. While (supra-)national data privacy laws regulate the acquisition and usage of personally identifiable information on the legislative level, given the noticeable trend towards highly interconnected data processing systems, there is a clear demand for technical solutions as well. At the present moment this is especially evident in the field of medical data processing. In light of the current global Covid-19 health crisis, being able to autonomously collect and distribute health data on a large scale has become profoundly relevant. Nonetheless, given the privacy-sensitive nature of these data, the patients clearly need to remain in control of their information throughout this process. As a result, over the last few years lots of research regarding privacy-compliant medical data sharing has been conducted [1, 3, 6]. In general, data subjects have a legitimate right to monitor and control what personal information is being used in what way, even if the actual data processing is performed on a remote device operated by a third party.

These challenges regarding both data protection and data privacy can be subsumed under the term *data sovereignty*. Data sovereignty describes the approach of enabling data providers to monitor and control the use of their

information at all times, even when they are being used by remote stakeholders. During this process, the data in question can be business-related, or consist of personally identifiable information. To achieve data sovereignty in technical systems, there are some tasks to be considered. First of all, it is necessary to track data flows across systems and domain boundaries. This is called *data provenance tracking* and allows to reliably monitor data usage regardless of where the data is being processed. Besides passively observing data flows, providers also need to be able to actively control and prevent certain types of unwanted data usage. This can be done by applying *usage control* (UC) techniques.

In this work we investigate the general design of usage control and provenance tracking systems and analyze them with regard to four dimensions:

- Security goals
- Enforcement capabilities
- System architecture
- Attack vectors

Based on this analysis we classify usage control and provenance tracking systems into three different categories. We also identify generic use cases for these systems that help to understand what stakeholders are relevant and what attack vectors can occur in different scenarios. The remainder of this paper is structured as follows. Section 2 briefly introduces the design and functionality of usage control and provenance systems on a purely conceptual level. Afterwards in section 3 we identify and categorize several corresponding system architectures that are used as a basis for implementing these concepts. In section 4 we then identify relevant stakeholders and describe four generic use cases for usage control and provenance systems as well as important attack vectors that have to be considered. Finally in section 5 we conclude with a short recap and an outlook on future research.

2 Provenance Tracking and Usage Control

In order to establish a technical solution for data sovereignty, we need versatile provenance tracking and effective usage control frameworks. Both of these topics have been subject to a lot of research in the past. In this section we will briefly introduce the most common way of defining provenance tracking and usage control mechanisms.

2.1 Provenance Tracking Mechanisms

Data provenance allows data providers to track the usage of their digital assets and collect information about derivations that have been created as part of a data processing step. The most common formal model for provenance is the PROV standard [4], formerly known as the Open Provenance Model (OPM). This family of documents describes data formats and serializations for exchanging provenance information across heterogeneous environments. It does not, however, propose concrete mechanisms for implementing provenance tracking in data processing systems. For this, Bier [2] suggests a provenance tracking system mainly consisting of three distinct components (c.f. figure 2.1).

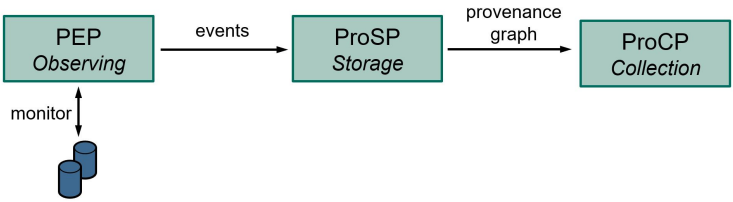

Figure 2.1: Provenance tracking components.

First of all, a *policy enforcement point (PEP)* is responsible for monitoring data accesses and creating events that represent data flows within the system. PEPs are usually implemented close to data processing applications and are capable of examining how data is being used. As data monitoring components, they are at the heart of each provenance tracking system. The generated events

containing data flow information are then relayed to a *provenance storage point (ProSP)*. The ProSP evaluates the events and aggregates all data flow information into a provenance graph. The nodes of this provenance graph correspond to representations of certain data at a specific point in time, while the edges describe linkages between data representations (i.e. data flows). In short, the provenance graph represents a comprehensive information flow history of the entire data processing domain. If sensitive data are shared across domain boundaries, a third level is established by including a *provenance collection point (ProCP)*. This component queries and aggregates multiple provenance graphs, thereby creating a coherent history of data that have been tracked across multiple systems.

2.2 Usage Control Mechanisms

In addition to tracking provenance information, achieving data sovereignty requires a mechanism for data providers to actively and continuously control the access to their information even after it has been disclosed. This can be done by applying usage control (UC) techniques. Usage control was developed over a decade ago as a generalization of attribute-based access control. In contrast to classical access control schemes, UC allows for continuous authorization of data accesses over a period of time. It also features the possibility to declare obligations that need to be fulfilled before, during or after a certain data usage, which is not covered by classical access control. This allows the definition of complex data usage strategies, such as limiting the number of views or the time of access to sensitive information. The most widely adopted formal usage control model is $UCON_{ABC}$, which has been introduced in 2004 by Park and Sandhu [8]. Even today $UCON_{ABC}$ provides the formal basis for many usage control systems. In terms of designing usage control architectures, most modern systems rely on a derivative of the XACML reference architecture [9]. Originally developed for attribute-based access control, the XACML components can be canonically extended to implement usage control policies. Figure 2.2 shows a generic usage control system based on XACML components.

As before, the central component of the system is a policy enforcement point (PEP), which closely interfaces the data processing applications and continuously generates events representing any data usage. However, unlike PEPs

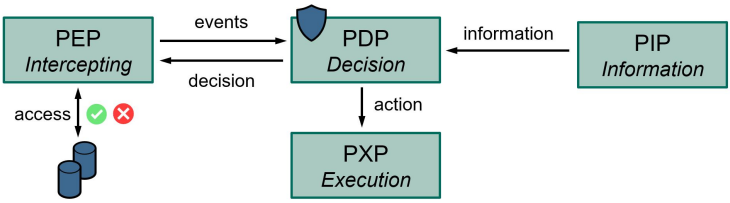

Figure 2.2: Usage control components.

implemented for provenance tracking, usage control PEPs must be capable of actively interfering with the data processing as well. It is not sufficient to just *observe* data usage anymore – usage control PEPs need to be able to actively *intercept* data usage events and potentially *modify* or *block* their execution based on the prevailing usage policies. We call an enforcement point capable of this *intercepting PEP*, in contrast to an *observing PEP*. Naturally every intercepting PEP is also an observing PEP. While observing PEPs are sufficient for provenance tracking, we require the definition of suitable intercepting PEPs in order to enforce usage control on sensitive data.

The other essential usage control component is the *policy decision point (PDP)*. The PDP holds a set of active usage control policies and receives the events from the intercepting PEP. The received events are then evaluated against the set of active policies, which results in a usage control decision. In addition to the classical binary access decision of `allow` versus `deny`, the PDP can also rule that the data usage described by the event should be `modified` prior to its execution. In the end the intercepting PEP receives the decision and enforces it on the data processing application.

Finally there are two more components involved in the usage control enforcement process. The *policy information point (PIP)* can be queried by the PDP for subject and object attributes, as well as generic information such as database entries or environmental properties. The *policy execution point (PXP)* is responsible for executing obligations demanded prior to a data usage, for example the incrementation of an access counter. Obligations are invoked by the PDP and have to be executed successfully before the PDP publishes a positive decision.

In the end it is the collaboration of all components that ensures proper usage control enforcement.

3 Classifying System Architectures

In the previous section we described the mechanisms and basic components of usage control and provenance tracking systems. However, this merely conceptual view on usage control and provenance does not yet describe how to actually apply these mechanisms in real-world use cases. Depending on the specific demands and requirements there are many ways of designing usage control and provenance architectures. In this section we explore and categorize different options in designing actual system architectures and discuss what real-world use cases they cover.

3.1 Usage Control Architectures

Local UC architecture. The simplest form of usage control systems are *local UC architectures*. Figure 3.1 shows an example of such an architecture. A local UC architecture consists of a single computer system that enforces a set of usage rules on local data without considering any external influences. The usage control components (PEP, PDP, PIP and PXP) all run as services on the local computer system that should be protected. Furthermore, both the sensitive data as well as the respective usage control policies are also stored directly on this system. During system operations, the usage rules are then enforced on the local database by the mechanisms described earlier (c.f. figure 2.2). Usually there is a fixed set of usage control policies that have been defined by the system administrator (analogous to mandatory access control). In addition to that, system users can also create protection policies for their own data (analogous to discretionary access control).

Figure 3.1 shows the four usage control components implemented as dedicated software modules instead of a single large monolithic software stack. Usually this design is preferred, because it respects separation of concerns and allows the

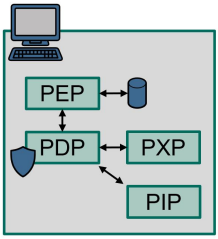

Figure 3.1: Local usage control architecture.

flexible extension of the usage control system. Nevertheless, all UC components run on a single system containing all data that are subjected to usage control.

The benefit of the local UC architecture is its simplicity and self-containment. The architecture does not depend on any other system and manages both the database as well as the policy set sovereignly. On the other side, the local UC architecture cannot enforce usage control anymore as soon as the data are being shared with another system. Hence it is not suitable for implementing any of the scenarios that have been described in the introductory motivation.

Cross-domain UC architecture. In order to support usage control enforcement across different stakeholders, multiple local UC architectures can be merged into a *cross-domain UC architecture*. As the example in figure 3.2 shows, a cross-domain architecture links together multiple remote UC systems that can share sensitive data as well as respective usage control policies. Each participating usage control system represents a single UC domain, i.e. it operates on a set of policies that are evaluated by a single decision point (PDP). Even though the cross-domain architecture now deals with multiple UC domains (unlike local architectures), each UC domain is still implemented as a single computer system.

The main difference of this architecture compared to a set of local UC architectures is that now data flows between usage control domains are being considered as well. Furthermore, the various UC domains are usually operated by different stakeholders. For example, figure 3.2 shows a data flow from the green system on

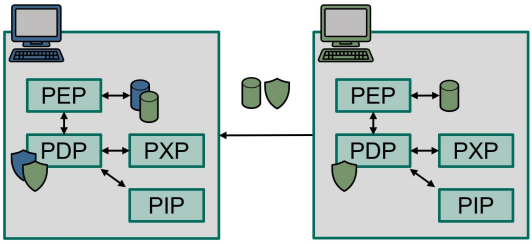

Figure 3.2: Cross-domain usage control architecture.

the right to the blue system on the left. Since the data provider wants to protect his sensitive information in the domain of the remote data receiver, the data flow is preceded by the deployment of a usage control policy regulating the data usage on the remote side. This deployment step is automatically initiated by the enforcement point (PEP) of the donating system (here: right side) whenever it observes an outgoing data flow. The actual policy transmission is then performed by the policy execution point (PXP) as part of a UC obligation. The donating enforcement point only allows the outgoing data flow if the deployment of the protection policy has been executed successfully. Finally the deployed policy is being continuously evaluated by the remote decision point on the data receiver (here: left side). At this point the remote PEPs ensure proper enforcement of the demanded usage restrictions even outside the data owner's usage control domain.

Being able to handle usage control between different stakeholders is the main advantage of cross-domain UC architectures over purely local architectures. Hence cross-domain architectures are suitable to implement the scenarios outlined in the introductions. On the other side, now the usage control systems must be able to remotely deploy and enforce protection policies. Furthermore, the used policy model has to be able to distinguish local from remote data usage. Finally the cross-domain architecture is still limited to a single computer system per usage control domain. This hinders scalability in generic and flexible use cases.

Distributed UC architecture. The most flexible type of usage control systems are designed as *distributed UC architectures*. In contrast to the cross-domain configuration, distributed architectures allow the deployment of a single usage control domain over several collaborating computer systems. Figure 3.3 shows an example of this type of usage control. As you can see, the usage control components previously running on a single computer system are now distributed across multiple devices. Most notably, there are now multiple PEPs running on user devices, while the policy decision point (PDP) runs centrally on a dedicated server. This allows support for use cases where several computer nodes are used for data processing inside a UC domain (e.g. when using thin clients in server environments). Depending on the scenario it is also possible to include multiple information points (PIPs) and execution points (PXP) running on dedicated hardware. This is very useful, since both of these components often attach to existing infrastructure such as databases or directory services. However, in order to avoid policy conflicts there is usually only a single decision point (PDP) per usage control domain.

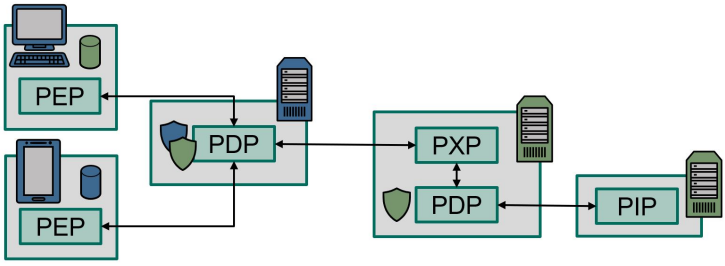

Figure 3.3: Distributed usage control architecture.

While in principal suitable for the same usage control scenarios as cross-domain architectures, the distributed form of usage control offers far greater flexibility than their monolithic counterparts. This is true even in scenarios with only a single UC domain and no data flows between different stakeholders. Being able to independently deploy PIPs and PXPs as dedicated components in existing server infrastructure enables a wider range of usage control applications. Supporting multiple PEPs within a single UC domain allows data processing applications

to be deployed independently of the rest of the usage control infrastructure, for example on mobile devices of employees. Furthermore, this also offers a level of scalability within a UC domain. It is now possible to add more enforcement, execution or information points into an existing UC domain whenever the need arises. On the other hand a distributed UC architecture is more complicated than a UC system running exclusively on a single computer system. It has to be ensured that all usage control components can communicate reliably and securely with each other, even when they are located within a single UC domain. Because of this increase in complexity there are broader attack vectors on distributed usage control architectures and their security properties must be inspected more closely.

3.2 Provenance Tracking Architectures

In addition to distinguishing different types of usage control architectures, provenance tracking systems can be classified in a similar fashion. However, unlike the UC architectures, provenance tracking systems should be classified according to the scope of the acquired provenance information rather than how the system components are deployed.

System-wide provenance tracking. The simplest way of tracking provenance information is to only focus on the data within a single computer system. As the example in figure 3.4 shows, such a system consists of at least one policy enforcement point (PEP) observing all data usages on the system, while a provenance storage point (ProSP) residing on the same system generates and stores provenance information. While there is always only one ProSP per system, it is possible to use multiple PEPs for monitoring data usage (e.g. one per data processing application). The generated provenance graph then attests to the history of data usage on this particular system, for example in order to prove compliance with data privacy laws.

Naturally, this type of architecture only tracks the provenance of data while it is being processed on a single computer system. As soon as the information leaves the system in question, no more fine-grained provenance tracking is

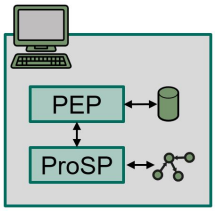

Figure 3.4: System-wide provenance tracking architecture.

possible. This is because the provenance storage points have only a local view on data processing and operate independently on multiple systems. Hence this architecture is only suitable for use cases where all relevant data processing is performed inside a single system.

Domain-wide provenance tracking. Domain-wide provenance tracking is used to track the provenance of data within a single domain. Unlike the system-wide provenance tracking, this means that data flows between multiple systems (i.e. multiple PEPs) within a domain are being tracked by a dedicated ProSP. However, it is still not possible to track data flows across multiple domains and different stakeholders. Figure 3.5 shows an example of this architecture.

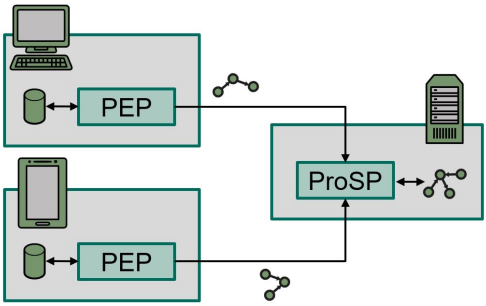

Figure 3.5: Domain-wide provenance tracking architecture.

Cross-domain provenance tracking. In contrast to the previous two architectures, *cross-domain provenance tracking* allows keeping track of data usage even across multiple domains and stakeholders. For this, multiple domain-wide (or system-wide) architectures are linked together and share provenance information. That way provenance tracking is possible even across the domains of different stakeholders. In addition, there is a global provenance collection point (ProCP) aggregating the provenance graphs of multiple local provenance storage points. This allows to generate a comprehensive history of data usage across multiple domains. Figure 3.6 shows an example of cross-domain provenance tracking.

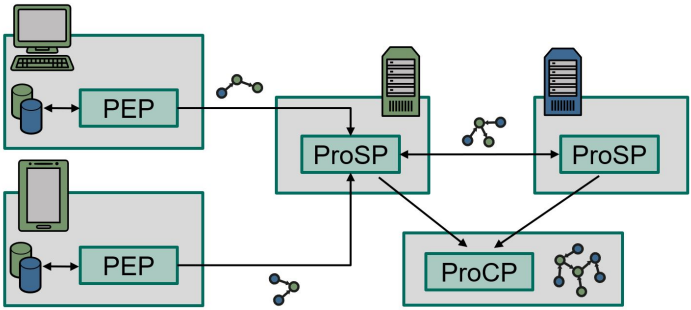

Figure 3.6: Cross-domain provenance tracking architecture.

As Bier pointed out in [2], usage control and provenance tracking can be combined. Similarly, usage control and provenance architectures can also be combined. Clearly, the system-wide provenance tracking architecture is compatible with both local and cross-domain UC architectures. Since there is at least one policy enforcement point in each of those architectures, all that is needed to implement system-wide provenance tracking is a provenance storage point for each system. However, system-wide provenance tracking cannot be used with distributed architectures, since there is no single computer system performing all relevant data processing anymore. On the other hand, domain-wide provenance tracking requires a distributed usage control architecture, while cross-domain provenance tracking is compatible with cross-domain and distributed usage control. Both of them are not compatible with purely local usage control,

because their PEPs cannot observe data flows across system and/or domain boundaries. Table 3.1 shows the possibilities of combining usage control and provenance architectures.

Table 3.1: Combining usage control and provenance architectures.

		Usage control		
		local	cross-domain	distributed
Provenance	system-wide	✓	✓	✗
	domain-wide	✗	✗	✓
	cross-domain	✗	✓	✓

4 Identifying Generic Use Cases

After describing the different possibilities of realizing usage control and provenance architectures, we are left to evaluate their security properties in different use cases. We do this by first identifying the stakeholders that have an interest in usage control and provenance systems of different flavors. As we will see, depending on the concrete goal of the protection systems, the stakeholders' motives can change somewhat and they even have to be considered attackers. Afterwards we describe four different generic use cases for provenance and usage control that demonstrate what attack vectors are to be expected and what security guarantees the various options ultimately yield.

4.1 Stakeholders

There are four main stakeholders to be considered when designing usage control and provenance architectures.

Data owner. This stakeholder holds the rights on a certain set of data that is being disclosed. Usually the data owner has either a monetary or personal interest in monitoring and regulating the usage of his information. For this

purpose the data owner defines usage control policies that specify what may or may not be done with the disclosed information. Furthermore, the data owner may demand tracking the provenance of his information to maintain transparency. If the disclosed data contain personally identifiable information, a data owner is often also called *data subject*.

System user. A system user operates computer systems that run data processing applications in the usage control infrastructure. This stakeholder has legitimate access to information previously disclosed by a data owner and uses it to achieve a certain task. While doing so, the usage control infrastructure enforces the restrictions provided by the original data owner. The system user's access and distribution of protected information may also be logged by the provenance tracking infrastructure. Crucially, the system user does not have privileged access to the systems he operates or the protection components running there. Depending on the scenario, a system user may be motivated to bypass the usage control protection and/or provenance tracking for his personal benefit. Hence this stakeholder must be considered as a possible adversary.

System owner. The system owner is responsible for operating computing systems that run data processing applications as well as the local usage control and provenance infrastructure. Usually this stakeholder has an interest in receiving sensitive information from external data owners and use them outside the boundaries specified by the data owner's usage rules. Unlike the system user he has privileged access to all parts of the managed infrastructure and may use this power to manipulate usage control and provenance components. However, as we will see there are also scenarios where the system owner is the primary data owner. In this case we can trust the system owner to setup all protection systems correctly and not bypass the usage control enforcement, since it is his own interest to enforce protection rules against non-privileged system users.

Supervisory authority. The supervisory authority is a stakeholder only relevant to infrastructures with provenance tracking. This stakeholder is not directly involved in any data sharing, but instead has an interest in verifying

the legitimacy of the data usage with regard to data privacy laws. Usually this stakeholder is a government agency, but it may also be a trusted third-party verifying the privacy-compliance for all participants.

4.2 Provenance Compliance

Provenance compliance is a use case where stakeholders want to track the usage of sensitive information throughout the whole data processing infrastructure. Their goal is to verify the legitimacy of the data usage and its compliance to contractual agreements or data privacy laws. We can distinguish between *internal compliance* and *external compliance*.

Internal compliance means that a system owner intends to conduct provenance tracking only within his own local infrastructure and on his own data. An example for this use case would be a company tracking data flows within their own infrastructure for process optimization purposes or to verify that their employees act in compliance with company-internal standard operating procedures. In this case the system owner (i.e. the company) simultaneously acts as the data owner and the supervisory authority. Notably there are no external data owners present in this scenario. The only other relevant stakeholders are the system users, which in this example would be company employees using the IT infrastructure to perform their tasks as usual, thereby generating provenance information. While the company acts as supervisory authority by analyzing the provenance on their own data, the generated provenance graphs are not intended to serve as evidence for any external supervisory authority. For the use case of internal compliance both system-wide and domain-wide provenance architectures are suitable. Regarding the security properties of internal compliance, the most important adversaries are the system users (i.e. the employees). They might try to hide illegitimate or undesired data usages from their employer by blocking or forging provenance information. Hence the system owner needs to make sure that the provenance tracking components (mainly PEP and ProSP) are properly protected from tampering. In case of a domain-wide provenance architecture it also has to be ensured that no forged provenance information can be sent to the provenance storage point. However, even though a system user may be

motivated to tamper with provenance information in order to hide data accesses from his employer, there is usually no direct monetary incentive for him to do so.

External compliance on the other hand is conducted with the explicit goal of proving compliance to external data owners or supervisory authorities. In this case the system owner receives sensitive information from external data owners in order to process it in his own infrastructure. He is required to track the usage of this data in his domain and report the respective provenance information back to the data owners for transparency. All three discussed provenance tracking architectures are suitable for this task. The most important difference to internal compliance is that the system owner now has a clear interest of forging provenance information in order to deceive the original data owners and supervisory authorities. He can do this by manipulating either the enforcement points or the provenance storage points on his own systems. In order to mitigate this problem, we need to establish trust in the system owner's infrastructure, either by contractual agreements or technical measures (c.f. section 5).

4.3 Usage Control Enforcement

Besides provenance compliance, the other important use case is enforcing usage control on data processing applications. Depending on the goals that should be achieved by the protection system, once again we can distinguish between *internal enforcement* and *external enforcement*.

Internal usage control enforcement, similar to internal compliance, is conducted by a system owner solely in his own infrastructure. For example, a company has a valuable pool of business data and wants to safeguard data usage in their own local infrastructure. The system owner can do this by establishing either a local or distributed UC architecture (in the latter case only with a single domain). In such a system intercepting enforcement points will oversee all data usages in the companies infrastructure and query the central decision point for each data access. The company can then deploy proper usage restrictions in form of policies at this decision point. Once again, since in this case the system owner is simultaneously also the data owner, we can trust him to properly setup and operate the necessary usage control infrastructure. However, the system users (i.e. employees) that are being subjected to usage control enforcement when

working with the protected data, may be motivated to bypass the protection components and access the sensitive data without restriction. Hence we have to view system users as the most important possible adversaries and properly protect all the deployed usage control components from tampering by non-privileged system users.

External usage control enforcement is the most important use case that a UC architecture should address. In this case a data owner wants to impose his own usage restrictions on *remote* data processing applications running within a remote UC domain. This requires either a cross-domain or distributed usage control architecture. The data owner can then deploy his own usage control policies into the remote UC domain, before transmitting sensitive data. The remote usage control components (either within a single computer system or as distributed services) then intercept all data usages in the remote system and enforce the deployed usage restrictions on them. As before, from a security perspective the most important adversary is the system owner of the receiving side. This remote system owner has a clear interest of bypassing the usage control components in his own domain and access the foreign data without restrictions. Furthermore the system owner has full control over the enforcing UC system and can manipulate the components to either ignore the deployed policies, or bypass the interception at the enforcement point. Just as with external compliance, we hence need to establish trust in the system owner's *remote* infrastructure, either by contractual agreements or technical measures.

Table 4.1 shows the four identified use cases and their characteristics.

Table 4.1: Use cases for usage control and provenance.

	Compliance		**UC-Enforcement**	
	internal	**external**	**internal**	**external**
PEP capability	observing	observing	intercepting	intercepting
Architecture	system domain	system domain cross-dom.	local distributed	cross-domain distributed
Attacker	user	owner	user	owner

5 Conclusion

In this work we investigated the functionality and properties of usage control as well as provenance tracking systems. We classified three different variants of both UC and provenance tracking architectures, and described in what way they can be combined. Finally we identified generic use cases of usage control and provenance tracking systems that help to understand how these systems can operate in practice and what attack vectors are to be expected.

As pointed out in the previous section, there has to be a way of enforcing the correct application of these techniques on the remote side. This is especially important for the use cases of external compliance as well as external usage control enforcement. This part of the problem is often overlooked, but it is essential for the security guarantees of the resulting system. While most existing systems make do with non-technical solutions such as operational and contractual agreements, a possible technical solution for this issue relies on *trusted computing* to cryptographically attest to the integrity of usage control and provenance tracking components. However, correctly applying trusted computing to this problem is not trivial and still an area of active research [10]. Since there are many ways of ending up with insecure systems when not properly considering how to establish trust in remote usage control and provenance components, this is an important area for future research.

References

[1] Arno Appenzeller et al. “Enabling data sovereignty for patients through digital consent enforcement”. In: *Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments*. 2020, pp. 1–4.

[2] Christoph Bier. “How usage control and provenance tracking get together-a data protection perspective”. In: *2013 IEEE Security and Privacy Workshops*. IEEE. 2013, pp. 13–17.

[3] Eleonora Ciceri et al. “PAPAYA: A Platform for Privacy Preserving Data Analytics”. In: *Digital Health* (2019), p. 42.

[4] Paul Groth and Luc Moreau. “PROV-overview. An overview of the PROV family of documents”. In: (2013).

[5] Matthias Jarke, Boris Otto, and Sudha Ram. *Data Sovereignty and Data Space Ecosystems*. 2019.

[6] Xiaoguang Liu et al. “A blockchain-based medical data sharing and protection scheme”. In: *IEEE Access* 7 (2019), pp. 118943–118953.

[7] Boris Otto et al. *IDS Reference Architecture Model*. Tech. rep. International Data Spaces Association, 2018.

[8] Jaehong Park and Ravi Sandhu. “The UCON ABC usage control model”. In: *ACM Transactions on Information and System Security (TISSEC)* 7.1 (2004), pp. 128–174.

[9] OASIS Standard. *extensible access control markup language (xacml) version 3.0*. 2013.

[10] Paul Georg Wagner, Pascal Birnstill, and Jürgen Beyerer. “Challenges of Using Trusted Computing for Collaborative Data Processing”. In: *International Workshop on Security and Trust Management*. Springer. 2019, pp. 107–123.