Karlsruher Schriften
zur Anthropomatik

Band 68

Paul Georg Wagner

Trustworthy Distributed Usage Control Enforcement in Heterogeneous Trusted Computing Environments

KIT Scientific Publishing

Paul Georg Wagner

Trustworthy Distributed Usage Control Enforcement in Heterogeneous Trusted Computing Environments

Karlsruher Schriften zur Anthropomatik
Band 68
Herausgeber: Prof. Dr.-Ing. habil. Jürgen Beyerer

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buchs.

Trustworthy Distributed Usage Control Enforcement in Heterogeneous Trusted Computing Environments

by
Paul Georg Wagner

KIT Scientific Publishing

Karlsruher Institut für Technologie
Institut für Anthropomatik und Robotik

Trustworthy Distributed Usage Control Enforcement
in Heterogeneous Trusted Computing Environments

Zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften von der KIT-Fakultät für Informatik des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Paul Georg Wagner, M.Sc.

Tag der mündlichen Prüfung: 10. Juni 2024
Erster Gutachter: Prof. Dr.-Ing. habil. Jürgen Beyerer
Zweiter Gutachter: Prof. Dr.-Ing. Felix Freiling

Impressum

KIT Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2024 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1863-6489
ISBN 978-3-7315-1390-2
DOI 10.5445/KSP/1000175307

Abstract

In the age of ubiquitous data acquisition, protecting local databases against attacks and data theft is a preeminent challenge for many organizations and businesses today. At the same time, being able to quickly establish collaborative data processing chains across multiple enterprises often becomes a prerequisite for economic success. Since this usually includes sharing valuable business data with competing stakeholders, it is of utmost importance to effectively secure any shared information against illegitimate use and theft, especially when they are being processed in remote systems. In addition to regulatory measures such as contractual obligations, non-disclosure agreements, and security audits, this necessitates competent technical solutions as well. Two powerful tools for achieving data security in such scenarios are distributed usage control and provenance tracking. Usage control mechanisms allow to continuously safeguard critical data during their life cycle and proactively restrict their usage to legitimate applications. Provenance tracking, on the other hand, denotes the approach of recording the history of data transmissions and usages for later analysis and verification. A hitherto unsolved challenge in applying these technologies to the use case of collaborative data processing is how to properly enforce them against malicious tampering or removal by data receivers. This task is complicated by the fact that the data processing infrastructure is usually operated by the data receivers themselves, who may have economic interests in bypassing issued usage restrictions or provenance tracking.

To solve the described problem, this dissertation presents and evaluates a system for *trustworthy* distributed usage control and provenance tracking in heterogeneous environments. The designed system is based on several trusted computing technologies responsible for transparently and continuously protecting the integrity and trustworthiness of its distributed components against malicious manipulations and attacks. As core contribution, this thesis describes the design and implementation of a suitable system architecture for distributed usage control and provenance tracking, which encompasses a concept for the automated remote attestation and authentication of the required distributed system components in a transitive fashion. By means of a comprehensive security analysis, we show that our concept can effectively establish the identity and integrity of all participating system components. As a result, malicious manipulations or attacks against any part of the usage control system invariably leads to the revocation of the granted usage rights. Furthermore, we also develop an extension for the widespread policy language ODRL, which allows to acquire provenance information directly via usage control policies and subsequently leverage them for expressing new usage rules.

Further research contributions presented in this thesis pertain to the trusted computing technologies necessary for securely operating the developed system architecture. Since the infrastructure of distributed usage control systems is usually provided by the participating stakeholders themselves, it is often infeasible to globally agree on a single system-wide technological foundation. To alleviate this issue, we develop and evaluate a *heterogeneous* attestation protocol capable of conducting remote attestations between Trusted Platform Modules (TPMs), Intel SGX, and ARM TrustZone. Furthermore, we design and analyze a novel TPM-based attestation protocol to attain the necessary security goals for our system. Our protocol can establish mutually attested and encrypted communication channels without being vulnerable to nonce-data attacks by malicious system operators. In addition, we achieve support for embedded platforms by implementing a suitable trusted boot process on ARM TrustZone devices.

Finally, a trustworthy distributed usage control system must also give data owners a notion to what degree the specified usage rules can in fact be enforced in the current application context. This primarily depends on the current system state, the usage control components required for the enforcement of the policy, as well as the deployed trusted computing technologies. In order to give data owners a comprehensible assessment of the reached level of protection, we develop a quantitative score that represents the ability of the distributed system to successfully enforce usage control policies in its current state. We conclude this thesis by validating the developed score, as well as the entire implemented usage control system, using a practical application scenario from the realm of smart manufacturing.

Kurzfassung

Im Zeitalter der Datenverarbeitung stellt der Schutz der eigenen Datenbasis gegen Angriffe und Datendiebstahl für viele Unternehmen eine zentrale Herausforderung dar. Gleichzeitig sind unternehmensübergreifende Datenverarbeitungsketten im Zuge vernetzter Geschäftsprozesse jedoch immer häufiger die Grundlage des wirtschaftlichen Erfolgs. Da in diesen Prozessen wertvolle Geschäftsdaten bisweilen auch mit konkurrierenden Unternehmen ausgetauscht werden müssen, ist es von elementarer Bedeutung, die missbräuchliche Verwendung geteilter Daten selbst in fremden Systemen effektiv zu verhindern. Hierzu sind neben organisatorischen Maßnahmen wie vertraglichen Vereinbarungen, Non-Disclosure-Agreements und Security-Audits auch technische Lösungen nötig. Zwei wichtige Werkzeuge zur Erhöhung der Datensicherheit in diesen Anwendungsfällen sind verteilte Nutzungskontrolle und Provenance-Tracking. Nutzungskontrolle ermöglicht es, Daten auch während ihrer Verarbeitung kontinuierlich zu überwachen und proaktiv auf eine legitime Verwendung einzuschränken. In Ergänzung dazu bezeichnet Provenance-Tracking den Ansatz, die Historie der erfolgten Datenverarbeitung aufzuzeichnen und verifizierbar zu machen. Eine bislang ungelöste Herausforderung bei der Anwendung dieser Technologien ist jedoch die technische Durchsetzung der Kontrollmaßnahmen gegenüber bösartigen Datenempfängern. Erschwert wird dies vor allem dadurch, dass die beteiligten Systeme in der Regel von den Datenempfängern selbst betrieben werden und diese motiviert sind, die Nutzungsregeln und das Provenance-Tracking durch Manipulation der Schutzkomponenten zu umgehen.

Zur Lösung dieses Problems wird in der vorliegenden Dissertation ein Gesamtsystem für *vertrauenswürdige* verteilte Nutzungskontrolle und Provenance-Tracking in heterogenen Umgebungen konzipiert, realisiert und evaluiert. Wesentlicher Bestandteil des entwickelten Systems sind mehrere Trusted-Computing-Technologien, welche die Integrität und Vertrauenswürdigkeit der verteilten Komponenten transparent und kontinuierlich gegen Manipulationen und Angriffe schützen. Hierbei besteht der Kernbeitrag dieser Arbeit aus dem Entwurf und der Implementierung einer geeigneten Systemarchitektur für verteilte Nutzungskontrolle und Provenance-Tracking, welche mittels eines transitiven Authentifizierungs- und Attestierungskonzepts die Identität und Integrität aller beteiligten Systemkomponenten sicherstellt. Anhand einer umfangreichen Sicherheitsanalyse zeigen wir, dass Manipulationen und Angriffe an sämtlichen Stellen des konzipierten Systems zum Widerruf der erteilten Nutzungsberechtigungen führen. Zudem entwickeln wir eine Erweiterung der verbreiteten Policysprache ODRL, durch die Provenance-Informationen direkt mittels Nutzungskontrollpolicies erhoben und im Rahmen der Policyauswertung nutzbar gemacht werden können.

Weitere in dieser Arbeit erzielte Forschungsbeiträge betreffen die für den sicheren Betrieb der entwickelten Systemarchitektur benötigten Trusted-Computing-Technologien. Da die Infrastruktur verteilter Nutzungskontrollsysteme in der Regel dezentral durch die beteiligten Unternehmen selbst gestellt wird, ist es meist nicht möglich, sich auf eine einzige systemweit zu verwendende Technologie festzulegen. Daher wird in dieser Arbeit ein *heterogenes* Attestierungsprotokoll entwickelt und evaluiert, das interoperable Attestierungen zwischen Trusted Platform Modules (TPMs), Intel SGX und ARM TrustZone ermöglicht. Um ein ausreichendes Schutzniveau für den Anwendungsfall der verteilten Nutzungskontrolle zu erreichen, entwerfen und analysieren wir außerdem ein neuartiges TPM-basiertes Attestierungsprotokoll. Dieses kann beidseitig attestierte und verschlüsselte Kommunikationskanäle aufbauen, ohne dabei anfällig für Nonce-Data-Angriffe durch bösartige Systembetreiber zu sein. Zur Unterstützung eingebetteter Plattformen realisieren wir darüber hinaus einen vertrauenswürdigen Boot-Prozess für Geräte mit ARM-TrustZone-Technologie.

Schließlich muss ein vertrauenswürdiges Nutzungskontrollsystem Datenbesitzern auch Rückmeldung darüber geben, inwieweit die spezifizierten Nutzungsregeln im aktuellen Anwendungskontext tatsächlich durchgesetzt werden können. Dies hängt maßgeblich vom momentanen Systemzustand, den zur Durchsetzung der Policy benötigten Nutzungskontrollkomponenten, sowie den dort eingesetzten Trusted-Computing-Technologien ab. Um eine nachvollziehbare Bewertung des erreichten Schutzniveaus abgeben zu können, entwickeln wir im abschließenden Teil dieser Arbeit einen quantitativen Score, welcher die Fähigkeit des verteilten Systems zur erfolgreichen Durchsetzung der Nutzungskontrollpolicies im aktuellen Zustand repräsentiert. Zur Validierung der entwickelten Methode, sowie des implementierten Gesamtsystems, wird schließlich ein praktisches Anwendungsbeispiel aus dem Bereich des Smart Manufacturing umgesetzt und evaluiert.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor Prof. Dr.-Ing. habil. Jürgen Beyerer for guiding the development of this dissertation over the past five years. Without the constructive discussions and helpful comments, this work would not have taken form. I am also grateful to Prof. Dr.-Ing. Felix Freiling for taking an interest in my dissertation and agreeing to assume the role of second reviewer.

Furthermore, I want to thank all my colleagues – both at Fraunhofer IOSB and the Karlsruhe Institute of Technology – for creating the appreciative and supportive work environment that I am fortunate enough to experience here every day. On this note, I am especially grateful to Dr.-Ing. Pascal Birnstill for mentoring me in my academic endeavors since their beginning. A special thanks also goes to Dr.-Ing. Arno Appenzeller for sharing not just an office, but the journey towards our doctorates as well.

Last, but certainly not least, I want to thank my family for their continuous support and encouragement.

Karlsruhe, July 2024
Paul Georg Wagner

Contents

Appendix

1 Introduction

In the age of ubiquitous computing, data undoubtedly play an essential role in the economic success of many business ventures.[1] While large tech companies such as Google, Microsoft, and Amazon have innovated data-driven business models for over two decades and are now among the world's most valuable public enterprises,[2] in recent years the opportunities of big data have impacted many established business segments as well. Motivated by the advancement of digitization, computerization, and the expansion of digital infrastructures, more and more enterprises rely on data-driven and globally interconnected value chains. Especially the possibility of using data analytics to extract previously hidden knowledge from business data further incentivizes the acquisition and management of large datasets.[3]

One domain increasingly reliant on automated data processing as a key factor for economic success is *industrial production* and *smart manufacturing* [Win21]. Today, many production facilities are equipped with a multitude of sensors capable of collecting data from all stages of a production process. While still primarily used to monitor and control the operation of production equipment as part of an industrial control network [Gal12], this information can also be exploited to improve the efficiency of the production processes and serve as basis for new and innovative business models. For example, such data can help to design assembly lines in a more flexible, modular, and self-organizing way [Büt17, Bur17], improve logistical processes [Ten14, Jun17,

[1] https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data *(accessed on 12/08/2023).*

[2] https://www.statista.com/statistics/263264/top-companies-in-the-world-by-market-capitalization/ *(accessed on 12/08/2023).*

[3] https://www2.deloitte.com/de/de/pages/trends/data-analytics.html *(accessed on 12/08/2023).*

Gün17], and allow just-in-time delivery of production materials [Bub14, Tre17]. Furthermore, smart factories of the future will utilize acquired production data to facilitate collaborative interaction between production machinery and human operators [Tho17], as well as provide workers with personalized assistance functions during challenging manufacturing tasks [Jos17, Len20]. As a result of these developments, it is likely that both the amount as well as the variety of data generated in smart manufacturing applications will only grow in the future.

1.1 Virtual Data Spaces

For many new and innovative data-driven services it is not sufficient to merely generate, process, and analyze data in the local application infrastructure. Instead, to fully exploit the potential of big data, a collaborative approach combining data from multiple different sources is necessary [Oli19]. In the context of smart manufacturing, for example, collaborative predictive maintenance requires the aggregation of process data from several companies operating the same machinery [Jar19]. To facilitate data sharing across different actors in such scenarios, the concept of *data spaces* gained traction in recent years [Ott22b]. In general, data spaces encompass organizational and technical means to bring data providers, data consumers, and analytics companies together in a shared virtual data ecosystem. To achieve this, a data space interconnects the databases of its participants via standardized interfaces and provides mechanisms to automatically negotiate data exchanges [Ott18]. Usually, data space architectures rely on established cloud computing concepts, while also adhering to decentralized data storage and peer-to-peer data exchanges as fundamental design principles [Pet22]. Furthermore, data spaces often provide its participants integrated data analytics applications, as well as a framework to ensure legal compliance of data exchanges [Nag22]. Figure 1.1 illustrates the concept of a decentralized data space infrastructure.

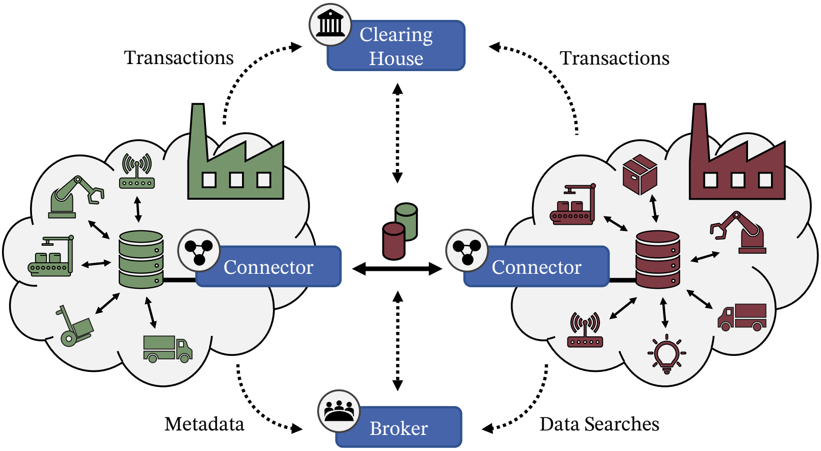


Figure 1.1: Concept of a decentralized data space. Brokers and clearing houses are responsible for negotiating and invoicing data exchanges. Own illustration after [Opr22].

Currently, the *International Data Space (IDS)* is arguably the most notable implementation of the data space concept [Pet22]. Under development since 2015, the IDS specifies a comprehensive information model together with a standardized reference architecture for connector systems, which serve as entry points into the virtual data space [Kir22, Ott19]. Today, there are several actively maintained, open source, and IDS-compliant connector implementations available [Sch18a, Kir22, Kan22]. Building on the experiences gained by the IDS, in 2020 the French and German Ministries for Economic Affairs launched the *GAIA-X* project, aimed at developing a secure and privacy-compliant European data space based on federated cloud computing infrastructures [Tar22, Ott22a].

A major challenge when implementing applications for collaborative data processing is to achieve a level of security and privacy that is acceptable to all data providers. Even though the mutual sharing of information is ultimately beneficial for all participants as long as everyone acts in good faith, in general companies are reluctant to disclose their own valuable data. This is because as soon as data have been disclosed to a third party, the data owner is no longer fully in control of it. For example, the data owner cannot be certain if the data receivers are taking sufficient measures to secure their local data processing infrastructure against cyber-attacks and data theft. Furthermore, there is always the danger of data receivers deliberately misusing shared data for purposes other than previously agreed, since they usually pursue different economic goals than the original data owner [Krä22]. Especially in the realm of industrial production and manufacturing, companies regularly abstain from data sharing approaches due to the risk of data theft [Usl22]. After all, production data often allow the inference of details about the manufacturing process, such as the current equipment utilization. Since this can be highly valuable information, especially for competitors, it must not fall into the wrong hands. Additionally, data captured in smart manufacturing applications may also contain personally identifiable information about human workers, for example video images or body poses. This presents an additional privacy risk that must be adequately mitigated [Man19, Wag21b].

1.2 Towards Data Sovereignty

To address these issues concerning security and privacy, data spaces try to achieve the goal of *data sovereignty* for their users [Jar19]. While there is currently no universally accepted definition of the term, data sovereignty is often regarded as empowering individual data owners, be it natural or juridical persons, to execute their informational self-determination at all times [Jar19, Opr22, Lau22]. Recently, Krämer et al. identified *control* and *transparency* as two fundamental prerequisites for achieving data sovereignty [Krä22]. The authors understand *control* as the ability of a data owner to grant or deny data receivers access to their information, while they see *transparency* as the ability of a data owner to ascertain information about the usage of their data and the resulting profits. In general, there are two complementary approaches for implementing control and transparency in virtual data ecosystems: regulatory measures and technical measures. Currently, companies mainly rely on regulatory measures such as contractual obligations, non-disclosure agreements, and security audits to ensure their data sovereignty in virtual ecosystems [Opr22]. However, due to the high degree of automation in data processing, regulatory measures alone are not flexible enough anymore for many modern use cases. Hence, virtual data spaces incorporate more and more technical measures to improve the level of data sovereignty.

One powerful technical measure to enhance data security in virtual data spaces is *usage control (UC)*. Usage control provides mechanisms to continuously regulate and restrict the processing of sensitive data during their entire life cycle [Par02]. In contrast to classical access control, UC can safeguard critical information not only at the time of the initial data request, but also before and afterwards. This allows data owners to define fine-grained rules about both admissible and illegitimate data usage scenarios. These rules are usually specified in form of *usage control policies*, which are distributed to all applications and then continuously enforced at every point of the data processing chain. In recent years, a lot of research has been conducted in the field of usage control, mainly concerning usage control models, architectures, and policy languages [Laz10, Aka22]. A major step forward was the development of *distributed usage control (DUC)* by Pretschner et al. [Pre06, Kel13]. Distributed usage control allows to conduct usage control operations across multiple networked computer systems, instead of just as a locally confined security feature. Together with the fact that providing usage control technology encourages data sharing in collaborative environments [Opr21], this makes distributed usage control a prime candidate for a technical measure enhancing the level of data sovereignty in virtual data spaces [Opr22, Dui22]. Indeed, usage control mechanisms have already been included in some IDS connector implementations as a central security feature [Ste21].

In addition to actively controlling information by enforcing usage rules, *provenance tracking* is a technical measure that can provide transparency of data usages in distributed systems. The term “provenance” originally describes the tracing of the origin and ownership of historical artifacts and artworks [Bun10]. In the field of data science, provenance tracking has since been established as a method of recording the history of data generation, distribution, and processing [Her17]. Provenance information can help data users to understand where a certain piece of data originated, how and by whom it was created, and in what ways it has been processed and modified since [Zaf17].

This makes provenance tracking useful for collaborative data spaces as well, since it can reveal the distributed usages of shared data to the original data owners and/or supervisory authorities [Ste21]. Especially when sharing personally identifiable information, as it is the case in many smart manufacturing applications, being able to reliably track the provenance of data is necessary to comply with data privacy laws [Ujc18, Bie21].

Implementing usage control and provenance tracking as technical measures for data sovereignty requires the operation and configuration of suitable protection components in the data space infrastructure. These protection components are responsible for distributing and enforcing usage control policies, tracking data flows, and logging provenance information. As such, their correctness and integrity are crucial prerequisites for the secure operation of the virtual data space. This directly leads to the question of how to secure the deployed usage control and provenance tracking components against malicious interference and manipulation. Since data spaces are a prime target for cyber-attacks due to their high volume and quality of data, we must assume that attackers constantly try to compromise the distributed data processing components. Even worse, in many cases *legitimate* data receivers themselves might have an interest in bypassing usage restrictions or preventing the collection of provenance information on foreign data, for example when they are collaborating with business competitors. Since in a decentralized data space architecture all participants operate their own respective connectors (see fig. 1.1), malicious data receivers could easily manipulate or switch off usage control and provenance tracking components in order to siphon off valuable data unchecked. Hence, to establish a trustworthy virtual data space, we need technical measures that can actively *enforce* usage control and provenance tracking even against potentially malicious data receivers.

1.3 Research Gaps and Objective

One type of technology that can provide the required security guarantees is *trusted computing*. The idea behind trusted computing is to enhance the confidentiality and integrity of critical software systems using hardware-based trust anchors [Yan20]. These trust anchors consist of tamper-resistant hardware modules attached to the platform, which provide various security services such as generating and managing cryptographic keys. Since these functionalities are implemented exclusively in hardware, software-based attackers (e.g., malware) cannot manipulate or steal protected information. A very useful feature of many trusted computing technologies is *remote attestation*. Remote attestation allows external parties to verify the integrity of a trusted platform by retrieving and validating a list of fingerprints describing the software stack currently executed on the platform. The list of fingerprints is authenticated directly by the hardware-based trust anchor, which even platform owners with physical access to the attested machine cannot easily manipulate. Because of this property, remote attestation is an effective tool to prevent malicious modifications of critical software, and hence has been suggested as a mechanism to establish trust in virtual data spaces [Hub22].

Previous research into using trusted computing hardware to secure usage control components is focused mainly on *Trusted Platform Modules (TPMs)* [Zha08, Nei11b]. TPMs are specialized hardware security modules similar to smart cards, which are already pre-installed in many modern computer systems and can be used to perform attestations of remote software stacks. Primarily due to their ubiquity and ease of use, data spaces increasingly rely on TPMs as a means to secure the integrity of data protection components [Sch18a, Ott19]. However, there are still some unsolved problems when using TPMs for this purpose, especially when considering malicious administrators as potential attackers [Wag19b, Wag20]. We explore these problems further in chapter 4 of this thesis and develop an improved solution for TPM-based remote attestations that can protect usage control components even against malicious administrators. In addition, more modern trusted computing technologies such as *Trusted Execution Environments (TEEs)* should also be considered for this task. TEEs allow to run entire software stacks inside carefully protected execution environments, and as such promise an increased level of security compared to TPMs. Important TEEs available today are Intel's Software Guard Extensions (SGX), AMD Secure Encrypted Virtualization (SEV), and ARM TrustZone. Even though Intel SGX has recently been proposed to secure the enforcement of data protection policies in cloud infrastructures

[Djo20, Mey21], to our knowledge there is currently no TEE-based solution to protect *distributed* usage control and provenance tracking infrastructures. We present a more detailed picture of the current state of the art and analyze the scope and limitations of existing research in section 3.1 of this thesis.

A related challenge that has not been sufficiently researched yet is the question of deploying trusted computing in *heterogeneous environments*. Due to their heavy reliance on hardware-based trust anchors, trusted computing solutions are often constrained to one specific hardware platform (e.g., Intel, AMD, or ARM). As a result, usually only one single trusted computing technology is being considered as technical measure to secure the integrity and confidentiality of data protection infrastructures. However, these technologies all have different benefits and drawbacks, and can hence provide security in different areas of application [Sch22]. For example, implementing TPM-based remote attestation is usually rather simple as it does not require any modifications of existing application software, but on the other hand it cannot provide equally high security guarantees as modern TEEs. Being able to design distributed systems that rely on multiple trusted computing technologies to protect their components increases flexibility and allows for the combination of technological benefits. Furthermore, especially in data space scenarios with multiple participants, it might not even be feasible to unanimously agree on one single technology among all stakeholders. Finally, there are often technical reasons for being committed to a certain trusted computing technology as well. Especially in the field of production and smart manufacturing, devices are quite commonly designed as ARM-based embedded systems, while the backbone infrastructure still consists mostly of standard x86 servers. Deploying distributed usage control in such heterogeneous environments inevitably requires solutions to link together platforms using different trusted computing technologies, e.g., ARM TrustZone and Intel SGX.

Another open question that has not been adequately researched so far concerns the *trustworthiness* of distributed usage control and provenance tracking systems. While it is clearly necessary to protect usage control and provenance tracking components against malicious tampering using technical measures, focusing on this aspect alone does not suffice in terms of trustworthiness.

We also need to answer the question of how policy issuers can *know* that the integrity of the remote usage control system has been verified and that it is indeed capable of enforcing all deployed policies and/or tracking the provenance of shared data. This primarily depends on the current protection state of the distributed system as well as the deployed trusted computing technologies, since they may be unsuitable to protect the specific security interests of the data owner. Besides these objective factors, the current level of trustworthiness is also influenced by subjective aspects, such as the degree of trust that the policy issuer places in certain trusted computing technologies or in the participating stakeholders themselves. Since the state of a distributed usage control system is dynamic and changes regularly, it is not feasible to develop a usage control architecture that is secure and trustworthy for just one particular use case. Instead, we must develop a method that can dynamically assess the trustworthiness of a distributed usage control and provenance tracking system in its current state from the point of view of (potential) data providers, considering their specific usage control policies. The data providers should then be given qualified feedback about the current trustworthiness assessment to allow informed decisions regarding the sharing of valuable data. To our knowledge, these questions have not been answered so far.

Finally, the approach of combining distributed usage control with provenance tracking also requires more research. While there is previous work regarding the *architectural* combination of these technologies [Bie13, Bie21], the question of how provenance tracking can be integrated into an actual usage control *enforcement process* still remains open. Achieving this has some clear advantages. First, it would allow provenance tracking to be mandated as part of normal usage control rules. A data owner could specify that the usage of a particular data asset is admissible only under the condition that the data provenance is reliably tracked. The distributed usage control system is then responsible for enforcing this obligation. Being able to manage provenance tracking via policies would give much greater flexibility to the data owners compared to tracking provenance either selectively on application level, or globally for all processed data. Furthermore, a close integration of usage control and provenance tracking would also enable policy issuers to reference the processing history of data assets for the specification of new usage rules. For example, a data owner may want to enforce the policy that the distribution of certain data assets is allowed only if they have been previously anonymized. Usage restrictions of this nature could be expressed much more easily with direct access to the previously tracked provenance data.

Considering these research gaps, the main objective of this thesis is to develop and evaluate a framework for *trustworthy* and *distributed* usage control and provenance tracking, which can be securely operated in *heterogeneous* trusted computing environments. We clarify our objective further by defining the following five major goals for the framework.

Goal 1: The framework should give data providers the possibility to monitor and control their data even when they are shared with remote stakeholders. To achieve this, the framework should offer distributed usage control mechanisms with integrated provenance tracking capabilities. To maximize the framework's utility, it should support flexible and generic application scenarios, instead of being designed for a singular purpose or use case.

Goal 2: The framework must include suitable protection mechanisms to enforce usage control and provenance tracking on a technical level. Malicious data receivers must not be able to manipulate the enforcement of shared usage rules or disturb the tracking of data provenance. We use trusted computing technologies as a building block to achieve this.

Goal 3: The framework should support heterogeneous infrastructures. Hence, it must not rely on just one platform and trusted computing technology, but instead provide technological interoperability. To realize use cases from the field of smart manufacturing, we require support for TPMs, Intel SGX, and ARM TrustZone.

Goal 4: The framework should include a method to offer data providers qualified feedback concerning the trustworthiness of the usage control system in its current state.

Goal 5: The system performance should be sufficient to realize practical use cases from the realm of smart manufacturing.

1.4 Research Questions and Contributions

To achieve our objective of developing a trustworthy distributed usage control and provenance tracking framework that fulfills goals 1 to 5, we have to address research questions in three different areas. First, we must define a distributed system architecture that (i) appropriately combines usage control and provenance tracking, and (ii) is capable of enforcing these mechanisms even against malicious system participants. Second, we need to identify trusted computing technologies and develop corresponding remote attestation protocols that can achieve the security guarantees necessary to realize the desired technical enforcement. Finally, we must develop a method to give data providers quantitative feedback about the trustworthiness of the distributed usage control system based on the adequacy of the deployed technologies.

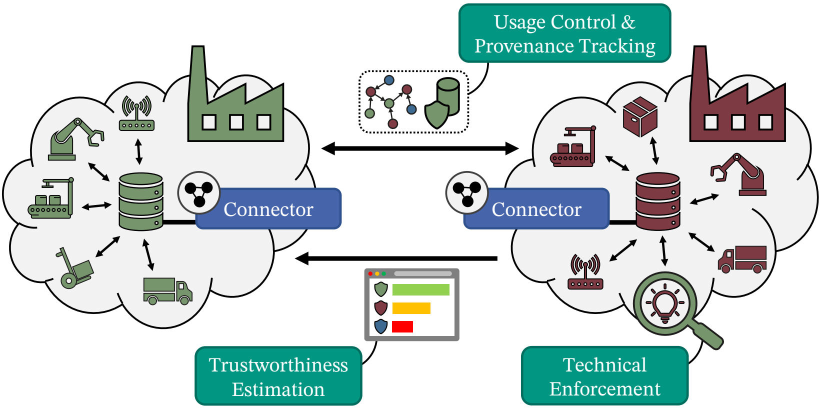

Usage Control & Provenance Tracking
Connector
Connector
Trustworthiness Estimation
Technical Enforcement

Figure 1.2: Research areas addressed in this thesis.

Figure 1.2 illustrates these three areas of research in the application domain of smart manufacturing, which we use to evaluate the developed framework. In the remainder of this section, we refine the addressed research questions and present the concrete research contributions accomplished in this thesis.

1.4.1 Usage Control and Provenance Tracking

Developing a combined usage control and provenance tracking system that is both distributed *and* trustworthy entails two major research questions. The first one deals with the definition of a suitable distributed system architecture that can provide the necessary security guarantees against potentially malicious usage control participants.

Research Question 1: *How can we monitor and control the usage of shared data across multiple remote domains? How can shared usage control policies be reliably enforced even in the presence of malicious data receivers?*

We address this research question by designing a distributed usage control architecture that leverages remote attestations to protect the integrity of the usage control enforcement process against any outside influence. All further requirements for the concrete system design are derived from the previously determined goals 1 to 5.

Contribution RC1: *We develop a conceptual system architecture for trustworthy distributed usage control enforcement. Our design proposal includes a transitive remote attestation concept that protects the integrity of all system components, as well as a hierarchical authentication scheme to prevent the impersonation of usage control components. We show the usefulness of our system design by conducting a comprehensive security analysis based on the assumption of idealized remote attestation protocols.*

The second research question to be addressed deals with the proper combination of provenance tracking and distributed usage control.

Research Question 2: *How can we effectively obligate the collection of provenance information with usage control policies? How can we utilize previously collected provenance information in the specified usage rules?*

We answer this question by conceptually integrating provenance tracking into the distributed usage control enforcement process. Furthermore, we implement our concept by extending the widely used Open Digital Rights Language (ODRL) policy model [Ian18b] with provenance tracking capabilities.

Contribution RC2: *We develop an extension of the ODRL policy language that allows policy issuers to manage provenance tracking in remote domains and utilize the collected information in the usage control enforcement process. We also implement and evaluate a corresponding Policy Decision Point (PDP) capable of processing such policies.*

1.4.2 Technical Enforcement

In the second part of this thesis, we solve the challenge of realizing a secure technical enforcement of the defined usage control and provenance tracking methods based on trusted computing mechanisms. To achieve this, two more research questions must be addressed.

Research Question 3: *How can we leverage trusted computing technologies to enforce the correct behavior of usage control and provenance tracking components? Which concrete remote attestation protocols can we use to securely instantiate our distributed system design? Is the performance of those remote attestation protocols sufficient for our purposes?*

We answer this question by developing and evaluating protection concepts for distributed usage control and provenance tracking infrastructures based on TPMs, Intel SGX, and ARM TrustZone. Our analysis reveals two shortcomings when using TPMs and ARM TrustZone as underlying trusted computing technologies for our system architecture. First, we show that the existing TPM-based remote attestation protocols currently used in virtual data spaces are vulnerable against nonce-data attacks by malicious administrators, who are the main adversaries in distributed usage control systems. Furthermore, we also require a mechanism that allows to conduct integrity measurements of dynamically loaded data processing applications in *both* the normal and the secure world of ARM TrustZone devices. We contribute solutions to these issues as part of our trustworthy usage control framework.

Contribution RC3: *We develop and evaluate a novel TPM-based remote attestation protocol that uses a hardware-protected key exchange to establish mutually attested and encrypted communication channels without being vulnerable to nonce-data attacks by malicious system administrators.*

Contribution RC4: *We implement and evaluate a trusted boot process for ARM TrustZone platforms that uses a firmware-level TPM to conduct integrity measurements of dynamically loaded applications in both worlds of the device.*

The second relevant research question in this area concerns the protection of distributed usage control and provenance tracking infrastructures in *heterogeneous* trusted computing environments.

Research Question 4: *How can we conduct remote attestations in heterogeneous execution environments, i.e., between different trusted computing technologies? Are there significant performance overheads when using a heterogeneous remote attestation protocol compared to single-technology protocols?*

We address this challenge by implementing a heterogeneous remote attestation protocol that bridges the technological gap between the three technologies used to protect our usage control and provenance tracking system.

Contribution RC5: *We develop and evaluate a heterogeneous remote attestation protocol that allows to establish mutually attested and encrypted communication channels between TPM-protected software stacks, Intel SGX enclaves, and ARM TrustZone devices.*

Based on these contributions, we finally integrate all developed protection concepts and attestation protocols into a combined framework for trustworthy distributed usage control enforcement and provenance tracking.

Contribution RC6: *We implement a combined usage control and provenance tracking framework that supports TPMs, Intel SGX, and ARM TrustZone to transparently verify the integrity of distributed system components and thus enforce policy compliance on a technical level. We evaluate our framework in terms of functionality and performance using an exemplary application scenario from the realm of smart manufacturing.*

1.4.3 Estimating Trustworthiness

The final part of this thesis is concerned with giving data providers qualified feedback about the trustworthiness of the usage control system in its current state. To achieve this goal, we must address another research question.

Research Question 5: *How can the trustworthiness of a distributed usage control system be estimated in relation to the policies that should be enforced, from a particular data provider's point of view?*

The perceived level of trustworthiness largely depends on the current system state, the usage control components required for the enforcement of the analyzed policy, as well as the trusted computing technologies and remote attestation protocols protecting them. Hence, we develop a method to derive a quantitative estimation for the level of trustworthiness based on those properties. This estimation can then be communicated to (potential) data providers in order to support informed decisions about the disclosure of usage-controlled critical data.

Contribution RC7: *We define a trustworthiness score representing the ability of a distributed usage control system, which is protected by trusted computing mechanisms, to successfully enforce policies in its current state. We integrate our score into a web-based trust dashboard, which is implemented as part of our usage control and provenance tracking framework.*

1.5 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces relevant preliminary work regarding usage control architectures, provenance tracking models, and trusted computing technologies. In chapter 3 we provide a brief overview of the current state of the art and then develop our conceptual system design for trustworthy distributed usage control enforcement and provenance tracking. We also conduct a security analysis of the developed proposal under the assumption of an idealized remote attestation protocol.

Based on the gained insights, chapter 4 then discusses the benefits and drawbacks of concrete technical protection mechanisms, namely Trusted Platform Modules, Intel SGX, and ARM TrustZone. To meet the identified security requirements, we develop a novel TPM-based attestation protocol that is not vulnerable to nonce-data attacks, as well as a heterogeneous attestation protocol that can provide interoperability between all three considered trusted computing technologies. Chapter 5 integrates all previous findings into a trustworthy distributed usage control and provenance tracking framework, and describes its implementation. Afterwards, in chapter 6, we develop and evaluate a method to estimate the trustworthiness of distributed usage control systems. In chapter 7 we then demonstrate the feasibility of the developed framework and evaluate its performance by implementing an example from the application domain of smart manufacturing. Finally, this thesis closes in chapter 8 with some concluding remarks and an outlook on future work.

2 Preliminary Work

This chapter introduces relevant preliminary work that this thesis builds on, including distributed usage control (section 2.1), provenance tracking (section 2.2), and trusted computing (section 2.3).

2.1 Distributed Usage Control

Usage control (UC) is a technology that allows the continuous monitoring and safeguarding of data [Par02]. While classical access control mechanisms protect resources only until a requested operation is initially granted, usage control can restrict sensitive information even after it has been disclosed. Because of this property, usage control can be seen as the evolution of classical access control [Laz10]. The core model of usage control is UCON$_{\text{ABC}}$, developed by Park and Sandhu [Par04]. UCON$_{\text{ABC}}$ generalizes the classical attribute-based access control model (ABAC), which defines acting agents (*subjects*) and sensitive resources (*objects*) by associating them with descriptive attributes. With ABAC, access decisions are determined by evaluating the granted *access rights* based on the current attribute values of the implicated subjects and objects [Hu15]. This process is also called *authorization.* UCON$_{\text{ABC}}$ expands this traditional model with the concepts of *obligations* and *conditions.* Obligations are specific requirements that subjects must fulfill for the requested operation to be granted. Conditions define environmental circumstances, independent of subjects or objects, which need to be satisfied for a positive authorization. By combining attribute-based authorizations (A) with obligations (B) and conditions (C), UCON$_{\text{ABC}}$ allows to specify flexible usage strategies, such as restricting data usage to a certain time and location (condition), or demanding the deletion of sensitive information after a certain time interval (obligation)

[Aka22]. Figure 2.1 illustrates the various parts of the UCON$_{ABC}$ usage control model as specified by Park and Sandhu.

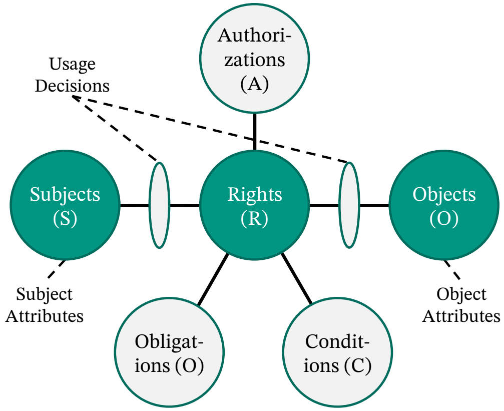


Figure 2.1: The UCON$_{ABC}$ usage control model after Park and Sandhu [Par04].

Another novel facet of usage control is the concept of *continuous enforcement* [Aka22]. In contrast to ABAC, usage control allows to continuously monitor and safeguard protected objects not only at the time of the initial access, but also during and after the subsequent data usage. UCON$_{ABC}$ achieves this kind of continuous enforcement by including a notion of consecutiveness in the model. Authorizations, obligations, and conditions can be defined and evaluated before (*pre-*), during (*ongoing-*), and after (*post-*) any data usage [Par04]. If the usage rules are violated at any of these points in time, continuous access to the data will be revoked. Furthermore, subject and object attributes can be updated at those times as well (*attribute mutability*). Continuous enforcement and attribute mutability allow the specification of very flexible and fine-grained usage rules that can distinguish legitimate data processing scenarios from malicious data usages.

Pretschner et al. developed *distributed usage control (DUC)* as an important enhancement of the core usage control model [Pre06, Kel13]. Unlike UCON$_{ABC}$, distributed usage control deals with use cases where critical data are being transferred from a data provider to a remote data consumer. In such cases, data owners specify their own preferences and requirements regarding the usage of their data in the form of usage control policies. These policies are then transmitted alongside the protected data to the remote receiver, where the specified usage rules are being continuously enforced. Figure 2.2 illustrates this fundamental concept of distributed usage control.

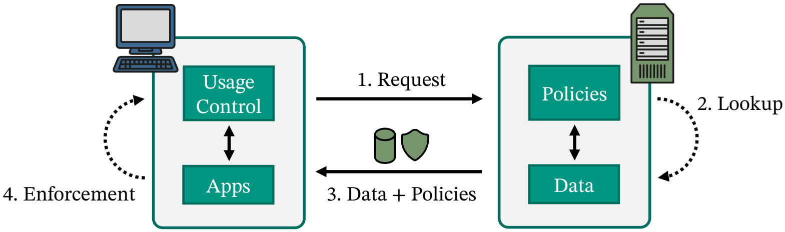


Figure 2.2: The principle of distributed usage control. Own illustration after [Hil07a].

Distributed usage control distinguishes between a *preventive* and a *reactive* type of enforcement [Pre09a]. Preventive usage control enforcement actively supervises and filters data processing applications to inhibit illegitimate data usages according to the specified policies. This requires installing capable enforcement mechanisms in the data processing applications on the receiver's side. However, preventive policy enforcement can be quite disruptive for legacy applications and hence is not always feasible or desired. In contrast, reactive usage control enforcement does not directly prevent policy violations, but ensures that the policy issuer is notified about any digression. The data owner can then react accordingly to the violation, for example by penalizing the offending data receiver, demanding a proper compensation, or initiating legal action. The challenge addressed in this thesis – establishing trustworthiness in distributed usage control systems – is relevant for both of these enforcement types. Preventive usage control enforcement requires mechanisms allowing policy issuers to verify the remote data processing applications, while reactive enforcement requires a reliable back channel that cannot be influenced or manipulated by the data consumer. However, most existing usage control frameworks rely on the preventive type of policy enforcement. Since its development, the distributed usage control model has been formalized [Hil07b, Pre08, Pre09b], extended by mechanisms for information flow tracking [Pre11], as well as extensively evaluated in practical applications [Kel13, Bir16, Kel18]. As a result, this model today serves as the theoretical baseline for implementing usage control in distributed systems.

2.1.1 The XACML Architecture

While UCON$_{ABC}$ and the distributed usage control model can formally represent usage restrictions, these models do not yet consider the architectural design of a usage control system capable of enforcing these restrictions. The earliest implementations of usage control systems relied on a monolithic design approach, where a single protection component is responsible for monitoring and supervising applications, evaluating and enforcing usage control decisions, as well as managing the set of active policies [Agr07, Pre07]. However, such large monolithic system designs have quickly proven to be infeasible for more complicated applications in terms of flexibility and scalability. Similar problems in the field of access control led to the development of the *XACML* reference architecture. XACML (short for *Extensible Access Control Markup Language*) is an OASIS standard for attribute-based access control systems, which among other things includes an information model, an XML-based policy language, and a corresponding reference architecture [OAS13]. Since usage control is a generalization of attribute-based access control, an extended variant of the XACML reference architecture is often adopted to implement (distributed) usage control systems [Jun14, Kel18, Mar19, Aka22]. This XACML-based usage control architecture provides separation of concerns by defining several independent components, each with a specific task in the usage control system. The components are interacting via well-defined interfaces and continuously share information about the current state of the usage control system. Usage control enforcement is then achieved by the collective effort of all involved system components. Figure 2.3 illustrates the components of XACML-based usage control systems, as well as their individual functions and interactions.

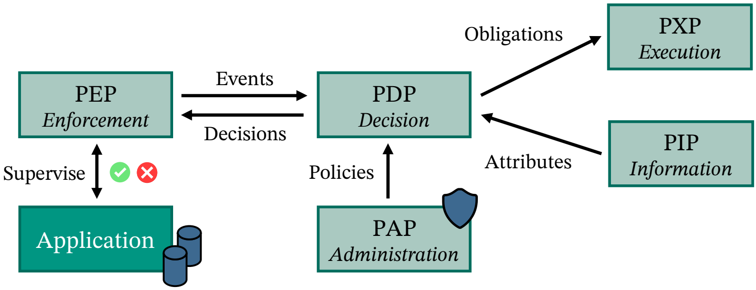


Figure 2.3: An XACML-based usage control system architecture.

Policy Enforcement Point. The Policy Enforcement Point (PEP) is responsible for enforcing usage restrictions and hence is one of the most important components in the architecture. A PEP supervises data processing applications by anticipating and intercepting data usages, generating *events* describing the intercepted usages, and disseminating them into the usage control system. Based on those events the system generates usage decisions, which are then transmitted back and enforced by the PEP. Since enforcement points require far-reaching access to the protected data for their responsibilities, they are usually implemented as part of the data processing applications themselves [Laz14, Bir16], as a proxy module filtering data streams [Riz19, Mun20], or as a low-level operating system module [Bai10, Kel18].

Policy Decision Point. The Policy Decision Point (PDP) is responsible for generating the usage decisions that are enforced by a PEP. For this, the PDP evaluates the received events against the set of currently active usage control policies, and decides if the requested data usage is legitimate. In addition to either granting or denying a particular data usage, the decision point can also demand the requested usage to be modified or delayed before it can be allowed [Jun14]. Like enforcement points, the PDP is a mandatory component in every usage control system. However, usually only a single PDP is used in each usage control domain, whereas multiple PEPs can be deployed to enforce the resulting decisions in the individual applications.

Policy Information Point. A Policy Information Point (PIP) is providing decision points with additional information that is required for the evaluation of usage control policies. This mainly includes subject and object attributes, but also independent information such as database entries or environmental properties. Information points have also been used to store a data flow model that can be queried by decision points for alternative representations of the data to protect [Kel13, Bir16]. PIPs are very useful as information sources for complex usage control scenarios, but unlike PEPs and PDPs, they are not mandatory components.

Policy Execution Point. A Policy Execution Point (PXP) is responsible for executing obligations that are demanded during the evaluation of a usage control policy. For example, a PXP can be used to increase an access counter, log the data usage, or send a notification to the original data owner before the data usage is allowed. Like information points, PXPs are very useful components when implementing complex data usage strategies, but they are not mandatory components either.

Policy Administration Point. Finally, a Policy Administration Point (PAP) manages and controls the set of active usage control policies. Usually, PAPs offer a user interface that allows system administrators to check the currently active policies and deploy or revoke them at the decision point. In distributed usage control scenarios, PAPs may also receive and automatically deploy policies from remote data providers. Especially in the distributed case, this component is sometimes referred to as a Policy *Management* Point (PMP) instead [Jun14, Kel18].

2.1.2 Decentralized vs. Cross-Domain Usage Control

The XACML-based usage control architecture specifies a logical design for distributed usage control systems, which encompasses several collaborating components. However, the actual implementation and deployment of those logical components is not addressed by the XACML standard. Sometimes the different XACML components are realized as part of a usage control framework running on one single computer system, receiving and enforcing distributed usage control policies [Bai10, Kel13]. In contrast, Kelbert et al. proposed to implement distributed usage control systems in a fully decentralized way, meaning that the different XACML components are realized as independent (sub-)systems communicating with each other over the network [Kel14, Kel15]. This has the main advantage of increased flexibility and scalability. In some use cases a decentralized architecture is even required, for example when enforcing usage control with PEPs located on multiple clients that should be linked to the same decision point.

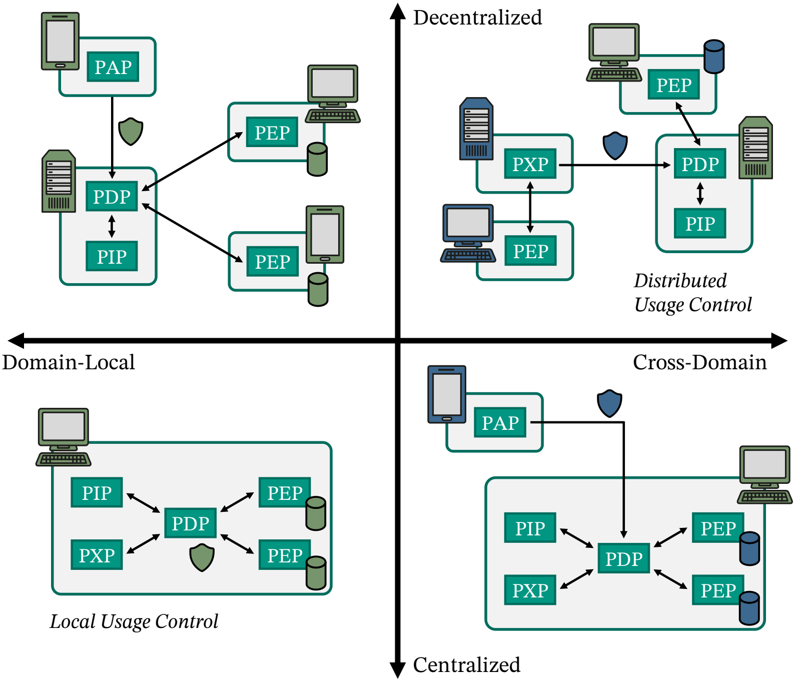


Figure 2.4: Decentralized vs. cross-domain usage control.

In the scientific literature, the term *distributed* usage control is sometimes used ambiguously to mean either usage control between multiple participants, or a usage control architecture implemented over multiple networked computer systems. To clarify the terminology for the remainder of this work, we will instead use the terms *cross-domain* and *domain-local* to refer to usage control that is conducted between multiple participants or in a single domain, respectively. As introduced by Kelbert et al., we use the terms *decentralized* and *centralized* to refer to usage control architectures that encompass multiple networked components in contrast to a single computer system [Kel14, Kel15]. We furthermore consider usage control to be *distributed* if it is both decentralized and cross-domain. Figure 2.4 illustrates this terminology as it is used in the remainder of this thesis.

2.1.3 Policy Languages

In addition to capable usage control models and system architectures, the third vital part of any usage control framework is the policy language adopted for the specification of usage rules. The chosen language needs to be expressive enough to adequately represent the desired usage restrictions, while simultaneously being suitable for the enforcement strategies supported by the implemented usage control architecture. Over the years, several usage control languages with different purposes and goals have been developed. Recently, Akaichi and Kirrane compiled a comprehensive survey that includes a list of all policy languages proposed so far [Aka22]. In this section, we briefly present and compare a selection of the most important languages that are suitable for the purposes of this thesis.

XACML. Since the XACML reference architecture serves as a basis for many usage control systems (see section 2.1.1), its corresponding policy language has also been adopted for usage control purposes. While originally developed for attribute-based access control, XACML already supports many features necessary for a usage control policy language. For example, it allows the specification of obligations and supports using PIPs as additional information sources in the policy [OAS13]. However, attribute updates or the continuous evaluation of conditions is not supported by XACML. Furthermore, specified obligations can only be executed by the enforcement points themselves, not by external PXPs. There are some Java-based implementations of XACML policy decision points available, both as commercial[1] and open source[2] projects. To make the XACML policy language compliant with the UCON_{ABC} usage control model, several extensions to the standardized language have also been proposed [Col10, El 15, Di 18]. However, so far none of these extensions have been fully implemented or widely adopted.

OSL. The Obligation Specification Language (OSL) has been developed by Hilty et al. as a policy language specifically designed to express usage control requirements [Hil07b]. OSL employs the common *event-condition-action (ECA)* pattern, which defines authorizations as a triple consisting of an event, one or more conditions, and an action. Events are used to describe usage requests, which are matched to OSL rules according to the unique event name. The conditions of any matching OSL rules are then evaluated using the event parameters. Only if the specified conditions are fulfilled, the associated action is finally authorized. OSL has originally been specified in the logic language *Z*, which allows to formally verify OSL rules against the underlying usage control model. However, implementation-level representations of OSL policies are typically serialized as XML documents [Kum12, Wüc12].

ODRL. The Open Digital Rights Language (ODRL) is a comprehensive policy language that has been specified as a recommendation by the World Wide Web Consortium (W3C) [Ian18b]. Originally developed for Digital Rights Management (DRM) applications, ODRL now offers a generic and highly extensible usage control language. The official ODRL specification includes a core information model, a vocabulary document, and several policy serialization syntaxes. ODRL policies are most often serialized in JSON-LD, but the

[1] https://www.axiomatics.com/ *(accessed on 12/08/2023).*

[2] https://github.com/authzforce/ *(accessed on 12/08/2023).*

specification also supports XML as well as RDF, a resource description format. Similar in nature to XACML, ODRL allows the specification of rights, conditions, and obligations under which the usage of digital resources may be permitted or prohibited. However, unlike XACML, ODRL does not rely on a specific reference architecture, which is why by default it does not support the definition of external PIPs or PXPs. On the other hand, the advantages of ODRL include its highly expressive nature, an extensible information model, and a compact specification with clear semantics. Due to its status as a W3C recommendation, ODRL can be seen as a de-facto industry standard for the specification of digital rights. Consequently, ODRL has been chosen as the default language to disseminate usage rules in the International Data Space [Ste21]. Furthermore, there are also some projects implementing the ODRL specification in Java[1] and JavaScript[2].

MyDataControl. MyDataControl is a Java-based distributed usage control framework developed by Jung et al. [Jun14, Jun22]. The framework is designed after the XACML reference architecture and provides several usage control components as standalone web services, alongside a dedicated policy language. Like OSL, the MyDataControl policy language is based on the event-condition-action pattern, which authorizes data usages by evaluating policy conditions on the intercepted system events. In addition, MyDataControl supports delaying and modifying events before they are authorized. Since this language has been developed specifically for the MyDataControl framework, it already includes comprehensive support for both information and execution points. PIPs and PXPs can be directly referenced from policies, which allows the realization of complex data usage restrictions. MyDataControl uses an XML-based policy representation and is also considered in the International Data Space [Ste21]. However, while the policy specification is publicly available,[3] the corresponding decision point implementation is closed source and held under a proprietary license.

[1] https://github.com/oeg-upm/licensius *(accessed on 12/08/2023).*

[2] https://github.com/nitmws/odrl-wprofile-evaltest1 *(accessed on 12/08/2023).*

[3] https://developer.mydata-control.de/language/ *(accessed on 12/08/2023).*

LUCON. Logic-Based Usage Control (LUCON) has been developed by Schütte and Brost as a mechanism for restricting data flows in message-based systems [Sch18b]. LUCON features a Domain Specific Language (DSL), which can be used to control how data may be routed, combined, and processed across distributed services. As such, LUCON is not a generic usage control language and can only be applied sensibly in message-based environments. Nevertheless, the LUCON language allows the specification of rights and obligations that are matched against services and messages. Static and dynamic data flow analyses of message routes are performed by compiling the LUCON policies into first-order logic terms and evaluating them using Prolog. The LUCON reference implementation is Java-based and has been integrated into the Apache Camel message routing of the IDS-compliant Trusted Connector [Sch18a]. As such it is available under the Apache 2.0 license.[1]

Table 2.1: Comparison of usage control policy languages.

Name	Purpose	PIPs	Obligations	Serializing	Code
XACML	ABAC	Yes	PEP only	XML	Java
OSL	UC	No	PEP only	Z, XML	None
ODRL	DUC, DRM	No	PEP only	JSON-LD, XML, RDF	Java, JavaScript
MyDataControl	DUC	Yes	PXP	XML	Java
LUCON	DUC	Yes	PEP only	DSL	Java

Table 2.1 provides a comparison of the discussed policy languages in terms of capability and area of application. Due to the comprehensive nature of the underlying framework, at the time of this thesis MyDataControl can be seen as the state of the art policy language for distributed usage control. However, since the associated reference implementation is not available as open source, in this thesis we will focus on the ODRL policy language instead. Most importantly, ODRL provides an expressive and extensible usage control language with a small overhead, which is convenient for our purposes. As a W3C recommendation, ODRL can also be considered an accepted industry standard.

[1] https://github.com/Fraunhofer-AISEC/trusted-connector *(accessed on 12/08/2023).*

In the course of this thesis, we extend ODRL with provenance tracking mechanisms and then use it to demonstrate our proof of concept for a trustworthy distributed usage control system (see research contributions RC2 and RC6).

2.2 Provenance Tracking

Provenance data are a type of metadata that describe the history and evolution of data sets. They usually include information about the processes, technical entities, and people that have been involved in the acquisition and the usage of data sets [Gro13]. Provenance information can be used to derive knowledge about the origins of data and how they have been distributed, processed, altered, and combined since their generation. This is useful to decide whether certain information can be trusted and how they may be integrated into larger data sets. *Provenance tracking* is the concept of automatically capturing provenance information in data processing systems. Provenance tracking is especially useful in areas of application where large amounts of data are being continuously evaluated [Her17]. For example, when conducting data-driven scientific experiments (i.e., in the field of physics or biomedical research), provenance tracking mechanisms can help to trace the origin and the subsequent evaluation of scientific data [Bar18b, Car18]. This is a crucial requirement for reliably identifying systematic or computational errors and achieving reproducible scientific results [Sah19]. Furthermore, provenance tracking is useful for maintaining database management systems [Bun06, Che09a], curating training data for machine learning models [Sou19, Wer22], and even for detecting cyber-attacks by analyzing the provenance of security audit logs [Bat19]. Another key area of application for provenance tracking is privacy protection. Ujcich et al. show how provenance models can be used to verify compliance of data processing systems with European data privacy law [Ujc18]. Bier leverages provenance tracking to implement a data protection information system that can provide transparency for data subjects [Bie21]. Provenance tracking mechanisms have also been integrated into virtual data space architectures [Ste21], where they keep a reliable log of conducted data transmissions and provide data owners with the information of where and for what purpose their data are being used.

2.2.1 The PROV Model

Designing a system that can track and evaluate the provenance of data requires a suitable *provenance model* as a logical foundation. The provenance model defines how the collected provenance information should be represented, serialized, stored, and exchanged between computer systems. It is usually technology-independent and provides an abstract definition of the provenance semantics. The current state of the art provenance model is the *PROV* family [Gro13]. Standardized by the W3C in 2013, the PROV family consists of several related documents giving both users and developers comprehensive guidelines about how to specify, access, and interpret provenance information. The core of the PROV family is the provenance data model (PROV-DM) [Bel13]. PROV-DM defines a vocabulary together with a graph-based data model that generically represents provenance information using three different base classes and seven types of relations. Concrete instances of these classes are represented as vertices in a provenance graph, while relations between two instances are represented as directed edges. Figure 2.5 shows the base classes and relation types defined in the PROV-DM data model.

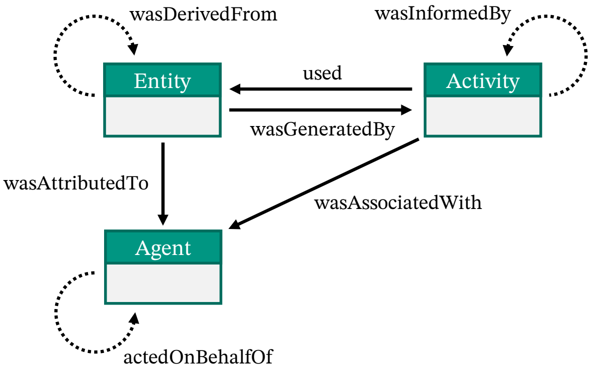


Figure 2.5: The PROV-DM data model [Bel13].

The three base classes of PROV-DM are called *entities*, *activities*, and *agents*. Entities are the “things” that the provenance should be collected of, for example physical objects or digital assets such as data. They can be derived from other entities, which is represented by the *wasDerivedFrom* relation. Activities describe generic processes that run for a certain duration, for example a particular step in a data processing chain. Activities can generate and use entities, which is represented by the *wasGeneratedBy* and *used* relations. Finally, the concept of agents is used to describe persons, organizations, or systems that are associated with certain entities and/or activities. An agent can be the originator of an entity (*wasAttributedTo*) and bear some responsibility for an activity (*wasAssociatedWith*). Agents can also delegate their authority to other agents (*actedOnBehalfOf*). All elements of the PROV-DM model, including the relations, can be described further by associating them with attributes, such as the duration of an activity or the purpose of a data usage. PROV-DM is a domain-agnostic model and meant to be extended by deriving new classes, relations, and attributes to better describe the semantics of a particular use case. There are a number of PROV-DM extensions that are covering various specific domains of provenance tracking, for example in big data applications [Gao20] or to represent data privacy requirements [Ujc18]. The PROV family also specifies several serializations of the provenance data model, including PROV-XML and a human-readable notation of provenance information (PROV-N) [Gro13]. Furthermore, PROV includes a definition of interfaces for the web-based lookup and retrieval of stored provenance information (PROV-AQ). Because PROV is an expressive but also relatively lightweight standard, which can be easily extended for the specific needs of many different application domains, it serves as a foundation for most provenance tracking systems today.

2.2.2 System Architectures

In addition to a data model and corresponding serialization schemes, implementing provenance tracking in practice also requires the design of a suitable system architecture. The system architecture defines by what technical means and processes provenance information is collected, handled, and exploited.

Provenance tracking systems usually distinguish three different phases of the provenance life cycle [Zaf17, Bie21].

(i) *Collection:* As a first step, provenance information about the supervised entities is generated and collected at the data processing applications. This has to be done at a level of granularity that fits the purpose of the respective provenance (e.g., files, tables, or individual data items). Provenance collection can be conducted indirectly by analyzing existing log files, or directly by including provenance tracking modules into the used applications and operating systems.

(ii) *Storage:* Second, the collected provenance information must be stored, either together with the original data or in separate databases. Depending on the type of collected information, external provenance storage is usually implemented using relational databases [Gla10], tuple stores [Pat10], or graph databases [Geh12]. There are also proposals that store provenance information using blockchains [Lia17, Nei17], as well as using cryptographic solutions such as encryption and group signature schemes [Asg11, Alh12].

(iii) *Dissemination:* Finally, suitable interfaces must be provided that allow users to query and analyze the captured provenance data. This is usually done using the query language of the respective database that has been used for storage, such as SQL [Gla10] or SPARQL [Pat10].

Existing provenance tracking frameworks usually integrate these three phases into a single monolithic system architecture [Hu20]. Inspired by the concept of distributed usage control, Bier instead proposes a provenance tracking architecture based on distributed system components [Bie21]. In this system design, provenance information is collected by evaluating a separately constructed data flow model. The provenance information is then stored using a set of distributed Provenance Storage Points (ProSPs) and disseminated by a Provenance Dissemination Point (ProDP). The advantage of this approach is that it can be rather easily included in distributed usage control architectures [Bie13]. Because of this, we adopt a simplified version of Bier's provenance tracking system design for this thesis (see section 3.2.2).

2.3 Trusted Computing

One cause of many security problems today is the fact that the software stack running on computing devices (and by extension the device behavior) can be modified rather easily at all times of the device's life cycle. For example, as soon as a device is infected with malware, it can tamper with the system's configuration and binaries to hide from detection while stealing user data from the compromised device. To prevent this, there are many levels of software isolation available on modern computer systems. Operating systems and hypervisors are responsible for separating processes and virtual machines from each other to prevent data leakage. However, since these isolation mechanisms tend to be complex and have large attack surfaces, vulnerabilities are fairly common and often quite dangerous [Tan19]. Furthermore, software-based isolation mechanisms are also vulnerable against attackers with physical access to the device, since they can tamper with the installed operating system or the device's firmware. Especially if platform owners have to be considered as potential attackers (like in the case of distributed usage control), another type of security feature is necessary.

The term *trusted computing* describes the concept of establishing trust in computing devices by making the device behavior predictable and verifiable [Wag18b]. This is achieved by installing specialized hardware modules as trust anchors (also called *roots of trust*) to enhance the security and trustworthiness of a computing device. The rationale behind this is that hardware – unlike software – cannot be easily modified, even with privileged and/or physical access to the device. The hardware roots of trust are then used as a building block to provide secure cryptographic functionalities such as encryption and digital signatures. Most trusted computing devices are also capable of using roots of trust to verify the software that is currently being executed on the protected system. This extends the initial trust that is placed in the immutable hardware trust anchors to software components, for example single processes or even the whole software stack. The entirety of the trusted computing hardware and the protected parts of the software is then called a Trusted Computing Base (TCB) [Tru19b, p. 21]. In recent years, the field of trusted computing has been researched extensively both in academia and the industry [Sch22]. Various trusted computing approaches and technologies are available from different manufacturers, often pursuing contrasting protection philosophies. In the remainder of this section, we give a brief introduction to the most important trusted computing technologies used today.

2.3.1 Trusted Platform Modules

One of the first widely available trusted computing technologies was the *Trusted Platform Module (TPM)*. TPMs are tamper-resistant hardware chips that provide computers with a cryptographic co-processor, similar to a smart card. They allow the secure generation and storage of cryptographic keys that can be used to encrypt confidential data and create digital signatures. The TPM architecture and functionality is standardized by the Trusted Computing Group (TCG), a non-profit organization consisting of security experts from both academia and the industry.[1] The current version 2.0 of the TPM specification has most recently been revised in November of 2019 and is publicly available [Tru19b]. The TPM 2.0 specification brought many improvements over their predecessors and by now has been widely adopted by TPM manufacturers and system integrators. However, TPM 2.0 devices are not backwards compatible with earlier TPM versions. Since earlier TPM versions are hardly ever used anymore, in this thesis we only deal with TPMs in version 2.0.

Architecture. Hardware TPMs consist of a standard microcontroller architecture that includes main memory, I/O interfaces, and a central processing unit responsible for executing the TPM functions. However, TPM chips also contain several security-specific components [Wag18b]. This includes volatile and non-volatile memory modules that are reserved for cryptographic purposes such as key storage and session handling. An authorization component is responsible for storing and evaluating policies that can be used to limit

[1] https://trustedcomputinggroup.org/ *(accessed on 12/08/2023).*

access to keys and other TPM objects. The TPM also contains separate engines responsible for implementing hash functions, as well as symmetric and asymmetric cryptography. This type of modular architecture yields *algorithm agility*, which is a major improvement of TPM 2.0 over previous versions. The idea behind algorithm agility is to separate TPM functionality from the used cryptographic mechanisms. It allows TPM users to select from a wide range of implemented cryptographic algorithms, instead of being limited to predefined algorithms for certain purposes. While the TCG defines a core set of mandatory algorithms that a TPM must support, manufacturers are free to implement additional algorithms as well. For a full description of the TPM 2.0 architecture, we refer the reader to the TPM specification documents [Tru19b].

Key Management. A core functionality of Trusted Platform Modules lies in the generation, management, and usage of cryptographic keys. For this, each TPM is provisioned with a set of unique *platform secrets*, which are not known to anyone and cannot be read out of the TPM chip. These secrets are used internally to derive deterministic *primary keys*, which can be certified by the TPM manufacturer. This certification proves to a TPM user that the primary keys indeed belong to a genuine TPM and are protected by it. Since they are deterministic, the primary keys should not be used directly to encrypt or sign data. Instead, the primary keys are used as roots of a *key hierarchy* [Art15]. Each new cryptographic key that the TPM generates is *wrapped* (i.e., encrypted) by a parent key in order to securely store it outside the TPM. To use a wrapped key, it first needs to be loaded into the TPM's protected memory, where it gets decrypted by its respective parent key. This allows the TPM to manage a virtually unlimited number of cryptographic keys with very little internal memory. The memory regions that are used to store platform secrets and loaded private keys are completely isolated and can only be accessed by the TPM itself. This is commonly referred to as a *Root of Trust for Storage (RTS)*. The RTS can protect both confidentiality and integrity of stored information. A TPM distinguishes primary keys for different purposes, such as for storage, endorsement, and to control the platform firmware. It also associates a set of attributes with each generated key, which can be used to control properties like the key's migratability, duplicability, and its access control policy. A full description of the TPM key management is given in [Tru19b].

Integrity Measurement. Trusted Platform Modules can also be used to measure the integrity of the software stack that is being executed on a computer system. This is achieved by collecting fingerprints of loaded binaries and configuration files in form of hash digests. These digests are called *measurements* and together represent the entire software state of a trusted platform. Integrity measurements allow platform owners to verify that the executed software is indeed benign and has not been misconfigured or infected by malware. Two challenges have to be solved in order to conduct secure and trustworthy integrity measurements.

(i) The list of collected measurements must be stored completely isolated from the measured software.

(ii) The measurement process may not be influenced or manipulated by anyone outside the TPM trust anchor.

If this were not ensured, malicious software could hide from detection either by removing the suspicious fingerprints from the stored measurements, or by preventing its own measurement in the first place. However, the TPM takes care of both of those requirements.

Platform Configuration Registers. To securely store integrity measurements, the TPM provides special-purpose volatile memory called the *Platform Configuration Registers (PCRs).* The PCRs are part of the TPM's Root of Trust for Storage and hence cannot be directly accessed from outside the TPM.[1] The PCRs are initialized to either all zeros or all ones at every system restart and cannot be cleared or reset during the system operation. Since the TPM only allows to add but not delete digests in the PCRs, they can be used to store a sequence of measurements without risking manipulation by malicious software. However, the TPM only allocates a very limited amount of secure

[1] Because the PCRs are not confidential, the RTS only protects their integrity.

memory for the PCR banks. To ensure that each PCR can attest to a virtually unlimited number of measurements, its entries are recursively chained together using a cryptographic hash function H. The only way to change the value of a specific PCR is to invoke its *extend* operation. This operation adds a new measurement digest d to the k-th PCR by calculating

$$PCR[k] \coloneqq H(PCR[k] \,\|\, d).$$

The resulting PCR value depends on not just the newly added digest d, but all the previously added digests already included in the old PCR value as well. Since this hashes together all measured digests, the individual measurement values have to be stored in an external measurement log. Measurement verification is then performed by recalculating the intermediate hash values using all logged measurement digests and comparing the final value with the value stored inside the respective PCR. That way an attacker is still unable to tamper with the externally stored measurement log, since any modification of measurements would lead to a PCR mismatch. Following the concept of algorithm agility, the TPM allows its users to choose which hash algorithm should be used for H. Since the various hash functions use different digest sizes, TPMs often contain multiple PCR banks with different word lengths. Furthermore, each PCR bank usually contains at least 24 registers to allow multiple independent measurement processes and simplify the verification of PCR contents.

Trusted Boot. Populating the PCRs with trustworthy measurements that could not have been manipulated by malicious software is the responsibility of a *trusted boot* process. Trusted boot[1] is not to be confused with *secure boot*, which is a security feature that checks digital signatures of boot loaders. Trusted boot, on the other hand, defines a measurement process that consecutively loads boot stages and extends their digests to the PCRs before executing them. This load-measure-execute sequence starts at a tiny bit of trustworthy code called the *Core Root of Trust for Measurement (CRTM)*, which

[1] Sometimes the term *measured boot* is used instead.

is the very first part of the firmware that gets loaded and executed by the processor during system start. Beginning with the CRTM, each subsequent boot stage is then responsible for measuring the next boot stage, all the way up to the operating system and the user applications. This establishes a *chain of trust (CoT)* rooted in the CRTM that extends over all measured software components. Malicious software in any stage of the boot processes cannot hide from this chain of measurements without first compromising the initial CRTM code. The CRTM together with the freshly initialized processor forms the *Root of Trust for Measurement (RTM)*. If the RTM is trusted, then all subsequently loaded software included in the chain of trust can be trusted as well. The actual boot measurements of the firmware and operating system modules are extended into PCRs 0 to 9. Measuring the user applications is then the responsibility of a kernel-level *Integrity Measurement Architecture (IMA)*, which usually extends its measurements to PCR 10. Figure 2.6 illustrates this process and the resulting chain of trust. A more detailed picture of a TPM-based trusted boot process with IMA support is given in [Wag18b].

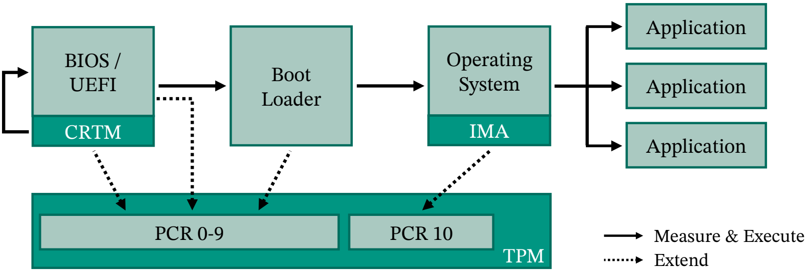


Figure 2.6: The TPM-based trusted boot process. Own illustration after [Wag18b].

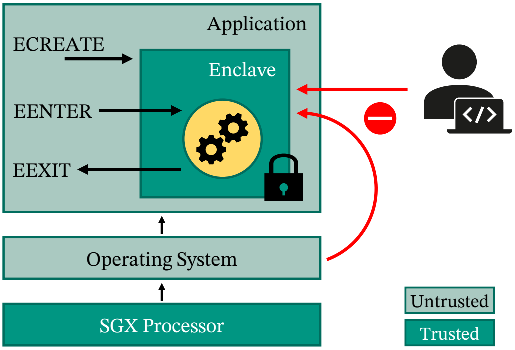
In conclusion, even though TPMs have been around for over two decades, they are still very important security modules and are widely used in a multitude of different applications. Because of this, in the remainder of this thesis we build on TPMs as one of the key technologies to protect distributed usage control architectures.

2.3.2 Trusted Execution Environments

Even though TPMs allow the measurement and verification of a protected system's software stack, they do not provide any security guarantees for the code itself while it is being executed. *Trusted Execution Environments (TEEs)* fill this gap by enhancing the trusted computing concept with the possibility of cryptographically isolating software while it is running. TEEs can be distinguished by the scope of the provided isolation, which directly corresponds to the size of their TCB. *Process-based* TEEs allow to run (parts of) single processes in encrypted containers, which are usually called *enclaves*. *VM-based* TEEs cryptographically protect different virtual machines from each other, while other TEEs provide one or more secure *worlds* or *realms* for the execution of privileged software. In the remainder of this section, we briefly introduce and motivate the most important Trusted Execution Environments. A full list of currently available hardware-based TEEs has recently been compiled by Schneider et al. [Sch22].

2.3.2.1 Intel Software Guard Extensions

Intel's *Software Guard Extensions (SGX)* are a set of processor instructions extending the classical x86 architecture with a process-based Trusted Execution Environment [Cos16a]. SGX allows untrusted processes to launch isolated *enclaves* inside the normal address space using the new ECREATE processor instruction. These enclaves consist of encrypted memory pages, which are only readable in plain-text by a genuine SGX processor. The SGX processor then uses hardware-based access control mechanisms to prevent any untrusted code from accessing enclave memory, and it clears all remaining secret information from the CPU before untrusted code is executed again. This allows the SGX processor to completely isolate the executed enclaves from the rest of the computer system and protect the confidentiality of enclave data as well as the integrity of trusted code. Other system processes, privileged software like the operating system, and even platform owners with physical access to the system cannot read out any enclave memory or manipulate its trusted code base. The untrusted processes can interact with secure enclaves only via an interface defining explicit enter and exit calls. The EENTER processor instruction is used to put the SGX processor into the protected enclave mode and transfer control authority to the trusted code. The EEXIT instruction hands the control flow back to the untrusted process. The TCB of SGX enclaves is very small compared to TPMs, because it only contains the enclave code along with the SGX processor itself. On the other hand, SGX processors are much more complex than TPM chips, which drastically increases the attack surface and has led to vulnerabilities such as address-translation and cache attacks [Fei21]. The principle architecture of SGX enclaves is illustrated in fig. 2.7. A full architectural description of SGX is given by Costan and Devadas [Cos16a].



Untrusted
Trusted

Figure 2.7: The SGX enclave architecture. Own illustration after [Bre18].

Since SGX enclaves are always launched by untrusted processes, the integrity of an enclave needs to be verified before it can be trusted. SGX achieves this with a measurement process that automatically establishes the enclave's code identity. This process uses a hash chaining approach similar to the TPM-based integrity measurement (see section 2.3.1). However, instead of using a set of PCR registers to represent the entire system configuration, SGX uses a single *enclave measurement* digest (MRENCLAVE) to describe the code identity of a running enclave. The MRENCLAVE value is constructed during the launch of an enclave by a sequence of EADD and EEXTEND processor instructions. First, the EADD instruction is used to load and encrypt the next enclave page from untrusted memory. This instruction also appends the page metadata to MRENCLAVE by calculating the SHA-256 hash digest of its concatenation with the previous measurement value. Afterwards, a number of EEXTEND instructions are used to measure the page's memory content in the same way. This measurement sequence is then repeated for every page of the enclave's memory image. The final MRENCLAVE value uniquely represents the entire initial memory image of the executed enclave. Figure 2.8 illustrates this measurement process. Besides the measurement value, an SGX enclave is also represented by the MRSIGNER value. This value identifies the authority that digitally signed the loaded enclave image, which is usually the enclave author. It is used to authenticate keys that securely provision secret data into the enclave during runtime.

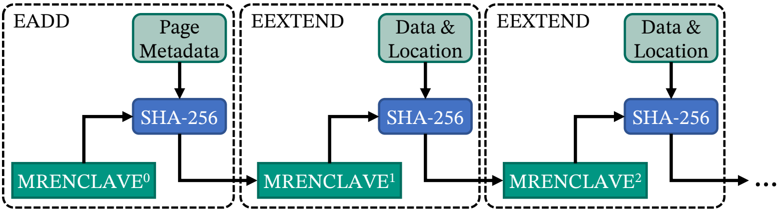


Figure 2.8: The SGX enclave measurement process [Les16].

Intel SGX has been released in 2015 as part of the Skylake processor architecture.[1] In 2021, Intel shifted the SGX technology away from consumer CPUs to the Xeon product line of server processors [Rao22], where it is available ever since. Furthermore, Intel recently expanded their security portfolio by introducing a new TEE technology called *Trust Domain Extensions (TDX)*. While the core functionality of TDX is very similar to SGX, it integrates with Intel's virtualization technology to provide VM-based instead of process-based isolation [Int23b]. However, Intel has announced that they will keep supporting both SGX and TDX for confidential cloud computing use cases in the future [Rao22].

[1] https://ark.intel.com/content/www/us/en/ark/products/series/88392/6th-generation-intel-core-i7-processors.html *(accessed on 12/08/2023)*.

2.3.2.2 AMD Secure Encrypted Virtualization

Secure Encrypted Virtualization (SEV) is a virtualization-based TEE technology developed by AMD for their product line of x86-based processors. Similar to Intel TDX, AMD SEV cryptographically isolates virtual machines from each other by transparently encrypting their main memory with a VM-specific encryption key [Kap21]. The resulting TCB contains the AMD processor along with the entire virtual machine, including the virtualized operating system. This has the advantage of making software deployment much easier compared to Intel SGX, since existing software must not be modified to be used in SEV-protected environments [Mof18]. However, including an entire operating system in the TCB greatly increases the available attack surface and the likelihood of vulnerabilities in the trusted code. On the other hand, the underlying VM hypervisor is not part of the TCB and must not be trusted [Kap21]. SEV-protected virtual machines can also choose to keep certain memory pages unencrypted in order to communicate with other VMs or the hypervisor. AMD SEV was introduced in 2016 and has since been updated with additional security features including memory integrity protection [AMD20a] and data leakage prevention [Kap17]. A comprehensive description of the AMD SEV architecture and API can be found in [Kap21, AMD20b].

2.3.2.3 ARM TrustZone

ARM TrustZone is a TEE technology that has been introduced in 2004 for the Armv6-A application processor family [Pin19]. Since then, it has been adopted for the ARM Cortex-M microprocessor profile as well, albeit with a slightly different internal design than the original Cortex-A version. As it has been around for a while, many security features in the embedded and mobile sector are based on TrustZone, including the Samsung Knox[1] technology

[1] https://docs.samsungknox.com/admin/fundamentals/whitepaper/the-samsung-knox-platform/ *(accessed on 12/08/2023).*

and the Android secure operating system[1]. ARM TrustZone achieves software isolation by partitioning the entire protected system into a *normal world* and a *secure world*. Hardware-based access control is used to prevent normal world code from accessing any resources assigned to the secure world, including main memory and I/O devices [Pin19]. However, unlike AMD SEV, TrustZone does not support the transparent encryption of memory pages that are mapped to the secure world. The normal world is called the *Rich Execution Environment (REE)*, while the secure world implements the TEE. The new security state introduced by TrustZone is orthogonal to existing privilege levels, most importantly to the processor's kernel-mode and user-mode [Li19]. Both the normal and the secure world operate completely independently of each other and have their own stack of firmware, boot loader, operating system, and applications. The operating system used in the secure world is called *trusted operating system* and is significantly smaller and less complex than the *rich operating system* running in the normal world. This reduces the attack surface on the *Trusted Applications (TAs)* running in the secure world. Figure 2.9 illustrates the ARM TrustZone architecture on a high level. A more detailed description can be found in the TrustZone specification documents for the respective processor architecture [ARM20].

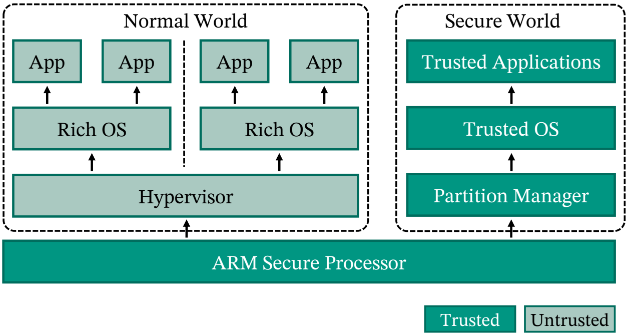


Figure 2.9: The ARM TrustZone architecture [ARM20].

[1] https://source.android.com/docs/security/features/trusty *(accessed on 12/08/2023).*

Since TrustZone utilizes a simplified trusted operating system with a small footprint, the secure world Trusted Applications running on top of it usually are not as complex as legacy applications. However, TrustZone TAs are still more capable and easier to implement than SGX enclaves, which receive no operating system support at all. As a result of this, TrustZone's Trusted Computing Base is bigger than the TCB of an SGX enclave, but still smaller than the TCBs of entire virtual machines protected by AMD SEV. Unlike Intel SGX and AMD SEV, but rather similar to TPMs, TrustZone also supports a specific boot process tailored to establish trust in the secure firmware and operating system [ARM21b]. This process is called *Trusted Board Boot (TBB)* and is closely related to the secure boot concept on x86 systems. During a Trusted Board Boot, the secure firmware verifies the digital signatures of all boot loader stages and the trusted operating system before executing them. However, the TrustZone architecture currently does not include native support for conducting integrity measurements of executed Trusted Applications. As this is a prerequisite for our use case, in section 4.4 we discuss how comprehensive integrity measurements can still be achieved on ARM TrustZone platforms. Recently, ARM presented the specifications of a new TEE technology for Armv9-A called the *Confidential Computing Architecture (CCA)*. ARM CCA extends the TrustZone architecture by introducing multiple *realms* on top of the existing secure world [Li22]. That makes it possible to dynamically isolate multiple Trusted Execution Environments from each other, instead of having all critical software running in a single secure world. In addition, CCA will include support for native measurements of the trusted applications running inside the different realms.

2.3.2.4 RISC-V Physical Memory Protection

In 2017, a security extension called *Physical Memory Protection (PMP)* has been specified for the RISC-V processor architecture [Wat17, Che22]. PMP implements a hardware-based access control mechanism for memory pages similar to the memory protection unit used in ARM TrustZone. The PMP mechanism can be used to isolate software from each other, and several proposals leverage it as a memory protection unit to define process-based TEEs for RISC-V architectures [Cos16b, Wei19, Lee20, Fen21]. Furthermore, it is also possible to use PMP to conduct and verify integrity measurements of isolated process memory [She21]. However, due to being an open source instruction set architecture, TEEs based on RISC-V are being considered mostly in academia and are currently not very widespread in the industry.

2.3.3 Remote Attestation Protocols

Automatically measuring the software stack executed on a trusted computing platform can prevent malware or malicious system users from tampering with critical applications unnoticed. However, for many applications it is not sufficient to verify the resulting integrity measurements just locally. For example, platform owners might want to dynamically provision critical data into trusted software stacks over the internet. This requires a mechanism that can be used to check the state of a specific trusted computing platform without being physically present. This concept is called *Remote Attestation (RAT)*. Remote attestation protocols can be used to verify conducted integrity measurements from outside the trusted platform itself. The trusted platform under test is usually called a *prover* or *attester*, while the remote party is called a *verifier*. Figure 2.10 illustrates the fundamental procedure of a remote attestation.

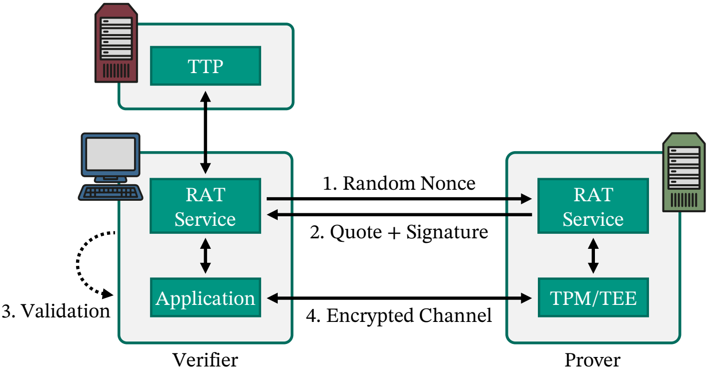


Figure 2.10: The concept of remote attestation protocols.

First, the verifier sends a randomly drawn, unique *nonce* to the prover as a challenge. The prover then creates a report attesting to the current platform state (usually called a *quote*), which contains at least the received nonce, the platform measurements, and possibly also other platform information depending on the used trusted computing technology. The created quote is signed by the trusted platform with an *attestation key*, which certifies that the generated quote is a genuine attestation report representing the prover's current platform state. The signed quote is then transmitted back to the verifier, where it can be validated in two steps. First, the quote signature must be validated to establish the authenticity of the report. The nonce contained in the received quote must also match the previously provided challenge to prevent a malicious prover from reusing old reports. Second, the verifier can extract the platform measurements from the authenticated quote and use them to assess the integrity of the attested platform, usually by comparing the measured binary fingerprints to known good values. Depending on the use case, these known good values are either provided directly by the application developer or get provisioned via an external measurement repository such as a *Trusted Third Party (TTP)*. In addition to the remote attestation step, most protocols also establish an encrypted channel between the verifier and the attested trusted platform, for example by integrating the attestation into a TLS handshake [Kna18].

The exact remote attestation process, as well as the content and structure of the quote, differs for every trusted computing technology. TPMs use a quote structure that includes a selection of PCR digests for the attestation. These quotes are signed with a restricted attestation key that can only be used for TPM-internal data structures. Since the TPM-based integrity measurement process includes the entire system software (see section 2.3.1), attestation protocols usually transmit the quote together with the collected measurement logs. The verifier can authenticate the measurement logs by using them to recalculate the PCR values and comparing the results with the attested PCR digests in the quote. The authenticated measurement logs then give an overview of all software that is running on the attested TPM-protected system. A more detailed picture of the TPM-based remote attestation process is given later in section 4.2.2. SGX-based reports, on the other hand, mainly include the `MRENCLAVE` and `MRSIGNER` values, as well as additional information about the SGX platform capabilities and the initial enclave configuration. SGX reports are signed by a special *quoting enclave* with a secret attestation key that never leaves the protected enclaves. The verification process is then similar to TPM-based remote attestation. However, since single enclaves are much simpler entities than entire software stacks, separate measurement logs are not required. Besides remote attestation, the SGX architecture also offers a *local attestation* process. Local attestation is basically a simplified version of remote attestation, where SGX reports are authenticated locally using the SGX instruction set instead of the attestation key in the quoting enclave. More information about the SGX-based attestation protocols is given later in section 4.3.2. AMD SEV and RISC-V based TEEs also define similar remote attestation protocols. Since ARM TrustZone does not natively support integrity measurements, no standard remote attestation protocol exists on that platform. However, there are some proposals that migrate TPM-based remote attestation protocols to TrustZone platforms [Wan20].

One major challenge in defining a secure remote attestation protocol is to properly link the attestation keys used to sign quotes with the trusted platform itself. For this, all attestation keys need to be rooted in the trusted hardware, and there must be a way for verifiers to authenticate them as genuine. The trust anchor required for this is called the *Root of Trust for Reporting (RTR)*. Different trusted computing technologies vary in the way the RTR is implemented. TPMs use attestation keys derived in the endorsement hierarchy, which can be directly authenticated as genuine using an *endorsement certificate* provided by the platform manufacturer. Intel SGX instead defines a provisioning process that establishes certified attestation keys at local quoting enclaves. Verifiers can ensure the authenticity of the presented attestation keys by validating their certificates with a trust anchor at Intel. However, directly connecting attestation keys to a specific trusted platform also has privacy implications that need to be addressed. If there is a way to deduce information about the identity of the attested platform from the public part of the used attestation key, the privacy goal of *unlinkability* would be violated. A cryptographic solution supported by both TPMs and Intel SGX is to generate attestation keys using group signature schemes [Bri10]. Another way of resolving this issue is to introduce a *privacy broker* as an additional level of indirection. The privacy broker is assumed to be trusted and can remove the direct link between an attestation key and the trusted platform by providing an intermediate attestation certificate.

Finally, there are also some remote attestation concepts that are not based on trusted computing hardware at all. Since this area of research mostly considers very resource-constrained IoT devices and embedded systems, it is also called *device attestation*. Software-based device attestation techniques measure certain device responses that are prone to subtle changes when the device is modified, for example response timings for different inputs [Ses04, Li10, Dus20]. The advantage of this approach is that it does not require any hardware-based trust anchors, but the resulting protocols are suitable only for limited use cases. Hardware-based device attestation mechanisms, on the other hand, often utilize *Physically Unclonable Functions (PUFs)*. The output values of PUFs are susceptible to minor flaws during the manufacturing process of the physical devices and cannot be easily predicted. Because of this, PUFs are physically linked to a specific device and cannot be copied or transferred. A number of device attestation protocols exploit this property to remotely measure the memory contents of embedded devices [Sch11, Kon14, Ama20, Qur21]. The drawback of this attestation technique is that it is device-specific and usually does not offer any Trusted Execution Environments.

Table 2.2 provides an overview of the currently used trusted computing technologies and their capabilities. In this thesis, we focus on TPMs, Intel SGX, and ARM TrustZone to protect distributed usage control and provenance tracking infrastructures. At the time of this thesis, these three technologies are the most widespread of all and cover all necessary requirements for implementing trustworthy distributed usage control. AMD SEV is a VM-based technology mainly designed to protect virtual machines in cloud computing applications, which can be cumbersome to use for the protection of small distributed usage control components. Intel TDX and ARM CCA have only been announced recently and are not yet widely available in practice. RISC-V based TEEs are used mainly in academia, while device attestation techniques focus only on very resource-constrained devices.

Table 2.2: Comparison of trusted computing technologies.

Name	Isolation	Attestation	TCB	Domain	Year
TPM 2.0	None	Remote	Large	Universal	2014
Intel SGX	Process	Local, Remote	Small	Server	2015
Intel TDX	VM	Remote	Large	Server	2023
AMD SEV	VM	Remote	Large	Desktop, Server	2016
ARM TrustZone	Realm	None[1]	Medium	Embedded, Mobile	2004
ARM CCA	Realm	Remote	Medium	Embedded, Mobile	2022
RISC-V (PMP)	Process	Remote	Medium	Embedded, Mobile	2017
Device Attest.	None	Remote	Large	Embedded	-

[1] Remote attestation is possible with a firmware-level TPM [Wan20].

3 Concept and System Design

This chapter presents a generic design for a trustworthy distributed usage control and provenance tracking system. Section 3.1 provides an overview of the current state of the art regarding trustworthy usage control and provenance tracking infrastructures. In section 3.2 we propose a suitable remote attestation concept for decentralized usage control architectures and present our trustworthy system design. Section 3.3 provides an attacker and a trust model for the described attestation concept, on which we subsequently base the security analysis of our system design (section 3.4). Finally, in section 3.5 we discuss some possible design alternatives, before ending the chapter with a brief conclusion.

Some of the results presented in this chapter have been partially published in previous research papers. In [Wag19b] we analyzed the challenges and security requirements of distributed usage control systems that are protected by trusted computing technologies. Furthermore, in [Wag21a] we discussed possible system architectures for joint usage control and provenance tracking systems, and evaluated existing attack vectors.

3.1 State of the Art

In this section we provide an overview of existing proposals to ensure the trustworthiness of usage control and provenance tracking systems, and discuss their limitations. We identify four different technical approaches to establish trust in and protect the integrity of usage control and provenance tracking components.

3.1.1 Certification Processes

A relatively simple method of establishing trust in usage control and provenance tracking components is to define a suitable certification process. This works by having a trusted Certification Authority (CA) evaluate the software components and the operational environment of a usage control and provenance tracking system. If the evaluation result is positive, the CA issues a digitally signed certificate asserting the security compliance of the system. This certificate usually includes information about the system operator's identity, the operated components, and the applied security mechanisms to protect the system infrastructure. Potential data providers can then verify the certificate to decide if the remote system is trustworthy and sufficiently protected before authorizing a data transfer to it. The certificate also authenticates the public key that data providers can use to securely communicate with the certified usage control and provenance tracking system. One example of such a certification process is implemented by the International Data Space (IDS) [Ste19]. The IDS certification process establishes confirmed and verifiable information about IDS connector systems and the companies operating them [Hub22]. It also ensures that certified IDS connectors correctly implement the IDS standards, including usage control and provenance tracking.

While certification is a relatively simple solution from a technical point of view, there are some clear disadvantages of this approach. First, a certification process assumes the existence of a global Certification Authority that is known and trusted by all system participants. This is not always the case, especially in decentralized and distributed environments. Furthermore, the exact evaluation process used by the CA must be specified in detail and agreed upon by all participants to make the resulting certificate meaningful and descriptive. This usually requires a standardized reference model describing the expected features and behavior of system components, which is not always available either. Finally, this approach only asserts the system integrity at the time of the certification. As soon as the system becomes operational, its integrity can no longer be enforced. Hence system operators could get complacent over time and deviate from the required security procedures, or even willingly manipulate components that have already been certified.

3.1.2 Reputation Systems

While certification ensures the initial trustworthiness of system components by organizational and cryptographic means, reputation systems instead focus on deriving the trustworthiness of system components based on their behavior. To achieve this, reputation systems continuously collect direct and indirect evidence about the past behavior of system components [Has17]. The collected evidence is then used to calculate an aggregated reputation score that gives users an estimation about the reliability and integrity of the different system components. This mechanism has been proposed for establishing trust in usage control systems. Yang and Cemerlic integrate a Dirichlet-based reputation system into the UCON_{ABC} usage control model [Yan09]. This allows participants to join the usage control system only if their reputation is high enough. Alnemr et al. introduce the concept of reputation objects for usage control, which allows to specify and evaluate reputation in specific contexts, instead of globally [Aln10]. Baldini et al. [Bal13] and Neisse et al. [Nei15] present usage control frameworks for "Smart City" applications, which include a trust model based on reputation evidence. Finally, Truong et al. describe a comprehensive trust service platform for usage control systems that includes collected reputation evidence and trust recommendations, as well as knowledge based on evaluations conducted by trust brokers [Tru16a].

Even though the presented proposals provide a notion of reputation and trust in usage control systems, several issues still remain unsolved. One issue of using reputation systems for trust estimations is that they require a significant amount of input data to make any prediction with acceptable reliability. However, especially in distributed usage control systems it is necessary to conduct sufficient integrity verifications *before* any valuable data are being released. Furthermore, most reputation systems rely on receiving binary assessments (cooperates vs. defects) about the behavior of supervised components. However, for many usage control operations it is unclear how an honest component could be distinguished from a malicious one purely based on their responses. Finally, reputation systems do not achieve a continuous enforcement of correct system behavior either. Malicious system owners could simply manipulate a well-reputed system component after the critical data have already been released.

3.1.3 Distributed Ledgers

Another useful tool for securing usage control and provenance tracking applications are blockchain-based distributed ledgers. Distributed ledgers essentially provide a decentralized database that cannot be manipulated against the majority of honest ledger participants. This property can be exploited to secure the integrity and trustworthiness of collected provenance information without having to resort to a centralized trusted authority. In recent years, there have been many proposals for blockchain-based provenance tracking frameworks that are focusing on different application scenarios. Neisse et al. propose a provenance tracking framework that is based on the Ethereum distributed ledger, which allows to log the sharing and usage of personal data across different domains [Nei17]. Liang et al. implement blockchain-based provenance tracking for files stored in cloud environments [Lia17]. Cui et al. use the Hyperledger platform to track the provenance of electronic parts for trustworthy supply chains [Cui19]. Ramachandran and Kantarcioglu implement a provenance tracking framework for scientific data by defining a distributed voting process based on the Ethereum ledger [Ram18]. Sigwart et al. [Sig19] and Javaid et al. [Jav18] rely on Ethereum-based smart contracts to implement provenance tracking frameworks for IoT use cases.

While these proposals show that blockchain-based methods are useful to protect the integrity of provenance information, there are also drawbacks with this approach. Distributed ledgers only ensure that the provenance information is not being maliciously manipulated after it has been stored, but they cannot enforce the correct collection of the provenance data in the first place. Also, distributed ledgers are by nature a public data structure, which results in confidentiality issues especially when collecting the provenance of personally identifiable information. Such applications require the additional definition of suitable anonymization strategies, which can protect the confidentiality of the tracked data without destroying important provenance information.

Even though not as useful as for provenance tracking, blockchain-based mechanisms can also help to secure usage control applications. Cirillo et al. propose a usage control framework that relies on the Hyperledger platform to synchronize and record policies as well as usage control decisions between multiple parties [Cir20]. This enhances the transparency of the usage control system by making usage control decisions verifiable for the policy issuer. However, the framework still does not provide any technical enforcement of the recorded usage control decisions. Furthermore, logging every usage control operation on the distributed ledger results in an increased enforcement latency.

3.1.4 Trusted Computing

The main drawback of the proposals discussed so far is that they do not provide any mechanisms to reliably detect and prevent the manipulation of critical system components by malicious operators. To fix that, trusted computing has been proposed as a technical measure to protect the integrity of usage control and provenance tracking infrastructures. Most existing solutions rely on Trusted Platform Modules (TPMs) for this task. Kyle and Brustoloni implement a TPM-protected usage control module for the Linux kernel [Kyl07]. They utilize the Linux kernel IMA for integrity measurements and provide data confidentiality by encrypting the file system with a key sealed to the TPM. However, their proposal only focuses on a centralized usage control model with a monolithic system architecture. Agreiter et al. [Agr07] and Zhang et al. [Zha08] propose TPM-protected usage control systems capable of cross-domain XACML policy enforcement. For this, Agreiter et al. rely on Java-based communication gateways, while Zhang et al. conduct usage control enforcement by transforming received XACML policies to low-level SELinux policies. Both proposals use a centralized usage control architecture and do not specify any suitable remote attestation protocols. Neisse et al. propose a TPM-protected usage control framework that provides enforcement of OSL policies both inside a message broker and at operating system level [Nei11b, Nei11a]. The authors also specify a remote attestation protocol for TPM 1.2, but still rely on a centralized usage control architecture.

More recently, the International Data Space (IDS) reference architecture relies on TPMs as a hardware-based trust anchor to protect the integrity of IDS connector systems, which usually include both usage control and provenance tracking components [Ott19]. The IDS specifies a TPM-based remote attestation protocol that provides encrypted and mutually attested communication channels for IDS connectors of the highest security level [Bro22]. This protocol is used to validate the integrity of IDS connector systems and transmit both usage control policies and shared data over a secure channel. However, as we show in chapter 4, there are still some unsolved security issues when using this approach to protect distributed usage control systems against potentially malicious administrators.

Besides applying TPMs to usage control systems, there are also some proposals that instead focus on Intel SGX as a protection mechanism. Birell et al. propose the enforcement of usage control policies with monitoring components that are implemented as SGX enclaves [Bir18b]. The authors define three possible system architectures that place the policy enforcement either directly at the data source, remotely at the data processing applications, or at a delegated monitoring component. However, the proposal does not make use of the standard XACML-based distributed reference architecture and instead proposes monolithic enclaves that enforce pre-provisioned policies. Djoko implements an SGX-based usage control framework for cloud storage systems [Djo20]. This proposal defines a logic-based formal policy language that operates on subject and object attributes. It also follows a centralized usage control approach that implements the enforcement point, information point, and decision point all inside a single enclave. In contrast, Meyer zum Felde et al. propose a fundamentally different approach to usage control enforcement in SGX-based environments [Mey21]. The authors define a multi-enclave architecture allowing data owners to approve customized code templates that are instantiated by data consumers. Remote attestation is utilized to verify the integrity of a central management enclave, which then locally attests the consumer's data processing enclaves. Critical data is only released to the attested enclaves if the instantiated code templates match the measurements as approved by the data provider.

Trusted computing has also been proposed to protect provenance tracking infrastructures from malicious tampering. Lyle and Martin describe how TPM-based attestation can be used to ensure the integrity of applications that are collecting provenance data [Lyl10]. In contrast to most TPM-protected usage control systems, the conducted attestations are not immediately validated. Instead, the generated quotes and the respective measurement logs are stored together with the collected provenance information for later verification. Demsky uses TPM-based remote attestation to secure a provenance tracking framework that applies binary-rewriting to automatically track data flows across multiple applications [Dem11]. Taha et al. apply TPM-based remote attestation to protect the integrity of the "Progger" provenance logging tool [Tah15]. Finally, Kaaniche et al. propose a provenance tracking architecture that relies on Intel SGX enclaves for the provenance collection, and on the blockchain-based Hyperledger platform for provenance storage [Kaa20].

3.1.5 Conclusion

In conclusion, there are several approaches to protect the integrity and ensure the trustworthiness of usage control and provenance tracking infrastructures. However, existing proposals still have clear limitations in terms of their scope and security guarantees. Using certification processes or reputation systems has the major drawback that the achieved integrity protection cannot be guaranteed against malicious system operators. Even though blockchains can provide a decentralized database without the need for a single trust anchor, which has been proven useful for provenance storage, they cannot enforce the proper collection of provenance data. Hence, in recent years trusted computing emerged as the most promising approach for protecting usage control and provenance tracking systems. Today, most existing solutions rely on TPM-based remote attestation, including the prominent International Data Space. While there are also a few SGX-based proposals, so far the focus is only on a centralized notion of usage control. To our knowledge there is currently no general solution for a trustworthy *distributed* usage control and provenance tracking system.

3.2 Trustworthy System Design

In this section we build on the presented research to develop a system design for trustworthy distributed usage control and provenance tracking that is based on trusted computing technologies. For this, we first propose a suitable remote attestation concept for distributed usage control systems. Then we present our conceptual system design and describe the necessary interactions between the components. Finally, we discuss the proper handling of measurements during remote attestation and develop a concept for provisioning and authenticating components in distributed usage control systems.

Unlike the previous proposals presented in section 3.1.4, we keep our system design independent of specific trusted computing technologies by assuming an ideal remote attestation protocol that can protect the integrity of usage control components and the confidentiality of transmitted data. Later, in chapter 4 of this thesis, we discuss the application of specific trusted computing technologies and resolve the remaining security issues that occur when using concrete remote attestation protocols.

3.2.1 Remote Attestation Concept

Using trusted computing to secure the integrity and trustworthiness of usage control and provenance tracking requires a suitable remote attestation concept. How this attestation concept looks like depends primarily on the chosen usage control approach. The existing proposals presented in section 3.1.4 all follow a centralized usage control approach. Since in this case all usage control components are deployed on a self-contained computer system, the necessary remote attestation concept is rather straightforward. As fig. 3.1 illustrates, the data provider can simply perform a single remote attestation of the entire usage control system whenever critical data should be shared. After verifying the attested system measurements, data and corresponding usage control policies can be transmitted to the remote system.

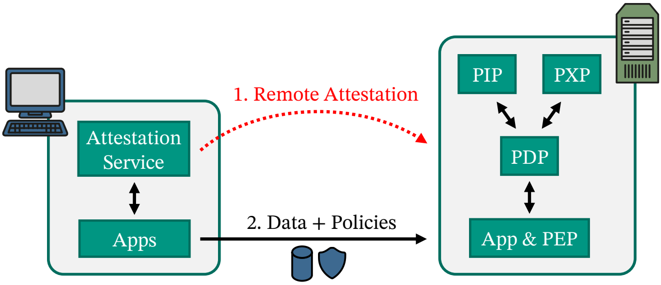
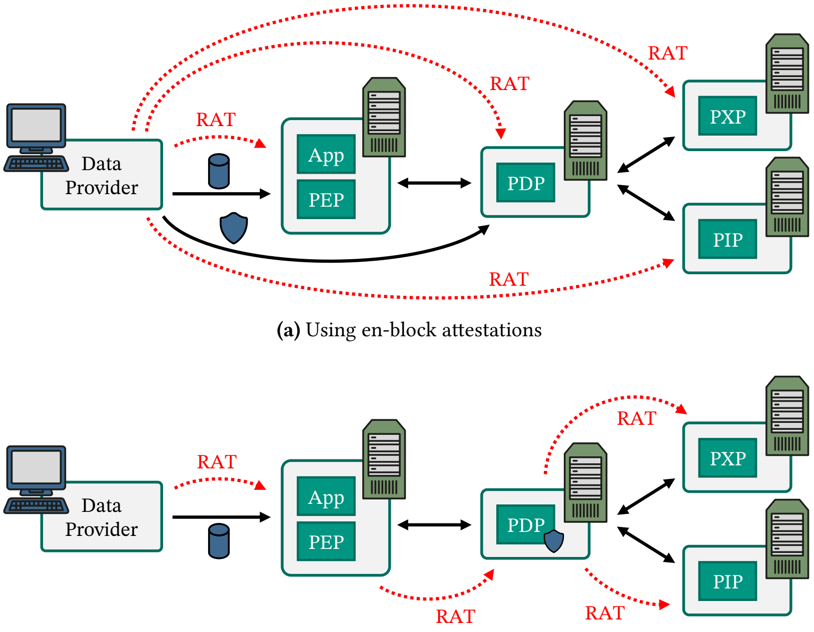


Figure 3.1: Remote attestation of a centralized usage control system.

A distributed usage control system, on the other hand, relies on a *decentralized* system architecture (see section 2.1.2). To establish trust in such a usage control system, we need to define a remote attestation concept that allows the verification of multiple components at once. In principle there are two different approaches to achieve this. Figure 3.2a shows the *en-block* attestation strategy for decentralized usage control systems. For this, the data provider must remotely attest *all* systems that are running usage control components required for the enforcement of the desired policies. Only if all attestations are successful and the attested measurements have been correctly verified, data and protection policies may be transmitted to the respective components. The advantage of this approach is that the resulting trust dependencies are rather simple. The data providers themselves directly verify the integrity and trustworthiness of all remote usage control components. However, there are also two issues with this solution. First, the data providers performing the attestations need to know which usage control components (and by extension which systems) are in fact required for the enforcement of their particular policies. Since data providers usually only communicate with the applications that receive the shared data, they do not have any background knowledge about the topology of the decentralized usage control system. Second, with this attestation concept the data provider only verifies the integrity of all participating usage control components. It remains unclear to the data provider if the communication *between* those remote usage control components is secure as well.



(a) Using en-block attestations

(b) Using transitive attestations

Figure 3.2: Remote attestation of a decentralized usage control system.

The alternative is to establish an attestation concept that uses *transitive* remote attestations, as is depicted in fig. 3.2b. Using trusted computing to create transitive trust relationships has been introduced by Kuntze and Schmidt for the use case of mobile device authentication [Kun06]. We propose using transitive remote attestations to protect the individual components of distributed usage control systems. With such a transitive attestation concept, the data provider only attests the very first contacted remote usage control component. This is usually an enforcement point attached to a data processing application. The attested component then transitively performs its own remote attestations of all additional usage control components that are required for the current task. For example, an enforcement point needs to attest the decision point that it uses for policy evaluation. The decision point in turn may attest additional information points and/or execution points, depending on what the evaluated policy dictates. This establishes a transitive chain of trust that spans the remote usage control system and is always rooted at the data provider. The advantage of this attestation concept is that the initial attester does not need to know which specific usage control components (e.g., PIPs and PXPs) exist and are required for the current operation. Instead, they only need to establish trust in the first component of the usage control operation, which then performs subsequent attestations as required. This approach also solves the second issue of the en-block attestation concept, since the transitive attestations between remote usage control components already provide the required secure communication channels. The drawback of this concept is that the resulting trust dependencies are more complicated than with en-block attestations. Still, due to the clear advantages we rely on transitive attestations for our trustworthy usage control system design. We identify the resulting trust dependencies and discuss the security of this solution in sections 3.3 and 3.4.

3.2.2 Distributed System Architecture

We base our distributed usage control system design on the XACML reference architecture as introduced in section 2.1.1. To achieve trustworthy usage control enforcement and provide provenance tracking capabilities, we extend the XACML base architecture with four additional components. Figure 3.3 shows the resulting system architecture.

Policy Retrieval Point. Similar to the MyDataControl framework [Jun14, Jun22], we include dedicated Policy Retrieval Points (PRPs) into our system design. Our PRPs are responsible for managing and securely storing the currently active usage control policies. Furthermore, they retrieve policies that have been transmitted from remote usage control participants alongside the shared data. During enforcement, decision points can query their PRP for all policies that are associated with the data in question, and hence need to be evaluated. A detailed view of the policy deployment, enforcement, and management process is provided in the following sections 3.2.3 to 3.2.5.

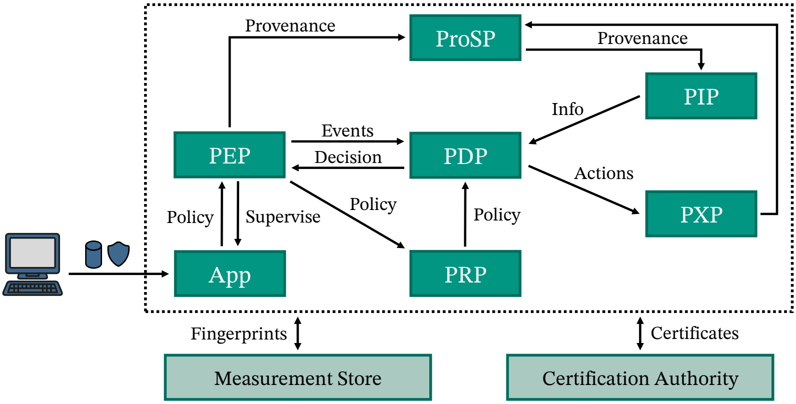


Figure 3.3: Design of a trustworthy usage control and provenance tracking system.

Provenance Storage Point. In order to support provenance tracking in conjunction with distributed usage control, our system architecture provides Provenance Storage Points (ProSPs). Originally proposed by Bier [Bie21], ProSPs are responsible for receiving and storing provenance information that has been collected from data processing applications. We adopt this approach, but additionally integrate the provenance tracking directly into the usage control enforcement process. This is achieved by designing our ProSP to receive provenance information through usage control obligations that are executed by a PXP (see fig. 3.3). That way the collection of provenance information can be regulated with usage control policies. Alternatively, the provenance collection can also be triggered directly by the applications via their respective enforcement points. In addition, the provenance data collected at the ProSP can be referenced in usage control policies via an information point. This allows data providers to specify usage rules that are conditional on the provenance history of controlled data items. The exact method of provenance collection and policy evaluation in our system design is presented in section 3.2.6.

Measurement Store. Furthermore, we introduce the concept of Measurement Stores (M-Stores) in our proposed system design. The M-Store is not a dedicated usage control or provenance tracking component. Instead, the M-Store is responsible for managing and providing trustworthy fingerprints that are used during the validation of conducted remote attestations between the system components. A detailed discussion about the management and dissemination of trustworthy system fingerprints, as well as the deployment of multiple M-Stores for different trust domains, is given in section 3.2.7.

Certification Authority. In addition to conducting remote attestations, a trustworthy distributed usage control system also requires mechanisms to identify and authenticate individual system components. To achieve this, we employ a Public Key Infrastructure (PKI) consisting of multiple Certification Authorities (CAs). The CAs are responsible for provisioning certificates that uniquely identify and authenticate system components during usage control operations. We motivate the use of a PKI and present a suitable component certification and authentication scheme for distributed usage control systems in section 3.2.8.

The design presented in fig. 3.3 only shows the logical system components and their interactions with each other. However, these components must also be instantiated and deployed on actual computer systems. In this regard, we do not make any assumptions about the concrete deployment of the usage control components. Since our system design is decentralized in nature, each component can be independently deployed on separate computer systems, or multiple components together on one machine. In addition to decentralized component deployment, our distributed system architecture allows multiple stakeholders to operate independent usage control systems consisting of the components illustrated in fig. 3.3. To facilitate data sharing, our design allows to seamlessly interconnect these individual usage control systems. This allows the use of PIPs and PXPs from other stakeholders to receive policy information and execute obligations across domain boundaries. We call the set of all instantiated and deployed usage control components that are operated by a certain stakeholder in their own infrastructure a *usage control domain*. A single usage control domain typically includes multiple instances of enforcement points, information points, and execution points. While there is usually just one decision point per usage control domain, in some cases the operation of multiple PDPs can be advantageous. For example, a stakeholder could operate more than one PDP for scalability purposes, or to support the evaluation of multiple policy languages. However, we assume the deployment of exactly one ProSP and PRP at each usage control domain to avoid the fragmentation of provenance information and deployed policies. Figure 3.4 gives an example of a distributed usage control system with two usage control domains.

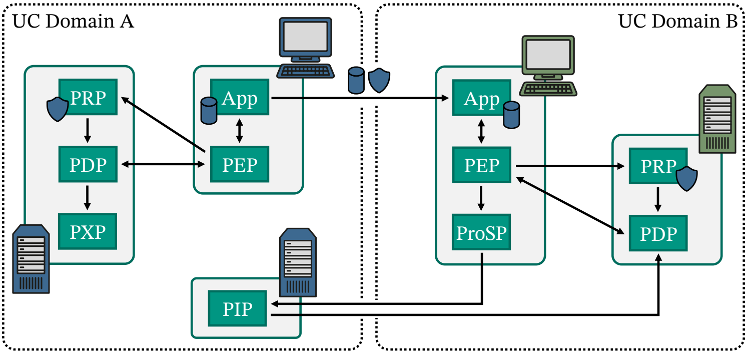


Figure 3.4: Example of a distributed usage control system instantiation with two usage control domains. Remote attestations between the components are not shown.

Furthermore, we employ transitive remote attestations to protect the proposed system design against malicious tampering by dishonest component operators. The transitive attestation concept is implemented by having all system components automatically conduct attestations whenever they need to communicate with another system component. This includes the initial transmission of policies and data, as well as all subsequent communication regarding usage control and provenance tracking operations. The expected system fingerprints of trustworthy code bases are provided by local Measurement Stores, which are then compared with the attested measurements. Together, the conducted attestations establish trust in the integrity of the distributed usage control and provenance tracking system. In addition to verifying the code integrity of the distributed system, certificates are used to authenticate individual components during the usage control enforcement. For the conceptual system design, we abstract from concrete trusted computing technologies by assuming the use of an ideal remote attestation protocol that is capable of establishing secure channels between the attested system components. We assume that the established secure channels perfectly protect confidentiality and integrity against both external attackers and malicious component operators. In section 3.4 we show that the presented architecture cannot be manipulated under these assumptions, which establishes the soundness of our system design. Naturally, for the actual deployment of a trustworthy usage control and provenance tracking system, concrete technologies and remote attestation protocols must still be specified. We discuss the options for concrete attestation technologies and protocols, as well as their limitations and the resulting security guarantees, in the next chapter.

In the remainder of this section, we describe the various system operations and the necessary interactions between system components in greater detail. The concrete implementation of the designed components, as well as the specification of a policy language for joint usage control and provenance tracking, is presented later in chapter 5.

3.2.3 Policy Deployment

In principle there are two different philosophies of handling the deployment of usage control policies [Mio19]. With a *traditional* deployment concept, policies are transmitted directly to a decision point and are valid for the entire usage control domain. This means that whenever an enforcement point requests a usage decision, all deployed policies are evaluated against the observed event. The decision is then derived from the subset of matching policies. A prominent example for a usage control system that uses this deployment concept is the MyDataControl framework [Jun14, Jun22]. The advantage of a traditional policy deployment is that it keeps the number of required policies low. This is because it allows the specification of general usage rules for certain *types* of data, instead of individual data sets. However, the main drawback of this approach is the unclear relation between the protection policies and the shared data. For one, this makes the automated sharing of protected data difficult. For each data set that should be transferred to a remote domain, the data owner needs to identify and re-deploy the subset of policies that are relevant for the protection of this particular data set. Information flow tracking has been proposed to mitigate this problem, since it allows to specify policies that can be evaluated independently of a concrete data representation [Pre11]. However, this complicates both the policy deployment and enforcement process. Furthermore, in terms of protecting distributed usage control enforcement against malicious influences, the traditional policy deployment approach can also lead to security issues. We elaborate on this point later in our security analysis (see section 3.4.1).

A different deployment philosophy that has recently gained traction is relying on *sticky policies* [Mio19]. Sticky policies are directly attached to data and are always transmitted alongside them throughout the entire data life cycle. This makes data transfers between different usage control domains much easier, since there is already a direct association between the data asset and the relevant protection policy. As a result, the sticky policy concept is especially well suited for distributed use cases with a high number of data transfers. Furthermore, this concept simplifies conflict handling during policy evaluation, since usually only one sticky policy must be considered for each asset. On the other hand, using sticky policies has the disadvantage of having to declare separate and independent policies for each data asset. This results in a larger number of policies that need to be managed and transferred (policy duplication). Dynamic policy updates also become more difficult with this approach. Nevertheless, since sticky policies are better suited for distributed use cases, and are beneficial from a security standpoint (see section 3.4.1), we integrate a sticky policy concept into our system design. We leverage the PRP to create a level of indirection that alleviates problems such as policy duplication. In addition, using sticky policies over traditional policy deployments allows us to design *stateless* decision points, which is advantageous in terms of flexibility and scalability. As a result, instead of having to manage a set of active usage control policies at the PDP, our decision points dynamically receive the policies for each asset from the querying PEPs themselves.

Besides adequately distributing policies, a trustworthy usage control system must also ensure the integrity of policies during transmission. Once again, there are two ways of achieving this using remote attestations. A straightforward option is to protect the integrity of policies by deploying them over secure, attested channels. For this, a policy issuer performs a remote attestation of the receiving component, validates the platform measurements, and then transfers the policies over the attested channel. The attestation protocol ensures that a malicious attacker cannot modify the policy during transfer. Afterwards, the attested trusted computing platform is responsible for protecting the integrity of the deployed policies during enforcement. Since here the policies are deployed over secure channels, we call this method *in-band* policy deployment. The alternative to in-band deployment is to transfer policies over other, non-attested channels (*out-of-band* deployment). In that case, the integrity of the policies cannot be directly ensured at the time of the deployment. Instead, the receiving system must now include the deployed policies in the local platform measurements (usually of the PDP). The data owner can then later verify the policy integrity by conducting a remote attestation and checking the measurements at the time of the data transfer. There are examples for both approaches in existing usage control systems. Kyle and Brustoloni [Kyl07], as well as Agreiter et al. [Agr07], use in-band policy deployment over TPM-attested channels. Zhang et al. [Zha08] and Neisse et al. [Nei11b] use an out-of-band policy deployment approach, which separately authenticates the policies over the platform measurements. However, out-of-band policy deployment has the drawback of being rather inflexible, especially when many policies need to be authenticated. It also makes the revocation of deployed policies difficult, since platform measurements can usually only be extended, but not deleted. For these reasons, and because we transmit data and policies together anyway due to the sticky policy approach, we use in-band deployment for our system design.

Figure 3.5 illustrates the resulting policy deployment and data transmission process across two usage control domains, as it is implemented in our proposed system design. First, the data provider attests and authenticates the remote application that should receive the protected information. This ensures that the receiving application is trustworthy and will itself perform the required remote attestations of subsequent usage control components in a transitive manner. Then, both the data and the associated policy are transferred over the attested channel. Usually, the attestation and data transmission is initiated directly by a local application that wants to share data with a service in the remote domain. The remote application then derives a unique asset ID to identify the received data in the local usage control domain. The exact format of this asset ID usually depends on the policy language. For example, ODRL uses Internationalized Resource Identifiers (IRIs) to uniquely identify assets [Ian18b]. The receiving application then forwards the transferred policy to the enforcement point, which stores it at the local PRP and obtains a unique policy ID in return. This policy ID may be a randomly drawn UUID or a cryptographic hash of the policy's textual representation. Finally, the enforcement point saves the association between the data asset and its policy as the tuple (aid, pid). The stored policy ID is later used to identify the policy during the enforcement process (see section 3.2.4).

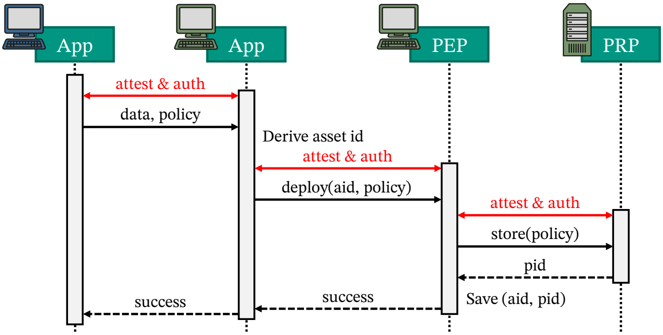


Figure 3.5: Sequence diagram of a cross-domain policy deployment.

Figure 3.5 shows how policies and data are transferred between applications in different usage control domains. This process can be simplified when sharing data internally between applications of the same usage control domain. In that case no explicit policy transfer is necessary, because the policy is already known at the local PRP. Instead, only the policy ID must be given to the enforcement point of the receiving application.

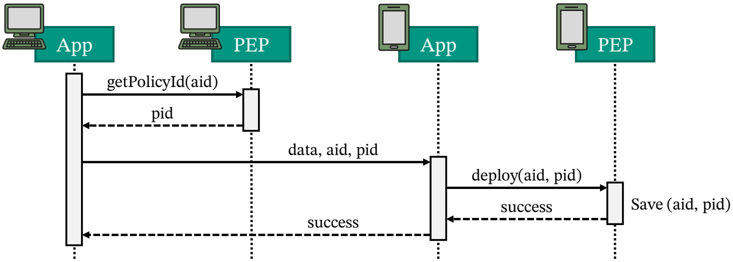


Figure 3.6: Sequence diagram of a domain-internal policy deployment. Remote attestations between components are not shown.

Figure 3.6 shows this process of domain-internal policy deployment. Since in our system design all communicating components always authenticate each other and conduct (mutual) remote attestations, in the following sections we condense the sequence diagrams by omitting the explicit depiction of these steps. We describe the details of the remote attestation and authentication procedure in sections 3.2.7 and 3.2.8, respectively.

3.2.4 Policy Enforcement

After the successful deployment of policies in remote usage control domains, the specified usage rules must be enforced on the shared data. Figure 3.7 illustrates how this process works in our proposed system design. The application that has been remotely attested by the data owner during data transmission is now responsible for notifying its enforcement point about any data usages that take place. This is done by disseminating event objects, which usually contain the name of the event together with parameters that are describing the data usage. The data assets involved in the event are referenced by their unique asset ID. The enforcement point now identifies the (sticky) policy that protects the asset and notifies the decision point about the event. The decision point then uses the policy ID to retrieve the relevant policy from the local PRP and evaluates it against the intercepted event. During policy evaluation, the decision point may contact PIPs for parameter evaluation and PXPs for the execution of obligations. The details of these lookups, especially which parameters and actions can be referenced, is mostly application-specific. We give concrete examples of supported PIP and PXP lookups later in section 5.3. Note that the decision point is stateless in our design, unlike when following a centralized usage control approach, which has benefits in terms of scalability and security (see section 3.4). Once the policy has been successfully evaluated, the decision point transmits the resulting decision back to the enforcement point. Now the PEP either grants, denies, or modifies the event to meet the specified usage restrictions. If any step of this enforcement process fails, for example because a PXP execution or a remote attestation is unsuccessful, the failure gets forwarded all the way to the enforcement point, which then denies the data usage by default. Note that fig. 3.7 only shows the general sequence of the policy enforcement. We give more detailed information regarding the concrete implementation of the policy evaluation and enforcement process as part of our framework description in chapter 5.

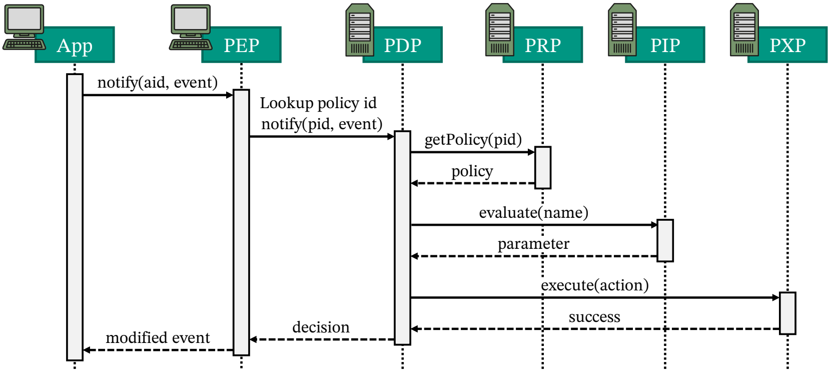


Figure 3.7: Sequence diagram of a policy enforcement. Remote attestations between components are not shown.

3.2.5 Policy Update and Revocation

Besides deploying protection policies to remote usage control domains, data owners also require mechanisms to update the set of usage rules and revoke policies that grant access to previously shared data. In a traditional policy deployment model this is normally achieved with Policy Administration Points

(PAPs), which first authenticate the original policy issuers and then disseminate policy updates or revocation requests in the usage control domain. In case of a sticky policy model, however, updates and revocations need to be handled differently. Since sticky policies are always directly associated with a data asset, policy revocation is achieved by requesting the deletion of the shared data from the receiver's systems. This process is illustrated in fig. 3.8.

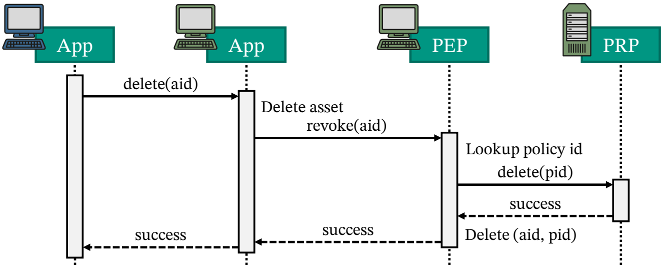


Figure 3.8: Sequence diagram of a policy revocation. Remote attestations between components are not shown.

The original policy issuer (i.e., an application of the data provider) transmits a deletion request containing the relevant asset ID to the data receiver. The application on the data receiver's side then deletes the identified asset and revokes the deployed policy by notifying both its PEP and the local PRP. If the original policy permitted the re-distribution of the shared data to other endpoints or even an entirely different usage control domain, this data deletion process can also be requested transitively across multiple applications. If the policy should be updated instead of revoked, the data owner can then re-deploy the asset after its deletion with a modified policy as previously illustrated in fig. 3.5. Note that the contacted application in fig. 3.8 cannot deny compliance with the deletion request, since it has already been remotely attested and deemed trustworthy during the previous policy deployment. However, a malicious data receiver could still block the deletion request at a network level. Similar issues are well-known also in the traditional policy deployment model. We discuss this further in our security analysis in section 3.4.

A special case of updating policies is the re-distribution of data with a modified set of rules. This is useful if a data provider wants to share information with a certain application using a relatively light rule set, but enforce much stronger usage restrictions when the asset is disseminated further to other applications or domains. To realize this, the deployed policy needs to contain event modification rules (cf. fig. 3.7) that cause the enforcement point to attach a new policy, before allowing the transfer of the asset to further data receivers. We explain the concrete mechanism that our usage control framework provides for this later in section 5.3.3.

3.2.6 Provenance Collection

One goal of our system architecture is to integrate provenance tracking mechanisms into the usage control enforcement process. Bier proposed to use dedicated Provenance Storage Points (ProSPs) to store the provenance information collected in one usage control domain [Bie21]. We build on this approach and employ ProSPs with a similar function in our design as well. The ProSP designed by Bier collects provenance information from a data flow model that is being provided by a PIP specifically implemented for this purpose. Since in our system design we rely on a sticky policy concept associating each data asset with its own usage rules, we do not require a dedicated data flow model as part of the usage control enforcement. Furthermore, our goal is to give data providers direct control over the tracking of their shared data by means of the deployed protection policies. Hence, we need to perform the provenance tracking on the level of usage permissions specified in the deployed policies, as opposed to the level of data flows. This requires a ProSP design that is integrated into the normal usage control enforcement process.

Figure 3.9 shows how we accomplish this in our system architecture. Provenance information can be sent to a ProSP either directly by the data processing applications, or indirectly via the evaluation of usage control policies. Integrating provenance tracking into the usage control enforcement process has the advantage of keeping the data processing applications independent from the provenance tracking mechanisms.

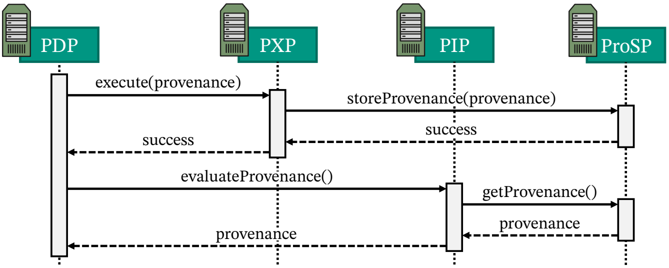


Figure 3.9: Sequence diagram of a provenance collection. Remote attestations between components are not shown.

As fig. 3.9 illustrates, we use PXP actions to populate the ProSP with provenance information during policy evaluation. This allows data owners to specify usage control obligations that are updating the provenance stored at the ProSP whenever a data usage is permitted. If the provenance collection should be enforced without any usage control restrictions, the data owner can just deploy policies that unconditionally allow all data usages. In that case the used PEP does not even require the capability of enforcing usage control decisions. For the provenance collection it is sufficient to only observe the occurring data usage events. Figure 3.9 also shows how the collected provenance information can be retrieved back from the ProSP to be referenced in usage control policies. We use PIPs to make the collected provenance available at the decision point during policy evaluation. This allows data owners to specify usage restrictions that are based on the origin and history of data assets. We present a concrete policy scheme that integrates this type of provenance collection and retrieval into the ODRL usage control language in section 5.3, and give concrete application examples in chapter 7.

3.2.7 Attestation and Measurement Handling

Our proposed system design uses transitive remote attestations to check the integrity of all distributed system components that are relevant for the current usage control operation. As presented in section 2.3.3, a remote attestation protocol provides the verifier with a set of cryptographically signed platform measurements that describe the software stack as well as the used platform configuration of a remote prover. To confirm that the remote platform has not been maliciously modified, the verifier needs to compare the attested measurements with known good values describing a legitimate software stack. As mentioned before, our usage control system design is independent of concrete trusted computing technologies and remote attestation protocols. Nevertheless, we still need to specify how each distributed usage control component knows the list of trustworthy platform measurements that should be used to verify the conducted remote attestations. This is especially challenging with a transitive remote attestation scheme, because *all* of the attestations belonging to a transitive attestation chain must be verified against a list of measurements that is considered trustworthy by the original data provider.

A simple solution to provide verifiers with trustworthy measurements of remote software stacks is to include them into the Trusted Computing Bases, e.g., by directly compiling the expected measurements into the deployed binaries. This solution allows to conduct transitive remote attestations, because the list of trustworthy measurements is itself part of the trusted software stack. Hence, verifiers can check that a remotely attested component also uses trustworthy measurements when conducting subsequent attestations further down the attestation chain. However, the main drawback of this solution is its low flexibility and scalability, especially in distributed systems. Updating the implementation of even a single system component would change the list of measurements and hence require a re-deployment of all system components. Even more importantly, in our use case the code identities of components like enforcement points are often not yet known at the time of system deployment. Furthermore, this solution is not suitable for applications that require support for bi-directional remote attestations, because it would create circular dependencies between the trusted software stacks [Che20]. Due to these issues, directly incorporating trustworthy measurements into the system components is not a suitable solution for a distributed usage control system design.

We solve these problems by incorporating Measurement Stores (M-Stores) into our usage control system design. An M-Store is a repository of trustworthy platform measurements that system components can access to verify remote attestations. Instead of concrete measurements, system components now only have to include an M-Store certificate in their trusted software stacks, which is used for authenticating the correct M-Store during attestation. This introduces a level of indirection that solves the problems with scalability and circular dependencies between software stacks. Measurement repositories are sometimes designed as globally trusted, external parties that disseminate whitelisted measurements provided by system manufacturers [Sch16]. However, using a single system-wide M-Store is not realistic in distributed usage control systems due to the many different participants and stakeholders. It also has the disadvantage of adding an additional trust anchor. To avoid this, we instead design our Measurement Stores to be operated by individual system participants in their own usage control domain. These local M-Store instances then only offer platform measurements that are considered trustworthy in that particular usage control domain. This way no additional global trust anchor is necessary. In practice, usage control participants will populate their local M-Stores with signed component fingerprints provided by the system manufacturer. Nevertheless, participants could also use approaches such as reproducible builds to determine trustworthy component fingerprints independently.

Since every domain has a local M-Store instance, relying on fixed M-Store certificates integrated into the trusted software stacks is now too rigid for our application. Instead, one option is to use the more flexible component configuration to set up a list of M-Store network URLs for each domain, as well as corresponding certificates for authentication. However, in that case the initial verifier at the root of the transitive attestation chain still has to explicitly check that each subsequently attested component is using a correctly configured M-Store for fingerprint validation. Alternatively, the M-Store identity that is deemed trustworthy by the initial verifier can also be directly forwarded to all subsequent components in the transitive attestation chain. This allows to validate all participating components with the “correct” (i.e., local) Measurement Store, while avoiding additional communication and configuration overhead. Figure 3.10 shows how a remote attestation process using this verification method looks like for the example of a cross-domain policy deployment.

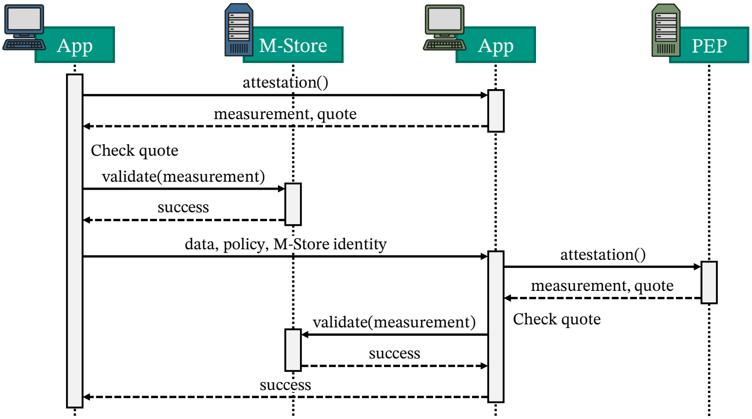


Figure 3.10: Sequence diagram of a transitive remote attestation during policy deployment. The establishment of a secure channel is not shown.

First, the initial verifier (in this case the data provider's application) requests the attestation of the remote component (i.e., the receiver's application). During the attestation, the prover transmits its platform measurements and a signed quote to the verifier. The verifier then checks the received quote and contacts the M-Store in the local usage control domain to validate the attested measurements. Once the attested channel is established, the verifier relays the used M-Store identity to the attested prover, in this case together with the data and the deployed policy. The transmitted M-Store identity consists of the used M-Store certificate and its URL. The remote component then requests its own attestation for the next component in the transitive attestation chain, in this case the remote enforcement point. However, the resulting enforcement point measurements are now also validated using the initial M-Store in the data provider's domain, instead of the receiver's local M-Store. By transitively relaying the M-Store identity, the initial data provider can specify what M-Store should be used for the entire attestation chain, even if the attestations are actually verified in the remote domain. This ensures that all conducted remote attestations are verified with measurements that are trustworthy from the original data provider's point of view.

Note that fig. 3.10 illustrates the remote attestation process on a conceptual level only. For example, the establishment of the attested secure channel between verifier and prover is not shown. We present the concrete design and implementation of the heterogeneous remote attestation protocol that protects our proposed usage control system later in chapter 5.

3.2.8 Component Authentication and Provisioning

Remote attestations allow to measure and verify the code bases and configurations of usage control components. However, this alone is not sufficient to achieve a trustworthy distributed usage control system. We must also be able to uniquely identify and authenticate individual system components. For example, during policy evaluation a PDP has to retrieve information from a *particular* PIP instead of just any PIP with a valid code base. Hence, we need to complement the attested *code identities* of usage control components with unique *component identities* that can be used to authenticate particular component instances. This problem of identifying and authenticating system components has not yet been fully solved in the existing proposals for trustworthy usage control architectures (see section 3.1). However, the MyDataControl framework uses unique component IDs to distinguish specific usage control components from each other. These IDs are registered at a central Policy Management Point (PMP), which can subsequently be queried for connection details of any usage control component [Jun14]. While this solves the problem of component identification, it does not yet provide a suitable authentication mechanism that protects against malicious component operators in distributed usage control scenarios. For example, a malicious system participant could easily operate multiple instances of a particular PIP and alternatingly register them at the PMP under the same component ID.[1] This would allow the malicious participant to bypass policy restrictions that rely on the state of a particular PIP, for example stored access counters.

[1] At the time of this thesis, the MyDataControl API allows to update registered PIPs. See: https://management.dev.mydata-control.de/swagger-ui/index.html#/Components/updatePip *(accessed on 12/08/2023)*.

To prevent these types of vulnerabilities in distributed usage control systems, we include a dedicated component authentication scheme in our trustworthy system design. There are three main requirements for this scheme.

(i) **Component identification:** Each system component must have an unambiguous and discernable component identity in addition to its code identity.

(ii) **Component authentication:** Each component identity must be cryptographically verifiable by other components. No attacker may fabricate or take over a remote component identity.

(iii) **Component uniqueness:** Provisioned component identities must be unique to a single component instance.

To achieve the first requirement, we adopt a similar approach as the MyDataControl framework and identify each deployed component instance by associating it with a Uniform Resource Identifier (URI). This component URI is then used to unambiguously reference individual usage control components both in protection policies and in system configurations. Usually, component URIs are assigned according to a hierarchical and descriptive naming scheme (i.e., as URNs). However, for convenience purposes the identifiers can also be chosen equal to the components' network locators (i.e., the URLs). This has the benefit of allowing users to directly reference network endpoints in configurations and usage control policies. We fulfill the second requirement of cryptographic authentication by provisioning all usage control components with digital certificates. Digital certificates are signed data structures that bind cryptographic public keys to certain identity information. In our case, a certificate attests to a specific component identity by certifying the component's URI. To prevent attackers from forging certificates, their authenticity must be ensured. This could be done by using a centralized trusted party similar to the MyDataControl PMP, which is responsible for binding component URIs to authentication keys. However, for scalability purposes we use a hierarchical Public Key Infrastructure (PKI) instead. PKIs are the most widely used method to issue and verify certificates. Figure 3.11 illustrates how we use a PKI to certify distributed usage control components.

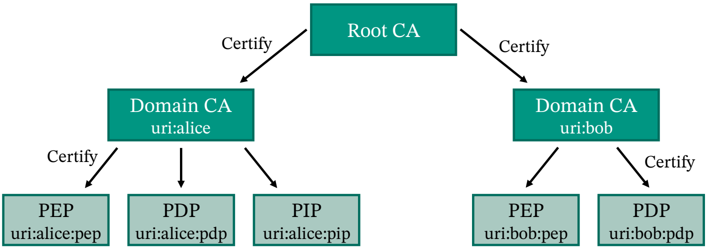


Figure 3.11: Hierarchical PKI with two usage control domains.

The PKI consists of a tree of Certification Authorities (CAs), which are responsible for verifying certificate requests and issuing certificates that are signed with the CA's private key. The *root CA* is located at the base of this tree and must be trusted by all participants. In our case, the root CA does not directly issue signed certificates for usage control components. Instead, it certifies individual *domain CAs*, which are operated locally by the various usage control system participants. Each domain CA is then in turn responsible for certifying the usage control components of one particular usage control domain. That way new usage control components can be certified locally without having to communicate with the root CA. Furthermore, domain CAs establish a hierarchical naming scheme by enforcing that every certified component identifier must either be prefixed by the local domain name (if URNs are used), or that its host name is part of the local sub-domain (if URLs are used). Components can then be authenticated by verifying the component's certified URI as well as the certificate chain to the root CA.

Finally, the third requirement demands the uniqueness of provisioned component identities. This means that a certificate for a particular component URI must be issued to exactly one component instance and may not be used by any other component for authentication (not even by one with the same code base). While this property is usually not relevant for standard PKIs, in our case the uniqueness of component identities is necessary to ensure proper policy enforcement. If multiple component instances could authenticate themselves under the same URI, the evaluation of policies would not be well-defined anymore. We solve the problem of component uniqueness by implementing the domain CAs such that they issue the certificate for a certain component URI only once. For this, the domain CA remembers all issued certificates and denies any further certificate requests for the same URI by any other component. Only the original certificate holder is allowed to request a new certificate for that particular URI, e.g., when the old certificate expires. The same policy is also used by the root CA when certifying domain CAs to ensure the operation of a single CA in each usage control domain. Furthermore, to prevent malicious system participants from bypassing these certification rules, we protect the local domain CAs with trusted computing technologies as well. During the initial registration process, the root CA remotely attests each domain CA before issuing the intermediate certificate, to ensure that the domain CA properly enforces the uniqueness property. Finally, all certified components are required to keep the private key associated with the issued certificate confidential by protecting it with the underlying trusted computing technologies.

Figure 3.12 illustrates the resulting certificate provisioning and subsequent component authentication process. Whenever a domain CA launches for the first time, it generates a new asymmetric key pair (sk, pk) together with a Certificate Signing Request (CSR) for this key pair, which contains the local domain URI as well as the newly created public key. Usually the CSR is self-signed to provide a proof of private key possession. The domain CA then transmits the generated CSR to the root CA in order to acquire a signed certificate. This is done over an attested channel to ensure the integrity of the domain CA, and to verify that the private key associated with the CSR is indeed being protected by a TCB. The root CA then checks that the received CSR does not violate the uniqueness rule, i.e., that no CA already exists for this particular usage control domain. Only if this is successful, the root CA issues a new certificate for the requested usage control domain, which is digitally signed with the root CA's private key. Now the local domain CA is ready to itself provision usage control components using the same basic process. For this, all components that are deployed in this usage control domain also create new asymmetric key pairs and transmit corresponding CSRs containing their respective component URI to the domain CA over an attested channel. The domain CA then checks the received certificate requests and the uniqueness of their URIs in the same manner as before. Additionally, the domain CA also enforces a hierarchical naming scheme by verifying that the requested URI indeed belongs to its usage control domain, before issuing a component certificate that is digitally signed with the domain CA's private key. Once all usage control components have requested their individual certificates, the provisioning process of the new usage control domain is complete. Now any (local or remote) usage control component can authenticate another component using a simple challenge-response protocol. For this, the requester transmits a random challenge to the peer and expects a signature under a valid component certificate in return. By verifying the certificate chain to the root CA, the component's URI can be authenticated.

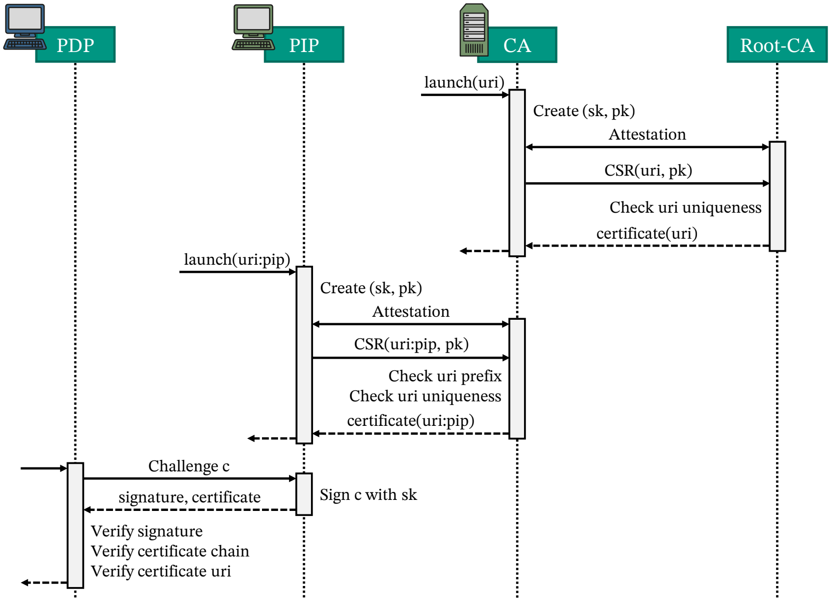


Figure 3.12: Sequence diagram of certificate provisioning and component authentication.

Note that the sequence diagram in fig. 3.12 shows the proposed provisioning and authentication process of usage control components on a conceptual level only. In our proof of concept implementation, presented later in chapter 5, we use a dedicated communication protocol that combines both remote attestation and component authentication into a single handshake.

3.3 Security Model

After presenting the concept for a trustworthy distributed usage control and provenance tracking system, in the remainder of this chapter we discuss the security aspects of our proposal. In this section we establish our security model by identifying the protection goals that our system design should fulfill and developing a suitable attacker model. We furthermore discuss what trust dependencies exist between the usage control components, which need to be covered by remote attestations for all system operations.

3.3.1 Protection Goals

The main responsibility of our system proposal is to enforce deployed usage control policies on shared data assets and track the provenance of data usages across domains. We identify a total of five protection goals that need to be fulfilled in order to achieve this task. Most importantly, data owners expect the system to protect the *confidentiality* of their shared data assets against any unauthorized use. Data confidentiality must be achieved against both external adversaries as well as legitimate data receivers trying to circumvent the imposed usage restrictions. It also requires the prevention of any unauthorized data transfers outside the distributed usage control system, where the deployed usage rules can no longer be enforced. Besides data confidentiality, in many applications the *integrity* of shared data must be protected as well. For example, if data owners share their information with a remote system in order to collaboratively train a common machine learning model, they have a clear interest in preventing any unauthorized modifications of their data in the remote domain. To what extent and purposes confidentiality and integrity of shared data must be protected depends on the concrete use case and is individually specified for each data asset by its respective usage control policy. In addition to confidentiality and integrity, the main protection goal concerning provenance tracking is *transparency*. Transparency is achieved if data owners are reliably notified of any permitted usages of their data in remote domains by the provenance tracking modules. Furthermore, the system must ensure that data receivers cannot credibly deny any of the logged data usages later on. This protection goal is commonly called *non-repudiation*. Finally, the system must also ensure the *availability* of the collected provenance data for later validation by the data owners.

Besides the main protection goals concerning the entire usage control and provenance tracking system, we can also identify additional *component-level* protection goals. This is useful for analyzing the security of single system components in greater detail. Table 3.1 gives an overview of these component-level protection goals. Most importantly, all system components have to protect the *code integrity* of their own software stack. This protection goal ensures that the components always behave as expected during the system operation. While this has been an a-priori assumption in many previous solutions, in our proposal we enforce this protection goal by utilizing remote attestation as a technical measure. In addition to code integrity, the individual system components must also protect the *confidentiality and integrity of usage control metadata* to various degrees. Both PEPs and PDPs mainly have to ensure the integrity of the usage events and decisions that they process. Otherwise, an adversary could tamper with the usage control system by influencing the captured events or the resulting decisions. The confidentiality of usage events may also be a required protection goal, for example in scenarios where the information about data usages is itself valuable. Since policies are usually not confidential, it is sufficient for PRPs to only protect their integrity. Similarly, PIPs also need to protect the integrity of their stored information, as it is used for evaluating policies. Whether the stored information is also confidential depends on the specific use case. In contrast, PXPs usually do not store any confidential information and mainly require code integrity to ensure that all obligated actions are executed correctly. However, depending on the specific application, they may also retain data that is of interest to a usage control participant, for example process log files. In that case the PXP must ensure that this information cannot be manipulated (i.e., provide data integrity), and additionally protect its *integrity against deletion*. Finally, ProSPs always need to protect the stored provenance information against both malicious tampering as well as unauthorized deletion. Similar to PEPs and PDPs, ProSPs may also be required to protect the confidentiality of the stored provenance data, if the information about particular data usages is considered secret.

Table 3.1: Component-level protection goals.

Component	**Confidentiality**	**Integrity**		
	Data	Code	Data	Deletion
Applications	●	●	◐	◐
Enforcement Points	◐	●	●	○
Decision Points	◐	●	●	○
Retrieval Points	○	●	●	○
Information Points	◐	●	●	○
Execution Points	○	●	◐	◐
Provenance Storage Points	◐	●	●	●

Note that achieving the component-level protection goals listed in table 3.1 is a required, but not a sufficient precondition for a secure overall system providing the described main protection goals. For example, in order to achieve a transparent distributed usage control and provenance tracking system, it is necessary to use ProSPs that are capable of protecting the integrity of both code and data, as well as preventing unauthorized data deletion.

3.3.2 Attacker Model

Besides defining the required protection goals, we also need to specify what types of attackers we expect on our proposed usage control and provenance tracking system.

For our security analysis we distinguish three types of *attacker goals*.

- **Curious attacker:** This attacker intends to break data confidentiality by accessing protected data outside the usage restrictions specified by the data owner. The attacker can achieve this goal either by breaking the policy protection or by extracting the data from the usage control system altogether. In cases where the information about data usages itself is considered confidential, we also assume that the attacker wants to capture provenance information and usage events.

- **Modifying attacker:** This attacker intends to manipulate the protected data without detection, thereby breaking the protection goal of data integrity. Furthermore, the modifying attacker also aims to tamper with captured provenance data in order to hide or forge illegitimate data usages. This breaks the protection goals of transparency and non-repudiation.
- **Destructive attacker:** Unlike modifying attackers, this attacker intends to to destroy protected information rather than tamper with it. As before, this attacker can target both shared data as well as the captured provenance tracking information. Hence, this attacker breaks the protection goals of availability and transparency.

In addition to the attacker intentions, we also distinguish three levels of *attacker capabilities*.

- **Network attacker:** This attacker has access to the network that the usage control components use to exchange information. Hence, a network attacker can read, intercept, and modify all traffic between usage control components, but does not have any access to the applications and system components themselves.
- **Software attacker:** This attacker has software-level access to the devices that operate applications and usage control components. A software attacker can inspect unencrypted data and launch new processes, but has no privileged access to the device.
- **Privileged attacker:** This attacker has privileged access to the devices that operate applications and usage control components. A privileged attacker is in complete control of the executed software stack and also has physical access to the device. Usually, privileged attackers are malicious component administrators. However, we assume that even such privileged attackers cannot tamper with the hardware-based trust anchors introduced by trusted computing technologies.

Based on this definition of attacker goals and capabilities, our attacker model consists of four different types of adversaries that we expect on distributed usage control and provenance tracking systems. *External attackers* include all adversaries outside the actual usage control system. These adversaries only have access to the network and generally act as both curious and destructive attackers, since they want to steal valuable data and/or cause damage by deleting them. However, external attackers usually have no incentive to specifically modify protected data. *Curious users* are system-internal adversaries that are mainly interested in accessing protected data outside their usage restrictions. Unlike external attackers, they have software-level access to some of the system components, most often the data processing applications. Curious users generally have no interest in manipulating or deleting data assets. However, they might try to manipulate protection components in order to bypass the usage control protection. In contrast to curious users, *malicious users* also have an interest in modifying or destroying information stored in the system. For example, disgruntled employees could try to tamper with usage-controlled information in order to sabotage data processing applications. Furthermore, malicious users may try to modify or delete captured provenance information in order to cover up illegitimate data transfers or usages. However, just like curious users, they still only have restricted attacking capabilities on the device. User-space malware can be seen as a special type of curious/malicious user with similar motivations and capabilities. Finally, *component administrators* are the most capable adversaries that we expect on distributed usage control and provenance tracking systems. As privileged attackers, they may tamper with all usage control components under their control to try and bypass remote usage restrictions on shared data. We assume that system administrators act on behalf and in the interest of their respective employers. Hence, they constitute malicious adversaries only against the interests of *other* usage control system participants. As such, component administrators are motivated to break the confidentiality of shared data, but usually have no interest in deleting or manipulating the received information. However, they still act as both modifying and destructive attackers against the collected provenance information describing the usages of shared data in their domain.

3.3.3 Trust Dependencies

As third part of the security model, we identify the existing trust dependencies between usage control and provenance tracking components. This is especially important when using a transitive attestation concept, in order to determine which remote attestations are required during the system operation. Figure 3.13 shows the identified trust dependencies as a directed graph.

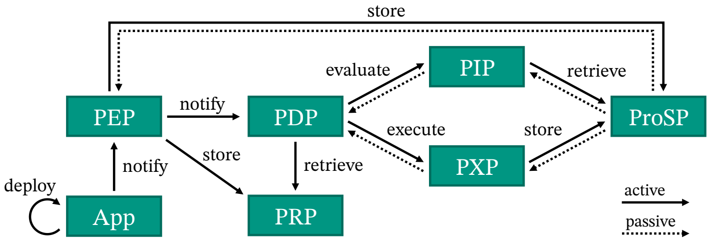


Figure 3.13: Trust dependencies between system components. Dependencies to the Measurement Store and CAs are not shown.

In general, we distinguish between *active* and *passive* trust dependencies. Active trust dependencies directly correspond to the usage control functions that have been illustrated in fig. 3.3. Each time one component requires the support of another component to accomplish its tasks, the resulting trust dependency must be authenticated and covered by a transitive remote attestation. For example, an enforcement point must trust its decision point to provide correct usage decisions, while the decision point itself depends on the subsequent execution points and information points. In contrast to active trust dependencies, passive dependencies are necessary in cases where a contacted component has to trust the requester, instead of the other way around. Hence passive trust dependencies point in the other direction, against the flow of the usage control operations. For example, an execution point needs to know that a requested action indeed originates from a legitimate decision point as part of a policy evaluation process. Otherwise an attacker could contact the PXP to execute any arbitrary actions in the usage control system. Similarly, ProSPs need to determine that the receiver of requested provenance information is in fact a trustworthy PIP, which will not disclose or abuse the requested information. In the implemented system we use bi-directional remote attestations and certificate-based authentication to cover these types of passive trust dependencies. Furthermore, all usage control and provenance components always have an active trust dependency to the local Measurement Store used for fingerprint validation, as well as to their respective domain CAs. To simplify fig. 3.13, these dependencies are not explicitly shown in the illustration.

3.4 Security Analysis

In this section we conduct an informal security analysis of our proposed usage control and provenance tracking system design. Our methodology is to establish the soundness of the design proposal under the assumption that the system components are protected by idealized Trusted Computing Bases (TCBs). Hence, we build our security analysis on four main assumptions.

(i) We assume that all system components (i.e., the data processing applications as well as the usage control and provenance tracking components) are implemented correctly and are free of bugs. This includes the assumption that the enforcement points can reliably intercept data usages and that the applications do not disclose any received information unless permitted by a usage control policy.

(ii) We assume all system components to be secured by a trusted computing technology that protects the code integrity of their TCBs, as well as the confidentiality and integrity of the processed data against all attackers. We also assume the used trusted computing technologies to protect the state integrity of components against all adversaries (i.e., to prevent rollback, reset, and duplication attacks).

(iii) We assume an idealized remote attestation protocol that protects the confidentiality and integrity of transmitted data against all attackers by establishing encrypted, bi-directionally attested, and replay-protected communication channels.

(iv) We assume that all existing trust dependencies between system components, as identified in fig. 3.13, have been authenticated and are covered by the conducted remote attestations.

In the remainder of this section, we identify the attack vectors on distributed usage control and provenance tracking systems, and motivate why our system design achieves the required protection goals under the presented attacker model given the assumptions (i) to (iv). Naturally, real-world remote attestation protocols and trusted computing technologies cannot perfectly guarantee the security of a TCB and the data processed on it. We investigate the application of various *concrete* trusted computing technologies to protect our proposed system design, as well as the resulting security implications and limitations, in the next chapter.

3.4.1 Attacks on Data and Policies

First, we consider direct attacks on the data and protection policies that are exchanged between distributed usage control components. These attacks mainly concern the confidentiality and integrity of usage-controlled data, as well as the transparency of provenance information captured in remote domains via the deployed usage control policies.

Data Interception Attacks. The simplest way of gaining illegitimate access to protected data is with an interception attack. There are generally two places where attackers could intercept shared data. External attackers have access to the network traffic between the system components. However, the remote attestation protocol prevents data interception during transfer by establishing secure communication channels between the applications. Internal adversaries, such as malicious users or administrators, additionally have access to the devices receiving the protected data. Still, direct access to the shared information is not possible because we assume that the deployed trusted computing technologies protect the confidentiality and integrity of all data transmitted to these devices.

Policy Suppression and Modification Attacks. Breaking the confidentiality and integrity of shared data can also be achieved by suppressing or modifying the deployed usage control policies. There are several ways for an attacker to achieve this. First, an external attacker could try to suppress the deployment of a policy by simply blocking the network packets containing the policy during the deployment step. The motivation behind this attack is that if the usage control policy is completely removed during transfer, any data access would be granted without usage restrictions in the remote system. This attack vector also concerns the protection goal of transparency, because blocked policies could contain usage rules that include provenance tracking obligations. However, our system design is not susceptible to this type of suppression attack, because we always transfer policies and data together over a single attested and encrypted connection (sticky policy concept). Assuming the security of the underlying communication channel, removing policies in that way is not possible without blocking the entire data transfer. Furthermore, any failures during the subsequent policy deployment process with the remote PRP (cf. fig. 3.5) will be signaled back to the trustworthy (i.e., attested) enforcement point, which reacts by denying any usage requests for this data. Another possible attack vector concerns the malicious modification of usage control policies. For example, an external attacker could try to add comprehensive usage permissions to policies during transfer, and hence indirectly gain access to protected data against the wishes of the data owner. However, modifying usage control policies during transfer is not feasible either, since they are directly deployed by the data owners and are once again protected by the remote attestation protocol during their transition through the entire usage control system. Finally, even internal attackers are unable to tamper with policies that are already stored at the PRP, since their integrity is secured by the deployed trusted computing hardware.

Policy Deployment Misdirection Attacks. A related but much more subtle attack vector to suppress the deployment of policies exploits the distributed nature of the system architecture. Most usage control architectures offer dedicated policy deployment functionalities for data owners. For example, the MyDataControl distributed usage control framework provides interfaces at the PDP for data owners to deploy their policies in anticipation of any data transfers [Jun14, Jun22]. The problem with this approach is that it separates the policy deployment and the subsequent data transmission into two independent operations. This makes it difficult for the data owner to know at the time of data transmission, if the remote enforcement point indeed communicates with the same decision point that has previously been used for the policy deployment. A malicious system administrator could trick data owners into deploying their policies at a different (albeit legitimate and attested) PDP, which does not serve the enforcement points in question. As a result, even a successful and remotely attested policy deployment may not ensure that the protection policy is adhered to. While this attack vector is certainly relevant for traditional distributed usage control systems, we completely avoid this issue by leveraging a sticky policy concept in our system proposal. Jointly transferring data together with their policies *by design* avoids any ambiguity for data owners between the target of the policy deployment and the target of the data transfer. As a result, in our distributed usage control system the deployed policies always reach the correct PDP/PRP.

Policy Enforcement Prevention Attacks. In addition to suppressing or manipulating usage control policies, attackers can also try to disrupt the policy enforcement process. Since a policy enforcement operation requires the collaboration of multiple distributed usage control components, attackers may impede the enforcement process by tampering with the network traffic between them. A successful policy enforcement prevention attack would break the protection goals of data confidentiality and integrity, as well as transparency. There are several ways in which adversaries can exploit this attack vector. One option is to outright *block* important messages during the policy enforcement process. External attackers could inhibit the event notification messages between the PEPs and the PDP, or the policy transmission from the PRP to the PDP. They could also prevent the PXP from receiving obligated actions. However, in either case our design is not susceptible to blocked messages, because our enforcement points inhibit the usage of data by default if the communication with any required usage control component fails (cf. section 3.2.4). Another attack option would be to *modify* messages during policy enforcement instead of blocking them. Suitable candidates for this would be the decision being returned from the PDP to the enforcement points, or any requested information from the PIPs. Similar to the already mentioned policy modification attacks, this is prevented by the integrity protection features of the used remote attestation protocol. However, in addition to preventing direct message manipulation, it is crucial that the remote attestation protocol also protects against message replays. Otherwise, an attacker could inject a previously captured ALLOW decision into the network traffic for a new event notification. In the next chapter we ensure that the used real-world remote attestation protocols provide suitable integrity protection mechanisms including replay protection. Finally, a third viable attack vector is to *redirect* the network messages concerning policy enforcement to other legitimate endpoints. For example, an external attacker could intercept the policy lookup requests transmitted by a PDP and forward them to another (unmodified) retrieval point in order to influence the enforcement process. Similarly, event notifications transmitted by PEPs could be redirected to a different PDP, or information requests and obligated actions to different PIPs and PXPs. However, our system design is robust against such redirection attacks. Redirecting event notifications to different PDPs has no impact on the enforcement process anyway, because we designed our decision points to be stateless. Any redirection of policy lookups to a different PRP than originally used by the decision point will be detected, because the unique policy ID demanded in the request will not be known to another retrieval point. Finally, requests to PXPs and PIPs cannot be redirected either, because these components are uniquely identified and authenticated by the URI referenced in the respective policy (cf. section 3.2.8). In all cases, the redirection attempt will be detected as an authentication failure and signaled back to the enforcement point, which denies further data usages by default.

Policy Revocation Prevention Attacks. Similar to the suppression of policy deployments, an attacker could also try to prevent policy revocations. This is a relevant attack vector because policies might contain usage permissions in addition to usage restrictions. Hence, an external or internal attacker could be motivated to maliciously prevent updates or revocations of permissive policies. The easiest way for an attacker to achieve this is to sever the network connection between the data processing applications and the remote data owner. That way no policy update requests can reach the remote usage control domain. Attacks of this nature are usually mitigated by enforcing a heartbeat protocol during enforcement. If no connection with the original data owner can be established after a certain amount of time, all further data usages are inhibited by default. However, in practice such a heartbeat solution tends to be complicated and error prone. Even innocent and transient network disconnects would lead to data locks, which could seriously hamper the legitimate use of data in productive scenarios. Furthermore, attackers could exploit such a heartbeat solution to execute denial-of-service attacks on the distributed usage control system. Finally, the damage potential of this attack vector is also rather limited. Even if policy updates and revocations are completely prevented, the usage control system will keep enforcing the original usage rules on the protected data. The protection policy cannot be removed without a legitimate policy update request by the original data owner (cf. section 3.2.5). Because of these reasons, we choose not to include a heartbeat solution to prevent this type of attack in our proof of concept implementation (see chapter 5).

3.4.2 Attacks on Usage Control Components

Besides targeting network traffic such as transmitted policies and data, attacks on the usage control components themselves must also be considered.

Component Manipulation Attacks. While external attackers are limited to intercepting network traffic, internal attackers such as malicious users or administrators have the opportunity to directly manipulate usage control components. This can be achieved by tampering with the deployed application binaries that are implementing the usage control functionality. For example, malicious users could try to remove the enforcement points bound to data processing applications in order to bypass the usage control enforcement. Component administrators could also manipulate the implementation of the local decision point to always return ALLOW decisions if usage restrictions of remote data providers are concerned. In general, successfully manipulating the implementation of usage control components could break all of the protection goals identified in section 3.3.1. We prevent these manipulation attacks in our system design by applying trusted computing hardware and remote attestation protocols to protect the code integrity of the entire usage control system. As described in section 3.2.1, we use a transitive attestation concept to detect any malicious manipulations of usage control components. Two aspects are important for a secure realization of this attestation concept. First, it must be ensured that all necessary attestations are being conducted during the system operation. We identified the required attestation targets by analyzing the trust dependencies between distributed usage control components (see fig. 3.13). Furthermore, in our proof of concept we enforce the integrity verification of all usage control components by implementing a communication protocol that transparently conducts bi-directional remote attestations (see chapter 5). Second, it is important to ensure that attestation failures are always reliably forwarded back to the beginning of the respective attestation chain, i.e., to the data provider during policy deployment and to the PEP during policy enforcement. There the failed attestations must be handled accordingly. A data provider reacts to attestation failures by denying the policy and data transfer (cf. section 3.2.3), while enforcement points deny the data usage in such cases (cf. section 3.2.4).

In addition, internal attackers could also try to manipulate policy enforcement by tampering with the *configuration* of a usage control component instead of its software stack. For example, the configuration of a decision point includes the URI of the retrieval point that should be used. To prevent the manipulation of component configurations, we always include the configuration in the attested TCB as well. This is feasible, because component configurations usually do not change very often. Finally, we also need to consider the possibility of attackers manipulating component code or configurations *after* the initial chain of remote attestations has been conducted. For example, a component administrator could manipulate a decision point after a remote data provider has already verified the usage control system and disclosed critical data during the policy deployment step. These type of attacks are known as *Time-of-Check Time-of-Use (TOCTOU)* attacks [Bra08]. Even though the used trusted computing technologies can offer a various degree of protection against TOCTOU attacks, we also consider this issue in our system design. For one, we attest the relevant usage control components not just at the time of the initial policy deployment, but also later during the policy enforcement process (cf. section 3.2.4). Furthermore, our system also intermittently conducts re-attestations during the policy enforcement to ensure that the distributed system is still in a correct state. If any of those attestations fail, the responsible enforcement point falls back to a safe state by denying any data usages. More details about the implementation of re-attestation in our proof of concept is discussed later in section 5.2.1.

Component Impersonation Attacks. In addition to manipulating the software stack of deployed components, the impersonation of legitimate component identities must also be considered as an attack vector. For example, an attacker could launch a new PIP instance with a valid code base and then use it to inject invalid information in the policy enforcement process. As discussed in section 3.2.8, this type of attack is prevented by our component authentication scheme. All usage control components are identified by their URI and authenticated using a signed certificate. Domain CAs issue certificates for each URI exactly once, so new component instances cannot impersonate an already existing component identity. Furthermore, the private keys associated with legitimate certificates are protected by the original component instances and are not available outside their TCBs.

Component Reset Attacks. A related attack vector concerning the integrity of usage control components are reset attacks. Instead of launching and provisioning a new component instance as part of an impersonation attack, a malicious administrator could also try to reset an existing component back to a fresh state. Depending on the system implementation, this may be achieved as easily as by deleting the locally saved state blob of a usage control component and then triggering a reboot. A successful reset attack on an existing usage control component could once again severely impact the usage control enforcement process. For example, the attacker could mount this attack against a PIP to get rid of unfavorable attribute values, or even remove entire policies by attacking a PRP. Fortunately, reset attacks are prevented by our component authentication scheme as well. This is because resetting a component back to a completely fresh state also removes the private keys that have been certified during the component provisioning step. As a result, the attacked component loses its unique certificate identity and cannot participate in the usage control enforcement any longer. This ultimately causes the enforcement points to deny any requested data usage by default. Naturally, this argument assumes that the component's TCB cryptographically protects the component state in a way that makes selective data removal impossible. We ensure this property in our proof of concept implementation (see chapter 5).

Component Rollback and Duplication Attacks. Besides resetting an existing usage control component entirely, internal adversaries could also try to roll back the state of a component to an earlier (albeit legitimate) version. For example, if a policy limits data usages based on an access counter stored inside a PIP, malicious users or administrators could bypass this restriction by rolling back the PIP to a previous state with a lower counter. Similarly, attackers could also try to duplicate the PIP instance and then bypass the access counter restriction by routing update messages to one instance and read requests to the other [Mat17]. Another interesting target for rollback and duplication attacks are the domain CAs. If an attacker could duplicate or reset a domain CA to an earlier state, multiple certificates for the same URI could be generated. Unlike with reset attacks, our component authentication scheme alone does not effectively protect against the rollback and duplication of component states. This is because both the component certificate as well as its sealed private key remain intact during the rollback and duplication process. Preventing this type of attack is the responsibility of the underlying trusted computing technology that is protecting the integrity of usage control components. Existing trusted computing technologies offer various primitives that can be used to detect both rollbacks and duplications of sealed data [Mat17]. For the security analysis of our system design, we hence assume that the used trusted computing technologies provide a suitable level of protection against these attacks. In the following chapter we analyze to what extent the concrete trusted computing technologies used for our system implementation can in fact achieve this requirement. Furthermore, in chapter 5 we also develop a combined solution for duplication and rollback protection (RBP) and include it in our distributed usage control framework.

3.4.3 Attacks on Provenance Tracking

Finally, attackers can also target the provenance tracking capabilities of our distributed usage control system design.

Provenance Suppression Attacks. The provenance tracking components included in our usage control system design are responsible for logging the history of data usages. As such, they ensure the protection goals of transparency, non-repudiation, and availability. In order to disturb the provenance tracking process, network attackers can block the provenance updates that are transmitted throughout the distributed system. This can be achieved either by severing the connection to the ProSPs, or by intercepting the provenance obligation messages to the PXPs if the provenance tracking is controlled by a usage control policy (cf. section 3.2.6). However, our system design protects against these provenance suppression attacks. In both cases the enforcement points that initiated the provenance updates will notice the failed communication with either the ProSP or the PXP. As a result, the PEPs will deny the requested data usage by default. Hence it is ensured that no data usages can occur without the proper tracking of data provenance, should it be mandated.

Provenance Deletion Attacks. Besides suppressing provenance update messages, internal attackers such as component administrators can also delete captured provenance information at their own ProSPs. Similarly, malicious administrators could remove stored log messages at local PXPs that have been executed as policy obligations. Protecting against such deletion attacks is difficult, especially when dealing with internal attackers. The integrity protection functionalities of the used trusted computing technology do not prevent these attacks. Even though sealed data cannot be read or modified outside the TCB, internal attackers with access to the platform storage can always delete it. Rollback protection primitives can help to detect the data deletion, but the captured information will still be irreversibly destroyed, thereby breaking the protection goals of transparency, non-repudiation, and availability. There are some previous proposals to mitigate deletion attacks on provenance data. Bier distributes captured provenance information across multiple stakeholders by using Provenance Dissemination Points (ProDPs) in addition to the local ProSPs [Bie21]. As presented in section 3.1.3, several provenance tracking systems instead rely on distributed ledgers for this task. While these solutions solve the problem of deletion attacks, they also add a layer of complexity and performance overhead to the provenance tracking. In our system design, we choose a simpler approach to deal with this challenge. We do not try to prevent the deletion of provenance information in potentially hostile environments or disseminate threatened data across multiple stakeholders, but instead store them inside a trusted domain where no destructive attackers are expected in the first place. For example, if we fear that a remote participant will delete the captured provenance information about our shared data, we use a protection policy that demands the use of a ProSP in our own trusted domain. Similarly, we can also demand the use of a local PXP if we expect destructive attackers on the remote PXPs. The trustworthy usage control system is then responsible for enforcing this policy. While this approach greatly simplifies the handling of captured provenance data and avoids complicated dissemination techniques, it requires data providers to consciously choose which usage control components to use. To facilitate this, we must enable policy authors to easily determine if a certain policy can fulfill its protection goals, or if it relies on PXPs or ProSPs in a dangerous domain. We solve this issue in chapter 6 of this thesis by developing a trustworthiness score that can fulfill this requirement. Later, in chapter 7, we then evaluate the effectiveness of the developed score to identify system states that are vulnerable to provenance deletion attacks.

3.4.4 Summary

Our design proposal for distributed usage control and provenance tracking relies on trusted computing as building block to establish a trustworthy infrastructure. In this section we showed that our design is robust against the expected attacks on such a distributed infrastructure. The integrity of all distributed components, as well as their individual configuration, is secured by a transitive remote attestation concept. Any direct network interception of data and policies is prevented by establishing secure communication channels between attested endpoints. Our design proposal is also robust against attackers that are blocking the network messages during usage control enforcement or provenance tracking. Such attacks always result in the affected enforcement points denying the usage of the concerned data by default.

Table 3.2: Summary of identified attack vectors and mitigations.

Attack	Attacker	Protection Goals[1]					Mitigation
		C	I	T	N	A	
Data Interception	All	●	●	○	○	○	RAT, TCB
Policy Blockage	All	●	●	●	○	○	Deny usage
Policy Modification	Internal	●	●	●	○	○	RAT, TCB
Deployment Misdirection	Admin	●	●	●	○	○	Sticky policy
Enforcement Blockage	All	●	●	●	○	○	Deny usage
Enforcement Modification	All	●	●	●	○	○	RAT
Enforcement Redirection	All	●	●	●	○	○	Authenticat.
Revocation Prevention	All	●	●	○	○	○	None
Component Manipulation	Internal	●	●	●	●	●	RAT, TCB
Component Impersonat.	Admin	●	●	●	○	●	Authenticat.
Component Reset	Admin	●	●	●	○	○	Authenticat.
Component Rollback	Internal	●	●	●	○	○	TCB, RBP
Provenance Suppression	All	○	○	●	●	●	Deny usage
Provenance Deletion	Admin	○	○	●	●	●	Trust domain

[1] **C**onfidentiality, **I**ntegrity, **T**ransparency, **N**on-repudiation, **A**vailability.

Furthermore, we apply a PKI-based component authentication scheme to prevent impersonation and message redirection attacks. Finally, deletion attacks targeting stored provenance and logging information are prevented by allowing data providers to move this information into trusted domains. Table 3.2 summarizes all identified attack vectors and the implemented mitigations.

3.5 Design Alternatives

In this section we briefly discuss the advantages and drawbacks of possible design alternatives regarding two specific sub-problems of the trustworthy distributed usage control architecture presented in this chapter.

Alternative policy enforcement methods. In section 3.2.3 we motivated the advantages of the sticky policy concept for our trustworthy usage control infrastructure. Consequently, our resulting system design relies on the joint deployment of assets together with associated usage control policies, which are then evaluated at a stateless Policy Decision Point before each data usage. However, the sticky policy concept can also be implemented in alternative ways. One possibility is to leverage cryptographic means for this purpose [Mio19]. Technologies such as *attribute-based* (ABE) and *identity-based* (IBE) encryption schemes allow data owners to encrypt both the critical data, as well as their policies, with a special key that is bound either to certain domain-specific attributes or some identity information [Mio19]. A receiving system can then decrypt the information together with the usage rules only if these additional key constraints are met. Furthermore, there are also specialized techniques for the enforcement of sticky usage control policies available. The *Degree (D°)* system provides a domain-specific programming language, which allows to embed usage control policies directly into executable files [Bru18]. When the resulting D° binaries are executed, they dynamically enforce the defined usage restrictions on the processed data during runtime [Bru21]. One advantage of these approaches is their relative simplicity compared to a dedicated usage control system stack. On the downside, however, they usually offer limited flexibility and suffer from a reduced expressiveness of the supported policy languages. By linking together applications and usage rules, programming-based enforcement approaches such as D° also require the deployment of separate application binaries for each usage control scenario, as well as re-compilations on policy updates. Using cryptographic mechanisms, on the other hand, does not consider the problem of providing a trustworthy execution environment for secure policy evaluation and enforcement. Our approach avoids these issues by using a standard – albeit extended – decision point implementation (see section 5.3), which is then protected by trusted computing hardware against manipulations and attacks.

Consolidation of code and component identities. In section 3.2.8 we describe an authentication scheme for our distributed usage control architecture, which introduces certificate-based *component identities* in addition to the remotely attested *code identities*. While this design choice appropriately separates the concerns of two different authentication goals, it is also possible to consolidate both identities into a single entity. For example, this can be achieved by including not only the component URI, but also an attestable description of the component's TCB into the provisioned certificate. Since our domain CAs conduct explicit remote attestations during the certificate provisioning anyway, we can add this to our system design rather easily. When additionally provisioning the component private keys as *implicit* attestation keys (see section 4.2.2 for more details), both identities could even be simultaneously verified by authenticating the component against the resulting certificate. The main advantage of this approach is that it saves the additional challenge-response handshake, which separately authenticates the component identity during the establishment of the secure communication channel (see fig. 3.12). However, as we will describe later in section 5.2, the secure communication protocol used in our proof of concept is designed to support multiple endpoint identities anyway. As a result, this modification would not significantly improve either our system architecture or its implementation. In addition, combining both identities would also complicate the provisioning and update process of component certificates. Because of these reasons, we keep both identities conceptually separated in our proof of concept.

3.6 Conclusion

In this chapter we achieved research contribution RC1 by presenting and analyzing a technology-independent design for a trustworthy distributed usage control and provenance tracking system. Our design is centered around a sticky policy concept to avoid any ambiguity between the targets of policy deployments and data flows. Furthermore, we integrate provenance tracking into the usage control enforcement process by leveraging ProSPs as additional distributed system components. Our system design also includes several protection mechanisms ensuring the reliable enforcement of shared usage rules even against malicious component operators in remote domains. Most importantly, we propose a transitive remote attestation concept that dynamically verifies the code integrity of all relevant usage control and provenance tracking components. The established transitive attestation chains are always rooted either at the original data provider, or at enforcement points requesting a usage decision or provenance operation. In addition, to prevent adversaries from influencing the usage control enforcement by impersonating system components, we also include a certificate-based component authentication scheme in our design. If either the remote attestation or authentication of any required system component fails, the responsible enforcement points will always fall back to denying the requested data usages by default.

In the second part of this chapter, we conducted a comprehensive security analysis of the developed system design. In our analysis, we first identified five main protection goals that the usage control and provenance tracking system should fulfill, as well as four additional protection goals for individual system components. Furthermore, we presented an attacker model that distinguishes external network attackers from internal attackers of various capabilities, such as curious/malicious users and component administrators. During our security analysis we found that the confidentiality and integrity of shared data is threatened mainly by message interception and blockage attacks on the usage control and provenance tracking process. Furthermore, the required protection goals are also jeopardized by adversaries manipulating or impersonating system components. We show that our system design is robust against these attacks by virtue of the proposed attestation and authentication concept, as well as the underlying trusted computing technologies. However, due to the limited attack vector and the drawbacks of countermeasures, we choose not to mitigate the malicious prevention of policy revocations. Finally, destructive attackers such as component administrators could subvert the protection goals of transparency, non-repudiation, and availability by deleting stored provenance information. Since a technical protection measure proves difficult, we mitigate this attack vector by moving imperiled components, such as ProSPs and PXPs, into trusted domains. Later, in chapters 6 and 7 of this thesis, we also develop and evaluate a trustworthiness estimation method that allows policy issuers to verify the effectiveness of this mitigation against deletion attacks in the current system state.

To summarize, in this chapter we laid the conceptual foundation for our desired trustworthy usage control and provenance tracking framework by developing and evaluating an appropriate distributed system design. Since our proposal allows the realization of flexible and generic use cases while protecting the shared data against attacks even in remote domains, we have achieved goal 1 of the thesis objective. However, so far we kept our system design independent of concrete trusted computing technologies and conducted our security analysis under the assumption of an idealized remote attestation protocol. Hence, we still need to identify to what extent *real* remote attestation protocols and their underlying trusted computing technologies fulfill the security requirements that are necessary to reliably protect our system design. Most importantly, the used trusted computing technologies must be able to protect the confidentiality and integrity of data on the trusted platform. Furthermore, as we discussed in our security analysis, our design also requires suitable mechanisms to prevent rollback and duplication attacks. The remote attestation protocol must be able to establish mutually attested and encrypted communication channels that offer strong replay protection. In the following chapter we instantiate our proposed remote attestation concept with real trusted computing technologies and attestation protocols, concretely TPMs, Intel SGX, and ARM TrustZone.

4 Technical Enforcement

In this chapter we develop technical mechanisms that are suitable to enforce distributed usage control and provenance tracking based on trusted computing technologies. For this, in section 4.1 we identify the security requirements that both the remote attestation protocols as well as the underlying trusted computing technologies need to fulfill. In sections 4.2 to 4.4 we then discuss the security properties of TPMs, Intel SGX, and ARM TrustZone, and analyze the existing proposals for remote attestation protocols on these platforms. We uncover that the currently used TPM-based remote attestation protocols have security issues when protecting distributed usage control infrastructures. To mitigate this, we develop and evaluate a remote attestation protocol that conducts a TPM-internal key exchange. Furthermore, we also describe a trusted boot process for ARM TrustZone platforms that allows to conduct measurements both in the normal and the secure world of the device. In addition, enforcing usage control and provenance tracking in distributed environments requires a remote attestation protocol that allows to interconnect trusted platforms that are protected by different technologies. To accommodate this, in section 4.5 we extend an existing key exchange protocol with support for heterogeneous remote attestations. Finally, in sections 4.6 and 4.7 we discuss some alternative attestation approaches and close with a brief conclusion.

Some of the contributions presented in this chapter have been partially published in previous research papers. In [Wag18a] we analyzed the capabilities and limitations of TPMs and Intel SGX to enforce usage control policies. In [Wag20] we presented a first version of our remote attestation protocol featuring a TPM-internal key exchange. Finally, in [Wag22c] we presented an initial concept for a heterogeneous remote attestation handshake between TPMs and Intel SGX enclaves.

4.1 Security Requirements

During the security analysis of our proposed system design (see section 3.4), we made several idealizing assumptions regarding the technical protection mechanisms that our remote attestation concept relies on. In this section we translate these assumptions into concrete security requirements for the used remote attestation protocols, as well as the underlying trusted computing technologies. These security requirements must be fulfilled in order to sufficiently protect the usage control and provenance tracking components of our distributed system design against the identified attack vectors. We have already described the attacker model for our scenario in section 3.3.2. Most importantly, the strongest adversary expected to try and break the presented security requirements are malicious component administrators.

First, we identify five security requirements T1 to T5 for the underlying trusted computing technologies.

(T1) **Code integrity:** The trusted computing technology must offer a measurement process that unambiguously establishes the code identity of the Trusted Computing Base (TCB). Furthermore, there must be a mechanism to protect the integrity of the measured code identity against malicious tampering by internal attackers. This serves as the technological basis for the remote attestation.

(T2) **Data protection during storage:** The trusted computing technology must be able to protect the confidentiality and integrity of data that are stored outside the TCB (e.g., on the file system).

(T3) **Data protection during processing:** The trusted computing technology must be able to protect the confidentiality and integrity of data while they are being processed on the trusted platform.

(T4) **Duplication protection:** The trusted computing technology must offer mechanisms to detect or prevent the duplication of the trusted platform and its protected components.

(T5) Rollback protection: The trusted computing technology must offer mechanisms to detect or prevent the rollback of the trusted platform and its protected components.

Furthermore, we define five additional security requirements R1 to R5 for the remote attestation protocols. As described in section 2.3.3, remote attestations are conducted between two network endpoints. During the attestation process, the *prover* endpoint relies on the underlying trusted hardware to convince the *verifier* of its TCB integrity. There are some commonly used security requirements for remote attestation protocols available in the literature [Wag20, Ban21], which we adopt and extend for our use case.

(R1) TCB authentication: The code identity of the attested TCB must be unambiguously determined during the attestation handshake. No adversary (be it external or internal) must be able to impersonate or forge the code identity of a particular TCB. The authentication property convinces a verifier of the attested platform's code base.[1]

(R2) Mutual attestation: A single protocol handshake must be able to authenticate both endpoints simultaneously.

(R3) Replay protection: The attestation protocol must ensure the freshness of the conducted attestation by preventing dishonest provers from replaying previously intercepted information.

(R4) Secure channels: The attestation protocol must protect the confidentiality and integrity of transmitted data against interception and malicious tampering by internal attackers. This is usually achieved by establishing a shared secret between both endpoints that is bound to the authenticated code identities (i.e., that is known only to the mutually attested TCBs).

(R5) Forward secrecy: Disclosing a long-term secret used to authenticate the attestation and/or the secure channels must not compromise previously established shared secrets.

[1] Note that we do not see the validation of the authenticated code identity (i.e., the *authorization* of an attested TCB) as part of the attestation protocol.

In the following sections we discuss what mechanisms and functions are responsible to fulfill the requirements T1 to T5 on trusted platforms based on TPMs, Intel SGX, and ARM TrustZone, respectively. Furthermore, we analyze to what extent the existing remote attestation protocols can fulfill the requirements R1 to R3. Note that we do not yet analyze published attacks on the security properties promised by the different trusted computing technologies. We discuss this later in chapter 6 as part of our trustworthiness estimation method. The concrete application of these mechanisms and remote attestation protocols to implement a trustworthy usage control and provenance tracking system is described in chapter 5.

4.2 Using Trusted Platform Modules

As introduced in section 2.3.1, Trusted Platform Modules (TPMs) are tamper-resistant hardware modules that are available on many computer systems today. TPMs serve as hardware-based trust anchors and offer a wide variety of different cryptographic functionalities, including remote attestation. Since TPMs are passive co-processors instead of complete execution environments, attesting the integrity of TPM-protected software stacks does not require any modifications to the legacy applications. Because of this, TPMs are a convenient technology for securing distributed usage control and provenance tracking systems. However, as we see in the remainder of this section, there also are some drawbacks when using TPMs for this purpose. In this section we first discuss the security properties of TPMs and identify to what extent they fulfill our requirements. Then we give an overview of existing TPM-based remote attestation protocols and point out their limitations, especially in the light of nonce-data attacks. Finally, we present and evaluate a novel attestation protocol that further improves the achieved security properties of TPM-based remote attestation. For additional information about the TPM structures and commands referenced in this section, we refer the reader to the respective specification documents [Tru19c, Tru19d].

4.2.1 Security Properties

Code integrity. As described earlier in section 2.3.1, TPMs allow to securely collect integrity measurements of the software stack that is being executed on the local platform. While the TPM does not actively conduct these measurements itself, its Platform Configuration Registers (PCRs) can be used to protect the measurement integrity against internal attackers. Even malicious component administrators cannot influence this measurement process, because it is rooted in the trustworthy hardware. Furthermore, they cannot tamper with the measurements after their collection, since the PCRs can only be extended with further entries (cf. `TPM2_PCR_Extend`). Together, these properties of the PCRs are sufficient to ensure code integrity (requirement T1).

Data protection during storage. TPMs allow to securely store confidential information in such a way that they can be decrypted only by the trusted platform itself. This concept is also called *sealing*. Sealing can be achieved by using the TPM's enhanced authorization capabilities to create cryptographic keys that are bound to certain PCR values (cf. `TPM2_PolicyPCR`). These keys can then only be used on platforms that are in a pre-defined, trustworthy state. This allows to protect the confidentiality and integrity of data that is stored outside the TCB (requirement T2).

Data protection during processing. TPMs are passively used hardware modules, which do not offer a dedicated Trusted Execution Environment. Hence, TPMs cannot isolate user software or its data from the rest of the system during processing. This results in a very large TCB, which includes the firmware, the operating system, and *all* executed applications. Because of this, TPMs can protect data confidentiality and integrity during processing (requirement T3) only in the sense that we can exclude malicious software on the trusted platform (cf. code integrity). However, data are being processed in plain text on the trusted platform, which results in lower security guarantees and additionally requires a-priori trust in the processing hardware (e.g., the CPU and main memory).

Duplication protection. A naive duplication of a single usage control component on the trusted platform itself is not feasible due to the conducted integrity measurements. Since the TCB on TPM-protected platforms encompasses all executed applications, any duplicated entity will be detected during the attestation. However, the duplication of single components (or even the entire system) to another physical platform must also be prevented. This can be achieved by protecting the component states using sealing keys that have the `fixedTPM` attribute set. This prevents the sealing key (and by extension the component states) from being duplicated to other TPM-protected platforms using the `TPM2_Duplicate` command (requirement T4).

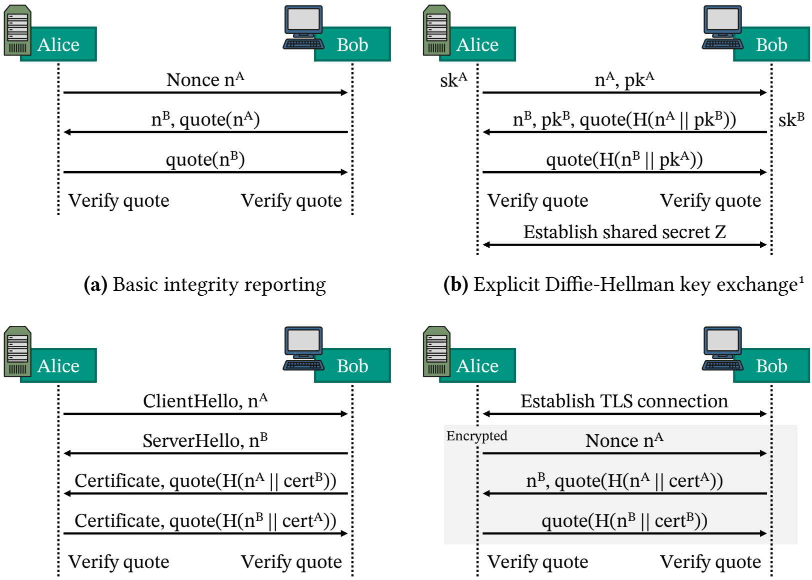
Rollback protection. TPMs offer non-volatile (NV) memory storage that can be used for rollback protection. The simplest solution to prevent rollbacks on a trusted platform is to use *monotonic counters*. These counters are stored in NV memory and may only be updated using the `TPM2_NV_Increment` command. The TPM does not allow to directly set or reset the counter, and its value is preserved through reboots. Hence, a component can protect its sealed state against rollbacks by including the current value of a monotonic counter and updating it each time the state changes. When the component is (re-)launched, it unseals its state and checks if the included value still matches the monotonic counter at the TPM. This allows the component to detect if it has been launched with an old state. The downside of this approach is that only a limited amount of NV memory is available on the TPM chip and it has a tendency to wear out quickly [Seg16, pp. 168–169]. Furthermore, this approach is not resilient against system crashes, because the counter update and storage process is not an atomic operation. If the platform crashes between the NV memory update at the TPM and the respective update in the sealed state, the protected component cannot be launched again. To solve this issue, instead of using monotonic counters, Parno et al. propose to store chains of state hashes inside the NV memory [Par11]. This allows to keep a log of all state updates, which in case of crashes can be used to recover from inconsistent states. With TPM 2.0, these hash chains can be created using the `TPM2_NV_Extend` command, which works very similar to the PCR extend operation. The downside of this approach is that it requires the validation of the entire hash chain instead of just one counter. Strackx et al. improve on this proposal by using additional guarded memory to avoid the expensive NV memory writes during state update [Str14]. However, the required guarded memory is usually only available on custom-tailored hardware platforms. Finally, Strackx and Piessens also propose another rollback protection method, which provides crash resilience even with monotonic counters by implementing them in a way that requires only single bit-flips [Str16]. To summarize, even though TPM-protected trusted platforms do support rollback protection (requirement T5), its correct implementation requires some care.

4.2.2 Remote Attestation Protocols

Besides discussing the security properties of the TPM itself, we must also identify suitable TPM-based remote attestation protocols. More concretely, we require an attestation protocol that fulfills the identified requirements R1 to R5 and that provides its security guarantees not just against network attackers, but also against internal adversaries such as malicious users or component administrators. As introduced in section 2.3.3, remote attestation protocols allow external verifiers to authenticate a trusted platform's code identity. In case of TPMs, the attestable code identity of a trusted platform is represented by its PCR values. Generally, TPMs offer two different variants to attest to code identities [Tru19a]. *Implicit* attestation leverages a signature key that is bound to a certain set of PCRs (e.g., using the `TPM2_PolicyPCR` command). If a trusted platform is able to provide a valid signature of a given challenge under this key, then a remote verifier implicitly knows about the prover's PCR state. Since implicit attestation uses bound signature keys, it is mainly suitable for cases where the trustworthy platform states do not change very often, and user data need to be signed anyway. However, most remote attestation protocols instead use *explicit* attestation by leveraging the `TPM2_Quote` command. This command creates a signed attestation report (i.e., a quote), which explicitly includes a digest of the current PCR values. As such, the created quote serves as cryptographic proof of the attested platform's current software state to remote verifiers. However, for explicit attestation to work it must be ensured that the attestation key used to sign quotes has the `restricted` attribute set. This makes the attestation key usable only on TPM-internal data structures, which prevents a malicious prover from using the TPM to create valid signatures over forged PCR digests.

Since TPMs have been around for a while, there are several proposals for TPM-based remote attestation protocols on offer. The earliest proposals for attestation protocols focus exclusively on reporting the integrity of remote software stacks [Sai04, Cok11]. For this basic attestation concept, the network endpoints simply exchange and then validate the generated quotes (cf. fig. 4.1a). This mutually authenticates the TCBs of both participating endpoints (requirements R1 and R2). Furthermore, the exchanged quotes also contain randomly drawn nonces for freshness (requirement R3). While this attestation approach is very straightforward, it has been found to be vulnerable against masquerading attacks [Stu06]. In addition, these protocols do not establish any encrypted communication channels between the attested endpoints at all (requirements R4 and R5), and hence are not suitable for our use case. In response to these issues, Stumpf et al. proposed to establish secure channels by conducting an ephemeral Diffie-Hellman key exchange (DHKE) that is bound to the attested platform's code identity [Stu06]. This is achieved by concatenating the generated Diffie-Hellman public keys[1] with the received nonce, and using the result as qualifying data for the quote (cf. fig. 4.1b). Both endpoints can then verify that the received public key has indeed been created on a trusted platform with the attested PCR values, which prevents adversaries from intercepting the key exchange. Finally, the authenticated DHKE establishes a shared secret between the attested platforms, which can then be used to derive a symmetric session key. Since the Diffie-Hellman public keys are freshly drawn for each attestation, this approach solves the problem of both secure channel establishment and forward secrecy (requirements R4 and R5). Because of these benefits, a few other attestation protocols have since been based on this proposal as well [Stu08, Gre11, Akr16].

[1] In this thesis we refer to the disclosed and retained parts of the generated Diffie-Hellman tuple as *public* and *private* keys, respectively. This is in accordance with the terminology recommended by NIST [Bar18a, pp. 2–13]. The Diffie-Hellman public and private keys are not to be confused with the symmetric *shared secret* Z that is established by the key exchange.



(a) Basic integrity reporting

(b) Explicit Diffie-Hellman key exchange[1]

(c) Integration into TLS

(d) Post-connection linkage

Figure 4.1: Concepts of TPM-based remote attestation protocols.

To simplify the use of TPM-based remote attestation, several authors propose to integrate attestation evidence into a TLS handshake (cf. fig. 4.1c). This has the benefit of leveraging the underlying encryption and integrity protection of the standard TLS protocol. Goldman et al. [Gol06] and Aziz et al. [Azi14] propose to link the TLS certificates used for the establishment of secure channels to the attestation key with the help of external Certification Authorities. Similarly, Gasmi et al. [Gas07] and Armknecht et al. [Arm08] use a modified TLS handshake that leverages an asymmetric TPM key, which is sealed to the current platform state and authenticated by an external CA. Cheng et al. [Che09b] propose to link the TLS handshake to the attested platform by hashing the TLS pre-master secret into the quote. However, this has the downside

[1] We denote the private and public Diffie-Hellman keys of a party A as sk^A and pk^A, respectively. This is to abstract from a concrete key agreement scheme, e.g., $(\mathrm{sk}, \mathrm{pk}) \coloneqq (x, g^x \bmod p)$ for Finite Field Diffie-Hellman and $(\mathrm{sk}, \mathrm{pk}) \coloneqq (d, dG)$ for Elliptic Curve Diffie-Hellman (see [Bar18a]).

of not providing forward secrecy (requirement R5). Lan et al. [Lan14] propose to link a TLS handshake to the trusted platform by extending the TLS protocol transcript into the PCRs. This has the drawback of changing the PCRs for every conducted attestation, and it also requires a modified TLS handshake. More recently, Walther et al. developed the RATLS protocol [Wal22], which adopts a TLS v1.3 handshake and links the established channel to the attested platforms by including the local TLS certificates into the quotes. Furthermore, Zhou and Zhang [Zho10] propose to use a password-based authentication scheme to establish secure channels in remote attestation protocols.

Finally, Brost develops the *International Data Space Communication Protocol (IDSCP)*, which leverages a different method of linking TLS certificates to the attested code identities [Bro22]. The idea behind IDSCP is to first open a mutually authenticated and encrypted TLS connection between both endpoints using a standard TLS handshake. Then the endpoints exchange their nonces and validate their quotes over the encrypted channel itself. To link the underlying TLS session with the attested platforms, each endpoint hashes the remote TLS certificate received during the handshake into their quote (cf. fig. 4.1d). During validation, the endpoints check that the received quote contains both the expected nonce and the hash of their own TLS certificate. This protects against man-in-the-middle attacks by ensuring that the attested peer is using the right TLS certificate for the channel authentication. The advantage of this post-connection certificate linkage is that it does not require any modification of the underlying TLS handshake. Furthermore, unlike the previously presented proposals, the IDSCP protocol implementation is publicly available under the Apache license.[1]

To summarize, there have been several proposals for TPM-based remote attestation protocols over the recent years. The most promising candidate for our purposes is IDSCP, because it is already actively used to protect usage control infrastructures in virtual data spaces.[2] However, as we see in the next section, there are still some open security issues that need to be solved.

[1] https://github.com/industrial-data-space/idscp2-jvm *(accessed on 12/08/2023).*

[2] https://github.com/Fraunhofer-AISEC/trusted-connector *(accessed on 12/08/2023).*

4.2.3 Attacks on Existing Protocols

In this section we discuss two attack vectors on TPM-based remote attestation protocols, which are especially relevant when protecting distributed usage control components. In combination, we find these attack vectors to be insufficiently addressed by the existing proposals.

Attacks by internal adversaries. Most proposed remote attestation protocols use the common Dolev-Yao threat model, which considers *network attackers* capable of intercepting and tampering with any transmitted messages. Under this attacker model, the goal of a remote attestation protocol is to prevent an adversary controlling the network from impersonating a trusted platform. However, in the case of distributed usage control we also have to deal with *internal attackers* such as malicious users and component administrators, who are motivated to bypass the enforcement of usage rules. These internal attackers are more powerful than network attackers, because they have complete access to the trusted platforms and must be assumed to know any long-term secrets that are not protected by the TPM, such as TLS private keys and passwords. Internal attackers can use these long-term secrets to intercept some of the proposed protocol handshakes, or outright decrypt the communication between the attested trusted platform and the remote verifier. More concretely, the proposals by Cheng et al. [Che09b], Goldman et al. [Gol06], and Aziz et al. [Azi14] allow internal attackers to passively intercept data transmitted over the attested channel by decrypting the TLS pre-master secret during the protocol handshake. IDSCP [Bro22] and RATLS [Wal22] instead use TLS handshakes with perfect forward secrecy, but only authenticate the long-term TLS certificates over the quotes, and are hence vulnerable to man-in-the-middle attacks by internal adversaries with knowledge of the corresponding TLS private keys. Finally, the proposal by Zhou and Zhang [Zho10] is susceptible to a man-in-the-middle attack that uses the long-term passwords to authenticate an internal adversary. Those approaches that use only TPM-internal long-term secrets [Gas07, Arm08] or rely exclusively on ephemeral keys [Stu06, Stu08, Gre11, Akr16] are not vulnerable to this attack.

Nonce-data attacks. Another threat that must be considered when using TPMs to protect trustworthy distributed usage control infrastructures are nonce-data attacks. Nonce-data attacks exploit the improper use of a quote's qualifying data for key authentication [Seg16, pp. 144–146]. Zhou and Zhang describe how this issue can lead to man-in-the-middle attacks in scenarios where trusted platforms are colluding with network attackers [Zho10]. In the case of distributed usage control enforcement, we face a scenario where a variant of this attack is applicable. This is because we must assume that trusted platforms may include additional attestation endpoints, for example as part of third-party data processing applications running on the usage-controlled trusted platforms. These attestation endpoints can be exploited as oracles providing quotes for arbitrary qualifying data. Figure 4.2 shows how a nonce-data attack can be mounted against the enforcement step of a distributed usage control system in such a scenario. We show the attack on an IDSCP handshake,[1] since it is the most recent attestation protocol that is being actively operated in usage controlled environments. However, earlier attestation protocols using qualifying data for key authentication, i.e., [Stu06, Stu08, Che09b, Gre11, Akr16], are also vulnerable to this attack.

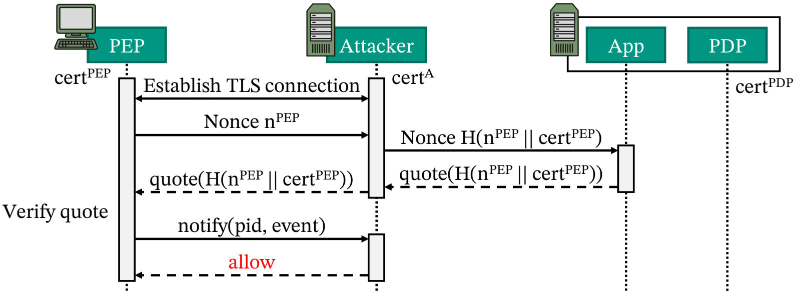


Figure 4.2: A nonce-data attack on IDSCP during usage control enforcement. Only one half of the mutual attestation is shown.

[1] The complete IDSCP protocol is described in [Bro22]. For convenience purposes, we provide the relevant excerpt of an IDSCP handshake in appendix A.

In this example, a data receiver intends to intercept the communication between a PEP and its PDP in order to bypass usage restrictions. For this, the attacker first establishes a TLS connection and then intercepts the PEP's nonce during the subsequent IDSCP handshake. Since the attacker uses an untrusted platform, a valid quote cannot be generated locally. Instead, the attacker calculates the hash of the intercepted nonce and the PEP's TLS certificate, and then uses the resulting digest as a nonce for a separate attestation request to the third-party application running on the same platform as the PDP. The contacted application then uses the TPM on the local platform to create a quote that includes the forwarded hash digest as qualifying data. Note that this hash digest is indistinguishable from a randomly chosen nonce. Finally, the attacker forwards the received quote to the PEP and completes the IDSCP handshake. During quote validation, the PEP checks if the received quote is signed by the correct TPM and if it includes the expected PCR values. Furthermore, the quote must contain the expected nonce and the local TLS certificate (cf. fig. 4.1d). Since all of this is true, the PEP accepts the established TLS channel and the attacker is considered to be authenticated. Now the attacker can easily bypass any usage rules by answering all evaluation requests with an `ALLOW` decision. Note that for this attack to succeed, we do not even need to assume an internal attacker with access to the TLS long-term secrets. The attacker can just use any valid TLS certificate cert^{A}, which is accepted by the PEP, to establish the initial communication channel and execute the attack. However, if we do assume that an internal attacker has access to the TLS private keys of the PEP, the attack becomes even easier. Such an attacker could impersonate the PEP to the PDP and thus receive a valid quote from the original PDP during the normal IDSCP handshake. This quote can then be used to complete the IDSCP handshake with the original PEP. In that case we do not even require a secondary attestation endpoint for the attack to succeed.

Table 4.1 gives a complete overview of the discussed protocols and their limitations. To summarize, most existing protocols are not suitable for protecting distributed usage control systems, because they are vulnerable either to man-in-the-middle attacks by internal adversaries, or to nonce-data attacks. Because of this, we still need to define a suitable TPM-based remote attestation protocol that can be used to protect our trustworthy usage control design.

Table 4.1: Overview of TPM-based remote attestation protocols.

Proposal	R1	R2	R3	R4	R5	Remarks
Sailer et al. [Sai04]	✓	✗	✓	-	-	No key exchange
Goldman et al. [Gol06]	✓	✗	✓	✗[1]	✗	
Stumpf et al. [Stu06, Stu08]	✓	✗	✓	✗[2]	✓	
Gasmi et al. [Gas07]	✓	✓	✓	✓	✓	TPM 1.2 only
Armknecht et al. [Arm08]	✓	✓	✓	✓	✓	TPM 1.2 only
Cheng et al. [Che09b]	✓	✗	✓	✗[1,2]	✗	
Zhou and Zhang [Zho10]	✓	✓	✓	✗[1]	✗	
Coker et al. [Cok11]	✓	✗	✓	-	-	No key exchange
Greveler et al. [Gre11]	✓	✓	✓	✗[2]	✓	
Aziz et al. [Azi14]	✓	✓	✓	✗[1]	✗	
Lan et al. [Lan14]	✓	✗	✓	✓	✓	
Akram et al. [Akr16]	✓	✓	✓	✗[2]	✓	
IDSCP [Bro22]	✓	✓	✓	✗[1,2]	✓	
RATLS [Wal22]	✓	✓	✓	✗[1]	✓	

4.2.4 The MSCP Protocol

In this section we present the *Mutually-Attested Secure Communication Protocol (MSCP)* as our proposal for a TPM-based remote attestation protocol that fulfills all identified security requirements R1 to R5, and hence is suitable for protecting distributed usage control systems. The MSCP protocol comes in two variants with different advantages and drawbacks. In order to avoid the presented security pitfalls of earlier proposals, we need to address two challenges. First, in order to prevent man-in-the-middle attacks by internal adversaries, we need to avoid any (long-term) secrets stored outside the TPM. Second, we need to properly bind the ephemeral keys that establish the encrypted channel to the attested code identities, such that it is not susceptible to nonce-data attacks. We achieve both challenges by leveraging the cryptographic primitives that TPM 2.0 provides for key agreement purposes.

[1] Vulnerable to man-in-the-middle attacks by internal adversaries.

[2] Vulnerable to nonce-data attacks.

Protocol description. We assume that prior to the start of the protocol, both participants have taken ownership of their respective TPMs and created a Storage Root Key (SRK), as well as an Attestation Key (AK) together with a corresponding certificate. The AK certificates are usually provisioned with the help of a trusted CA [Tru21b], but this process is out of scope for the MSCP protocol. Table 4.2 shows the MSCP protocol messages between two communication endpoints A and B. Note that the terminology used in table 4.2 closely follows the TPM 2.0 specification [Tru19c, Tru19d].

Table 4.2: The MSCP remote attestation protocol.

		Initiation phase	
A, B	:	$dhTemplate \leftarrow$ TPMT_PUBLIC(decrypt, KEY_SCHEME_ECDH)	
$A \rightarrow B$	:	Non-predictable nonce N^A and PCR selection $PCRSel^A$	(1)
$A \leftarrow B$	:	Non-predictable nonce N^B and PCR selection $PCRSel^B$	(2)
		Attestation phase	
A	:	$dh^A \leftarrow$ TPM2_Create($srk^A, PCRSel^B, dhTemplate$)	
A	:	$(certInfo^A, certSig^A) \leftarrow$ TPM2_CertifyCreation(ak^A, dh^A, N^B)	
$A \rightarrow B$	:	$akCert^A, dh^A.public, (certInfo^A, certSig^A)$	(3)
B	:	$dh^B \leftarrow$ TPM2_Create($srk^B, PCRSel^A, dhTemplate$)	
B	:	$(certInfo^B, certSig^B) \leftarrow$ TPM2_CertifyCreation(ak^B, dh^B, N^A)	
$A \leftarrow B$	:	$akCert^B, dh^B.public, (certInfo^B, certSig^B)$	(4)
		Verification phase	
A	:	Verify $(certInfo^B, certSig^B)$ is valid under $akCert^B$	
A	:	Verify $certInfo^B$ contains expected PCRs, N^A, and $dh^B.public$	
A	:	$Z \leftarrow$ TPM2_ECDH_ZGen($dh^A, dh^B.public$)	(5)
B	:	Verify $(certInfo^A, certSig^A)$ is valid under $akCert^A$	
B	:	Verify $certInfo^A$ contains expected PCRs, N^B, and $dh^A.public$	
B	:	$Z \leftarrow$ TPM2_ECDH_ZGen($dh^B, dh^A.public$)	(6)
$A \leftrightarrow B$	:	Establish encrypted channel using shared secret $k \coloneqq KDF(Z)$	

During the initial phase of the protocol, both participants create and exchange random nonces, as well as a selection of the PCRs that they want to verify (1,2). In the remainder of the handshake, we leverage the Elliptic Curve Diffie-Hellman (ECDH) key agreement functionalities provided by the TPM to establish an ephemeral shared secret between both participants. Furthermore, we conduct the mutual attestation by directly associating the ECDH key pairs created by the TPM with the current values in the PCRs registers, instead of signing them using the `TPM2_Quote` command. This can be done by calling the `TPM_Create` command with an ECDH key template, as well as the PCR selection requested by the peer [Tru19d, pp. 44–47]. Note that setting the `decrypt` flag in the key template is necessary for the TPM to create ephemeral ECDH key pairs that can be used for a key agreement scheme [Tru19c, p. 143]. We then use the `TPM_CertifyCreation` command to generate a certificate for the created ECDH public key, which now includes the received nonce, as well as the key's creation data containing the current PCR values [Tru19d, pp. 149–150]. This certificate is signed by the attestation key and serves the purposes of (i) attesting to the platform's PCR values, and (ii) authenticating the created ECDH public key by binding it to the attested platform. The attestation phase is concluded by mutually exchanging the AK certificates, the created ECDH public keys, and the ECDH certificates authenticating them (3,4). Finally, both participants verify that the received ECDH certificate is correctly signed under the remote attestation key. Furthermore, the certificate must contain the expected nonce, the expected PCR values, and the received ECDH public key. If the verification is successful, the received public key is correctly bound to the attested platform and can be used to calculate the shared secret Z using the `TPM2_ECDH_ZGen` command [Tru19d, pp. 94–95] (5,6). After applying a key derivation function (KDF), the determined shared secret can be used as seed for an authenticated encryption layer between both endpoints. Note that the MSCP handshake only specifies how to establish a mutually attested, shared secret between two TPM-protected platforms. The underlying transport security is not part of the protocol and should be realized using open source and well-reviewed cryptographic libraries.

Security discussion. The proposed MSCP protocol fulfills the security requirements defined in section 4.1. Most importantly, the protocol handshake includes an ephemeral ECDH key exchange to establish a shared secret between the attested endpoints. The key agreement is conducted by the TPM itself and the ephemeral Diffie-Hellman private keys never leave the protected hardware. Furthermore, we authenticate the ECDH keys by signing them directly with the attestation key using `TPM2_CertifyCreation`. This cryptographically binds the key exchange to the trusted platform. Since the attestation key never leaves the TPM, even internal attackers cannot intercept the handshake by forging a key signature. Hence the requirement of secure channel establishment is fulfilled (R4). Furthermore, the ephemeral ECDH key exchange also achieves perfect forward secrecy (R5). As a result, assuming that the used transport encryption engine is secure, confidentiality and integrity of transmitted data is ensured even against internal attackers with access to long-term secrets. The mutual attestation and authentication of the endpoints' TCBs (R1 and R2) is achieved by associating the ECDH keys with the platform PCR values at the time of their creation. For this, we configure the `TPM2_Create` command to include the requested PCR values into the creation data field[1] of the generated ECDH keys. Then the `TPM2_CertifyCreation` command not only signs the public parts of the ephemeral ECDH keys, but the associated PCR values as well. Since the attestation key is unknown even to internal attackers, the mutual attestation cannot be forged in our scenario either. Finally, the proposed handshake includes fresh nonces to prevent replay attacks against both the key exchange and the mutual attestation (R3). However, besides protecting the handshake itself, it is also necessary to prevent replay attacks on the data that is subsequently transmitted over the established secure channel (e.g., `ALLOW` decisions). This can be achieved by using an authenticated transport encryption such as AES-GCM [Jim22]. In addition to fulfilling the given requirements, MSCP also avoids the vulnerabilities identified in earlier proposals. Man-in-the-middle attacks by internal adversaries are not feasible, because the protocol handshake does not rely on any long-term secrets such as TLS certificates. MSCP is also not susceptible to nonce-data attacks, because it does not authenticate the ephemeral keys by hashing them into the quote.

To verify that our proposal is indeed secure against these attacks and fulfills our security requirements, we formally validated the protocol with the Tamarin protocol prover [Mei13]. The formal verification shows that MSCP

[1] See also [Tru19b, p. 182] and [Tru19c, p. 159] for more details about the contents and structure of the creation data field that is associated with TPM objects.

fulfills the protocol requirements R1 to R5. Furthermore, we explicitly modeled the nonce-data attack vector by introducing a quote oracle that gives attackers signed quotes for arbitrary attacker-chosen qualifying data. This allows us to recreate the nonce-data attack on IDSCP as depicted in fig. 4.2 with the theorem prover, and show that by contrast MSCP is not vulnerable in this scenario. The details of the formal verification, as well as the complete protocol formalizations, are given in appendices B.1 and B.2. In summary, MSCP avoids the issues of earlier proposals and as such is suitable to protect distributed usage control infrastructures. It also provides a rather simple handshake with just three TPM commands per endpoint. However, the downside of conducting TPM-internal key exchanges is the expected performance overhead, since asymmetric cryptographic operations (e.g., an ECDH key generation) are usually much slower on hardware TPMs than most CPUs.

Protocol variants. To provide a protocol alternative that avoids these particularly slow TPM operations, we also define a variant of MSCP that uses an *externally* conducted ECDH key agreement, instead of the previous TPM-internal one. This means that the ephemeral ECDH key pairs are not created by the TPM using the `TPM2_Create` command anymore, but are instead generated by a software library running on the much faster CPU. However, this once again opens up the challenge of properly binding the ECDH public keys to the attested platform. Since attestation keys are restricted to only sign TPM-internal objects, we cannot use the `TPM2_CertifyCreation` command to directly certify ephemeral keys that have been created outside of the TPM. As we have seen earlier, previous proposals such as IDSCP are vulnerable against nonce-data attacks due to an improper binding of the secure channel to the attested platform (see fig. 4.2). To prevent such vulnerabilities against nonce-data attacks, we authenticate the TPM-external key agreement using the PCRs instead of the quote's qualifying data. Table 4.3 shows the resulting MSCP protocol sequence using a TPM-external key agreement. During the attestation phase, a software library (e.g., OpenSSL) is used instead of the TPM to create new ephemeral ECDH key pairs (1,3). We then link the created ephemeral public keys to the attestation identity by extending their SHA-256 hash digests into the PCRs (2,4). We choose PCR 16 for this purpose, because it is one of the few PCRs that can be reset to zero before the extension [Seg16, p. 210]. This allows us to keep sequential handshakes independent from each other. It also avoids interfering with the platform integrity measurements, which are usually conducted using PCRs 0 to 10. We then use the `TPM2_Quote` command to sign the PCRs with the attestation key, thereby attesting to the platform state, as well as achieving the desired ECDH key binding. During the verification phase, the received ECDH public keys are authenticated by comparing their hash with the contents of the quoted PCR 16. Once the key agreement is completed by the software library (5,6), the shared secret can be used to establish a secure channel between the attested platforms.

Table 4.3: MSCP variant with TPM-external key agreement.

	Initiation phase		
$A \rightarrow B$	:	Non-predictable nonce N^A and PCR selection $PCRSel^A$	
$A \leftarrow B$	:	Non-predictable nonce N^B and PCR selection $PCRSel^B$	
	Attestation phase		
A	:	$dh^A \leftarrow$ ECDH_Create()	(1)
A	:	TPM2_PCR_Reset(16)	
A	:	TPM2_PCR_Extend(16, SHA256($dh^A.public$))	(2)
A	:	$quote^A \leftarrow$ TPM2_Quote($ak^A, N^B, PCRSel^B \cup \{16\}$)	
$A \rightarrow B$	:	$akCert^A, dh^A.public, quote^A$	
B	:	$dh^B \leftarrow$ ECDH_Create()	(3)
B	:	TPM2_PCR_Reset(16)	
B	:	TPM2_PCR_Extend(16, SHA256($dh^B.public$))	(4)
B	:	$quote^B \leftarrow$ TPM2_Quote($ak^B, N^A, PCRSel^A \cup \{16\}$)	
$A \leftarrow B$	:	$akCert^B, dh^B.public, quote^B$	
	Verification phase		
A	:	Verify $quote^B$ is valid under $akCert^B$	
A	:	Verify $quote^B$ contains expected PCRs, N^A, and $dh^B.public$	
A	:	$Z \leftarrow$ ECDH_ZGen($dh^A, dh^B.public$)	(5)
B	:	Verify $quote^A$ is valid under $akCert^A$	
B	:	Verify $quote^A$ contains expected PCRs, N^B, and $dh^A.public$	
B	:	$Z \leftarrow$ ECDH_ZGen($dh^B, dh^A.public$)	(6)
$A \leftrightarrow B$	:	Establish encrypted channel using shared secret $k := KDF(Z)$	

Like MSCP with TPM-internal key establishment, this protocol variant also fulfills all of our security requirements and is not vulnerable against nonce-data attacks or man-in-the-middle attacks by internal adversaries (see appendix B.2 for the formal verification). Furthermore, it has the benefit of leveraging the greater computational power of the CPU for the expensive ECDH key agreement. It also allows to choose from a wider variety of elliptic curves than what is usually implemented in the hardware TPM. However, by conducting the key exchange outside the TPM, we no longer profit from the secure hardware to protect the ECDH private keys. Note that since these are ephemeral keys instead of long-term secrets, this does not cause a vulnerability to the previously described attacks by internal adversaries. Nevertheless, we now must ensure that the used software library properly implements this key agreement, which increases the size of the TCB. We also need to make the additional assumption that the trusted software stack does not allow external adversaries to extend PCR 16 with arbitrary data. The result of these considerations is a trade-off between leveraging the secure TPM hardware to its full extent and using the improved performance of the CPU. We evaluate and compare the performance of both MSCP protocol variants later in this section.

Finally, another design alternative would be to not conduct a dedicated ECDH key exchange at all, but instead create self-signed TLS certificates at both endpoints and authenticate *them* using the presented mechanisms. Because TLS since version 1.3 always conducts ephemeral key exchanges anyway [Res18], this solution would essentially just add another level of indirection. While the protocol variant using PCR 16 to authenticate the public keys can be modified rather easily to accommodate this, supporting TPM-internal keys would require a TPM-aware TLS implementation. Since we later integrate the mutually attested key exchange of MSCP into a heterogeneous attestation protocol anyway (see section 4.5), we choose to not include TLS in our proposal.

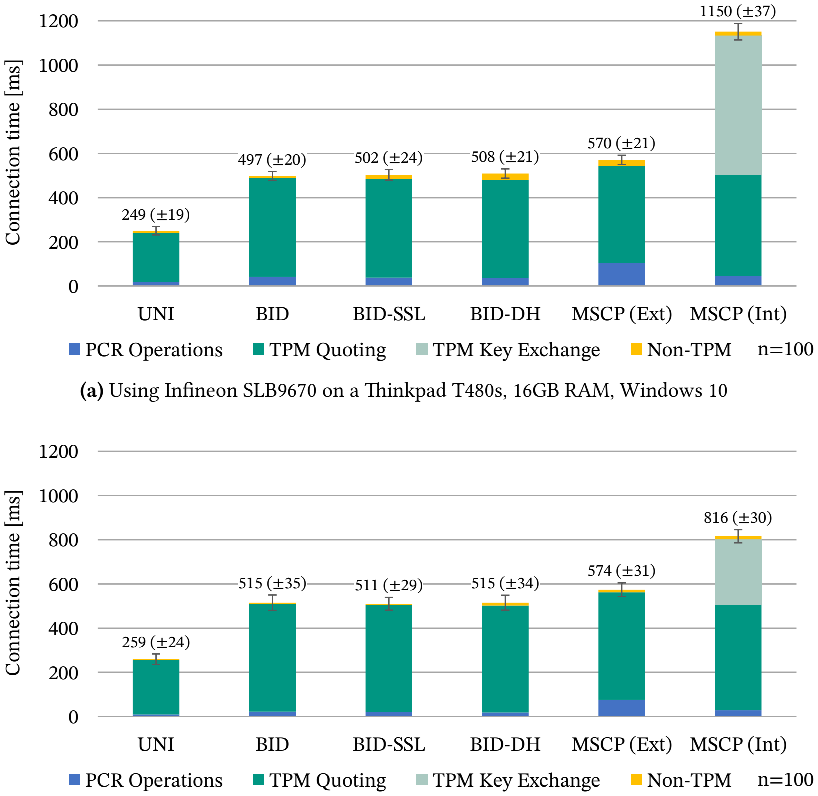
Protocol implementation and evaluation. We implemented and evaluated both proposed variants of MSCP in Java using the Microsoft TSS.MSR[1]

[1] https://github.com/microsoft/TSS.MSR *(accessed on 01/23/2024).*

API for TPM 2.0. To provide a usable MSCP protocol stack and achieve realistic handshake connection times, we integrated our attestation protocol into standard Java TCP sockets. When a new TCP connection is established between client and server, our implementation automatically conducts the attestation handshake and validates the received attestation evidence. For the ECDH ephemeral key agreement we use the NIST P-256 elliptic curve, because its support is mandated by the TPM 2.0 profile specification [Tru20]. After a successful attestation, all communication over the attested sockets is transparently encrypted using a 128 bit AES session key derived from the established shared secret with the PBKDF2 key derivation function. Our complete code is available online for verification purposes.[1] Furthermore, to allow for a comparison with existing proposals, we also implemented some of the previous attestation protocols discussed in section 4.2.2. In total, we implemented and evaluated six different attestation protocols.

1) Uni-directional attestation using `TPM2_Quote` without key establishment.
2) Bi-directional attestation using `TPM2_Quote` without key establishment.
3) Bi-directional attestation using `TPM2_Quote` over a TLS channel. The TLS certificates are authenticated by hashing them into the quote. This is the attestation mechanism used by IDSCP [Bro22].
4) Bi-directional attestation using `TPM2_Quote` with a dedicated Diffie-Hellman key exchange. The key exchange is authenticated by hashing the ephemeral public keys into the quote. This is the attestation mechanism used by the protocols that are based on the proposal by Stumpf et al. [Stu06, Stu08, Gre11, Akr16].
5) MSCP with a TPM-*external* Diffie-Hellman key exchange as defined in table 4.3. The key exchange is authenticated by extending the ephemeral public keys into PCR 16.
6) MSCP with a TPM-*internal* Diffie-Hellman key exchange as defined in table 4.2. The key exchange is authenticated by certifying the ephemeral public keys directly with the attestation key.

[1] https://gitlab.cc-asp.fraunhofer.de/tpm-20-commons/tpm-java *(accessed on 12/08/2023).*



(a) Using Infineon SLB9670 on a Thinkpad T480s, 16GB RAM, Windows 10

(b) Using Intel PTT on an i7-9700KF server, 32GB RAM, Windows 10

Figure 4.3: Mean connection times for TPM-based remote attestation protocols in milliseconds. The standard deviation is given in brackets.

Figure 4.3a shows the mean connection times over 100 handshakes using the widespread Infineon SLB9670[1] TPM 2.0 hardware module. As we can see, the majority of the connection time is expended for creating the attestation evidence using `TPM2_Quote`. We used an RSA 2048 bit attestation key with SHA-256

[1] https://www.infineon.com/cms/de/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/slb-9670vq2.0/ *(accessed on 12/08/2023).*

digests for all of our tests, which brings the complete handshake time for a bidirectional remote attestation to around 500 milliseconds. Our MSCP protocol variant with TPM-external key establishment takes about 60 milliseconds longer than the previous proposals using a Diffie-Hellman key exchange. This is because we require one additional `TPM2_PCR_Reset` and `TPM2_PCR_Extend` operation for each endpoint during the attestation. Finally, the MSCP handshake with TPM-internal key establishment takes about twice as long as with external key establishment. This is due to the overhead for creating and loading the ephemeral ECDH keys using `TPM2_Create`, as well as the final calculation of the shared secret using `TPM2_ECDH_ZGen`. Even though the internal key establishment avoids additional PCR operations, using the TPM for this task is much slower than using the CPU. In addition to the Infineon TPM, we also conducted a performance evaluation using Intel's Platform Trust Technology[1] (PTT). Intel PTT includes a TPM 2.0 implementation that is integrated into many current processors, making it a very widespread TPM module as well. As fig. 4.3b shows, on this platform the attestation protocols produce results that are generally comparable to the Infineon TPM, mainly due to similar execution times of the underlying TPM operations. However, the performance of the MSCP protocol with internal key exchange is noticeably improved by over 300 milliseconds compared to the previous configuration. This is caused by a lower overhead for the TPM-managed ephemeral key exchange, which is most likely the result of a more efficient internal implementation.

To conclude, our results show a noticeable performance impact on the attestation when using the TPM itself to conduct the ephemeral Diffie-Hellman key exchange. Given that hardware TPMs are much slower than CPUs especially for asymmetric cryptography, this is not unexpected. However, depending on the required frequency of re-attestation, the increased latency may still be acceptable for the benefit of protecting the key exchange with the secure TPM hardware. Alternatively, the MSCP protocol variant with external key establishment achieves a comparable performance to the previous proposals, with the added benefit of not being vulnerable to nonce-data attacks anymore.

[1] https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/enterprise-security-platform-trust-technology-white-paper.pdf *(accessed on 12/08/2023).*

4.3 Using Intel SGX

The Intel Software Guard Extensions (SGX) are a trusted computing technology for Intel's product line of server CPUs. In contrast to TPMs, Intel SGX is designed as a dedicated Trusted Execution Environment (TEE). SGX enclaves are completely isolated from the rest of the system, including the operating system and other user applications (see section 2.3.2). Because of this, SGX enclaves achieve a much smaller TCB than TPM-protected systems, which is a clear security advantage. Hence, SGX is a good candidate to protect the integrity of usage control and provenance tracking components. In this section we discuss the security properties of SGX enclaves and show that they fulfill our requirements. We furthermore give an overview of existing proposals for SGX-based attestation protocols.

4.3.1 Security Properties

Code integrity. As described further in section 2.3.2, SGX processors automatically conduct integrity measurements when launching enclaves. This is done by hashing the loaded memory pages of the enclave via the EADD and EEXTEND processor instructions (see fig. 2.8). Because of this, the resulting enclave measurement value (i.e., MRENCLAVE) uniquely represents the initial state of the enclave including its code and data. This value is securely stored inside the enclave's internal data structures and can only be altered by the processor's SGX implementation [Cos16a, p. 59]. Hence, the MRENCLAVE value is suitable to ensure code integrity (requirement T1).

Data protection during storage. Like TPMs, SGX enclaves also support sealing of confidential data to the TCB. For this, SGX processors allow enclaves to derive a unique *sealing key* using the EGETKEY instruction. This key is derived by the processor from the local platform secrets. It also includes the enclave measurements that have been collected during launch. Hence, this key is unique for the local platform and can only be calculated by that specific enclave itself. Alternatively, the EGETKEY instruction can also derive a sealing key from the MRSIGNER value. This results in a sealing key that is accessible to *all* enclaves signed with the same signing key, which is useful for provisioning secrets to multiple enclaves at once. However, unlike TPMs, SGX does not allow to arbitrarily create new sealing secrets. Instead, if multiple different sealing keys are required, a nested encryption approach is chosen. For this, the enclave internally generates new cryptographic keys for the required purposes, and then seals the respective private keys with the symmetric sealing key retrieved by EGETKEY. All in all, SGX enclaves can protect the confidentiality and integrity of data that is stored outside the TCBs (requirement T2).

Data protection during processing. Besides sealing, SGX enclaves also offer strong protection of data confidentiality and integrity during processing. This is due to the complete isolation of executed enclaves from the rest of the system. While SGX enclaves are being executed, the SGX processor is responsible for preventing any non-enclave code from accessing the processed data. If the enclave's memory pages are evicted from the processor cache, they are automatically encrypted with a processor-internal secret key. Furthermore, there is no way for other code to directly call into an enclave, including privileged components such as the operating system. Any communication with the enclave can only be conducted over well-defined interfaces using the EENTER and EEXIT instructions (see section 2.3.2). Because of this, each enclave can decide for itself under which conditions information may be released outside the TCB. Hence, the SGX architecture by design protects data confidentiality and integrity even during processing (requirement T3).

Duplication and rollback protection. SGX prevents the duplication of launched enclaves to other platforms by ensuring that sealed enclave states can only be read by one particular SGX processor containing the original platform secrets. However, enclaves can still be duplicated by simply creating multiple instances of a single enclave image on one platform. With the same memory image, these enclave instances will all have the same MRENCLAVE measurement value, and hence have access to the same sealed data. This would allow attackers to duplicate launched enclaves together with their sealed states, and henceforth operate multiple independent instances of deployed components. In the case of distributed usage control this could lead to serious security issues, for example when multiple independent CA enclaves for the same domain URI exist (see section 3.4.2). To mitigate this problem, local enclave duplication (requirement T4) can be detected using monotonic counters. Similar to TPMs, such hardware counters can only be increased and hence allow enclaves to detect if there are other active instances present on the platform that are operating on the same counter. However, unlike in TPMs, the support of monotonic counters is not guaranteed in all SGX processors, because it requires additional non-volatile memory banks in the hardware [Mat17]. Furthermore, monotonic counters are known to have issues with wear and performance as well [Mat17]. Besides duplication, SGX must also be able to protect against rollback attacks (requirement T5). Since this is a similar issue as duplication, it can also be detected using monotonic counters [Str16]. Furthermore, there are also proposals for rollback protection on SGX-based platforms that do not have access to monotonic counters. Strackx et al. mitigate the lack of monotonic counters by leveraging an external TPM together with some additional guarded memory [Str14]. The drawback of this solution is that it increases the size of the TCB by depending on TPM functionality, and still requires special hardware. Matetic et al. instead propose to use a cluster of enclaves storing integrity information about states [Mat17]. This allows to implement an all-or-nothing rollback policy in the cluster without additional hardware requirements. However, this solution requires a sufficiently large number of identical enclave instances, which must be assumed to not be attacked all at once. This is not necessarily the case in distributed usage control scenarios. Finally, Brandenburger et al. propose to store state integrity information at the clients of services running inside enclaves [Bra17a]. This proposal also considers duplication attacks, but has the drawback of requiring state-aware enclave clients. To summarize, SGX natively offers only limited support for duplication and rollback protection using monotonic counters. Because of this, rollback protection on SGX is usually done with the help of external trust anchors such as TPMs, other enclave instances, or clients.

4.3.2 Remote Attestation Protocols

Similar to TPMs, SGX-based architectures also provide the possibility to remotely verify the integrity of executed enclaves. The most basic primitive for SGX-based attestation is the `EREPORT` processor instruction. The `EREPORT` instruction can be used to conduct an attestation between two enclaves that are running on the same platform (i.e., *local* attestation). This instruction generates an attestation report, which attests to the code base of the currently executed enclave by including its `MRENCLAVE` and `MRSIGNER` measurement values, as well as information about the local SGX platform. Furthermore, to prevent enclaves from forging measurements, the attestation report is symmetrically signed with a processor-internal *report key*. Just like the sealing key, this report key is derived from the platform secrets, as well as the measurements of the target enclave that should receive the generated attestation report for verification. The target enclave can then use the `EGETKEY` instruction to recover the report key and thus validate the signature of the received attestation report, which authenticates the TCB of the attested enclave (requirement R1). However, this attestation procedure only works between enclaves that are running on the same physical platform, because otherwise the `EREPORT` and `EGETKEY` instructions will not use the same platform secrets and hence derive different report keys.

To allow the verification of attestation reports also by external and non-SGX platforms (i.e., *remote* attestation), the SGX architecture relies on the concept of *quoting enclaves*. Quoting enclaves are responsible for transforming SGX attestation reports into universally verifiable *quotes*. For this, a quoting enclave first conducts a local attestation of the target enclave that should be remotely verified. If the local attestation is successful, the quoting enclave signs the created attestation report with an asymmetric attestation key that has been previously provisioned to the SGX platform. The resulting quote is then transferred to the remote platform, where the quote signature can be verified using the public part of the attestation key. Depending on the type of the used attestation key, two variants of SGX-based remote attestation can be distinguished. The first attestation variant supported by SGX was the *Enhanced Privacy ID (EPID)* group signature scheme [Bri10]. EPID attestation keys are created locally on the SGX platform, and are then linked to a signature group by conducting a key joining protocol with a provisioning service centrally operated by Intel. The resulting attestation keys allow quoting enclaves to anonymously sign attestation reports, which can later be verified using a common public key for the entire group. The benefit of such a group signature scheme is that the created attestations cannot be traced back to individual SGX processors. However, since validating EPID signatures requires support of the Intel Attestation Service (IAS), it also introduces an additional trust anchor at Intel. In later versions of SGX, Intel introduced the *Data Center Attestation Primitives (DCAP)*, which allows platform owners to host their own attestation infrastructure [Sca18]. DCAP supports the use of custom quoting enclaves that generate traditional asymmetric attestation keys. The generated attestation keys are authenticated with a platform certification key that is derived from the local SGX platform secrets. The platform certificate in turn is signed with a root key managed by Intel, to endorse that the certification key is indeed protected by a genuine SGX platform.

While EPID and DCAP specify the process that generates, provisions, and authenticates attestation keys, we still need to define a remote attestation protocol that is capable of establishing mutually attested, secure communication channels. Several remote attestation protocols have been proposed since SGX was introduced. Aublin et al. [Aub17] propose to establish standard TLS connections between SGX enclaves. While this creates encrypted channels, it does not bind them to the enclave identities and hence does not conduct a remote attestation (requirements R1 and R2). Chen et al. [Che19b] instead propose a dedicated Diffie-Hellman key exchange between remote SGX enclaves, which is authenticated by the signed attestation report. The Intel SDK [Int23a, p. 118] also includes an example implementation for a similar attestation protocol. However, since both solutions focus on provisioning secrets to enclaves after they are deployed, these proposals do not support mutual attestation (requirement R2). Knauth et al. [Kna18] develop *RA-TLS*, which achieves remote attestation by linking TLS endpoints inside SGX enclaves to their respective attestation evidence. For this, each enclave generates new self-signed TLS certificates at every startup. The generated certificates are authenticated by including their fingerprints into the signed attestation reports. For convenience, the attestation reports are then embedded into the TLS certificates as a custom X.509 extension. This proposal supports mutual attestation and provides secure channels as well as replay protection (requirements R1 to R4). Forward secrecy can also be guaranteed if the underlying secure channel uses TLS in version 1.3 (requirement R5). RA-TLS has been adopted for some SGX frameworks as well, most prominently Gramine [Gra22]. Furthermore, King and Wang proposed *HTTPA*, which similarly integrates SGX-based attestation into HTTPS instead of TLS [Kin21]. However, this proposal establishes session keys by transmitting encrypted secrets instead of using an ephemeral key exchange, which is why it does not support forward secrecy (requirement R5).

Most recently, Google developed the *Enclave Key Exchange Protocol (EKEP)* [Asy21a]. EKEP is the standard attestation protocol for the Asylo[1] SGX framework. It is based on Google's Application Layer Transport Security (ALTS) protocol, which is itself a modified TLS implementation used by Google for secure RPC communication in their data centers [Gha17]. The main advantage of EKEP over previous proposals is that it guarantees mutual attestation and forward secrecy (requirements R2 and R5) by conducting a dedicated Diffie-Hellman key exchange, which is authenticated by the signed attestation report. Hence it fulfills all our requirements from section 4.1. The security of EKEP has also been formally verified using ProVerif [Roe22]. Furthermore, EKEP allows to transparently conduct both local *and* remote attestation using the same protocol handshake. An open source protocol implementation of EKEP is available on Github under the Apache license.[2] This implementation integrates into Google's gRPC remote procedure call library, which makes it particularly suitable for our use case of distributed usage control.

As table 4.4 shows, there are several proposals for SGX-based remote attestation protocols on offer that fulfill our requirements. We choose to build our proof of concept on EKEP, because it is suitable for our use case and can also be extended into a heterogeneous remote attestation protocol (see section 4.5).

[1] https://asylo.dev/ *(accessed on 12/08/2023).*

[2] https://github.com/google/asylo *(accessed on 12/08/2023).*

Table 4.4: Overview of SGX-based remote attestation protocols.

Proposal	R1	R2	R3	R4	R5
Aublin et al. [Aub17]	✗	✗	✓	✓	✓[3]
Chen et al. [Che19b]	✓	✗	✓	✓	✓
Intel SDK [Int23a, p. 118]	✓	✗	✗	✓	✓
RA-TLS [Kna18, Gra22]	✓	✓	✓	✓	✓[3]
HTTPA [Kin21]	✓	✓	✓	✓	✗
EKEP [Asy21a]	✓	✓	✓	✓	✓

4.4 Using ARM TrustZone

As introduced in section 2.3.2, TrustZone is a security extension of the ARM processor architecture that partitions the system into a normal world (REE) and a secure world (TEE). Both worlds have their own independent firmware, operating systems, and applications. Furthermore, platform resources such as main memory and I/O devices can also be divided up between both worlds. This allows to isolate security critical Trusted Applications (TAs) from the Rich Execution Environment and its often large attack surface. While TrustZone is not as comprehensive as other TEEs such as Intel SGX, it is still a very useful and widespread security technology on embedded platforms, where both SGX and hardware TPMs are usually not available.

In this section we first discuss the security properties of ARM TrustZone devices and point out existing limitations regarding the requirements for our usage control system design. Then we show in what ways usage control components can be deployed on TrustZone-protected devices and present a solution to achieve platform measurements in both the rich and the trusted environment. Finally, we discuss the existing proposals for remote attestation protocols on TrustZone platforms and propose a solution that is suitable for our proof of concept.

[3] Forward secrecy is guaranteed only if TLS v1.3 is used.

4.4.1 Security Properties

The ARM TrustZone technology itself comprises of the secure processor and the trusted firmware for the respective architecture, such as TF-A[1] for Armv7-A and Armv8-A class processors. However, leveraging the TrustZone technology for TEE functionalities still requires a comprehensive software stack that builds on the secure processor and its firmware. The most widespread framework for implementing Trusted Applications on ARM TrustZone devices is OP-TEE[2]. OP-TEE provides a secure operating system for the trusted world, as well as two APIs that facilitate the implementation of Trusted Applications. The TEE Internal Core API [Glo21] defines the main interfaces that Trusted Applications need to implement, and also provides them with several support functions such as storage access and cryptographic operations. In addition, the TEE Client API [Glo10] provides interfaces that allow to manage the implemented TAs from the normal world. It also includes functions to share information between the REE and the TEE over well-defined communication interfaces. Furthermore, the OP-TEE implementation provides several TrustZone-specific security features that are relevant for our use case.

Data protection during storage. While the TF-A trusted firmware itself does not provide any sealing functionalities, TrustZone devices may include a set of factory-programmed secret keys that are only accessible from inside the TEE. OP-TEE can use these platform secrets to derive several device- and TA-specific symmetric encryption keys [OP-19a]. These keys can then be used to seal critical data and application states to the trusted platform. Sealed data blobs can be stored either in the REE file system, or in a dedicated eMMC memory storage if the trusted device provides it. However, unlike TPMs and SGX processors, TrustZone devices are not required to include any hardware-based platform secrets. If no platform secrets are available, OP-TEE will instead use a constant seed for deriving the sealing keys [OP-19a], which is insecure.

[1] https://trustedfirmware-a.readthedocs.io/en/latest/ *(accessed on 12/08/2023).*

[2] https://optee.readthedocs.io/en/latest/ *(accessed on 12/08/2023).*

Hence, TrustZone can achieve data protection during storage (requirement T2) only if the used device provides hardware-based platform secrets.

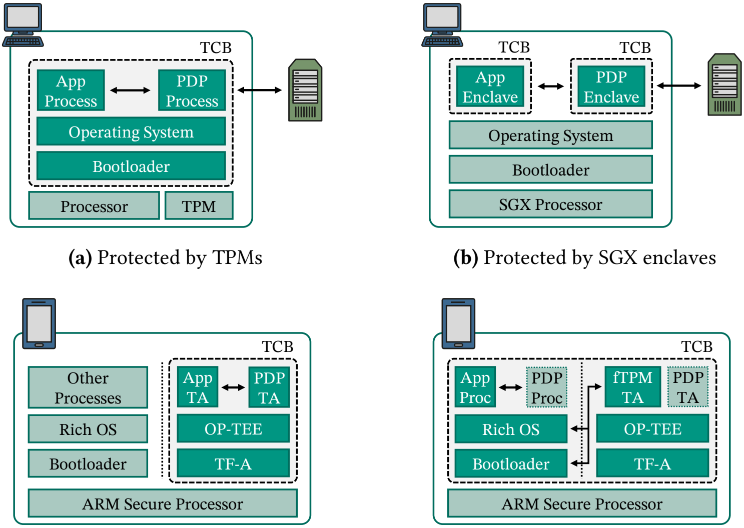
Data protection during processing. Like Intel SGX, TrustZone offers a dedicated Trusted Execution Environment (TEE). This means that the Trusted Applications running in the secure world are completely independent from the rich operating system and the rest of the REE. OP-TEE provides access to the TAs from the normal world via session-based communication channels. REE applications can use these sessions to invoke commands at the TAs. However, they have no direct influence over the execution of code in the secure world. Since TrustZone relies on hardware-based access control instead of memory encryption for the isolation, it is also possible to define shared memory between the REE and TEE that can be used for data exchange. Furthermore, unlike SGX enclaves, TrustZone TAs are full processes that are managed by the trusted operating system. While this allows TAs to rely on the trusted OS for support, it also results in a much larger TCB than when using SGX. Nevertheless, TrustZone devices still offer protection of critical data in the secure world during processing (requirement T3).

Rollback and duplication protection. Rollback protection on TrustZone devices is implemented using a Replay Protected Memory Block (RPMB). An RPMB provides non-volatile memory that can be accessed only after proper authorization. For this, OP-TEE derives a unique authorization key from the platform secrets and programs it into the RPMB controller [OP-19a]. Since this key is only available inside the TEE, no untrusted code can read out or modify the RPMB memory. OP-TEE uses a correctly configured RPMB to keep track of the version numbers of loaded TAs [OP-19b]. This protects the TA images against rollback attacks. Furthermore, the RPMB can be used to protect sealed data against rollback attacks as well. However, just like the platform secrets, TrustZone devices are not required to provide an RPMB. Without it, no secure rollback protection is possible (requirement T5). Furthermore, OP-TEE offers no dedicated protection mechanisms against TA duplication (requirement T4).

Code integrity. Finally, there is the open question of achieving code integrity on TrustZone platforms (requirement T1). The main goal of TrustZone is to define a TEE for operating Trusted Applications in a secure environment. However, TrustZone does not provide native support for measuring the integrity of trusted platforms. To resolve this, Raj et al. propose a firmware-level TPM (fTPM) for TrustZone devices [Raj16]. fTPM is a software implementation of the TPM 2.0 reference architecture, which can be executed as a TA service in the secure world of a TrustZone device. As such, it can simulate a TPM to both the REE and TEE, which allows us to use the PCR registers of the fTPM module as Roots of Trust for Storage and Reporting. During its operation, fTPM is protected by the TEE against malicious influence and tampering. Obviously, the fTPM implementation also requires hardware-based platform secrets, as well as a suitable RPMB, to securely implement the TPM standard [Raj16]. But even with these hardware-based trust anchors, fTPM cannot reach the same level of security as hardware TPMs, because it has a larger attack surface. Nevertheless, due to the lack of native solutions, using an fTPM module is currently the best option to make the code base of TrustZone platforms remotely verifiable. However, simply deploying fTPM as a Trusted Application does not automatically achieve code integrity. We still need to define a suitable measurement process that collects software fingerprints and establishes a comprehensive chain of trust rooted in the hardware-based trust anchor of the TrustZone platform. How to do that also depends on the scope of the TCB that should be protected. We discuss the caveats of conducting fTPM-based integrity measurements on TrustZone devices and develop a suitable solution for our use case in the remainder of this section.

4.4.2 Deployment of Usage Control Components

When using TPMs or Intel SGX to protect distributed usage control infrastructures, the scope of the usage control enforcement directly corresponds to the respective TCB. In case of TPMs, usage control components run as normal processes in the user space and are verified by the conducted integrity measurements (cf. fig. 4.4a). With SGX, the usage control components instead run as enclaves that are isolated from untrusted parts of the system (cf. fig. 4.4b).



(a) Protected by TPMs

(b) Protected by SGX enclaves

(c) Protected by TrustZone TEE

(d) Protected by fTPM

Figure 4.4: TCBs of deployed usage control components.

However, when using TrustZone we have two options regarding the deployment of usage control components, each achieving a different degree of isolation. One possibility is to conduct the usage control enforcement completely inside the TEE. This means that the usage control components must be executed as Trusted Applications, which have the advantage of being completely isolated from the normal world. As shown in fig. 4.4c, the resulting TCB encompasses only the secure world including the trusted firmware, the trusted operating system, and the TAs. The downside of this approach is that it requires to implement the data processing applications as TAs as well. Since TAs are meant to be small service daemons such as cryptographic modules, this is often not feasible. Especially legacy applications requiring support of the normal world operating system cannot easily be implemented as TAs. To address this problem, we can design the software stack to allow usage control enforcement in the REE as well. In this case the data processing applications are running as normal world processes, with the usage control components being located either in the REE or the TEE. This solution has the obvious disadvantage of not leveraging the TEE isolation for the data processing applications any longer. However, we can still protect the integrity of the usage controlled software stack by relying on the fTPM module as trust anchor in the secure world. As shown in fig. 4.4d, the resulting TCB then encompasses all required TEE components, including fTPM, as well as the complete software stack of the normal world. Furthermore, using fTPM allows us to leverage TPM-specific functionality in the REE, such as restricted signature keys, non-volatile memory, and monotonic counters. Because fTPM adheres to the TPM 2.0 specification, this approach greatly simplifies the implementation of remote attestation, as well as duplication and rollback protection on TrustZone platforms. On the downside, the data processing applications can no longer be isolated from the normal world operating system. Just as when using classical TPMs, security vulnerabilities in the normal world operating system could now be exploited to tamper with the usage control enforcement. As a result, the security guarantees of this solution are lower than when only protecting the software stack inside the TEE.

Nevertheless, since REE-based usage control components greatly increase enforcement flexibility, we consider both deployment variants useful for our system design. However, there are still some open questions about how fTPM can be used to protect the integrity of software stacks residing in *both* the REE and the TEE. For this we require an integrity measurement process that spans both the normal and the secure world. While the TF-A trusted firmware already provides a proof of concept for conducting fTPM-based integrity measurements of the secure world boot stages [ARM21a], we still need to configure a suitable normal world trusted boot process as well. Furthermore, since static boot-time integrity measurements are not sufficient for our use case, we must also enable dynamic load-time measurements of user applications both in the normal and the secure world. In the remainder of this section, we show how to define a suitable trusted boot process for TrustZone devices that establishes a chain of trust rooted in the ARM trusted firmware, while measuring the code integrity of both REE and TEE applications.

4.4.3 Conducting Both-World Measurements

In this section we show how integrity measurements can be conducted in both the normal and the secure world of TrustZone platforms. As motivated earlier, we achieve this by relying on an fTPM module running as a Trusted Application to establish a Root of Trust for Storage and Reporting. Our goal is to measure both the normal world applications as well as the TAs running in the secure world. Furthermore, a chain of trust spanning the complete TCB must be established, which includes the boot loaders and the operating systems of both worlds (cf. fig. 4.4d). The defined measurement process will then serve as the foundation for a TrustZone-based remote attestation protocol, which we discuss in the final section of this chapter. In the remainder of this section, we first introduce the default secure boot process of ARM TrustZone devices and present the existing support in the TF-A trusted firmware for measuring the integrity of the secure world boot stages. Then we show how this process can be extended to conduct load-time measurements in the normal world of TrustZone devices using the classical Integrity Measurement Architecture (IMA) of the Linux kernel. To reach our goal of measuring the entire TCB of TrustZone platforms, we also develop a solution for conducting measurements of dynamically loaded Trusted Applications in OP-TEE. Finally, we evaluate the proposed boot process by discussing the resulting chain of trust as well as the expected performance impact.

Note that the proof of concept presented in this section has been developed by Amblank and Wagner as part of a supervised Bachelor's thesis [Amb22].

Trusted Board Boot. The default secure boot process for Armv7-A and Armv8-A class processors is called Trusted Board Boot (TBB). TBB is implemented as part of the TF-A trusted firmware, and leverages the TrustZone hardware to authenticate boot images during device startup [ARM21b]. For this, the TBB boot process sequentially verifies the digital signatures of all boot loader stages, including the trusted operating system, before executing them. The signatures of the loaded boot images are validated using a certificate chain that is rooted in an immutable public key called *Root of Trust Public*

Key (ROTPK), which is burnt into the device during provisioning. Only the original platform owner knows the corresponding private key and can sign firmware images that the device will accept during boot. Figure 4.5 shows the principal flow of the Trusted Board Boot process. A complete description is given in the official TF-A documentation [ARM21b]. The very first boot loader stage that is loaded from ROM and executed on the device is called BL1. The only responsibility of BL1 is to load the next boot loader stage (BL2) and verify its signature using the ROTPK. BL2 then loads the secure world firmware (BL31), as well as the trusted operating system (BL32), and hands over control of the secure world to it. Finally, BL2 loads and executes the normal world boot loader (BL33), which initializes the boot process of the normal world operating system. Both secure world boot images (BL31 and BL32) are authenticated using the *Trusted World Public Key (TWPK)*, which is itself signed by the ROTPK. Similarly, the normal world boot loader (BL33) is signed with a *Non-trusted World Public Key (NWPK)*. By sequentially verifying all boot image signatures, TBB can extend the initial trust placed in the read-only BL1 and ROTPK to the entire firmware as well as the secure world operating system.

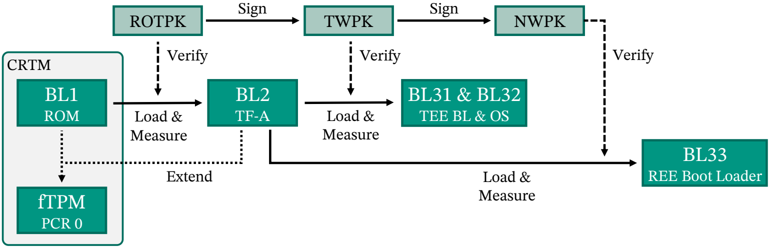


Figure 4.5: Measured Trusted Board Boot process using fTPM.

Measuring the boot process. Trusted Board Boot uses digital signatures to protect the firmware of TrustZone devices against unintended modifications caused by compromised devices or malware infections. However, this alone is not sufficient to prevent malicious platform owners from tampering with the code base of a trusted device. This is because platform owners must be assumed to control the ROTPK private key, which can be used to sign new boot images. To protect the integrity of a TrustZone-based software stack even in such scenarios, we need to conduct integrity measurements of the loaded boot images and establish a chain of trust that can later be remotely attested. Fortunately, the ARM trusted firmware supports the extension of TBB with fTPM-based integrity measurements. This allows us to execute a *measured* TBB boot process on TrustZone platforms. The full description of this process is given in the official TF-A documentation [ARM21a]. During measured TBB the BL1 and BL2 boot stages not only verify the digital signatures of loaded images, but also write their fingerprints into a TPM event log. This log follows the standard EFI event log format as specified by the TCG [Tru16b], and is stored in secure memory to prevent tampering by unmeasured code. As shown in fig. 4.5, once the fTPM module is loaded it extends the collected event log into PCR 0, which makes all loaded boot images verifiable using an fTPM-based remote attestation. While this measurement process is very similar to the classical trusted boot using hardware TPMs (see section 2.3.1), on TrustZone platforms we also have to include the fTPM module as part of the initially trusted Core Root of Trust for Measurement (CRTM). This is because we obviously cannot use fTPM to measure its own software integrity. Alternatively, trust into fTPM could be bootstrapped by having the BL1 boot stage directly load and authenticate the fTPM image. However, to our knowledge this is not yet included in the ARM trusted firmware at the time of this thesis.

Measuring normal world applications. The integrity measurements conducted by the modified TBB boot process end at the normal world boot loader (BL33). However, to also protect usage control components running in the normal world, we need to extend the fTPM measurements to the REE as well. Since conducting integrity measurements during boot is well-supported with classical hardware TPMs, we can largely rely on existing implementations to achieve this. First, we need to include a normal world boot loader that supports measured boot using fTPM. The TBB reference implementation uses TianoCore EDK2 and Grub as normal world boot loaders [ARM21a]. For our proof of concept, we replaced these modules with the U-Boot[1] universal boot

[1] https://u-boot.readthedocs.io/en/latest/index.html *(accessed on 12/08/2023).*

loader, which already provides support for fTPM-based measured boot according to the TCG EFI protocol specification [Tru16b]. This allows us to configure U-Boot to measure and launch the REE Linux kernel as an UEFI payload. Following the TCG firmware specification [Tru21a], U-Boot uses the PCR registers 2 and 4 to measure the UEFI drivers and payloads, respectively. Furthermore, in order to prevent measurement gaps, we also configure U-Boot to disable the boot command line. All in all, this extends the chain of trust from the ARM trusted firmware to the normal world operating system. Finally, the REE Linux kernel includes the Integrity Measurement Architecture (IMA), which we can use to conduct load-time measurements of launched user-space applications. By default the Linux IMA extends its measurements into PCR 10. Song et al. recently proposed to use a dedicated Trusted Application running in the secure world to collect and store the measurements of the normal world Linux IMA [Son22]. However, we find it more convenient to simply use fTPM as Root of Trust for Storage for the IMA as well. Fortunately, the Linux kernel already provides a driver[1] for fTPM. This allows us to extend the established chain of trust rooted in BL1 and fTPM over the normal world boot loader (BL33) all the way up to the user-space applications running in the REE. One remaining issue with the integration of the Linux IMA is that both the normal and secure world boot processes are running independently once the BL2 stage relinquishes control of the processor. As a result, the fTPM Trusted Application might not yet be available when the Linux IMA is initialized. While this issue could likely be resolved by delaying the IMA PCR extensions until the fTPM driver is available, we consider modifying the Linux kernel in such a way as beyond the scope of this work.

Measuring secure world applications. Finally, in order to achieve a measured boot process that is suitable for our use case, we also need to enable the load-time integrity measurement of Trusted Applications in the secure world. As illustrated in fig. 4.5, the chain of trust that TBB establishes in the secure world ends at the trusted operating system (BL32). We use OP-TEE as trusted

[1] https://www.kernel.org/doc/html/v6.7/security/tpm/tpm_ftpm_tee.html *(accessed on 01/21/2024).*

operating system, which is essentially a modified Linux kernel that has been stripped down in complexity to reduce its attack surface. The simplest way to include OP-TEE Trusted Applications into the established chain of trust is to compile them as *early TAs*. Early TAs are directly linked into a special section of the OP-TEE boot image, and as such are loaded and executed automatically as soon as the trusted OS launches [OP-19b]. Since this makes early TAs a part of the BL32 boot image, they are already included in the measurements conducted by BL2 during the Trusted Board Boot process (cf. fig. 4.5). However, compilation as early TA is feasible only for applications that almost never change, since every update of an early TA requires flashing the entire device boot image. Because of this, most OP-TEE Trusted Applications are realized as *file system TAs*. Just like normal world applications, these TAs are compiled into standalone binaries, which are then stored (either encrypted or unencrypted) in the REE file system [OP-19b]. The OP-TEE Client API can then be used to load the stored TA binaries from the file system and execute them in the secure world. Unfortunately though, OP-TEE does not natively support the measurement of dynamically loaded TAs.

To provide a comprehensive measured boot process for TrustZone platforms, we extend OP-TEE with a simple Integrity Measurement Architecture for the secure world. We achieve this by hooking into the `ree_fs_ta_read` method of the OP-TEE trusted OS, which is part of the TA loading routines, and extracting a digest of every TA image that OP-TEE loads from the file system. This digest is then passed to a dedicated IMA module, which we implemented as an early TA. To communicate with this IMA TA, we can conveniently use the `tee_ta_invoke_command` method provided by the OP-TEE kernel. Our IMA then assembles a new TPM command description in binary format, which uses the `TPM2_PCR_Extend` function to extend the received measurement digest into PCR 12. This TPM command is then issued to fTPM using the `TEE_InvokeTACommand` method of the TEE Internal Core API. Finally, our IMA also keeps a measurement log of the extended digests, which normal world applications can query for attestation purposes via a dedicated TA command. Figure 4.6 illustrates how we achieve conducting integrity measurements of dynamically loaded Trusted Applications in OP-TEE. Together with the static measurement of early TAs, this includes all secure world applications into the established chain of trust. Since our IMA module is integrated into OP-TEE as an early TA itself, it is also measured during the Trusted Board Boot as part of BL32.

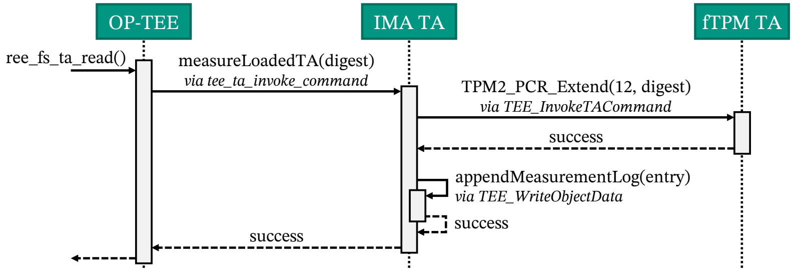


Figure 4.6: Conducting load-time integrity measurements in OP-TEE.

One technical difficulty to overcome when implementing a secure world IMA is that the fTPM TA by default only accepts a single OP-TEE communication session. If this session is already occupied by the fTPM REE driver, the `TEE_InvokeTACommand` shown in fig. 4.6 fails. Extending IMA measurements via the TBB event log in the secure memory is also not possible, because fTPM immediately reads and extends this log during initialization. There are two possible solutions for this problem. One option is to redirect the secure world IMA measurements via the normal world fTPM driver, instead of directly communicating with the fTPM TA. However, this exposes the TA measurements to the REE and hence muddies the chain of trust. Instead, we implement our IMA to act as a proxy for the fTPM TA, which accepts multiple OP-TEE sessions and simply forwards the received TPM commands. We then let the normal world fTPM driver communicate with the fTPM TA via our new IMA module by swapping out the TA's UUID in the fTPM driver configuration.

Implementation and evaluation. To demonstrate the feasibility of this solution, we implemented and evaluated a proof of concept for the described both-world measured boot process on TrustZone platforms. Our proof of concept is based on the Trusted Board Boot reference implementation with fTPM support, which has been provided by ARM as part of the OP-TEE framework.[1] Figure 4.7 illustrates the complete chain of trust for our proposed measured boot process. In contrast to classical measured boot processes with hardware-based TPMs, when measuring TrustZone platforms the resulting chain of trust is split between the normal and the secure world. The CRTM of our proof of concept consists of the BL1 image stored in the device's ROM, as well as the fTPM module. From there, the standard Trusted Board Boot process establishes trust in the secure world firmware and operating system (i.e., OP-TEE), as well as the normal world boot loader (i.e., U-Boot). Extending the chain of trust from U-Boot further to the REE Linux kernel achieves the required boot-time measurements in both worlds. Regarding load-time measurements, we use a dedicated IMA in each world to measure the integrity of dynamically loaded (trusted) applications. As a result, the proposed measured boot process spans over the entire software stack in both the normal and the secure world, and hence is suitable to protect the integrity of usage control enforcement infrastructures on ARM TrustZone platforms. Our complete proof of concept implementation and its documentation is publicly available.[2]

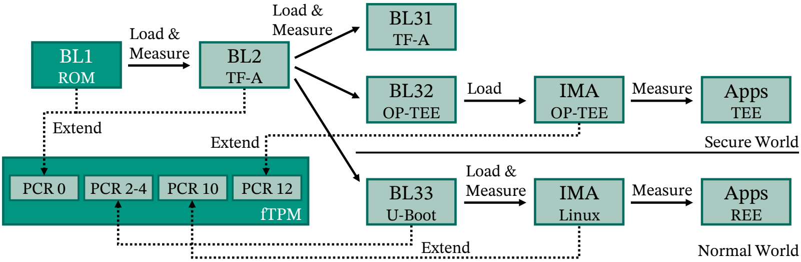


Figure 4.7: Complete chain of trust for both-world measurements on TrustZone platforms.

Finally, we also tested and evaluated the performance of this measured boot process with the widely used ARM Fixed Virtual Platforms[3] (FVP) simulator.

[1] https://github.com/OP-TEE/build/blob/master/fvp.mk *(accessed on 12/08/2023).*

[2] https://gitlab.cc-asp.fraunhofer.de/rafft *(accessed on 12/08/2023).*

[3] https://www.arm.com/products/development-tools/simulation/fixed-virtual-platforms *(accessed on 12/08/2023).*

FVP allows to conduct functionally accurate simulations based on detailed platform models of the various ARM processor architectures. We used the standard Armv8-A foundation model for our tests. Figure 4.8 shows the boot times of a non-measured secure boot process compared to conducting both-world measurements using fTPM. On average, the measured boot process takes about 15 seconds longer than the default boot process. This is mainly due to the overhead for calculating image fingerprints, as well as the delay caused by fTPM having to update the PCR values in the secure memory. Nevertheless, even though the performance impact is noticeable, using a measured boot process on TrustZone platforms is still suitable for our purposes. This is the case especially since the additional overhead influences only the boot and load times, but not the performance of the measured applications themselves.

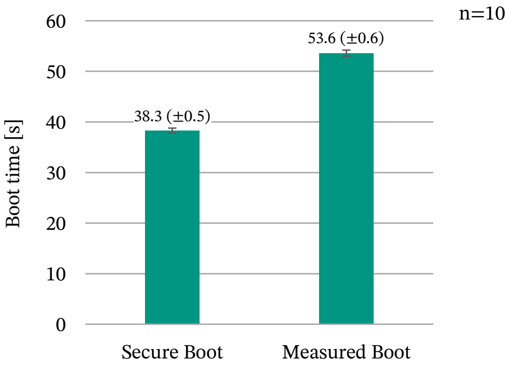


Figure 4.8: Mean TrustZone boot times in seconds using FVP with the Armv8-A model. The standard deviation is given in brackets.

To conclude, we have demonstrated that conducting both-world integrity measurements using fTPM is feasible on TrustZone platforms. While fTPM offers less security guarantees than hardware-based TPMs and requires additional security features such as an RPMB and platform secrets [Raj16], it is currently the best option to achieve comprehensive platform measurements on TrustZone devices. For the remainder of this section, we are left to discuss how we can remotely attest to the integrity measurements, which have been collected by the measured boot process and are stored inside fTPM.

4.4.4 Remote Attestation Protocols

Since ARM does not provide a native mechanism for remotely attesting TrustZone devices, in recent years several custom attestation protocols have been proposed for this purpose. While some of the proposals rely on fTPM as Root of Trust for Storage and Measurement, others use different attestation techniques. In any case, we require a remote attestation protocol that supports both-world measurements and fulfills the previously identified requirements R1 to R5. In this section, we discuss the existing proposals for remote attestation of TrustZone devices and identify a viable solution for our use case.

Shepherd et al. [She17] propose a remote attestation protocol for ARM TrustZone platforms that supports both uni- and bi-directional attestations between OP-TEE Trusted Applications. Furthermore, the attestation protocol uses a generic Diffie-Hellman key exchange to establish secure channels between attested devices. However, this proposal leaves open the question of how to create concrete platform measurements that should be attested. While the authors propose to verify TA binaries using an otherwise unspecified *trusted measurer*, integrity measurements of the normal world are not supported. Since in our use case we need to authenticate the entire usage-controlled TCB instead of just the Trusted Applications, this proposal does not completely fulfill requirement R1. Ahn et al. [Ahn20] leverage the Samsung Knox technology to define a remote attestation protocol for smartphone devices. While Samsung Knox is based on ARM TrustZone, it is a proprietary technology and requires additional trust in the Samsung Knox attestation infrastructure. Furthermore, this proposal does not support the establishment of secure channels to the attested devices (requirement R4). Wang et al. [Wan20] propose a uni-directional remote attestation protocol for TrustZone devices that relies on fTPM instead of Samsung Knox as trust anchor. Unlike the previous protocols, this proposal uses a probe-based integrity measurement architecture to dynamically attest normal world memory contents during runtime. However, this measurement process requires binary rewriting of the normal world applications. Furthermore, this proposal does not include a key agreement process and hence cannot be used to establish encrypted channels to attested software stacks (requirement R4). Ling et al. [Lin21] propose a remote attestation architecture for TrustZone platforms that supports page-based memory integrity measurements of normal world processes. For this, the authors implement a dedicated TA that periodically measures the memory pages of code segments loaded by the normal world operating system. While this proposal supports encrypted channels using a standard TLS connection, the attestation protocol does not link the used TLS certificates to the attestation identities. This makes the protocol vulnerable to man-in-the-middle attacks (see section 4.2.3). Furthermore, the protocol does not support mutual attestations. Finally, Ménétry et al. [Mén22] recently developed a runtime environment for WebAssembly-based Trusted Applications on TrustZone devices, which also includes a remote attestation protocol. The authors use a dedicated TA to load and measure WebAssembly applications in the secure world. The proposed remote attestation protocol establishes secure channels using an ephemeral Diffie-Hellman key exchange that is bound to the attested code identities. Furthermore, the protocol supports both TA-to-TA attestation as well as external (i.e., uni-directional) verification of TA identities. However, similar to the proposal by Shepherd et al., this remote attestation protocol is designed to verify only individual TAs and is hence not feasible for conducting normal world measurements. Since we need to authenticate the complete software stack of a TrustZone device for the application of distributed usage control, this proposal does not completely fulfill our requirement R1 either.

Table 4.5: Overview of remote attestation protocols for TrustZone devices.

Proposal	R1	R2	R3	R4	R5	Remarks
Shepherd et al. [She17]	✗[1]	✓	✓	✓	✓	
Ahn et al. [Ahn20]	✓	✓	✓	✗	-	Only Samsung Knox
Wang et al. [Wan20]	✓	✗	✓	✗	-	Memory attestation
Ling et al. [Lin21]	✓	✗	✓	✗	-	Memory attestation
Ménétry et al. [Mén22]	✗[1]	✓	✓	✓	✓	Only WebAssembly
MSCP (using fTPM)	✓	✓	✓	✓	✓	

[1] Only supports measurement of individual TAs.

Table 4.5 gives an overview of the discussed proposals for remote attestation protocols on ARM TrustZone platforms. In summary, none of the previous proposals fully support our requirements. In addition to the proposals specifically designed for TrustZone, we can also adapt existing remote attestation protocols designed for hardware-based TPMs. This is especially practical when relying on fTPM as RTS and RTM on the TrustZone device anyway. In that case we can leverage the both-world measured boot process described in the previous section to populate the fTPM platform registers, and then use a classical TPM-based attestation protocol to make the resulting PCR values remotely verifiable. Since we have already developed the MSCP protocol to protect distributed usage control infrastructures (see section 4.2.4), we can rely on it to verify the integrity of TrustZone devices as well. As shown previously, MSCP fulfills all of our requirements regarding remote attestation protocols. Hence, we can safely use an MSCP protocol instance that is backed by the fTPM module to attest to the PCR fingerprints of a TrustZone device, which represent the state of both the normal world and the secure world. A further benefit of this approach is that it already ensures protocol compatibility between TPM-protected platforms and TrustZone devices, which simplifies the challenge of specifying a heterogeneous remote attestation protocol.

4.5 Heterogeneous Remote Attestation

In the previous sections we used TPMs, Intel SGX, and ARM TrustZone to develop remote attestation mechanisms that are suitable for protecting the integrity of distributed usage control infrastructures. Now we are left with the challenge of conducting heterogeneous remote attestations *between* these technologies as well. In order to design a comprehensive and trustworthy distributed usage control system, it is essential to support the integrity verification of remote software stacks independently of the underlying trusted computing technologies. However, this issue has only recently become a topic of interest both in academia and the industry [Ott23, Int23c]. To our knowledge, at the time of this thesis there is no suitable attestation protocol available that can provide cross-technology integrity verification of trusted platforms protected by TPMs, Intel SGX, and ARM TrustZone. While the IDSCP attestation protocol introduced in section 4.2.2 also offers support for AMD SEV in addition to TPMs [Bro22], as we have shown in section 4.2.3 this protocol is vulnerable to attacks by internal adversaries when used to protect distributed usage control infrastructures. Very recently, Ott et al. proposed an attestation framework that supports TPMs, AMD SEV, and the ARM Platform Security Architecture [Ott23]. Furthermore, in 2022 Intel announced *Project Amber*[1]. The goal of Project Amber is to provide a confidential computing framework for the Intel ecosystem, which simplifies the development cycle of TEE-based cloud applications. This framework will also include a universal and technology-independent remote attestation service. However, Project Amber will initially be limited to Intel's SGX and TDX technologies [Int23c]. Finally, a similar approach has also been taken by Google with the Enclave Key Exchange Protocol (EKEP) [Asy21a]. As introduced in section 4.3.2, EKEP is the default attestation protocol for the Asylo framework. Asylo also has the goal of providing a technology-independent development environment for TEE-based applications. Because of this, the design of EKEP has been kept largely independent of a concrete TEE provider. However, currently EKEP is usable only for SGX-based local and remote attestation. Since the EKEP specification is publicly available and has already been formally verified [Roe22], we choose to adapt it for our use case instead of developing a dedicated heterogeneous attestation protocol from scratch.

In the remainder of this section, we specify and evaluate a heterogeneous remote attestation protocol that is suitable for our trustworthy distributed usage control framework. For this, we first identify additional requirements that such a heterogeneous attestation protocol needs to fulfill. Then we extend the existing EKEP protocol with heterogeneous attestation capabilities between TPMs, SGX, and ARM TrustZone platforms. Finally, we evaluate the resulting protocol regarding security and performance.

[1] https://projectamber.intel.com/ *(accessed on 12/08/2023).*

4.5.1 Additional Requirements

Our goal is to achieve a heterogeneous remote attestation protocol that allows the transparent integrity verification of software stacks protected by different trusted computing technologies. For this, in addition to the previously specified generic protocol requirements R1 to R5, we define three more requirements H1 to H3 that are specific to *heterogeneous* attestation protocols.

(H1) **Heterogeneous attestations:** A single instance of a heterogeneous attestation protocol must be able to deal with different trusted computing technologies on both endpoints. In order to realize distributed usage control scenarios, we require support for at least TPMs, Intel SGX, and ARM TrustZone. Furthermore, the protocol must be able to represent generic code identities independently of the underlying trusted computing hardware. For example, Intel SGX uses enclave fingerprints while TPMs and (fTPM-based) TrustZone platforms are represented by specific PCR values. Nevertheless, after a successful protocol handshake the individual code identities of both sides have to be mutually verified regardless of their representation.

(H2) **Mechanism negotiation:** The heterogeneous attestation protocol must support the automatic and transparent negotiation of the attestation mechanisms that should be used for the handshake. Classical attestation protocols implicitly know the attestation mechanism that is used by the remote peer. However, a heterogeneous attestation protocol must have the ability to explicitly negotiate the attestation mechanisms used by both endpoints during the protocol handshake. This is necessary to allow both peers to adequately validate the received attestation evidence.

(H3) **Hardware agnosticism:** Finally, the heterogeneous attestation protocol definition must be kept agnostic of the underlying trusted computing hardware. This means that there should be a common handshake sequence that is independent of the used attestation mechanisms. Only the exchanged attestation evidence itself should be hardware specific. This simplifies the protocol definition, improves flexibility, and ensures extensibility with additional trusted computing technologies in the future. Furthermore, the protocol should also be designed and implemented in a platform-independent fashion, in order to support heterogeneous attestations of a wide variety of different system architectures.

4.5.2 The EKEP Protocol

As previously mentioned, we build the heterogeneous attestation protocol for protecting the integrity of our distributed usage control system on the EKEP handshake. EKEP is used as an SGX-based remote attestation protocol in Google's Asylo framework [Asy21a]. Since EKEP is essentially a modified version of the ALTS transport encryption protocol, its main task is to establish mutually authenticated and encrypted communication channels. To achieve this, the EKEP handshake by default includes an ephemeral ECDH key exchange between the channel endpoints. However, unlike other transport encryption protocols such as TLS, EKEP does not determine by what exact mechanisms this key exchange should be authenticated. Instead, EKEP transmits generic *assertions* between the channel endpoints, which must be individually interpreted by the specific protocol implementation. In essence, EKEP assertions are cryptographic proofs of endpoint identities, which may be based on classical CA-issued certificates as well as attestation information. Furthermore, EKEP includes a mechanism for negotiating the types of assertions that should be exchanged during a handshake. This allows EKEP to support both local and remote SGX-based attestations in a single protocol instance. However, these properties also make EKEP a very good starting point for implementing a heterogeneous attestation protocol.

In this section we give a brief overview of the EKEP handshake, before describing how we adapt it to support heterogeneous attestations between TPMs, Intel SGX, and ARM TrustZone platforms. The complete protocol specification of EKEP can be found in [Asy21a].

The EKEP handshake. The EKEP handshake consists of three phases and a total of six messages between client and server. Figure 4.9 gives an overview of the protocol sequence. The first step of the handshake is the PRECOMMIT phase. In this phase, both the client and the server exchange fresh nonces and agree on the cipher suites as well as the assertion mechanisms that should be used in the remainder of the handshake. For this, the client initially sends a list of mechanisms that it can offer and a list of mechanisms that it is willing to accept from the server. The server then selects a suitable subset from the list of requested mechanisms and transmits them to the client as the server's offers. Similarly, an acceptable subset of the client's offers is selected as the server's requests. That way client and server negotiate a list of assertion mechanisms that is acceptable for both sides. If no agreement can be reached (i.e., if one of the subsets is empty), the handshake terminates.

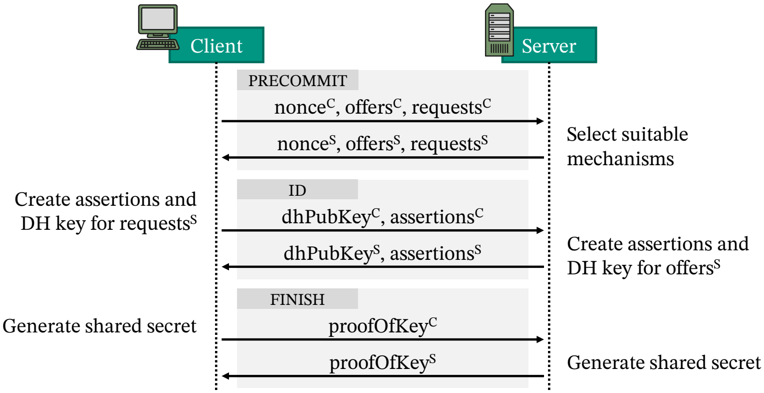

Client
Server
PRECOMMIT
nonce^C, offers^C, requests^C
nonce^S, offers^S, requests^S
Select suitable mechanisms
ID
Create assertions and DH key for requests^S
dhPubKey^C, assertions^C
dhPubKey^S, assertions^S
Create assertions and DH key for offers^S
FINISH
Generate shared secret
proofOfKey^C
proofOfKey^S
Generate shared secret

Figure 4.9: Overview of the EKEP protocol handshake. Own illustration after [Asy21a].

The second protocol phase then establishes the identities of both endpoints and conducts the ephemeral key exchange (*ID* phase). For this, both peers create new ephemeral Diffie-Hellman key pairs according to the previously negotiated cipher suite. Furthermore, the endpoints also generate the assertions that have been agreed on during the PRECOMMIT phase. To prevent man-in-the-middle attacks, these assertions must be bound to both the generated ephemeral keys and the previously exchanged nonces. EKEP then serializes the generated assertions into byte blobs and transmits them to the remote peer for verification. Finally, the last two messages of the handshake are responsible for establishing the encrypted channel (*FINISH* phase). First, both endpoints use the authenticated Diffie-Hellman keys to generate a shared secret and derive a symmetric session key. To provide a proof of key ownership, both endpoints also exchange a symmetric signature of the protocol transcript under the session key. Once these final checks have been completed successfully, the session key is used to establish a transparent, authenticated encryption layer based on 128 bit AES-GCM.

Supported assertion authorities. As shown in fig. 4.9, the EKEP handshake exchanges attestation information in the form of binary *assertions*. To manage the types of usable assertions over multiple handshakes, EKEP introduces the concept of *assertion authorities*. An assertion authority is identified by a unique name and is associated with a compatible assertion generator and verifier [Asy21a]. EKEP endpoints use these generators and verifiers to create and validate assertion objects that are transmitted during the protocol handshake. At the time of this thesis, EKEP supports three different assertion authorities. First of all, EKEP endpoints can assert their identity by proving ownership of a certificate. Similarly to common TLS certificates, these certificates are usually authenticated by a Public Key Infrastructure. However, EKEP does not require the certificate identities to be bound to the endpoint's network address or host name. This allows for a flexible, certificate-based authentication of EKEP endpoints. Furthermore, since EKEP is an SGX-based remote attestation protocol, it also includes assertion authorities for the local and remote attestation of SGX enclaves. Local SGX assertions mainly consist of the attestation report that is being created by the EREPORT SGX instruction. However, EKEP also includes a unique attestation domain identifier into local assertions, in order to distinguish different physical SGX platforms from one another. Remote SGX assertions consist of a signed data structure, which includes the attested enclave identity and the cryptographic signature created by the SGX quoting enclave. Since the EKEP protocol relies on the DCAP attestation primitives [Asy21c], endpoints can furthermore specify the acceptable intermediate certificates for the attestation key. The EKEP verifier then validates the specified certificate chain to the root certificate provided by Intel. Finally, EKEP supports two different quoting enclaves that can act as assertion generators when conducting remote attestations. Endpoints can either use the official quoting enclave by Intel, or rely on the more comprehensive Assertion Generator Enclave (AGE) that is included in the Asylo framework.

In the end, the concept of transmitting assertions during EKEP handshakes allows endpoints to transparently authenticate each other using either certificates or SGX enclave identities. Furthermore, the flexible design of EKEP makes it possible to easily configure SGX enclaves for both uni- and bi-directional, as well as local and remote attestation scenarios.

4.5.3 Achieving Heterogeneous Attestations

As we have seen, the EKEP protocol currently understands only SGX-based attestation evidence. Our goal is to extend EKEP with support for heterogeneous attestations across TPMs, SGX, and ARM TrustZone platforms.

Adding support for TPM 2.0 assertions. To support remote attestations between SGX enclaves and TPM-protected platforms, we need to specify an assertion generator for EKEP that is capable of creating and serializing TPM-based quotes. Furthermore, we also require a corresponding verifier that parses the received assertion blobs, extracts the original quotes, and finally validates the asserted TPM-based identities. This allows us to leverage the existing negotiation mechanism of EKEP to establish connections between TPM-protected systems and SGX enclaves. One pitfall to consider in this endeavor is the proper binding of the ephemeral Diffie-Hellman public keys established by EKEP to the TPM-specific assertions. By default, EKEP authenticates the ephemeral public keys exchanged in the `PRECOMMIT` messages by hashing them (together with the protocol transcript that includes the received nonce) into the qualifying data used for the SGX attestation reports.[1] However, as we

[1] See the EKEP handshaker implementation at https://github.com/google/asylo/blob/master/asylo/grpc/auth/core/client_ekep_handshaker.cc#L465 *(accessed on 12/08/2023)*.

have shown in section 4.2.3, binding an ephemeral key to a trusted platform by hashing it into a TPM-based quote can lead to vulnerabilities against nonce-data attacks. This is because on TPM-protected systems we need to assume the existence of a secondary attestation endpoint that attackers can use to generate a valid quote for arbitrary qualifying data. A single SGX enclave, on the other hand, does not give out any other valid attestation reports for externally chosen qualifying data. As a result of this, naively including heterogeneous attestation into EKEP by relying on the built-in method for ephemeral key authentication leads to an insecure protocol implementation.

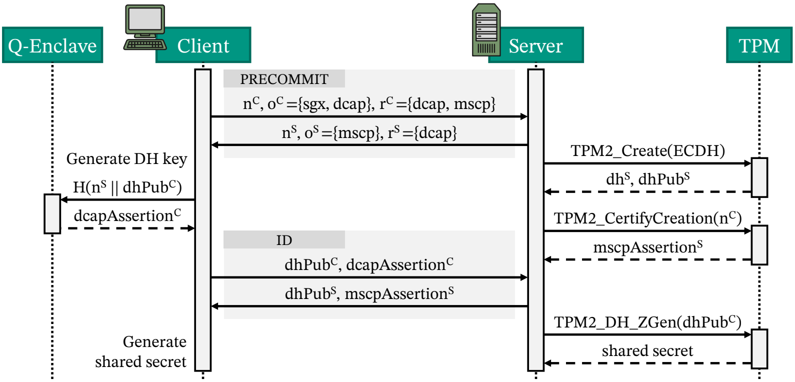


Figure 4.10: Heterogeneous remote attestation between TPMs and SGX enclaves using EKEP. The FINISH phase of the handshake is omitted.

We solve this issue by adding the complete MSCP handshake, as presented in section 4.2.4, to the EKEP protocol. This way the ephemeral Diffie-Hellman keys are properly authenticated even on TPM-based platforms, using either the PCRs (in case of TPM-external key establishment) or a dedicated key certification (in case of TPM-internal key establishment). Figure 4.10 shows the resulting handshake between an SGX enclave and a TPM platform using the MSCP variant with *internal* key establishment. During the PRECOMMIT phase of EKEP, both endpoints exchange their random nonces and negotiate the heterogeneous attestation mechanisms. Typically, SGX enclaves offer both local SGX reports and DCAP remote attestation evidence, while TPM-based endpoints can only offer MSCP assertions. After the negotiation, both peers generate their respective ephemeral ECDH key pairs and the corresponding remote assertions. For this, as implemented in the EKEP handshaker, the SGX endpoint contacts the local quoting enclave and uses a hash of the received nonce together with the ephemeral public key as qualifying data. The TPM endpoint instead calls `TPM2_Create` to generate its ephemeral ECDH key pair and `TPM2_CertifyCreation` to generate a remote assertion of the platform identity that is bound to both the received nonce and the created ephemeral key (cf. section 4.2.4). Finally, the handshake is completed by calculating the shared secret. While the SGX enclave can do that using the existing EKEP implementation, the TPM endpoint instead relies on the `TPM2_ECDH_ZGen` command.

One remaining technical issue when integrating MSCP into EKEP using TPM-internal key establishment, as shown in fig. 4.10, is the compatibility of the elliptic curves and cipher suites used for the key agreement. Most hardware TPMs only implement the common NIST P-256 and P-384 elliptic curves [Che23, pp. 10–12], because they are mandated by the TPM 2.0 platform profile [Tru20, p. 18]. EKEP, on the other hand, relies on the more modern Curve25519 [Lan16] for the ephemeral ECDH key exchange.[1] To ensure that the heterogeneous attestation endpoints always find a common elliptic curve for the key agreement, the NIST curves would have to be integrated into the EKEP implementation as well. Since EKEP includes cipher suite negotiation during the protocol handshake anyway, this integration is not a major problem. Alternatively, to avoid such compatibility issues altogether, we can also use the MSCP variant with *external* key establishment for the heterogeneous attestation. This handshake variant follows the same principle as shown in fig. 4.10, but uses `TPM2_Quote` instead of `TPM2_CertifyCreation` to generate the MSCP remote assertion (cf. section 4.2.4). This has the benefit of easier protocol integration, because it allows us to re-use the key establishment functions already available in EKEP. However, we still need to ensure the proper authentication of the generated ephemeral ECDH keys on the TPM platform by extending them into PCR 16, in order to prevent nonce-data attacks.

[1] See https://github.com/google/asylo/blob/master/asylo/grpc/auth/core/ekep_crypto.cc *(accessed on 01/22/2024)*.

Adding support for TrustZone assertions. Integrating MSCP remote assertions into the EKEP protocol stack allows us to establish mutually verified communication channels between SGX enclaves and TPM-protected platforms. Since we rely on an fTPM-based trusted boot process to measure the integrity of ARM TrustZone devices (see section 4.4), the described protocol modifications can principally be used to conduct heterogeneous attestations of TrustZone endpoints as well. However, there are some caveats to consider. As discussed previously in section 4.4.2, we can distinguish two different usage control deployment scenarios on TrustZone devices. In most cases, our data processing applications (i.e., the remote attestation endpoints) will reside in the normal world of TrustZone. For such REE-based endpoints we can simply re-use the described MSCP protocol implementation, since the fTPM TA can be accessed over the same interface as a normal hardware TPM by means of a normal world kernel module[1]. However, one important difference of using MSCP with a TrustZone-protected fTPM module instead of hardware TPMs is that the resulting assertions represent the software stack of *both* the normal and the secure world of the device. This is because in order to properly secure our usage control infrastructure, we always need to conduct integrity measurements of both the REE and the TEE applications (see section 4.4.3).

In those cases where we want to use MSCP to connect to data processing applications that are running inside the secure world (i.e., the TEE) of TrustZone, some additional issues must be considered. First, the communication between a TEE-based assertion generator and the fTPM module must be achieved differently. Since the secure world operating system does not provide the usual TPM command interface on kernel level, we now need to communicate with the fTPM module directly using the `TEE_InvokeTACommand` function provided by the OP-TEE Internal Core API [Glo21, pp. 101–102]. Furthermore, there is also the open question of how to distinguish attestation endpoints inside the TEE from those running in the normal world of the TrustZone platform. Since the fTPM module must be available in the normal as well as the secure world, both REE and TEE applications can use it to generate valid quotes that are

[1] https://github.com/OP-TEE/build/blob/master/br-ext/package/linux_ftpm_mod_ext/linux_ftpm_mod_ext.mk *(accessed on 12/18/2023).*

signed by the fTPM-managed attestation key. As a result, fTPM-based assertions alone do not reveal if the used communication endpoint is located inside the normal or the secure world of the attested platform. However, this distinction is relevant for the verifier because the TrustZone TEE obviously offers better isolation than the REE (cf. figs. 4.4c and 4.4d).

This issue can be solved by introducing an additional authentication mechanism to the generated MSCP assertions, which relies on non-fTPM keys that are not accessible outside the secure world. The best option for such an authentication is to leverage the TA-specific symmetric keys, which OP-TEE derives from the TrustZone platform secrets for the purpose of providing secure storage [OP-19a]. Since these keys are unique for each TA and are only available inside the TEE, they could be used to generate a symmetric HMAC signature of the transmitted EKEP challenge in addition to the actual attestation evidence generated by fTPM. The resulting assertion would then prove to the verifier not only that the attested endpoint is located on a TrustZone platform with a certain software stack, but also that the secure channel terminates inside the TEE instead of the REE. Unfortunately, to our knowledge OP-TEE currently does not provide the necessary APIs for Trusted Applications to directly derive such HMAC keys from the platform secrets in the user space of the secure world. This makes it difficult to implement such an HMAC authentication without deeper changes of the OP-TEE trusted operating system. Due to these additional complications, in this thesis we limit our proof of concept to REE-based heterogeneous communication endpoints on TrustZone devices, and leave the implementation of TEE endpoints for future work.

Implementation. Finally, we also implemented and evaluated our proposed EKEP-based heterogeneous remote attestation handshake. For this we mainly added the described assertion generators and verifiers for the MSCP assertion authority to the EKEP protocol implementation. However, since EKEP by default only supports SGX-based remote attestation, this also required the definition of new TPM-based data structures to be transmitted during the ID phase of the handshake (cf. fig. 4.10). More concretely, we defined data structures that can represent TPM-based platform identities, as well as the corresponding remote assertions corroborating these identities. During the EKEP handshake, these representations are serialized and transmitted as binary data blobs. Just like the SGX-based data types already used in EKEP, we rely on the Protobuf[1] library for the definition and serialization of these new data types. Listing 4.1 shows the TpmIdentity and TpmRemoteAssertion data structures that are used by our MSCP assertion generator and verifier.

Listing 4.1: TPM-based assertion and identity description for EKEP.

```
syntax = "proto2";

// A high-level representation of the identity of a TPM machine.
message TpmIdentity {
  // The TPM system fingerprint. Required.
  optional string system_fingerprint = 1;
  // The TPM system's PCR values as hex strings. Required.
  map<int32, string> system_pcrs = 2;
  // The stored IMA measurement logs. Optional.
  repeated string sml_entries = 3;
}

// A high-level representation of a TPM ECDH key certification.
message TpmKeyCertification {
  bytes cert_info = 1;
  bytes cert_data = 2;
}

// A cryptographically verifiable TPM remote assertion.
message TpmRemoteAssertion {
  // The asserted TPM identity, including the PCR values. Required.
  optional TpmIdentity identity = 1;
  // The signed attestation information over the TPM identity.
  // Can be a TPM quote or an ECDH key certification. Required.
  oneof attestation_info {
    bytes quote = 2;
    TpmKeyCertification key_cert = 3;
  }
  // The certificate of the used attestation key. Required.
  optional bytes attestation_cert = 4;
  // Proof of TEE residency. Optional.
  optional bytes proof_of_tee = 5;
}
```

[1] https://protobuf.dev/ *(accessed on 12/08/2023).*

Lines 4 to 11 of listing 4.1 define the TPM-based identity of an attested platform. The TPM identity includes the list of attested PCR values and a unique system fingerprint, which we calculate as the SHA-256 digest of the used attestation key. Furthermore, the assertion generator also has the option to include the collected measurement logs into the identity description. This avoids the need to separately transmit them to the verifier for integrity validation. Lines 20 to 33 define the TPM-based remote assertion that is transmitted during the ID phase of the handshake. In addition to the asserted platform identity, this data structure also includes the attestation evidence as generated by the TPM. Depending on the used MSCP variant, this consists of either a TPM quote (in case of external key establishment) or a key certification blob (in case of internal key establishment). In both cases the used attestation public key is included in the form of a verifiable certificate. Finally, in preparation of supporting TEE-based endpoints in the future, the assertion description also reserves a field for the additional symmetric signature proving TEE residency on TrustZone platforms.

We implemented the MSCP assertion generator and verifier modules for EKEP in C++. The resulting code is available online.[1] To connect the assertion generator to the local platform TPM and validate the resulting attestation evidence, we used the Microsoft TSS.MSR[2] software stack. Furthermore, we also patched our MSCP assertion verifier into the Asylo framework to allow existing SGX enclaves the heterogeneous attestation of remote TPM and TrustZone platforms. However, as mentioned before, so far our implementation supports only REE-based endpoints on TrustZone platforms. Also note that the resulting heterogeneous handshake encompasses only the cryptographic validation of the asserted platform identities, i.e., the PCR values themselves. We discuss how to validate the legitimacy of the attested PCR values later as part of chapter 5.

[1] https://gitlab.cc-asp.fraunhofer.de/datasov/lib/-/tree/master/smarttc *(accessed on 12/08/2023).*

[2] https://github.com/microsoft/TSS.MSR *(accessed on 01/23/2024).*

4.5.4 Protocol Evaluation

To evaluate the proposed heterogeneous remote attestation protocol, we first conduct a brief security analysis and then determine the handshake latencies that can be achieved using the different trusted platforms.

Security discussion. The modified EKEP handshake fulfills all security requirements that we expect from a generic remote attestation protocol. It is capable of mutually authenticating remote trusted platforms by exchanging and verifying attestation evidence (requirements R1 and R2). Furthermore, since EKEP always conducts a dedicated ECDH key agreement, it also establishes encrypted communication channels with perfect forward secrecy (requirements R4 and R5). Since the protocol relies on AES-GCM as an authenticated encryption mechanism for the channel communication, it can also protect the integrity of exchanged data against replay attacks (requirement R3). Regarding the additional requirements for heterogeneous attestation protocols, our proposal can transparently authenticate trusted platforms based on TPMs, Intel SGX, and ARM TrustZone (requirement H1). For the mechanism negotiation (requirement H2) we rely on the PRECOMMIT phase already defined by EKEP. Finally, the protocol handshake is also agnostic of the underlying hardware, since the different assertion types are serialized and transmitted as generic binary blobs (requirement H3).

We also considered the formal verification of our heterogeneous attestation protocol. The EKEP protocol handshake itself has already been formally verified by Roeder et al. using ProVerif [Roe22]. Their model of EKEP represents the entire protocol flow, including the key establishment phase, and can prove the security properties of (mutual) authentication and perfect forward secrecy. In their formal model, Roeder et al. abstract from concrete remote attestation mechanisms by representing trusted identities using secret HMAC keys. The protocol then creates valid attestation evidence only if each endpoint can prove its identity by providing a symmetric signature under these platform secrets. While this adequately models the SGX-based attestation process using a quoting enclave, it does not fit our TPM-based attestation mechanism. On TPM platforms, the attestation report is usually signed directly by the TPM instead of first being authenticated via a quoting enclave. To accommodate this difference, we updated the existing formal model of EKEP and included a dedicated TPM assertion generator that directly signs attestation evidence. Furthermore, we also included a new rule that extends the attacker model by providing TPM-based quoting oracles. As previously discussed in section 4.2.4 for the protocol formalization of MSCP, such a rule is necessary to be able to detect possible vulnerabilities against nonce-data attacks. By verifying the resulting formal protocol model using ProVerif, we show that our inclusion of heterogeneous attestations into EKEP does not break the protocol's security guarantees. We present and discuss our concrete modifications of the existing formal model of EKEP in greater detail in appendix B.3.

Protocol performance. To evaluate the performance of the implemented remote attestation protocol, we conducted several tests of the connection latencies using different platform configurations.

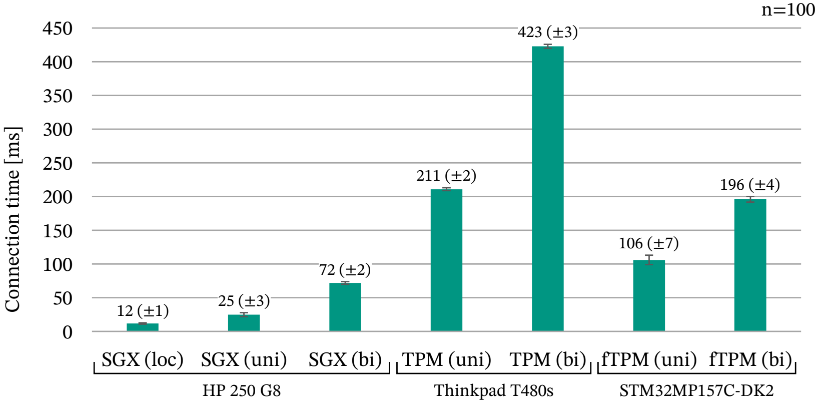


Figure 4.11: Mean connection times for TPM-, SGX- and TrustZone-based remote attestations in milliseconds. The tests have been executed on the indicated evaluation platforms. The standard deviation is given in brackets.

Figure 4.11 shows the results for TPM-, SGX-, and TrustZone-based remote attestations with a sample size of n=100. The first three tests have been executed on an HP 250 G8 (i7-1065G7, 8GB RAM, Ubuntu 22.04) using the SGX hardware in release mode. As we can see, an SGX local attestation takes about 12 milliseconds to complete, while an SGX remote attestation requires 25 milliseconds uni-directionally and 72 milliseconds bi-directionally. This overhead is caused by the additional communication with the quoting enclave and the creation of the asymmetric signature by the SGX remote assertion generator. We executed our tests for the TPM-based attestation on a Thinkpad T480s (i7-8550U, 16GB RAM, Ubuntu 22.04) using the on-board Infineon SLB9670[1] hardware TPM 2.0. The TPM-based assertions are generated according to the MSCP protocol with *external* key establishment. As expected, in this configuration the attestations take much longer due to the relatively slow TPM hardware. A TPM-based EKEP handshake takes about 211 milliseconds uni-directionally and 423 milliseconds bi-directionally, which is still adequate for our purposes. Finally, we also evaluated the remote attestation of TrustZone devices using the MSCP protocol together with fTPM. We executed these tests in the normal world of an STM32MP157C-DK2[2] prototyping board. This board features an ARM Cortex-A7 32 bit microprocessor that fully implements TrustZone.[3] With this setup, we find that the remote attestations can be conducted about twice as fast as when using hardware TPMs. Generating fTPM-based assertion evidence on the test platform takes about 106 milliseconds uni-directionally and 196 milliseconds bi-directionally. Note that this performance improvement compared to hardware TPMs is consistent with the original fTPM evaluation results by Raj et al. [Raj16]. However, for this we operated fTPM as a dedicated normal world service[4] instead of an OP-TEE TA[5]. Running fTPM as the latter resulted in increased attestation

[1] https://www.infineon.com/cms/de/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/slb-9670vq2.0/ *(accessed on 12/08/2023).*

[2] https://www.st.com/resource/en/data_brief/stm32mp157c-dk2.pdf *(accessed on 12/21/2023).*

[3] https://www.st.com/resource/en/datasheet/stm32mp157c.pdf *(accessed on 12/21/2023)*

[4] https://github.com/microsoft/ms-tpm-20-ref/tree/main/TPMCmd/Simulator *(accessed on 12/25/2023).*

[5] https://github.com/microsoft/ms-tpm-20-ref/tree/main/Samples/ARM32-FirmwareTPM *(accessed on 12/25/2023).*

times of about 1.1 seconds uni-directionally and 2.1 seconds bi-directionally. While some additional overhead is to be expected due to the necessary context switches between the REE and TEE, this significant deterioration in the fTPM performance stands in contrast to the results given in [Raj16]. Hence, we conclude that this issue is likely caused by an unoptimized build process when integrating the fTPM application into OP-TEE. Furthermore, we noted that the normal world fTPM service uses OpenSSL as underlying cryptographic library, while the OP-TEE build relies on WolfSSL[1] instead. This difference may also contribute to the observed performance discrepancy. In any case, we consider the optimization of the fTPM build process for OP-TEE as out of scope for this work.

In addition to the single-platform test cases, we also evaluated the performance of the implemented protocol when using it to heterogeneously attest TPM-protected systems, SGX enclaves, and ARM TrustZone devices. To achieve this, we connected the three presented evaluation platforms to a common network and executed the attestation protocol over Gigabit Ethernet. Figure 4.12 shows the mean connection times of the implemented heterogeneous attestation protocol for all three technological combinations.

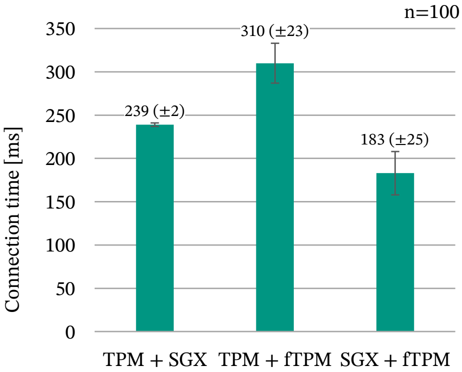


Figure 4.12: Mean connection times for heterogeneous remote attestations in milliseconds. The tests have been executed on the same platforms as shown in fig. 4.11. The standard deviation is given in brackets.

[1] https://www.wolfssl.com/ *(accessed on 01/14/2024).*

As we can see, the execution times for (bi-directional) heterogeneous remote attestations largely follow the previously determined (uni-directional) attestation times for the individual technologies (cf. fig. 4.11). Hence, a heterogeneous attestation between the TPM-based and SGX-based evaluation system takes about 239 milliseconds, while the attestation between the TPM and the TrustZone device requires 310 milliseconds. With 183 milliseconds, only the attestation between the TrustZone device and the SGX enclave requires a bit more time than the individual uni-directional attestations would suggest. This is likely caused by the necessary verification of the SGX-based assertion evidence on the relatively slow TrustZone board. Nevertheless, all heterogeneous attestations still execute fast enough for our purposes.

To conclude, in our evaluation we showed that the performance of the developed heterogeneous remote attestation protocol is sufficient for our use case of securing distributed usage control infrastructures. As expected, we found that generating assertions on SGX platforms is much faster than when using hardware TPMs. Still, our EKEP-based implementation of the MSCP handshake for attesting TPM-protected systems performs better than the previously described Java version of the protocol (cf. fig. 4.3). This is likely due to the more efficient implementation using C++ instead of the Java Crypto API. In addition, we showed that generating fTPM-based assertions on ARM TrustZone platforms is also faster than using hardware TPMs, which is consistent with the original fTPM evaluation results [Raj16]. The improved performance of fTPM compared to hardware TPMs can be explained by the use of a full application processor with access to fast memory. As a result, the efficiency of fTPM-based remote attestation directly depends on the capabilities of the underlying microprocessor. Because of this, our evaluation results are not representative for other ARM TrustZone devices. However, since the used STM32MP157C-DK2 board is an older model with a relatively weak processor, we can still conclude that the implemented heterogeneous remote attestation protocol is fast enough for our purposes. We evaluate the connection latencies in a complete distributed usage control scenario later in chapter 7.

4.6 Design Alternatives

This section briefly discusses the advantages and drawbacks of two alternative approaches in designing remote attestation protocols that can protect distributed usage control infrastructures.

Using implicit remote attestation. The attestation protocols developed and presented in this chapter all leverage *explicit* attestation to establish the code identity of remote endpoints. This means that they explicitly generate cryptographic proof of their code identity during each attestation handshake, for example using `TPM2_Quote` on TPM-protected systems or the `EREPORT` instruction (plus a signature from the quoting enclave) on SGX processors. However, remote attestations can also be conducted *implicitly*. As mentioned briefly in section 4.2.2, during an implicit attestation the used cryptographic key itself serves as proof of platform integrity. To achieve this, a TPM-based attestation key can be bound to the desired platform state using the `TPM2_PolicyPCR` command [Tru19d, p. 230]. SGX platforms, on the other hand, do not offer a specific implicit attestation functionality. Instead, implicit attestation can be achieved by initially provisioning some secret key material to an enclave via an explicitly attested channel, which is then sealed locally to the enclave identity. If a remote endpoint can subsequently authenticate itself using the previously provisioned secret, the enclave's code identity can be implicitly deduced without requiring another explicit remote attestation using `EREPORT`.

The main advantage of implicit over explicit attestation lies in its greater flexibility. For example, implicit attestation supports the use of encryption keys instead of signature keys as proof of platform integrity, which then allows to directly decrypt critical data with a private key that is usable only on a genuine trusted platform. Depending on the use case, this may save the need for establishing additional attested and encrypted communication channels. However, using implicit attestation also increases the scope and complexity of the attestation key provisioning process. Furthermore, since an implicit attestation key is cryptographically bound to the platform state, it must be updated each time the (legitimate) code base changes. While TPMs offer non-brittle

PCRs [Art15, pp. 34–35] to somewhat alleviate this issue, on SGX platforms such convenience functions are not available. Hence, updating SGX enclaves would require either the provisioning of a completely new implicit attestation key, or the implementation of a dedicated update mechanism to re-seal the previously provisioned key on the fly. Due to these drawbacks, and since classical explicit remote attestation is sufficient to protect distributed usage control systems, we avoid the additional overhead of implicit attestations.

Communication via domain sockets. The heterogeneous remote attestation protocol described in section 4.5.3 allows to establish attested communication channels between different trusted computing platforms. This includes conducting *local* instead of remote attestations if two communicating SGX enclaves are running on the same processor. However, due to the lack of a dedicated execution environment, TPM-based platforms have no concept analogous to local attestation. Because of this, two usage control components running on the same TPM-protected system will always conduct a full remote attestation over the network stack to communicate. From a performance standpoint it could be beneficial to allow such components a more direct way of communication, for example via the system memory. In Unix-based environments this can be achieved rather easily by configuring the use of *Unix domain sockets (UDS)*. Domain sockets can be used just like normal network sockets, except they transparently route the transmitted messages through the inter-process communication (IPC) layer of the kernel instead of the network stack. Because domain sockets can only be used to communicate locally, and since the TCBs of TPM-protected platforms consist of the entire system anyway, it may also be possible to safely omit the overhead of the attested and encrypted communication channel in these cases. However, this optimistically assumes that it is not feasible for a privileged attacker, for example a malicious administrator, to intercept the communication over domain sockets without impacting the platform measurements. Furthermore, the performance benefit of this modification would be limited to the initial attestation of new system components, since afterwards we can cache the established communication channels anyway (see section 5.2). Because of these drawbacks, we do not adopt the use of domain sockets for our proof of concept.

4.7 Conclusion

In this chapter we analyzed the capabilities of TPMs, Intel SGX, and ARM TrustZone to fulfill the security requirements necessary for our distributed usage control and provenance tracking system. Our analysis shows that existing TPM-based attestation protocols currently used in virtual data spaces are vulnerable against nonce-data attacks by malicious component administrators, who are the most capable adversaries expected in our system. To solve this issue, we proposed the use of a TPM-internal key exchange for establishing attested communication channels, and developed a corresponding protocol called MSCP. Our security analysis of MSCP, which includes a formal protocol verification using the Tamarin theorem prover, shows that the protocol effectively mitigates the issue of nonce-data attacks. To evaluate the performance of our protocol, we implemented it in a Java-based environment. Our results show that the TPM-internal key exchange causes a significant performance overhead compared to existing protocols. However, this can be alleviated by authenticating the key exchange with a resettable PCR register, while still retaining security against nonce-data attacks. In summary, these results constitute our research contribution RC3.

Furthermore, we also implemented a proof of concept for a trusted boot process on ARM TrustZone platforms that supports conducting load-time integrity measurements of applications in *both* the normal and the secure world of the device (research contribution RC4). As final contribution in this chapter, we designed a heterogeneous remote attestation protocol (research contribution RC5). We achieved this by building on the existing EKEP protocol for SGX environments and extending it with TPM-based attestation evidence designed according to our MSCP protocol. We integrated our proposal into the Asylo trusted computing framework and evaluated the performance of the resulting heterogeneous attestation protocol by connecting Intel SGX enclaves, endpoints protected by hardware TPMs, and fTPM-based ARM TrustZone devices. Our results show that the heterogeneous attestation does not cause any significant overhead compared to single-technology variants, and as such is feasible for our application scenario.

To conclude, in this chapter we developed the necessary technical measures to enforce distributed usage control and provenance tracking even against privileged adversaries in remote systems. As such, we achieved goal 2 of the overall thesis objective. In addition, we also developed a solution for applying these technical measures in *heterogeneous* trusted computing environments, which achieves goal 3 as well. With these results, we have now collected all building blocks that are necessary to implement a trustworthy distributed usage control and provenance tracking framework. However, our analysis also showed that the rollback prevention mechanisms provided by the considered trusted computing technologies are difficult to apply to our system. Hence we will also design a simple, dedicated rollback protection feature as part of our system implementation in the next chapter.

5 A Trustworthy Distributed Usage Control Framework

In this chapter we consolidate the research contributions achieved so far by describing our concrete proof of concept implementation for a trustworthy distributed usage control and provenance tracking system. Our implemented framework is called *DataSov*.

We begin in section 5.1 with a description of the DataSov framework's fundamental architecture and design. In section 5.2 we then show how our proposed component authentication and remote attestation scheme is integrated into the framework. Finally, section 5.3 presents a suitable usage control policy language for DataSov. For this, we extend the core information model of the widely used Open Digital Rights Language (ODRL) with support for external data sources and execution points. Our custom ODRL profile also allows the representation of provenance tracking information in usage control policies. We close this chapter in section 5.4 with a brief conclusion.

5.1 The DataSov Framework

In this section we present our proof of concept framework for trustworthy distributed usage control and provenance tracking called *DataSov*. The DataSov framework is built on our proposed usage control system architecture and the corresponding authentication and attestation concept as presented in chapter 3. We also utilize the technical protection mechanisms based on TPMs, Intel SGX, and ARM TrustZone developed in chapter 4.

5.1.1 System Architecture

In section 3.2 of this thesis, we described the necessary components for a trustworthy distributed usage control system, as well as their specific interactions. The DataSov framework is the implementation of this proposed concept. As illustrated in fig. 5.1, our framework consists of five main modules.

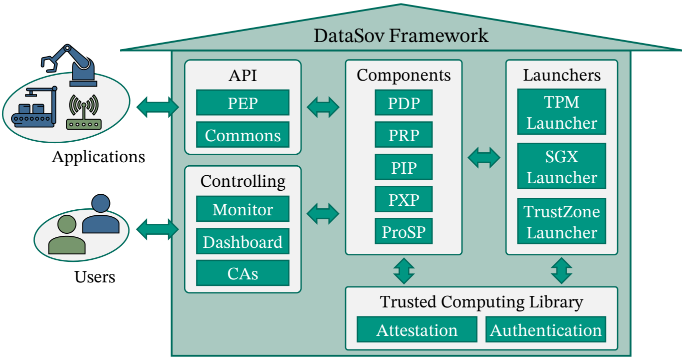


Figure 5.1: Overview of the DataSov framework architecture.

First, the *DataSov API* contains interfaces and classes that are necessary to integrate distributed usage control and provenance tracking into existing data processing applications. This mainly includes the DataSov Policy Enforcement Point (PEP), as well as common data types and structures that are used to interact with the framework. In addition, DataSov also provides several modules for the purpose of *controlling* the framework functionalities. This includes monitoring components that supervise the usage control enforcement and provenance tracking process, as well as a web-based dashboard that gives users feedback about the current system state and the collected provenance graphs. DataSov also includes default implementations for the Certification Authorities (CAs), which are necessary to securely provision new component instances and facilitate our proposed component authentication scheme.

At the heart of the DataSov framework lie the *core components*. These components implement the usage control and provenance tracking capabilities as described in section 3.2. This mainly includes our customized ODRL Policy Decision Point (PDP), as well as a Policy Retrieval Point (PRP) and a Provenance Storage Point (ProSP). We also include default implementations for Policy Information Points (PIPs) and Policy Execution Points (PXPs), which provide several basic functionalities. However, these components can also be extended with additional application-specific functions. Furthermore, the *launcher* module provides a suite of applications that facilitate the launch of DataSov components on different trusted computing platforms. As motivated in chapter 4, we support the execution of our usage control and provenance tracking components on TPM-protected systems, as well as inside Intel SGX enclaves and on ARM TrustZone devices. The DataSov launchers start the core components as individual server instances that are protected by the respective trusted computing technology. This design simplifies the deployment of the framework, because it decouples the functionality of the core components from their concrete execution context as separate applications on a trusted platform.

Finally, all DataSov components rely on a *trusted computing library* that implements most of the technical protection mechanisms developed in chapter 4. This mainly includes the heterogeneous remote attestation protocol and the component authentication scheme that is used to achieve secure communication channels between the distributed system components. In addition, this library also provides the core components with a persistence layer that includes a simple rollback protection mechanism. The idea behind this persistence layer is to offer DataSov components convenience methods for securely saving their individual states, while abstracting from the concrete and technology-dependent sealing functionalities used for the data encryption (cf. chapter 4). However, as of the time of this thesis, sealing is not yet fully implemented in the DataSov framework.

In the remainder of this section, we present a more detailed overview of the services that realize trustworthy usage control and provenance tracking in our proof of concept framework. We also give some insight into the design of the included protection mechanism against rollback attacks.

5.1.2 Components and Service Definitions

The design of the DataSov framework follows a distributed and service-oriented system architecture. All usage control and provenance tracking components are realized as independent services that are communicating with each other over the network. This allows us to dynamically deploy usage control components over multiple physical platforms and domains. To facilitate the network communication between distributed system components, the DataSov API defines service interfaces for all components, as well as common messages and data types that are exchanged by the services. Figure 5.2 gives an overview of the most important service interfaces and messages that are used in the DataSov framework. We discuss the concrete implementation of these services later in section 5.1.4.

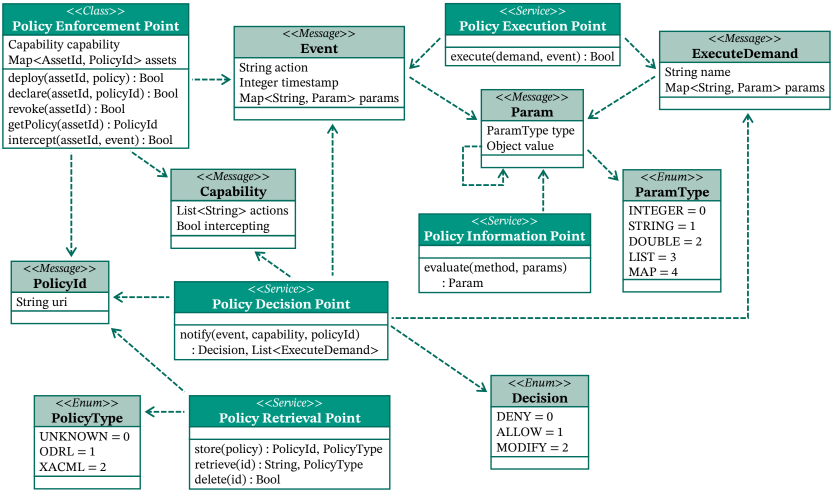


Figure 5.2: Overview of messages and service definitions concerning usage control in the DataSov framework.

The main entry point into the DataSov framework is the interface of the Policy Enforcement Point (PEP). The DataSov PEP is not a dedicated network service, but instead a small C++ code module that can be compiled into legacy data processing applications in order to enable support for usage control and provenance tracking. The PEP's main responsibility is to provide the legacy application with a simple interface for connecting to the distributed usage control system and requesting usage decisions. As such, it assembles notifications about occurring data usages and handles the subsequent communication with the policy retrieval and decision points. Before it can be used, each PEP needs to be set up with its *capability* definition. The capability definition includes a list of action names, for which this PEP can generate usage notifications. It furthermore includes the information if this PEP is capable of actively intercepting data usages, or if it can merely observe usages for provenance tracking. Afterwards, the PEP interface provides two methods that can be used to enable usage control enforcement and provenance tracking on the application's data. The `deploy` method attaches a new policy to a certain data asset that is processed in the application, in accordance with the previously motivated sticky policy concept. When this method is called, the PEP announces the new policy in the distributed usage control system and stores the association between the asset and the deployed policy in a local map. If a policy has already been deployed in the local usage control domain, the `declare` method can be used instead, which just creates the association between asset and the existing policy inside the PEP. Together, these two methods implement the concepts of cross-domain and domain-internal policy deployment, as introduced in section 3.2.3. Finally, the `intercept` method notifies the usage control system of an attempted data usage. The PEP then contacts the local decision point to request an evaluation of the relevant policy under the intercepted *event context.* The event context is automatically assembled from the name of the intercepted action, together with a set of parameters that further describes the data usage. Once the decision point has reached its verdict regarding the attempted data usage, the PEP module enforces the decision by either allowing, denying, or modifying the intercepted event accordingly.

Unlike enforcement points, the Policy Decision Point (PDP) is a dedicated service that provides its interface over the network. DataSov PDPs implement the `notify` method, which is called by enforcement points in order to evaluate a policy under a certain event context. This method also accepts a PEP capability definition, which is then used to determine remedies that must be executed if the decision cannot be enforced at the enforcement point (cf. section 5.3.1). Ultimately, the PDP returns a *decision* message that either allows or denies the data usage, which the PEP subsequently enforces. It is also possible to request the modification of certain event parameters before the event can be allowed. In that case a list of *execute demands* is returned alongside the decision. Execute demands are always identified by a name and may also contain additional parameters. The enforcement point must then successfully exercise all received execute demands before the data usage may be allowed. Later, in section 5.3.3, we show how PEP execute demands can be expressed as usage control obligations in ODRL policies. However, the DataSov framework is designed to be generally agnostic regarding the used policy languages. As a result, we can operate multiple decision points supporting different policy languages in our usage control domain. The advantage of policy agnosticism is that it allows us to individually express different types of usage rules for different applications and use cases. We achieve this by adhering to the sticky policy concept (i.e., one policy per asset), as well as by using a system design that keeps the defined data types and messages independent from a particular policy language. Furthermore, we use retrieval points that are capable of automatically determining the *policy types* of the stored usage rules. This allows the enforcement points to choose a correct decision point instance that is capable of evaluating the policy for the current asset. Currently, the DataSov framework includes only one type of decision point, which implements our custom ODRL policy language profile (see section 5.3). However, by re-implementing the PDP service interface, DataSov can be easily extended with support for additional policy engines.

Finally, the rules evaluated at the decision point may also contain references to Policy Information Points (PIPs) and Policy Execution Points (PXPs). In DataSov, information points are realized as services that provide the `evaluate` function. This function accepts a method identifier and a list of parameters as input values. Both of these properties are taken from the evaluated policy by the decision point. The PIP then determines the value of the requested attribute and returns it to the decision point as a parameter message. Similarly, DataSov execution points offer the `execute` function, which takes an execute demand together with the current event context as input parameters. The

PXP returns true if the requested demand has been executed successfully. If a demand could not be executed successfully, the decision point will treat the corresponding usage control obligation as not fulfilled and hence deny the usage request. The DataSov framework includes default implementations for both information and execution points, which support all methods that are defined as part of our custom ODRL policy profile (see section 5.3). However, since the DataSov framework is meant to be adapted for a wide variety of applications, users can easily add support for additional methods by extending the respective component service interfaces.

5.1.3 Provenance Tracking and Dashboard

In addition to distributed usage control, the DataSov framework also provides support for provenance tracking on the protected data assets. As motivated in section 3.2.6, our system design does not require a dedicated data flow model as basis for the provenance tracking, unlike the previous proposal by Bier [Bie21]. Instead, we implement the collection of provenance data directly via usage control obligations. The advantage of this approach is that it allows data owners to explicitly specify the desired provenance tracking semantics alongside the data asset's usage rules, instead of having to implicitly integrate them into the data processing applications. Furthermore, being able to reference provenance information in policies facilitates the expression of usage restrictions that are based on a data asset's processing history. Nevertheless, we still leverage the concept of Provenance Storage Points (ProSPs) to retain the provenance information that has been collected in a particular usage control domain.

Figure 5.3 shows the messages and service interfaces that are used to implement provenance tracking in DataSov. Each DataSov ProSP offers two methods that can be used to access the provenance. The `store` operation takes a provenance information object and merges it into the current provenance state. The `retrieve` operation returns the requested parts of the current provenance state. These methods are called during policy evaluation by execution points and information points, respectively. In section 5.3.4 we show how this form of provenance tracking can be controlled using obligations and constraints in ODRL policies. The stored provenance information itself is structured according to the W3C PROV data model [Bel13]. We identify each entity, activity, and agent with a unique URI. Furthermore, each provenance instance and relation can be associated with a class type and a list of generic properties. This enables policy issuers to build provenance graphs using any custom classes or relation types, instead of just the base types defined in the PROV data model (see section 2.2.1).

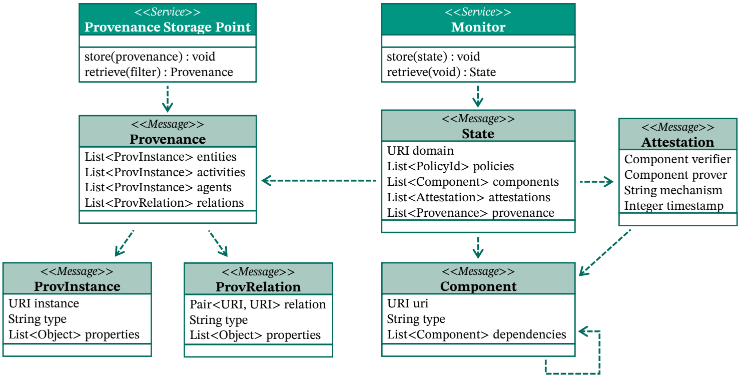


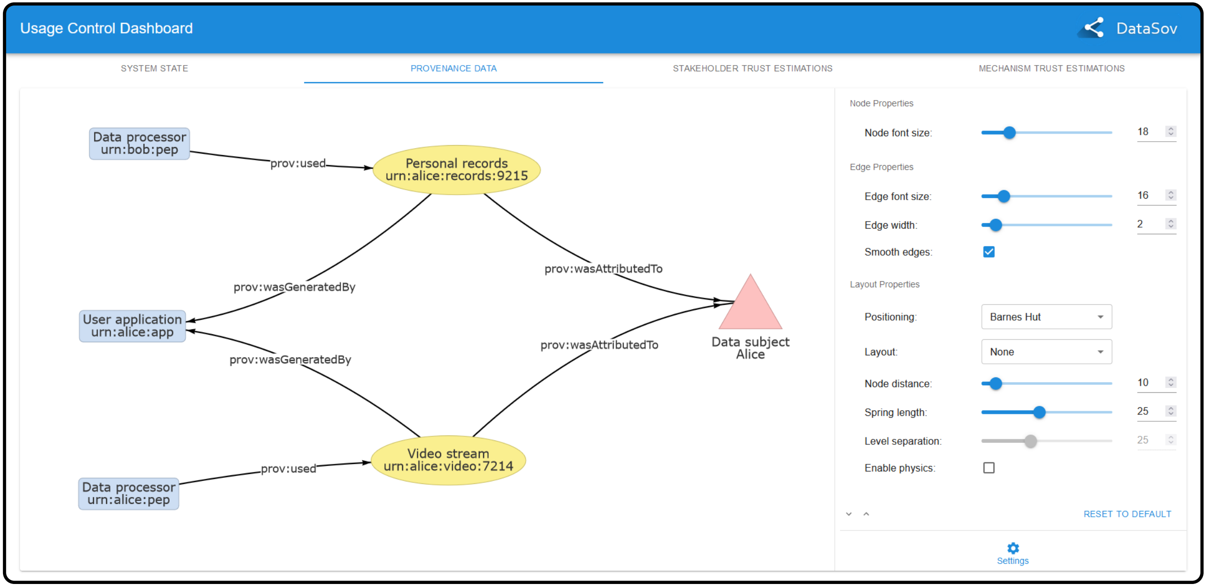
Figure 5.3: Overview of the messages and service definitions concerning provenance tracking in the DataSov framework.

After collecting and storing information about the usage history of monitored data assets, a provenance tracking system should also allow its users to query and analyze the captured information. In the DataSov framework, we implement this provenance dissemination step with additional *monitor* components and a dedicated *dashboard*. As fig. 5.3 shows, the DataSov monitors are services tasked with aggregating and disseminating information about the current state of the usage control system. Each monitor is associated with a single usage control domain and collects several pieces of information from the DataSov components running in that domain. The collected information include provenance data from the local ProSP, additional runtime information about the usage control components, the conducted remote attestations, and the deployed policies. In aggregated form, this information represents the current *state* of a particular usage control domain. While monitor components are implemented as network services, the DataSov dashboard is instead realized as a standalone web application. Dashboards are responsible for providing the DataSov users with feedback about the current state of the usage control and provenance tracking system. As such, they connect to one or more monitors and periodically retrieve the state information from multiple usage control domains that are of interest. The received state information is then integrated and visualized using a graphical user interface. Among other things, the dashboard provides information about the currently active usage control components, the deployed policies, and the provenance graphs of shared data assets. Figure 5.4 shows a screenshot of the DataSov dashboard displaying a small exemplary provenance graph. The visualization of the provenance graphs largely follows the illustrations defined in the PROV data model [Bel13]. Entities are represented as ellipses, activities as boxes, and agents as triangles. Relations are shown as labeled edges between the instances. With this depiction of the collected provenance graphs, the DataSov framework allows data providers to retrospectively review the usage and processing history of their shared data assets.

5.1.4 Implementation and Configuration

As a standalone web application, we realized the DataSov dashboard in JavaScript using Node.js together with the ReactJS ecosystem. In contrast, all of our distributed usage control and provenance tracking components are implemented in C++ as gRPC[1] services. gRPC is a cross-platform Remote Procedure Call (RPC) library developed by Google, which allows to define and connect services over the network. This library follows a client/server architecture and includes a compiler that generates code stubs for the custom gRPC message and service definitions. To transparently include heterogeneous attestation and authentication between DataSov components, we extended the implementation of the underlying communication channels

[1] https://grpc.io/ *(accessed on 12/08/2023).*



Usage Control Dashboard
DataSov
SYSTEM STATE
PROVENANCE DATA
STAKEHOLDER TRUST ESTIMATIONS
MECHANISM TRUST ESTIMATIONS
Data processor urn:bob:pep
prov:used
Personal records urn:alice:records:9215
prov:wasGeneratedBy
prov:wasAttributedTo
User application urn:alice:app
prov:wasGeneratedBy
prov:wasAttributedTo
Data subject Alice
Data processor urn:alice:pep
prov:used
Video stream urn:alice:video:7214
RESET TO DEFAULT
Settings

Node Properties

Node font size: 18

Edge Properties

Edge font size: 16

Edge width: 2

Smooth edges: ☑

Layout Properties

Positioning: Barnes Hut

Layout: None

Node distance: 10

Spring length: 25

Level separation: 25

Enable physics: □

Figure 5.4: Screenshot of the DataSov provenance dashboard.

provided by gRPC. Details about the realization of our remote attestation and authentication concept in DataSov are given later in section 5.2. We utilize the Bazel[1] build tool to compile our DataSov components for the different supported trusted computing platforms. Bazel by design always performs full dependency builds, which simplifies the deployment of the DataSov components, since the resulting binaries do not depend on any system libraries. We build the TPM-protected DataSov components as native C++ applications using the TSS.MSR[2] software stack by Microsoft to connect to the platform TPMs. Furthermore, our components can be built as SGX enclaves using the build chain provided in the Asylo framework. The resulting SGX enclave images are then loaded, configured, and executed by a dedicated launcher application. Crucially though, the component server is always located inside the SGX enclave, so as to ensure that the communication endpoints are terminated inside the enclave. Finally, we leverage the ARM toolchain provided in the OP-TEE environment to cross-compile DataSov for ARM TrustZone devices as well. More concretely, we provide build configurations for both the Raspberry Pi 3 and the STM32MP157C-DK2[3] TrustZone development board. For convenience purposes, our build process generates ready-to-use device images that contain the compiled binaries of all DataSov components. The code of the DataSov framework including all necessary build files is available online.[4]

The DataSov framework allows to dynamically deploy usage control and provenance tracking components on multiple devices, as required for the given application scenario. The only step necessary to set up a new component is to enroll a component certificate for the authentication scheme (see section 3.2.8). For this purpose, each DataSov component offers a *provisioning mode* that acquires such a certificate from the local domain CA. Unlike with most existing distributed usage control frameworks, no central registration, management, or lookup services are required. In terms of configuration,

[1] https://bazel.build/ *(accessed on 01/23/2024).*

[2] https://github.com/microsoft/TSS.MSR *(accessed on 01/23/2024).*

[3] https://www.st.com/resource/en/data_brief/stm32mp157c-dk2.pdf *(accessed on 12/21/2023).*

[4] https://gitlab.cc-asp.fraunhofer.de/datasov *(accessed on 12/08/2023).*

every component must be set up with their unique component URI, as well as the trusted root certificate for the component authentication scheme. Some components also require to specify the URIs of dependent services that should be used during operation. For example, PEPs are configured with a list of decision point URIs that determine which component should be contacted for policy evaluation, depending on the encountered policy language. Similarly, PDPs are configured with the URI of the local retrieval point. However, the URIs used to contact PIPs, PXPs, and ProSPs are never fixed in the component configuration. Instead, the decision point takes these URIs directly from the evaluated policy (see section 5.3). Finally, each component configuration is measured as part of the TCB to prevent undetected tampering by malicious system operators.

5.1.5 Integrated Rollback Protection

Authenticating distributed usage control components and verifying the integrity of their code bases with remote attestations is a central security mechanism of our trustworthy usage control framework. However, protecting the integrity of component *states* is equally important for a reliable usage control enforcement process (cf. section 3.4). This is because stateful components, such as PIPs and ProSPs, provide information that is relevant for the evaluation of usage control policies, and hence can influence usage control decisions. Take for example a PIP storing an access counter, which gets updated during the life cycle of a usage-controlled data asset. If a malicious platform owner could reset the PIP back to an earlier state with a lower counter value, active usage restrictions on that data asset may be violated as a result. To prevent such types of attacks, we need to integrate a rollback protection (RBP) mechanism for component states into the DataSov framework.

As a first step towards an effective rollback protection, we design each DataSov component to hold all information relevant for the usage control enforcement inside a single dedicated *state object*. This includes data such as the deployed usage control policies (PEPs and PRPs), information used during the policy evaluation (PIPs), collected provenance information (ProSPs), but also the component private keys that have been generated during the provisioning process (see section 3.2.8). In addition, each state object is associated with a unique *version number*, which is automatically incremented at every state update. We define these state objects as nested Protobuf[1] messages, which allows us to easily serialize component states into binary blobs. Now the question remains of how to protect the integrity of these serialized state objects in the DataSov framework, such that malicious state rollbacks are reliably detected and hence cannot negatively influence the usage control enforcement process. There are principally two ways of implementing such a mechanism. First, as discussed in chapter 4, we can rely on the underlying trusted computing technologies to provide hardware mechanisms that allow us to detect rollbacks of usage control state objects when they are loaded. The most promising hardware feature for this are monotonic counters. By updating a monotonic counter each time the component state changes, and then including it in the persisted state object (e.g., as the unique version number), the integrity of the state object becomes locally verifiable. To detect rollback attacks, we simply have to check if the version number of a loaded state object still matches the monotonic counter. While this is a relatively simple and commonly used solution for rollback detection, it also comes with some major drawbacks. Most importantly, monotonic counters are usable on TPM-protected systems (cf. `TPM2_NV_Increment`) and some TrustZone devices (using an RPMB memory module), but are generally not available on SGX platforms [Mat17]. In addition, there are known issues with both performance [Mat17], as well as increased wear of non-volatile memory [Seg16, pp. 168–169] when using monotonic counters for rollback protection. Finally, relying on specific trusted computing hardware for rollback protection also complicates the integration of new trusted computing technologies into our framework in the future. Because of these issues, we choose to implement the rollback protection mechanism in DataSov in a manner that is independent of the underlying trusted computing technologies.

The alternative way of realizing rollback protection is by means of a centrally trusted service. We implement this approach by introducing the *state store* as

[1] https://protobuf.dev/ *(accessed on 12/08/2023).*

a new component in the DataSov framework. The state store is responsible for keeping track of individual component states and ensures their integrity throughout the operation of the usage control system. During the provisioning phase, all stateful DataSov components register their individual component URI at the state store. The state store authenticates the components and their respective URIs via their individual component certificates (see section 3.2.8). During the subsequent component operation, for every update of the persisted state, the components calculate cryptographic hash digests of both the current and the new state, and send these digests to the state store. The state store then validates that the reported state digest still matches the deposited digest for this component, before updating it to the new value. If the validation fails, the component is notified over the encrypted channel and consequently terminates itself, which in turn invalidates any usage rules that rely on the manipulated component. Since any external modification of the locally persisted state will invalidate the digest deposited at the state store, and only the legitimate component is able to authenticate itself at the state store, this mechanism effectively prevents rollback attacks. Recovering from a detected rollback attack is possible either by restoring the original component state, or by enrolling a new component identity and updating the affected policies. Note that allowing recovery via a simple component reset is infeasible for our purposes, since reverting to an empty state could also lead to undesired side-effects during policy evaluation. For instance, in the previously mentioned example the access counter at the PIP would be reset to zero, which again could lead to policy violations.

The major advantage of using a central state store for rollback protection is that it works independently of any trusted computing technology, and hence offers a common mechanism that is accessible to all stateful DataSov components, regardless of their deployment. However, even though the state store does not save any secrets, the integrity of the deposited digests must still be ensured for the rollback protection to be reliable. Because of this, we assume the state store to be a trusted and globally available party in our proof of concept. Alternatively, the state store could also be realized as a distributed ledger over all participants of the usage control system. While this avoids the need for an additional trust anchor in the system, it also adds another layer of complexity. We leave the further exploration of this possibility for future work.

5.2 Remote Attestation in DataSov

In this section we discuss the inner workings of the remote attestation and component authentication methods as they are included in DataSov.

5.2.1 Implementing Heterogeneous Attestation

As presented in the previous section, we implement the core components of our DataSov framework in the form of gRPC services. Besides offering Remote Procedure Call (RPC) functionalities, the gRPC library also handles the serialization and transmission of corresponding service requests and responses over the network. Hence, to achieve a trustworthy distributed usage control and provenance tracking framework, we need to integrate our findings and developments regarding remote attestation and secure communication channels (see chapter 4) into the gRPC stack. Fortunately, the gRPC library is designed with the concept of independent *channel providers*, which decouple the implemented services from the underlying network transport layer. This allows us to implement a customized gRPC channel provider for DataSov, which establishes encrypted communication between the framework components based on heterogeneous remote attestation, as well as our component authentication scheme. More concretely, we build the secure gRPC channel provider for the DataSov framework on the Enclave Key Exchange Protocol (EKEP). EKEP is used in the Asylo trusted computing framework for SGX-based local and remote attestation [Asy21a]. Since Asylo is also a gRPC-based framework, the existing EKEP implementation already provides a protocol handshaker, as well as a corresponding encryption layer for gRPC communication channels. As presented in section 4.5, we achieve support for heterogeneous remote attestations in DataSov by extending the EKEP protocol implementation with assertion generators and verifiers for TPM-protected platforms and fTPM-based ARM TrustZone devices.

One technical issue that occurred during our implementation was that the SGX-based assertion generators included in EKEP cannot be compiled for non-SGX platforms. We resolved this by creating a fork of the underlying EKEP implementation and selectively patching out SGX-only libraries when compiling the gRPC communication stack for TPM-protected endpoints or TrustZone devices. The resulting heterogeneous remote attestation library for gRPC communication channels is implemented in C++. Additionally, the DataSov framework also contains Java wrappers for the default C++ library using the Java Native Interface (JNI). This makes our attestation and authorization framework usable in Java-based environments as well, which simplifies the integration of legacy applications into the distributed usage control framework. The complete code of our gRPC attestation library is available online as part of DataSov.[1]

Besides implementing a gRPC channel provider that supports heterogeneous remote attestations, we also need to consider how attestations can be properly configured in the DataSov framework. Most importantly, users have to decide *how often* attestations should be conducted between components. Since the gRPC channel concept decouples the establishment of network connections from the actual RPC calls, an already attested communication channel can be reused for an arbitrary number of service requests. However, indefinitely reusing a once attested channel increases the risk of encountering Time-of-Check Time-of-Use (TOCTOU) issues by missing updates of the remote component's active code base. This is an issue especially with trusted platforms that allow dynamic runtime measurements, e.g., when using TPMs together with the Linux kernel IMA. To prevent this, component re-attestations could be enforced proactively by establishing new gRPC channels for every single service request. However, this would result in an unreasonably high connection overhead and is usually not necessary, especially in scenarios with a large number of requests. As middle ground we implement a *channel manager* in DataSov, which allows users to flexibly configure the desired frequency of component re-attestations depending on the application scenario. Our channel manager caches successfully attested gRPC channels that have

[1] https://gitlab.cc-asp.fraunhofer.de/datasov/lib/-/tree/master/smarttc *(accessed on 12/08/2023).*

been established with remote DataSov components, and then automatically reuses them for subsequent requests. The channel manager can be configured to keep attested channels open either indefinitely, or for a certain period of time. By default, the DataSov components use a cache timeout of 60 seconds. After a cache timeout, the channel is closed and an automatic re-attestation takes place for the next connection. We examine the resulting communication round-trip times with a distributed usage control example both using cached and non-cached channels later in chapter 7.

5.2.2 Integrating Component Authentication

In addition to the heterogeneous attestation, we also need to automatically *authenticate* the DataSov components over the established gRPC channels, in order to prevent component impersonation attacks. As described in section 3.2.8, our proposed authentication scheme provisions component certificates via a hierarchical PKI. These certificates can then be used to unambiguously identify and authenticate each component via a unique URI. We integrate this certificate-based component authentication scheme into the described gRPC channel implementation for DataSov. To achieve this integration, we extend the EKEP protocol definition with the concept of DataSov *component identities*, in addition to the previously described *code identities* that facilitate the heterogeneous attestations. Component identities are represented by the unique component URI together with the signed certificate that has been provisioned for this URI. Furthermore, we add a suitable assertion generator and corresponding verifier to the customized EKEP protocol handshake, which use a simple challenge-response protocol to perform the cryptographic authentication of claimed component identities (cf. section 3.2.8).

Listing 5.1 shows the Protobuf messages used to exchange component identities and assertions as part of the gRPC channel establishment in the DataSov framework. Lines 4-9 contain the data structure representing the identity of a DataSov component, which consists of its URI and the corresponding component certificate. Lines 12-21 then define the assertion that is exchanged during the gRPC channel establishment to authenticate a particular component identity. For this, the asserted component uses its private key to sign the challenge transmitted by the verifier, which includes the nonce for freshness and a digest of the verifier's ephemeral Diffie-Hellman public key. Finally, the assertion also includes the complete certificate chain to the root CA.

Listing 5.1: DataSov component assertion and identity description for EKEP.

```
syntax = "proto2";

// A high-level representation of the identity of a DataSov component.
message ComponentIdentity {
  // The component URI. Required.
  optional string uri = 1;
  // The component certificate. Required.
  optional Certificate cert = 2;
}

// A cryptographically verifiable component assertion.
message ComponentAssertion {
  // The asserted DataSov component identity. Required.
  optional ComponentIdentity identity = 1;
  // The challenge requested by the verifier. Required.
  optional bytes challenge = 2;
  // The signature of the challenge under the component key. Required.
  optional bytes signature = 3;
  // The certificate chain for the component. Required.
  optional CertificateChain cert_chain = 4;
}
```

Conducting assertions of component identities as shown in listing 5.1 unambiguously authenticates a network peer as a specific DataSov component. Usage control components can then validate that they are communicating with the correct peers by checking the authenticated URI. Since EKEP supports the exchange of multiple assertions in a single protocol handshake, we can configure the channel establishment in DataSov to always use the assertion providers for *both* heterogeneous attestation and component authentication. This allows us to simultaneously verify the code integrity as well as the usage control identity of remote DataSov components during the establishment of a single gRPC communication channel.

5.2.3 Authorizing Asserted Component Identities

Finally, our custom gRPC channel implementation for DataSov must also integrate suitable methods for *authorizing* the asserted code bases and component identities. This means that DataSov components must be able to specify which peer identities are required and acceptable for each established gRPC channel. If the remote target does not provide proper assertions for the expected identities, the connection is unauthorized and must be closed. Typically, the authorization of *code identities* is done by comparing the attested TCB measurements with a set of expected (golden) values that the remote peer is supposed to have. Similarly, for the authorization of *component identities* we compare the asserted URI with the expected identifier of the target component.

Since EKEP is an SGX-based remote attestation protocol, it already offers features to authorize the code identities of remote SGX enclaves over gRPC channels. For this, the protocol uses Access Control Lists (ACLs) that can be attached to a gRPC channel context [Asy21b]. ACLs allow users to define a list of acceptable enclave identities that the network peer on this particular channel must successfully assert, in order to be considered authorized for communication. These ACLs are defined as Protobuf messages, which can be serialized into a human-readable text format for editing. Furthermore, individual ACLs can also be combined using boolean predicates to define complex authorization policies for a gRPC channel [Asy21b]. ACLs mainly consist of the expected peer identity together with a set of matching rules that determine which parts of the asserted enclave identity must match the expected identity description. This *identity matching* approach allows users to define authorization policies that only partially compare the properties of the asserted identities. For example, SGX-based enclave identities can be authorized by comparing the values of either the `MRENCLAVE` or the `MRSIGNER` field (or both). On the technical side, this concept is implemented using various *identity expectation matchers*, which evaluate the asserted identities against the active ACLs. The identity matchers are automatically called by the gRPC channel implementation as soon as a remote assertion is received and validated. To support channel authorization based on TPM code identities and DataSov component identities, we implement suitable ACL definitions and corresponding identity expectation matchers. Listing 5.2 shows the Protobuf messages defining the ACL format for TPM-based code identities.

Listing 5.2: ACL data structures for TPM-based identity expectations and matching.

```
1   syntax = "proto2";
2   message TpmPcrMatchSpec {
3     // The list of PCR registers to match.
4     repeated int32 system_pcrs = 1;
5   }
6   message TpmSmlMatchSpec {
7     // Treat expected measurement log as a blacklist instead of whitelist.
8     optional bool blacklist = 1;
9   }
10
11  // Specification of which fields from `TpmIdentity` to match.
12  message TpmIdentityMatchSpec {
13    // Selection between PCR matching and measurement log matching. Required.
14    oneof TpmIdentityMatchMode {
15      TpmPcrMatchSpec pcr_match_spec = 1;
16      TpmSmlMatchSpec sml_match_spec = 2;
17    }
18  }
19
20  // A verifier's expectation of a `TpmIdentity`.
21  message TpmIdentityExpectation {
22    // Reference identity matched against the target identity per `match_spec`. Required.
23    optional TpmIdentity reference_identity = 1;
24    // Specification of which properties from the target identity to match. Required.
25    optional TpmIdentityMatchSpec match_spec = 2;
26  }
```

Lines 21-26 define the main ACL data structure consisting of the TPM-based reference identity (cf. listing 4.1), as well as the identity matching specifier. We support two types of TPM identity matching. If the identity matching is conducted directly on the PCR values, then the ACL allows to filter the individual registers and banks to be compared (lines 2-5). However, this type of matching is suitable only for (relatively) static code bases, to avoid an excessively large list of expected PCR values. The other implemented identity matching type uses the IMA measurement log that is optionally included in the TPM-based code identities. This matching is performed by comparing all entries in the measurement log of the peer's asserted identity with the entries specified in the ACL's reference measurement log. For this comparison, the reference measurement log can be used either as whitelist or as blacklist (lines 6-9) and may also contain regular expressions.

Similarly, we also implement the ACL definition and identity matcher for the DataSov component identities. These ACLs allow to authorize remote components via their unique URI. Listing 5.3 shows the Protobuf messages defining this ACL format. We support URI matching both as exact string as well as using a regular expression (lines 3-10), allowing for the authorization of a set of acceptable components via a single ACL.

Listing 5.3: ACL data structures for DataSov component identity expectations and matching.

```
1   syntax = "proto2";
2   // Specification of which fields from `ComponentIdentity` to match.
3   message ComponentIdentityMatchSpec {
4     enum UriMatchType {
5       EXACT_URI = 1;
6       REGEX_URI = 2;
7     }
8     // Type of the URI matching. Required.
9     optional UriMatchType match_uri = 1;
10  }
11
12  // A verifier's expectation of a `ComponentIdentity`.
13  message ComponentIdentityExpectation {
14    // Reference identity matched against the target identity per `match_spec`. Required.
15    optional ComponentIdentity reference_identity = 1;
16    // Specification of which properties from the target identity to match. Required.
17    optional ComponentIdentityMatchSpec match_spec = 2;
18  }
```

Finally, we can use the boolean ACL predicates to combine our authorization rules regarding the expected code bases and component identities into a single channel policy. This avoids the need to “manually” check the asserted component URI at every location in the code where remote components are called. As described in section 3.2.7, we then distribute the resulting ACLs specifying the expected code and component identities via our Measurement Store (M-Store) concept. Hence, DataSov components do not have to be deployed with a set of fixed ACLs for one application, but can instead pull them dynamically from the local M-Store. A reference implementation for the M-Store component is also provided in the DataSov framework.

5.3 A Policy Language for DataSov

In this section we present a policy language that is suitable for our trustworthy usage control and provenance tracking framework. As mentioned earlier, the DataSov framework is generally policy-agnostic and can be used with any type of decision point. However, there are some requirements that a usage control policy language should fulfill in order to be useful in DataSov.

- **(P1) Distributed usage control:** The policy language should allow the definition of rich and expressive usage rules on digital assets that are distributed across multiple stakeholders. This necessitates the expression of issuer and subject of usage rules, and may even include mechanisms for policy negotiation and agreement.
- **(P2) General-purpose language:** The policy language should be applicable to generic use cases instead of being domain specific. This is usually achieved by defining an extensible information model that describes the semantics of expressed usage rules.
- **(P3) External information sources:** Since DataSov implements a usage control system that is both *distributed* and *decentralized* (see section 2.1.2), we require policy support for external information sources (i.e., PIPs). This means that the policy language must allow the definition of parameter lookups at remote repositories. Furthermore, to prevent malicious interference in the policy evaluation process, the language must also allow issuers to define the concrete component identities of information sources in the policy.
- **(P4) PXP obligations:** Besides supporting external information sources, the policy language must also allow the specification of obligations that should be executed on remote PXPs. Once again, policy issuers must be able to specify the concrete component identities of the execution points that should be used, so as to avoid component impersonation during the usage control enforcement process.
- **(P5) Representation of provenance information:** Finally, a policy language suitable for DataSov must be able to represent provenance

information. This allows policy issuers to both manage the desired provenance tracking as part of the policy enforcement, as well as referencing the usage history of data as constraints in usage rules.

In section 2.1.3 we presented a comparison of the most important usage control languages at the time of this thesis. Of those options, we rely on the Open Digital Rights Language (ODRL) as default policy scheme in the DataSov framework, mainly due to its widespread use and recommendation by the W3C. However, the core ODRL vocabulary does not allow to specify external information sources, PXP obligations, or provenance tracking information, and hence does not yet fulfill all of our requirements regarding a suitable policy language. Hence, in the remainder of this section, we first introduce ODRL in greater detail and then extend its core information model with a specialized ODRL profile, which allows policy issuers to reference external PIPs and PXPs, as well as represent provenance information in usage control policies. Finally, we also describe our implementation of a Policy Decision Point that can evaluate the newly specified ODRL profile for the DataSov system.

5.3.1 The Open Digital Rights Language

ODRL is a rights expression language that has been standardized as a W3C recommendation in 2018. The ODRL standard consists of a semantic information model [Ian18b], a corresponding vocabulary [Ian18a], and several options for serialization, most notably the JSON-LD format. While originally developed in the context of digital rights management, ODRL has recently received attention as a policy language for distributed usage control as well [Keb18]. In general, ODRL focuses on expressing usage semantics for digital assets in a technology-independent and normative fashion.

Figure 5.5 shows the most important classes and relationships of the ODRL information model [Ian18b]. At the core of the ODRL standard lies the concept of *policies*, which specify generic usage statements concerning one or more *assets*. Each ODRL policy consists of one or more *rules*, which are always associated with an *action* that may be executed on the protected assets. Rules can also contain several *constraints*, which further refine the rule using comparison as well as logical operators on pre-defined *operands*. Each rule can express either a *permission*, a *prohibition*, or a *duty* on the associated assets with regard to the defined action. A permission rule allows the action on the asset if all constraints are satisfied, while the prohibition rule disallows it. In contrast, a duty rule represents an obligation to exercise an action. Both permissions and prohibitions can also be associated with additional duty rules, which must then be fulfilled before the permission can be granted (duty), or in case an action infringes on a prohibition (remedy). Finally, each rule can be associated with one or two *parties*, which represent the issuer of the rule (*assigner*) and the intended recipient of the rule (*assignee*). Usually, policies express the currently applicable usage rules for certain digital assets. However, ODRL also allows to specify non-binding *offers* and *agreements* of usage rules between assigner and assignee. This is especially useful when implementing automated contract negotiations in preparation of data transmissions. Nevertheless, these features are out of scope for the DataSov framework and we do not consider ODRL offers and agreements in the remainder of this thesis.

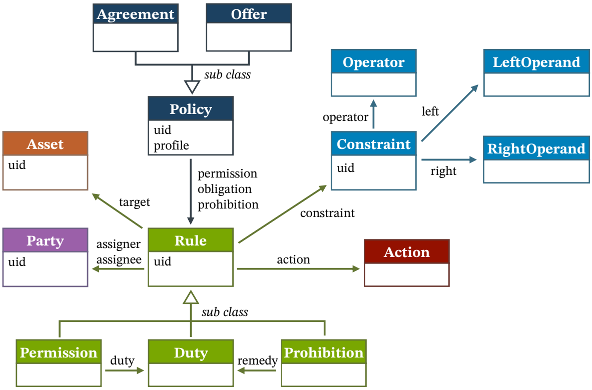


Figure 5.5: Simplified version of the ODRL information model. Own illustration after [Ian18b].

Since the ODRL information model is independent of concrete application domains and by design also considers asset distribution, it is a suitable foundational technology for expressing usage control policies in our framework (cf. requirements P1 and P2). However, due to its normative approach, the standard ODRL vocabulary does not allow to uniquely reference individual components such as information points and execution points. It also does not consider the representation of provenance data. As motivated earlier, these are necessary prerequisites for properly enforcing usage control and provenance tracking in the DataSov framework (requirements P3 to P5). Hence, in the remainder of this section, we extend the ODRL information model and vocabulary by specifying a new *ODRL profile* for DataSov that adds these missing language features.

5.3.2 Defining Information Sources

As shown in fig. 5.5, the core ODRL information model offers two *operand* classes to define constraints in ODRL rules. A *left* operand represents the property that the respective constraint is concerned with, such as the time of day or a specific attribute of the current asset. The *right* operands then define the concrete values to which the left operand should be compared, for example a specific number or string. Finally, a set of *operators* define the logical comparisons that should be conducted between both operands to evaluate the constraint. Listing 5.4 gives an example of an ODRL constraint representing a time restriction using the left and right operand concept.

Listing 5.4: Example of a core ODRL constraint with left and right operands [Ian18b].

```
"constraint": [{
    "leftOperand": "dateTime",
    "operator": "lteq",
    "rightOperand": { "@value": "2017-12-31", "@type": "xsd:date" }
}]
```

While the core ODRL information model defines the syntax and semantics of these operand types, unfortunately it does not allow to specify the expected sources of the referenced information, or how an ODRL decision point is supposed to actually retrieve them. However, this is an important requirement for trustworthy distributed usage control enforcement. Policy issuers must be able to not only define *what* information should be evaluated as part of a rule constraint, but also *from where* a trustworthy decision point may retrieve this information (cf. requirement P3). Otherwise, a malicious data receiver could interfere with the trustworthy policy evaluation process by manipulating the sources of referenced information, instead of attacking the decision point implementation itself. To achieve a policy language that can accommodate these requirements, we extend the core ODRL information model with a new profile specific to our DataSov framework. More concretely, we specify two new operand types in our custom ODRL profile: *context* operands and *information* operands. A *context* operand is used to reference information about the currently active policy evaluation context at the decision point. As described earlier, the DataSov PDP services are notified by the enforcement points about all data usage events that have been intercepted. These intercepted events also include a set of parameters that are describing the context of the requested data usage. We introduce the concept of context operands, in order to support directly referencing these event parameters in ODRL policies. Listing 5.5 shows the usage of a DataSov context operand in an ODRL constraint concerning the ID of an accessed asset.

Listing 5.5: Example of a DataSov ODRL constraint using a left context operand.

```
"constraint": [{
    "leftOperand": {
        "@type": "ods:ctxOperand",
        "key": "assetId"
    },
    "operator": "eq",
    "rightOperand": "uri:urn:example:asset:891"
}]
```

A context operand is defined as a JSON-LD object of type `ods:ctxOperand` with the `key` property as the only member. This property specifies the lookup key of the event parameter that will be evaluated on the current policy context by the ODRL decision point. DataSov context operands can be used both as a left-hand and a right-hand operand of an ODRL constraint. This allows policy authors to both define restrictions directly for event parameters (like in listing 5.5), as well as use them as target values for dynamic comparisons. Context operands are always evaluated purely locally at the decision point. The exact definition and semantics of the context operand, as well as the entire `ods:` namespace of the DataSov ODRL profile, is given in appendix C.

Furthermore, we also define the concept of *information* operands as part of our custom ODRL profile. In contrast to the locally evaluated context operands, information operands allow us to specify *external* information sources, such as Policy Information Points (PIPs), directly in our usage control policies. While the core ODRL information model and vocabulary mainly focus on specifying the *semantics* of certain operand identifiers, for our proposal of trustworthy usage control enforcement we also need to unambiguously determine the *source* of information that is referenced in policies. Otherwise, our policy evaluation process as presented in chapter 3 may be susceptible to man-in-the-middle attacks. In order to achieve sufficiently detailed policies, we need to add support for two new elements to our policy language.

(i) Allow policy authors to specify from where external information should be retrieved during the policy evaluation.

(ii) Allow policy authors to specify the concrete identity of the component that is expected to deliver the requested information, as it is being authenticated by the DataSov framework (cf. section 5.2.2).

To fulfill both of these requirements, we introduce the concept of information operands into the ODRL information model. Listing 5.6 shows an example of such an information operand at the right-hand side of an ODRL constraint.

Listing 5.6: Example of a DataSov ODRL constraint using a right information operand.

```
"constraint": [{
    "leftOperand": "recipient",
    "operator": "isPartOf",
    "rightOperand": {
        "@type": "ods:pipOperand",
        "uri": "uri:urn:example:pip",
        "method": "permittedRecipients",
        "params": {
            "assetId": {"@type": "ods:ctxOperand", "key": "assetId"}
        }
    }
}]
```

The DataSov information operand is an object of type `ods:pipOperand`, which contains three additional properties. The `uri` property defines the operand's target component that the decision point should contact during the policy evaluation. More concretely, it contains the expected component identity of the information point delivering the requested information. During policy evaluation, the decision point verifies that the contacted information point is indeed authenticated under the specified component URI (see section 3.2.8). This allows policy issuers to designate *specific* usage control components as trustworthy information sources, and thus prevents any evaluation ambiguity or man-in-the-middle attacks on the used information points during policy enforcement. Note that for convenience purposes, this property may also directly hold a network URL under which the information point is available, if the usage control system uses URLs as component identities. Otherwise, an additional mapping is necessary to look up the proper network target for the given URI. However, this mapping can be untrusted since the target's component identity, as specified in the policy, is verified on the established channel using the URI. Complementing the target URI, the `method` property identifies the lookup method that should be used at the contacted information point to retrieve the requested information. Which methods are available depends on the concrete information point. However, there are a number of core methods that all DataSov information points must support. The details concerning the supported PIP methods are specified in the DataSov ODRL profile given in appendix C. Finally, the `params` property is a key-value map of parameters that should be transmitted to the information point during the operand evaluation. Since we model these parameters to be ODRL operands themselves, they can either be constant values or consist of nested context and information operands. The example in listing 5.6 shows how an asset identifier can be taken out of the current policy evaluation context and then be used as input parameter for the retrieval of a list of permitted recipients. Just like context operands, information operands can also be used both on the left and right side of ODRL constraints.

5.3.3 Supporting External Obligations

In addition to external information sources, we also require our policy language to support the representation of external obligations (requirement P4). Generally, there are two ways to express obligations in the core ODRL information model. Policies can include obligations either (i) as a dedicated *duty rule*, or (ii) by means of the *duty property* that can be set inside a permission rule. Listing 5.7 illustrates these two types of obligations in an exemplary ODRL policy.

Listing 5.7: Example of an ODRL policy using two obligations. Modified from [Ian18b].

```
{"@context": "http://www.w3.org/ns/odrl.jsonld",
 "uid": "http://example.com/policy:42B",
 // Duty rule
 "obligation": [{
     "action": "delete",
     "target": "http://example.com/document:ABC"
 }],
 "permission": [{
     "action": "distribute",
     "target": "http://example.com/document:XZY",
     // Duty property
     "duty": [{
         "action": "attribute"
     }]
 }]
}
```

In both cases, the obligations are expressed using the *action* class of the ODRL information model. While this sufficiently determines the expected semantics of the included obligations, we require some additional information in the policy when using ODRL as a language for trustworthy distributed usage control. Most importantly, policy issuers must be able to explicitly specify the *target* components that should execute the defined obligations. Otherwise, we run into similar issues as with the information operands in the previous section. For example, a malicious decision point operator could interfere with the usage control enforcement by redirecting the triggered obligations to different (albeit legitimate) execution points than intended by the policy issuer. To prevent man-in-the-middle attacks of this nature, we introduce the concept of *PXP actions* as a subclass of the default ODRL action. Listing 5.8 gives an example of a PXP action that logs the distribution of a particular data asset.

Listing 5.8: Example of a DataSov ODRL obligation using a PXP action.

```
"action": [{
    "@type": "ods:pxpAction",
    "uri": "uri:urn:example:pxp",
    "method": "log",
    "params": {
        "message": "Asset distributed to example recipient.",
        "level": "WARNING"
    }
}]
```

The definition of our PXP action closely resembles the previously introduced definition of the information operand. A PXP action consists of a JSON-LD object of type `ods:pxpAction` with three additional properties. As before, the `uri` property expresses the component identity of the execution point that the obligation should be directed towards. Our decision point verifies this identity using the framework's component authentication scheme. The obligation itself is then refined by the `method` and `params` properties. The set of available PXP methods that can be referenced in obligations depends on the concrete implementation of the targeted component. A list of methods that must be supported by all DataSov PXPs is given as part of our custom ODRL profile in appendix C. Note that our ODRL profile can also be extended with additional methods that are implemented by customized PXP components.

In addition to referencing specific PXPs in obligations, we also define the concept of *PEP actions*. Unlike PXP obligations, PEP actions are automatically executed by the enforcement point that enforces the resulting decision.

This is useful for expressing obligations that can be executed directly on the usage-controlled data, such as data modifications or anonymizations. Hence, PEP actions constitute the mechanism that implements `MODIFY` decisions in the DataSov usage control system. Besides altering data, PEP actions can also be used to modify the *policy* of the associated data asset, which allows data owners to regulate policy updates in preparation of subsequent data transmissions (see section 3.2.5). The definition of PEP actions is essentially the same as when referencing PXPs, except that no dedicated URI property is required anymore. However, unlike in the core ODRL information model, policy issuers can still define a set of parameters for the enforcement point to use when executing the obligation. A list of all methods that are implemented by the DataSov PEPs is given as part of our custom ODRL profile in appendix C. Listing 5.9 shows an example for a PEP action that redacts the contents of a privacy critical event parameter as part of the usage control enforcement.

Listing 5.9: Example of a DataSov ODRL obligation using a PEP action.

```
"action": [{
    "@type": "ods:pepAction",
    "method": "modifyParam",
    "params": {
        "key": "employeeName",
        "match": "\.*",
        "mask": "[REDACTED]"
    }
}]
```

5.3.4 Representing Provenance Information

As final extension of the core ODRL information model, we also need to specify how provenance information should be represented in the policy language (requirement P5). In general, we have two goals regarding the support of provenance data in ODRL policies. First, we want to leverage the policy enforcement process as a means to control provenance tracking on usage-controlled data assets. Furthermore, the collected provenance information should be made available as input for the policy evaluation itself, in order to allow policy issuers to specify rules and constraints based on the previous usage history of data assets. We achieve the first of these goals by supporting the collection of provenance data though PXP obligations, as introduced in the previous section. For this, we include a provenance obligation method in all DataSov PXPs, which accepts a provenance object as input parameter and updates the local Provenance Storage Points accordingly. Data providers can then enforce provenance tracking as part of the usage control by simply including corresponding obligation definitions into their ODRL policies. Listing 5.10 gives an exemplary PXP action that collects provenance information for newly generated data assets attributed to a certain individual.

Listing 5.10: Example of a DataSov ODRL obligation for provenance tracking.

```
"action": [{
    "@type": "ods:pxpAction",
    "uri": "uri:urn:example:pxp",
    "method": "provenance",
    "params" : {
        "prosps": ["uri:urn:example:prosp"],
        "provenance": {
            "@type": "ods:provenance",
            "entities": [{"@type":"ods:ctxOperand", "key":"assetId"}],
            "activities": ["uri:urn:example:process:5778"],
            "agents": ["uri:urn:example:person:johndoe"],
            "relations": [
                [{"@type":"ods:ctxOperand", "key":"assetId"},
                  "prov:wasGeneratedBy",
                  "uri:urn:example:process:5778"],
                [{"@type":"ods:ctxOperand", "key":"assetId"},
                  "prov:wasAttributedTo",
                  "uri:urn:example:person:johndoe"]
            ]
        }
    }
}]
```

Our provenance PXP action, as illustrated in listing 5.10, accepts just two parameters. Most importantly, the provenance parameter holds a JSON-LD object representing the provenance information that should be recorded. To represent provenance information in our ODRL policies, we build on the existing PROV data model that has been standardized by the W3C [Bel13]. More concretely, a provenance object is defined by the type `ods:provenance` and contains a total of four lists representing the `entities`, `activities`, `agents`, and `relations`. As specified in the PROV data model, we identify each entity, activity, and agent by their unique URI. Furthermore, we represent the relations between these three PROV base classes by triples in the form of `[subject, relation, object]`. Note that we do not need to hard-code the identifiers of the referenced provenance objects, since our ODRL profile also allows to use context and information operands inside the parameter lists. For example, the action definition in listing 5.10 dynamically retrieves the identifier of the concerned data asset as provenance entity from the current evaluation context. Once the PXP executes this action as part of the policy evaluation process, the information represented in the specified provenance object is added to the provenance state at the local Provenance Storage Points. To determine which exact ProSPs should be updated, policy issuers can provide a list of component URIs in the `prosps` parameter. The execution point will then connect to all listed ProSPs and update the stored provenance information via the ProSP interface, as described in section 5.1. To prevent man-in-the-middle attacks, the DataSov PXPs always authenticate these connections against the specified URIs. If a provenance update fails for any reason, the execution point declares the corresponding obligation as *not fulfilled* in the ODRL information model. This in turn causes the ODRL decision point evaluating the policy to deny the requested data usage by default.[1] That way data owners can use the DataSov framework to enforce provenance tracking on shared assets in a reliable and trustworthy fashion.

In addition to acquiring provenance data during usage control enforcement, we also want to utilize the collected information in ODRL policies. For this, we leverage our previously described information operand concept to

[1] Alternatively, ODRL also allows to specify concrete *consequences* of an unfulfilled obligation [Ian18b]. However, this is not yet supported by the DataSov decision point.

retrieve provenance information from PIPs during policy evaluation. Listing 5.11 shows an example policy that uses our DataSov ODRL profile to express usage rules on data assets on the basis of their provenance history.

Listing 5.11: Example of a DataSov ODRL policy using an information operand to check provenance data as part of a constraint.

```
{"@context": "https://gitlab.cc-asp.fraunhofer.de/datasov/core/-/raw/master/pdp/profile/ods.jsonld",
 "profile": "https://gitlab.cc-asp.fraunhofer.de/datasov/core#ods-ttl",
 "uid": "uri:urn:example:policy:odrl:11",
 "prohibition": [{
   "target": "uri:urn:example:asset:891",
   "action": "delete",
   "constraint": [{
     "leftOperand": {
       "@type": "ods:pipOperand",
       "uri": "uri:urn:example:pip",
       "method": "provenance",
       "params": {
         "prosps": ["uri:urn:example:prosp"],
         "request": "relations",
         "match": ["\.*", "prov:used", "uri:urn:example:asset:891"]
       }
     },
     "operator": "neq",
     "rightOperand": []
   }]
 }]
}
```

This policy prohibits the deletion of a particular asset on the condition that is has been previously used by any activity. To express this restriction in ODRL, we use a constraint with a left operand of type `ods:pipOperand` and the method `provenance`. By using the parameter `request`, we can specify the desired type of provenance information, in this case the relations. The parameter `match` allows us to filter the requested provenance information at the PIP using regular expressions. In this case we filter for relations of type `prov:used`, as well as the concerned asset. Finally, with the `prosps` parameter we can again specify the expected component identities of the concrete ProSPs, which should be used as sources for the requested provenance information.

5.3.5 The DataSov Policy Decision Point

To summarize, we extended the core ODRL information model in order to define a suitable ODRL-based policy language for DataSov that meets our requirements. The DataSov ODRL profile introduces the concept of context and information operands to support referencing external information sources in policies (requirement P3). Furthermore, we added two new action classes to represent obligations that must be executed at the enforcement point and execution point, respectively (requirement P4). Finally, we included support for referencing provenance information in ODRL policies according to the the W3C PROV data model. This allows us to both control provenance tracking via usage control obligations, as well as rely on provenance data for usage control constraints (requirement P5). Figure 5.6 gives a complete overview of the extensions that our DataSov profile makes to the core ODRL information model. The complete definition of the profile's vocabulary, as well as a corresponding JSON-LD parsing context, is given in appendix C.

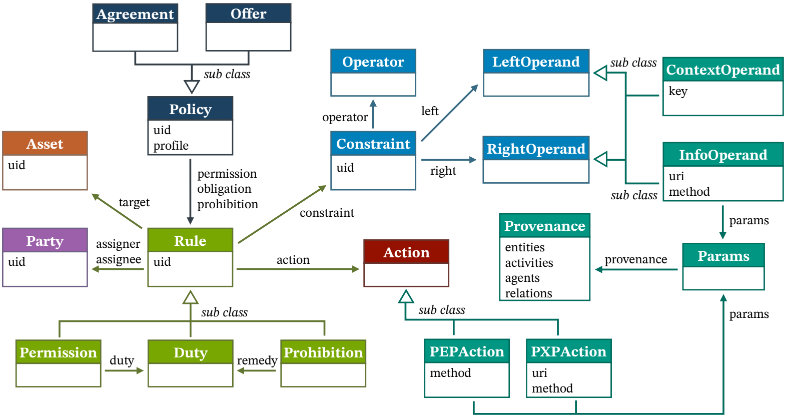


Figure 5.6: The extended information model of the DataSov ODRL profile.

In addition to extending the policy language, we also implemented a lightweight decision point engine for the DataSov framework that can parse and evaluate policies adhering to our ODRL profile. The developed policy engine is implemented in C++ using the RapidJSON[1] parsing library, and can be compiled into SGX enclaves using the Asylo framework. However, as a prototypical implementation we do not yet support the full ODRL information model. Most importantly, our decision point does not allow the expression of offers and agreements about digital rights, since usage rule negotiation is out of scope for the DataSov framework. The program code of our PDP is available online.[2]

In order to verify the feasibility of the developed solution, we tested the performance of the implemented ODRL decision point on an STM32MP157C-DK2[3] development board. We choose this platform since it is the slowest component used in our final evaluation scenario (see chapter 7). To provide a comparison with a more widespread ARM-based embedded device, we also tested our PDP on a Raspberry Pi 3 Model B+ single board computer. Figure 5.7 shows the determined policy evaluation times in milliseconds. We conducted our performance evaluation with four example policies of increasing complexity. The test policies contain between 10 and 25 rules (R), with each rule including either 10 or 25 constraints (C). Furthermore, all policies are constructed to avoid short-circuited rule and constraint evaluation, in order to ensure that the policy engine always has to evaluate all existing rules and constraints. To exclude the influence of the network stack and other components on the execution times, the example policies do not include any external PIP operands or PXP obligations. Our results show that a medium-sized policy with 10 rules and 10 constraints per rule can be evaluated in about six milliseconds on the ARM TrustZone board, and about two milliseconds on the Raspberry Pi. Larger policies, consisting of 25 rules with 25 constraints each, require about 35 and 14 milliseconds to evaluate, respectively. However, policies with more than 10 rules and

[1] https://github.com/Tencent/rapidjson *(accessed on 12/08/2023).*

[2] https://gitlab.cc-asp.fraunhofer.de/datasov/core/-/tree/master/pdp *(accessed on 12/08/2023).*

[3] https://www.st.com/resource/en/data_brief/stm32mp157c-dk2.pdf *(accessed on 12/21/2023).*

constraints rarely occur in realistic scenarios. Hence, we can conclude that our decision point implementation is fast enough for its intended purpose, even in applications with resource-constrained and embedded devices.

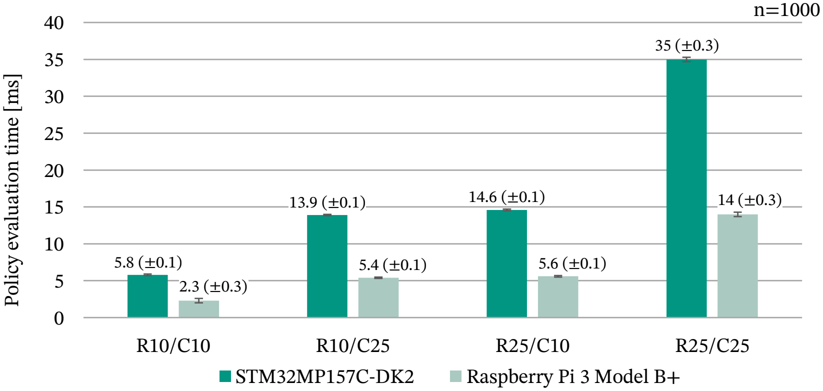


Figure 5.7: Mean evaluation times of DataSov ODRL policies on an STM32MP157C-DK2 and a Raspberry Pi 3 Model B+ in milliseconds. The policies differ in the number of evaluated rules (R) and constraints per rule (C). The standard deviation is given in brackets.

5.4 Conclusion

In this chapter we presented our proof of concept implementation for a trustworthy distributed usage control and provenance tracking framework called *DataSov*. The DataSov framework is based on the system design proposed in chapter 3 and integrates the results and contributions presented in chapter 4. As a result, the framework allows to dynamically deploy and provision distributed usage control components as Intel SGX enclaves, TPM-protected applications, and ARM TrustZone devices. During system operation, all deployed components are automatically authenticated using certificates, while being protected against malicious manipulations through heterogeneous remote attestations. We also designed and implemented a suitable rollback protection mechanism for our system, which is based on a central state store. Together, this achieves the first part of research contribution RC6. We cover the second part of this contribution, i.e., the evaluation of the framework in the realm of smart manufacturing, in chapter 7 of this thesis.

As second contribution in this chapter, we developed and evaluated an extension of the ODRL policy model for trustworthy distributed usage control (research contribution RC2). Our custom ODRL profile allows data providers to manage provenance tracking and reference the collected information directly via usage control policies. It also provides support for the expression of PXP obligations and PIP operands. Furthermore, we facilitate the enforcement of such policies in the DataSov framework by providing a corresponding ODRL decision point implementation.

6 Estimating Trustworthiness

In the previous chapters we described the concept and implementation of our trustworthy distributed usage control and provenance tracking system, which leverages trusted computing technologies to prevent malicious data receivers from tampering with critical system components. However, simply *offering* such technical measures to protect distributed usage control enforcement is not sufficient to reach all of our thesis goals. There must also be a way for (potential) data providers to actually *determine* the trustworthiness of a particular usage control system, in the sense that it indeed deploys adequately chosen trusted computing technologies that can enforce the specified usage rules on the shared data. To achieve this, two challenges need to be solved. First, we must define a method to estimate the trustworthiness of a distributed usage control system, i.e., give an indication to what extent the proper enforcement of a specific usage control policy can reasonably be expected in the current protection state. Furthermore, the determined estimations must be signaled back to the policy issuer, in order to facilitate informed data sharing decisions.

In this chapter we address these research questions by developing a trustworthiness estimation approach for distributed usage control systems and integrating it into our DataSov framework. For this, in section 6.1 we first motivate the need for a descriptive trustworthiness score in DataSov and lay out our specific goals and requirements for it. Afterwards, in section 6.2 we introduce a formal model that can represent the current state of the distributed usage control system and its protection mechanisms. Based on this formal model, in section 6.3 we then define a suitable trustworthiness score for deployed usage control policies and show that it fulfills our previously identified requirements. Section 6.4 describes how the developed trustworthiness score is integrated into the DataSov framework. Finally, we conclude this chapter in section 6.5 with a brief summary.

Some of the contributions presented in this chapter improve on our previous work, which we have partially published in three research papers. In [Wag19a] and [Wag22b] we discussed earlier versions of our formal model describing the protection state of distributed usage control systems and corresponding trustworthiness scores. Furthermore, in [Wag22a] we proposed a visualization concept for the determined estimations, which constitutes a precursor to the DataSov trust dashboard presented in section 6.4.

6.1 Motivation and Requirements

The DataSov framework relies on a transitive remote attestation concept that automatically verifies the integrity of distributed usage control components. However, this alone does not take into account the different capabilities, strengths, and weaknesses of the underlying (heterogeneous) trusted computing technologies that are protecting the individual system components. For example, while TPMs have a much larger TCB than Intel SGX enclaves and do not provide isolated execution environments for individual applications, they are also much more widely available and can be used seamlessly with unmodified legacy applications. Due to these specific benefits and drawbacks, component operators must regularly conduct a trade-off to make an adequate selection between the different technologies on offer. Likewise, data providers must decide if the available technological protection is sufficient for their individual security needs. However, which technologies should be deemed acceptable for which components depends on the value of the shared data, the level of risk aversion shown by the data owner, as well as the individual preferences and convictions of the various system participants, and thus cannot be universally decided. For example, some data providers might not trust Intel and hence want to avoid critical policies to be enforced by SGX enclaves, while other participants have different opinions and sentiments. Furthermore, in a distributed usage control system with many different participating stakeholders and component operators, the set of active system components and their individual protection states undergo constant changes. Hence conducting a static, one-time security analysis to determine the adequacy of the deployed components and technologies is usually not feasible. In addition, there may also be system components that must be trusted independently of the deployed protection mechanisms, in order to reliably enforce a particular usage control policy. As pointed out in section 3.4.3, trusted computing alone cannot adequately protect against the malicious deletion of critical information, such as collected provenance data or log messages. Because of this, we must find a way to ensure that such components are operated by trustworthy subjects, ideally by the policy issuers themselves, before allowing data and corresponding policies to be shared. Note that this determination also cannot be done statically for the entire system, because it depends both on the specific policy, as well as the degrees of trust that the data owner attributes to individual component operators.

We deal with these issues and challenges by developing a method to estimate the overall *trustworthiness* of the distributed usage control system. More concretely, we provide policy issuers in DataSov with a quantitative indicator that reveals how well the distributed usage control system in its current protection state is prepared to enforce a specific policy. The evaluated protection state includes the deployed usage control components, their operator reputation, the conducted remote attestations, and the applied trusted computing technologies. Furthermore, due to the aforementioned reasons, we conduct the trustworthiness evaluation dynamically during runtime and from the subjective point of view of the individual data providers.

Interpretation of trustworthiness. Over the past years, the term *trust* has been well researched in various scientific disciplines [Cho15]. For our purposes, we adopt a notion of trust as a measure of confidence in the behavior of a (technical) entity, which is an interpretation often applied in the realm of networking and information security [Cho15]. More concretely, Jøsang et al. define trust as "*the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends*" [Jøs07]. Subjective probabilities of this nature are usually interpreted as Bayesian *degrees of belief* in contrast to a frequentistic understanding of probability [Bey16]. In our scenario, we are interested in the level of trust that data providers have in the ability of a distributed usage control system to successfully enforce their usage rules. Even though the term *trustworthiness* is sometimes used synonymously with the term *trust* in the literature [Yan09, Nei15], we see value in distinguishing those two concepts. Cho et al. define trustworthiness as evidential statements that can be used to update (i.e., increase or reduce) previous trust assessments [Cho15]. In this sense, trustworthiness often originates from direct observations of data about the past behavior of the trustee, following a frequentistic interpretation of probability. However, in cases where no reliable data is available and/or a-priori estimations are necessary, trustworthiness can also be determined based on subjective degrees of belief, e.g., in the form of expert opinions [Hub16, pp. 37–38]. For the remainder of this section, we follow the latter interpretation and consider the trustworthiness of a distributed usage control system to be an upper limit for the degree of trust that a data provider can reasonably place in the enforcement of a particular usage control policy, given a number of expert estimations regarding the current technological protection state of the system.

Goals and requirements. Our goal in this chapter is to define a score that quantifies this notion of trustworthiness, and integrate it into our DataSov usage control framework. The trustworthiness score should represent to what degree a data provider can (at most) expect a specific policy to be enforced correctly by the distributed usage control system. More concretely, we require a function $t : \mathcal{P} \times \mathcal{S} \rightarrow [0,1]$, which maps a policy $P \in \mathcal{P}$ together with the current system state $S \in \mathcal{S}$ to the estimated trustworthiness score between 0 and 1. Evaluating this function allows data providers to distinguish system states that are favorable for the enforcement of their particular policies (score close to 1) from scenarios where the policy enforcement can likely be bypassed (score close to 0). Note that, although defined between 0 and 1, we see this score as a conveniently constructed quantitative indicator instead of a mathematical probability. We discuss the issues with interpreting this value as a probability later in section 6.3.4. In the following two sections, we describe the construction of our trustworthiness score t. We base this function on the capabilities of the deployed trusted computing technologies and the subjective level of trust that a data provider places in the operators of remote components. Furthermore, our score must fulfill the following five requirements in order to provide useful information to data providers.

(S1) Technological propriety: The trustworthiness score should indicate to what degree the trusted computing technologies currently deployed in the distributed usage control system are feasible to enforce the policy in question. Technological feasibility should be determined by experts based on the (partial) protection goals of the usage control components participating in the policy enforcement (see section 3.3.1).

(S2) Operator propriety: The trustworthiness score should indicate whether usage control components, which are either (i) unattested or (ii) cannot be sufficiently protected by the underlying trusted computing technologies, are operated by trustworthy stakeholders.

(S3) Minimality: The trustworthiness score of a policy must be limited by the technological adequacy of the least strongly protected usage control component, as well as the policy issuer's opinion of the least trusted component operator that is relied on for the policy enforcement.

(S4) Monotony: If a policy P_1 relies on a superset of usage control components and operators for its enforcement compared to policy P_2, its trustworthiness score must be equal or lower: $\forall S \in \mathcal{S} : P_1 \supseteq P_2 \implies t(P_1, S) \leq t(P_2, S)$.

(S5) Language agnosticism: The trustworthiness score should be constructed independently of a concrete policy language.

Reputation systems. Trustworthiness estimations can be based not just on expert opinions, but also on concrete data that has been collected about the entities in question. *Reputation systems* follow the latter approach by analyzing the past behavior of assessed entities to determine their trustworthiness [Has17]. The rationale behind this is that, if an entity has shown cooperative behavior in the past, it will likely continue to do so in the future and hence can be considered trustworthy. As elaborated in section 3.1.2, reputation systems have been used previously in the context of usage control systems [Yan09, Bal13, Nei15, Tru16a]. However, they are not feasible for our purpose of estimating trustworthiness in distributed usage control systems that are protected by remote attestations. The main issue is that reputation systems require an adequate set of observations about a component's behavior to make reliable predictions. This also includes assessments about the nature of the observed actions, i.e., if the component acted cooperatively or maliciously. However, in the case of distributed usage control enforcement, we need to provide reliable trustworthiness estimations *before* critical data is being shared. Furthermore, in our scenario a hostile data receiver could initially build a good reputation score by operating honest components, and then "spend" it on bypassing the protection policies of valuable data. Hence, our trustworthiness score should focus on the *current* protection state of the considered system components, instead of their past actions. Finally, reputation systems are designed to predict the most likely future behavior of entities. However, with remote attestation we already have a technical mechanism in place that achieves this purpose. Our notion of trustworthiness is instead aimed at determining if the available trusted computing mechanisms can adequately protect the enforcement process of the deployed usage control policies.

For these reasons, we define our trustworthiness score based on expert opinions regarding the capabilities of trusted computing technologies to achieve the protection goals of individual usage control components. We also consider the (subjective) trustworthiness of remote component operators from the data provider's point of view.

6.2 Formal Model

This section introduces the formal model that we use to represent a particular state $S \in \mathcal{S}$ of our distributed usage control system. Each system state consists of a component model, an attacker model, and a trust model. As mentioned in the introduction, the original publication of our formal system model, on which this section is based, can be found in [Wag22b].

6.2.1 Component Model

The purpose of the component model is to represent all instances of usage control components that are currently deployed in the system, as well as their security properties, remote attestations, and dependencies. We express the deployment status of the distributed usage control system with the help of an *instance graph.* The vertices of this graph represent the deployed component instances, i.e., the currently active PEPs, PDPs, PRPs, PIPs, PXPs, and ProSPs, as well as the usage-controlled data processing applications. The edges of the graph represent the existing trust relationships between individual component instances.

Definition 6.1 (Instance graph). *Let V be the set of component instances in a distributed usage control system. Let $E \subseteq V \times V$ be a set of directed edges over V, with $(v,w) \in E$ iff. usage control component v requires the cooperation of component w for its duties. We call the tuple $I := (V,E)$ instance graph of the distributed usage control system.*

The goal of the instance graph is to describe the current configuration of the distributed usage control system and to identify what concrete trust dependencies must be covered with remote attestations. Note that the instance graph is closely related, but not equivalent to the *trust dependency graph* we constructed during our conceptual security analysis in section 3.3.3. The graph in fig. 3.13 shows all necessary trust dependencies between the different *types* of usage control components. The instance graph, on the other hand, represents the concrete *instances* of these usage control components and their individual dependencies. As such the instance graph is not universal, but specific to a certain use case and it can change over time as new components are deployed in the distributed system. Furthermore, not all trust dependencies identified in fig. 3.13 must necessarily be included in the instance graph. For example, if a certain Policy Information Point does not provide any provenance information, its vertex in the instance graph does not have any dependencies to ProSPs. To illustrate this concept further, fig. 6.1 gives an example of a simple instance graph representing a usage control system with two participants Alice and Bob.

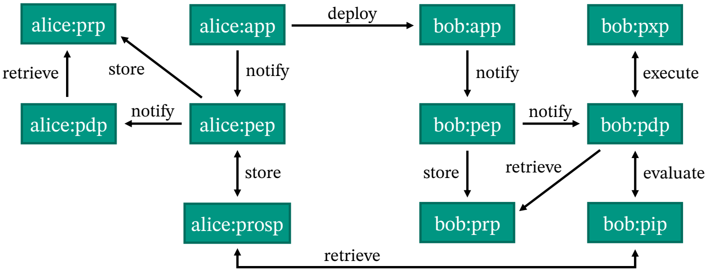


Figure 6.1: Example of an instance graph with two system participants Alice and Bob.

In addition to the dependencies between instances, we also need to represent the component properties on which the trustworthiness estimation should ultimately be based. For this, we introduce two further mappings on the instance graph. First, the *mechanism mapping* $m\colon V \to M$ associates each component instance with the specific trusted computing mechanism that protects it. Since we only consider TPMs, Intel SGX, and ARM TrustZone in the scope of this thesis, in our case the set of mechanisms results to $M \coloneqq \{\bot, \mathrm{TPM}, \mathrm{SGX}, \mathrm{TZ}\}$. However, in the future both our usage control framework, as well as the trustworthiness estimation method, can be extended with additional mechanisms. Furthermore, we define the set of usage control participants O and the *operator mapping* $o\colon V \to O$, which maps each component instance to the system participant that is responsible for operating it. We assume the operators to be in full control of both the component's hardware and software stack. For the example in fig. 6.1, the set of usage control participants results to $O \coloneqq \{\mathrm{Alice}, \mathrm{Bob}\}$. Finally, the component model must also include information about the conducted remote attestations. To represent this, we define the *attestation mapping* $\mathit{att}\colon V \to 2^V$. For each $v \in V$, this function specifies the set of component instances that have been successfully attested by component v until this point in time. The complete component model is then formed as the tuple of instance graph, mechanism and operator mapping, as well as the attestation mapping for all system components. Note that the resulting component model represents the state of the distributed system at one particular point in time. Hence the instance graph, as well as all three mappings that are based on it, will change during the system operation.

Definition 6.2 (Component model). *Let $I = (V, E)$ be the instance graph of a distributed usage control system and $\mathit{att} : V \to 2^V$ an attestation mapping over V. Let $m : V \to M$ be a mechanism mapping and $o : V \to O$ an operator mapping for I. We call the tuple $\mathcal{C} = (I, \mathit{att}, m, o)$ component model of the distributed usage control system.*

6.2.2 Attacker Model

Besides giving a description of the currently active system components, our formal model must also represent the expected attackers and their capabilities. As part of our security analysis in section 3.3.2, we have already introduced three basic types of attackers that a trustworthy usage control system must protect against. A *network attacker* has the ability to intercept and manipulate messages that are exchanged between distributed usage control components. A *software attacker* additionally has unprivileged access to the components, but is still restrained by the operating system. Finally, the *privileged attacker* has complete physical access to the component and can manipulate its software stack at will, including the boot loaders and operating system. However, we assume that the trusted computing hardware cannot be manipulated even by privileged attackers. We represent these capabilities in our formal model using a set of three attackers $A := \{\mathit{network}, \mathit{software}, \mathit{privileged}\}$. Note that there is a strict ordering of the attackers in A according to their capabilities: *network* < *software* < *privileged*. To describe which attackers are expected on which component instances, we introduce the *attacker mapping* $a : V \to A$. Due to the strict ordering, it is sufficient to only declare the *most capable* attacker on each component, with all less capable attackers being implicitly included. Also note that the attacker mapping expresses the *subjective view* of a single usage control system participant, and hence differs for each stakeholder. For example, each system participant will expect components that are under the influence of a competing organization to be attacked by malicious administrators, i.e., privileged attackers. Our own components, however, are usually not considered to be threatend by their administrators.

In addition to the expected attackers, we also need to define the protection goals that are relevant for the different usage control components. As part of our security analysis in section 3.3.1, we have already identified four (partial) protection goals for each type of usage control component. These protection goals include data confidentiality, code and data integrity, as well as protection against data deletion. Table 3.1 in section 3.3.1 shows which of these protection goals are relevant for the correct execution of each type of usage control component. However, as pointed out earlier, different trusted computing technologies provide different benefits and drawbacks. Because of this, it is useful to break down the protection goals even further in our formal model. In section 4.1 we distinguished trusted computing technologies based on their capabilities to secure data confidentiality and integrity in three different phases, namely, (i) during data transmission, (ii) while data is stored, and (iii) while data is actively processed. To represent these differences in our attacker model, we identify a total of eight different protection goals $G := \{C_x, I_x \mid x \in \{tra, proc, store\}\} \cup \{I_{code}, I_{del}\}$. This allows us to formulate a *goal mapping* $g : V \to 2^G$, which associates each system component with the set of protection goals that must be fulfilled at this component. Note that, due to the included data processing applications, this mapping is (partially) scenario-specific. We give an example of a concrete goal mapping as part of our evaluation scenario in section 7.3.2. Ultimately, the attacker model follows as the tuple of attacker mapping and goal mapping.

Definition 6.3 (Attacker model). *Let $I = (V, E)$ be the instance graph of a distributed usage control system. Let $A := \{network, software, privileged\}$ be the set of attackers and $G := \{C_x, I_x \mid x \in \{tra, proc, store\}\} \cup \{I_{code}, I_{del}\}$ be the set of protection goals. Let $a : V \to A$ be an attacker mapping and $g : V \to 2^G$ a goal mapping for I. We call the tuple $\mathcal{A} = (a, g)$ attacker model for the distributed usage control system.*

6.2.3 Trust Model

The final part of the formal model represents the initial trust statements on which to base the trustworthiness score. As mentioned before, our score relies on expert estimations concerning the suitability of trusted computing technologies for various tasks. In addition, the level of trust that data providers have in the operators of remote usage control components must also be considered. We formalize these initial trust statements using two *trust estimation functions.* First, the *operator* trust estimation function $t_o : O \rightarrow [0,1]$ describes the degree of trust that a potential data provider has in each component operator. This trust statement is quantified as a subjective degree of belief between 0 (known malicious) and 1 (fully trusted). Similarly, the *mechanism* trust estimation function $t_m : M \times G \times A \rightarrow [0,1]$ describes the degree of trust in the capability of each trusted computing mechanism in M to enforce a specific protection goal in G against an attacker in A. Sensibly, we demand that a well-defined mechanism trust estimation follows the previously specified attacker order, i.e., a stronger attacker cannot have a higher trust estimation than a weaker one for the same mechanism and protection goal. However, unlike the operator trust estimations, these degrees of belief usually cannot be decided by the data providers themselves. Instead, we assume the mechanism trust estimation function to be determined by technology experts, based on security reviews of the considered trusted computing mechanisms. Later, in section 6.4.3, we give exemplary values for this function, which we also use for our evaluation. The formal trust model is then represented by the tuple of both trust estimation functions.

Definition 6.4 (Trust model). *Let I be the instance graph of a distributed usage control system with the set of component operators O. Let A be a set of attackers, G a set of protection goals, and M a set of mechanisms. Let $t_o : O \rightarrow [0,1]$ be an operator trust estimation function and $t_m : M \times G \times A \rightarrow [0,1]$ a well-defined mechanism trust estimation function. We call the tuple $\mathcal{T} = (t_o, t_m)$ trust model for the distributed usage control system.*

Finally, a particular usage control system state $S \in \mathcal{S}$ can now be formally represented by a triple consisting of component model, attacker model, and trust model: $\mathcal{S} \coloneqq \{S \mid S = (\mathcal{C}, \mathcal{A}, \mathcal{T})\}$. Of those, the component model $\mathcal{C}$ is responsible for describing the distributed system at a particular point in time, including the component deployments and the conducted remote attestations. The attacker model $\mathcal{A}$ and the trust model $\mathcal{T}$ define the security expectations and the baseline trust statements on which the trustworthiness scores are based. Note that, unlike the objective component model, these parts of the system state express the subjective view of a particular usage control participant, and at least partially depend on the specific application context to which the usage control enforcement is applied. In section 7.3.2 we give a concrete example for a suitable attacker and trust model in a distributed application from the realm of smart manufacturing, as part of our framework evaluation.

6.3 A Trustworthiness Score

After formally representing usage control system states, in this section we introduce a trustworthiness score that fulfills our requirements S1 to S5. For this, we first describe how to identify the distributed system components that are relevant for the enforcement of a particular usage control policy. Then we present the definition of our trustworthiness score and assert that it indeed complies with the previously set requirements. Finally, we discuss what issues must be solved in order to interpret the defined score as a probability.

As mentioned in the introduction, this section is based on an earlier version of our proposed score construction, which we originally published in [Wag22b].

6.3.1 Usage Control Operations

The instance graph included in the formal component model represents the totality of all existing usage control components and their dependencies. However, for our trustworthiness score we are usually interested in just a small part of the distributed system, such as the subset of components that are required to enforce a particular usage control policy. We express this with the concept of *usage control operations.* A usage control operation is a connected subgraph of the system's instance graph, which represents the selection of components and trust dependencies that are necessary for successfully executing a particular system function (e.g., enforcing a policy). The particular system component which initiated a usage control operation (e.g., a data processing application) is called the *root* of the operation. Root components are characterized by having no incoming trust dependencies, and there must be exactly one root component in a usage control operation. Furthermore, note that not all trust dependencies identified in the underlying instance graph must also be included in the usage control operation.

Definition 6.5 (Usage control operation). *Let $I = (V, E)$ be the instance graph of a distributed usage control system. Let $J = (\bar{V}, \bar{E})$ be a subgraph of I with $\bar{V} \subseteq V$ and $\bar{E} \subseteq E \cap (\bar{V} \times \bar{V})$. We call J usage control operation on I, if*

$$J \textit{ is a connected subgraph} \tag{6.1}$$

$$\exists! x \in \bar{V} : \ \mathrm{indeg}(x) = 0 \tag{6.2}$$

$$(u, v) \in \bar{E} \Leftrightarrow \ u \textit{ depends on } v \textit{ in this operation.} \tag{6.3}$$

The unique component instance x is called root of the operation J. We denote the set of all usage control operations for the instance graph I by $Op(I)$.

In order to define our trustworthiness score based on usage control policies, we need to translate concrete policy definitions into formalized usage control operations. This process obviously depends on the policy language. Since we want to preserve language agnosticism (cf. requirement S5), we abstract from specific policy languages by introducing the function $ops : \mathcal{P} \times V^2 \rightarrow Op(I)$. This function takes a policy $P \in \mathcal{P}$, as well as the tuple of policy deployer and recipient $(d, r) \in V^2$, and returns the usage control operation that represents the enforcement process of that particular policy. This allows us to define our trustworthiness score exclusively on the formal model and independently of concrete policy languages. However, the function ops must still be specified for each supported policy engine. Listing 6.1 shows the algorithm that evaluates the function ops for the ODRL-based policy language used in DataSov. This algorithm iterates the provided instance graph and collects the relevant usage control components, together with their dependencies, according to the provided policy. Line 2 of listing 6.1 adds the initial dependency between the policy deployer and the recipient. Line 3 then gathers all dependencies from the recipient to the enforcement points. The PEPs are then iterated to determine the set of associated PDPs (lines 4-8). This loop also collects the dependencies from the PEPs to any directly used ProSPs, as well as to the retrieval points. To improve readability, we denote adding a set of edges together with the connected components using the arrow symbols $\overset{+}{\leftarrow}$ for uni-directional and ${}^{+}\!\!\leftrightarrow$ for bi-directional trust dependencies. Once the set of decision points is complete, it is itself iterated to extend the assembled usage control operation with the necessary dependencies to PXPs and PIPs. These dependencies are now taken from the provided ODRL policy. Line 11 collects all execution points that are specified in ODRL actions, while line 12 analyzes all ODRL constraints that include PIP operands. We identify these components in the policy via their unique URI (see section 5.3). The final two loops in lines 15-20 then iterate over the extracted PXPs and PIPs to collect the missing dependencies to Provenance Storage Points, which are referenced in the policy via the `prosps` parameter of ODRL provenance actions and constraints.

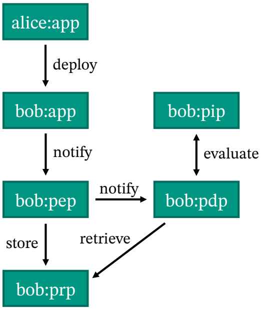
Listing 6.1: Definition of the function ops for the DataSov ODRL profile.

```
   **Input:** Instance graph $I = (V, E)$
   **Input:** ODRL policy $P \in \mathcal{P}$
   **Input:** Policy deployer and recipient $(d, r) \in V^2$
   **Output:** Usage control operation $J \in Op(I)$
1  $J \leftarrow (\emptyset, \emptyset)$; $pdp, pxp, pip \leftarrow \emptyset$;
2  $J \;{}^{+}\!\!\leftarrow \{(d, r)\}$;
3  $pep \leftarrow \{(u, v) \in E \mid u = r \wedge type(v) = \text{PEP}\}$;
4  **forall** $(a, b) \in pep$ **do**
5      $pdp \leftarrow pdp \cup \{(u, v) \in E \mid u = b \wedge type(v) = \text{PDP}\}$;
6      $J \;{}^{+}\!\!\leftarrow \{(u, v) \in E \mid u = b \wedge type(v) = \text{PRP}\}$;
7      $J \;{}^{+}\!\!\leftrightarrow \{(u, v) \in E \mid u = b \wedge type(v) = \text{ProSP}\}$;
8  **end**
9  $J \;{}^{+}\!\!\leftarrow pep \cup pdp$;
10 **forall** $(a, b) \in pdp$ **do**
11     $pxp \leftarrow pxp \cup \{(u, v) \in E \mid u = b \wedge v \in P.actions\}$;
12     $pip \leftarrow pip \cup \{(u, v) \in E \mid u = b \wedge v \in P.constraints\}$;
13     $J \;{}^{+}\!\!\leftarrow \{(u, v) \in E \mid u = b \wedge type(v) = \text{PRP}\}$;
14 **end**
15 **forall** $(a, b) \in pxp$ **do**
16     $J \;{}^{+}\!\!\leftrightarrow \{(u, v) \in E \mid u = b \wedge v \in P.actions.params.prosps\}$;
17 **end**
18 **forall** $(a, b) \in pip$ **do**
19     $J \;{}^{+}\!\!\leftrightarrow \{(u, v) \in E \mid u = b \wedge v \in P.constraints.params.prosps\}$;
20 **end**
21 $J \;{}^{+}\!\!\leftrightarrow pxp \cup pip$;
22 **return** $J$
```

To further illustrate this algorithm, fig. 6.2 shows an excerpt of a simple ODRL policy P together with the generated usage control operation $ops(P, (\texttt{alice:app}, \texttt{bob:app}))$ based on the instance graph example given earlier in fig. 6.1. Note that the policy deployer $d \in V$ (in this case alice:app) always constitutes the root of the generated usage control operation.

```
"permission": [{
  "target": "urn:alice:asset",
  "action": "use",
  "constraint": [{
    "leftOperand": {
      "@type": "ods:pipOperand",
      "uri": "urn:bob:pip",
      "method": "counter"
    },
    "operator": "lt", "rightOperand": 5
  }]
}]
```

(a) An example ODRL rule



(b) The resulting usage control operation

Figure 6.2: An example application of the algorithm in listing 6.1.

6.3.2 Score Definition

After introducing the concept of usage control operations, we can now proceed with the definition of our trustworthiness score. We base the trustworthiness estimation of a policy on the properties of the components that have been identified by the corresponding usage control operation graph. More concretely, all relevant components must either (i) be operated by stakeholders that are initially trusted to correctly provide the component's services, or (ii) be remotely attested by *all* of their predecessors in the operation graph. In the first case, we can evaluate the operator trust estimation function t_o to determine the extent that the respective components can be trusted. In the second case, the trustworthiness score is instead influenced by the confidence placed in the capabilities of the trusted computing mechanisms that are protecting the usage control component. Hence, we evaluate this part of the score using the mechanism trust estimation function t_m. In the remainder of this section, we show how the previously formalized system state can be used to determine a trustworthiness score based on this approach. First, we define the set of *critical stakeholders* $St_S(J)$ for each usage control operation $J \in Op(I)$. This set identifies all operators $o(v)$ of usage control components $v \in V$, which are the target of an unattested trust dependency in J and hence must be initially trusted for the operation to reliably succeed. Note that, since for our purposes *all* identified transitive trust dependencies must be covered in the operation graph, we consider a component to be unattested if just one of its predecessors in J did not conduct a remote attestation.

Definition 6.6 (Critical stakeholders). *Let $S = (\mathcal{C}, \mathcal{A}, \mathcal{T})$ be the state of a distributed usage control system with instance graph $I \in \mathcal{C}$, operator mapping $o \in \mathcal{C}$, and attestation mapping $att \in \mathcal{C}$. For each usage control operation $J \in Op(I)$, we define the set of critical stakeholders under the system state S as follows.*

$$St_S(J) := \bigcup_{(u,v)\in J} \{o(v) \mid v \notin att(u)\} \tag{6.4}$$

For the second part of the trustworthiness score, we define the set of *critical mechanism capabilities* $Cap_S(J \mid T)$. This set contains all trusted computing capabilities that we must rely on for the usage control operation J to be sufficiently protected, under the assumption that the components operated by the stakeholders given in T are already trusted. Analogously to the trust model, we represent each mechanism capability by a triple $(\mathbf{m}, \mathbf{g}, \mathbf{a}) \in M \times G \times A$, consisting of a trusted computing mechanism $\mathbf{m}$, a protection goal $\mathbf{g}$, and an expected attacker $\mathbf{a}$. The set of required mechanism capabilities is extracted from the usage control operation graph by evaluating the mechanism mapping and attacker model on it (cf. lines 4-13 of the algorithm in definition 6.7). Additionally, in order to get a useful set, we also need to remove duplicate combinations of mechanism and protection goals that only differ in the expected attacker. Sensibly, in such cases we retain the mechanism capabilities against the most powerful adversary according to the defined attacker order (cf. lines 8 and 11). The input set of initially trusted stakeholders T allows us to define which components can be considered a-priori trusted, and hence do not have to be analyzed for critical capabilities (cf. line 3). This aids our subsequent definition of the trustworthiness score.

Definition 6.7 (Critical mechanism capabilities). *Let $S = (\mathcal{C}, \mathcal{A}, \mathcal{T})$ be the state of a distributed usage control system with instance graph $I \in \mathcal{C}$, operator mapping $o \in \mathcal{C}$, attestation mapping $att \in \mathcal{C}$, mechanism mapping $m \in \mathcal{C}$, goal mapping $g \in \mathcal{A}$, and attacker mapping $a \in \mathcal{A}$. For each usage control operation $J \in Op(I)$, we define the set of critical mechanism capabilities under the system state S as the output of the following algorithm.*

```
   Input: Usage control operation $J \in Op(I)$
   Input: Initially trusted stakeholders $T \subseteq o(J)$
   Input: System state $S \in \mathcal{S}$
   Output: Critical mechanism capabilities $Caps_S(J \mid T)$
1  $R \leftarrow \emptyset$;
2  forall $(u, v) \in J$ do
3      if $o(v) \in T$ then continue;
4      $\mathbf{m} \leftarrow m(v)$;
5      $\mathbf{a} \leftarrow a(v)$;
6      forall $\mathbf{g} \in g(v)$ do
7          if $\exists \bar{\mathbf{a}} : (\mathbf{m}, \mathbf{g}, \bar{\mathbf{a}}) \in R, \bar{\mathbf{a}} < \mathbf{a}$ then
8              $R \leftarrow R \setminus \{(\mathbf{m}, \mathbf{g}, \bar{\mathbf{a}})\}$;
9          end
10         if $\nexists \bar{\mathbf{a}} : (\mathbf{m}, \mathbf{g}, \bar{\mathbf{a}}) \in R, \bar{\mathbf{a}} > \mathbf{a}$ then
11             $R \leftarrow R \cup \{(\mathbf{m}, \mathbf{g}, \mathbf{a})\}$;
12         end
13     end
14 end
15 return $R$
```

We can now define our trustworthiness score $t : V^2 \times \mathcal{P} \times \mathcal{S} \to [0{,}1]$ based on the sets of critical stakeholders and mechanism capabilities as specified in definitions 6.6 and 6.7. The value of the function t should reflect to what degree the distributed usage control system, characterized by the system state $S \in \mathcal{S}$, can be expected to enforce a policy $P \in \mathcal{P}$ that is deployed and received by the two components $(d, r) \in V^2$. For the policy P to be enforced properly under these circumstances, all component instances that are included in the corresponding usage control operation $J \coloneqq ops(P, (d, r))$ need to behave correctly. To achieve this in the current system state S, it must be ensured that all critical stakeholders $St_S(J)$ are honest and that all critical capabilities $Cap_S(J \mid St_S(J))$ are adequate. Consequently, we can obtain a suitable quantitative indicator by defining our trustworthiness score as the product over the respective operator and mechanism trust estimations t_o and t_m of the identified critical stakeholders and capabilities. However, note that even fully attested components may already be covered by the initial level of trust placed in their operators, if it is higher than what is achievable by the trusted computing mechanisms. To accommodate this in our construction, we maximize the resulting product over all possible sets $T \subseteq o(J)$ of component operators that include *at least* the required critical stakeholders.

Definition 6.8 (Trustworthiness score). *Let $S = (\mathcal{C}, \mathcal{A}, \mathcal{T})$ be the state of a distributed usage control system with instance graph $I = (V, E) \in \mathcal{C}$, operator mapping $o \in \mathcal{C}$, operator trust estimation $t_o \in \mathcal{T}$, and mechanism trust estimation $t_m \in \mathcal{T}$. Let $P \in \mathcal{P}$ be a usage control policy, $(d, r) \in V^2$ the tuple of policy deployer/recipient, and $ops : \mathcal{P} \times V^2 \to Op(I)$ a suitable usage control operation mapping. We define the trustworthiness score for policy P, policy deployer d, and policy recipient r under system state S as*

$$t(d, r, P, S) := \max_{T \in PT} \left(\prod_{s \in T} t_o(s) \cdot \prod_{c \in Cap_S(J|T)} t_m(c) \right) \tag{6.5}$$

with $J := ops(P, (d, r))$ as well as $PT := \{T \in 2^{o(J)} \mid T \supseteq St_S(J)\}$.

Note that in eq. (6.5) the power set PT contains all possible selections of component operators $T \subseteq o(J)$, which fully cover the required critical stakeholders $St_S(J)$. The set $Cap_S(J \mid T)$ then captures all critical mechanism capabilities of attested components that are still required for the enforcement of the policy, under the assumption that the operators in T are already trusted.

Since both the attacker and trust model included in the system state S are defined from the point of view of a particular policy issuer, the resulting trustworthiness score $t(d, r, P, S)$ is also subjective to that specific data provider. Following our notion of trustworthiness as outlined in section 6.1, the value of the function t can be interpreted as an upper limit for the level of trust that the data provider can reasonably place in the ability of the distributed usage control system to properly enforce the given policy. This estimation is based on the degree of trust that the data provider initially places in the component operators, as well as the aptitude of the deployed trusted computing mechanisms to uphold the required protection goals against the expected attackers. However, note that the defined trustworthiness score is generally not equal to the enforcement probability of the policy itself. Later, in section 6.3.4, we discuss the issues of interpreting this value as a probability in greater detail.

6.3.3 Requirement Compliance

We now show the compliance of our trustworthiness score, i.e., that its definition fulfills the requirements S1 to S5 as introduced in section 6.1. The practical evaluation of our trustworthiness score by means of an exemplary use case from the realm of smart manufacturing is presented in the next chapter.

Technological and operator propriety. Requirement S1 (technological propriety) demands that a useful trustworthiness score must indicate to what degree the deployed trusted computing technologies are suitable to protect the enforcement of a particular policy. In addition, requirement S2 (operator propriety) demands that the defined score must also show whether usage control components, which are not sufficiently protected by trusted computing mechanisms, are operated by trustworthy stakeholders. Together, these two requirements ensure that the constructed score adequately reflects the ability of the distributed usage control system to enforce a specific protection policy. This is the most important property for our score to be useful as a quantitative indicator assisting data providers with data sharing decisions.

To achieve both technological and operator propriety in our proposed trustworthiness score, we define a formal model that captures the deployed trusted computing platforms and their operators, as well as the (partial) protection goals and the expected attackers for every usage control component (cf. definitions 6.2 and 6.3). By means of a usage control operation graph we then identify all relevant system components and trust dependencies of the policy in question (cf. definition 6.5). Furthermore, our trust model (cf. definition 6.4) provides expert estimations regarding the protection capabilities of trusted computing technologies (i.e., the function t_m), as well as assessments about the initial trustworthiness of other usage control participants (i.e., the function t_o). Finally, the definition of our trustworthiness score is based on the idea that, in order to ensure a successful policy enforcement, every component in the corresponding usage control operation graph must be covered either by a high initial trustworthiness placed in its operator (determined with t_o), or by the adequacy of the trusted computing mechanisms used to protect this component (determined with t_m). Hence, we construct our score by multiplying together the respective baseline estimations of all required component operators and mechanism capabilities (cf. definition 6.8). This weighs in all estimations that are relevant for the enforcement of this policy. More concretely, the construction in definition 6.8 ensures the following three properties.

(i) The score always includes the operator trust estimations $t_o(s)$ for all critical stakeholders $s \in St_S(J)$, who operate unattested components in the usage control operation J, and hence *must* be initially trusted (cf. definition 6.6). This is ensured by maximizing the score only over those sets of component operators $T \subseteq o(J)$ that fully include all critical stakeholders.

(ii) As a result of this maximization, the score may additionally include the estimations $t_o(s)$ for such (trustworthy) stakeholders $s \in T$, who operate components that are attested but cannot be adequately protected by the deployed trusted computing technologies (i.e., the values $t_m(\cdot)$ for the associated mechanisms are outweighed by the operator trustworthiness).

(iii) The remaining components of the usage control operation J, which cannot be adequately covered by the initial trustworthiness of their operators, must then be protected by trusted computing mechanisms. Our construction ensures that the resulting score includes the mechanism trust estimations $t_m(c)$ for *all* critical capabilities $c \in Cap_S(J \mid T)$, on which these components rely (cf. definition 6.7).

While (i) and (ii) jointly fulfill operator propriety, (iii) ensures that the score indicates the feasibility of the required trusted computing technologies to enforce the necessary protection goals against the expected attackers, and hence fulfills technological propriety. As a result of achieving both requirements, our score can identify (un)trustworthy component operators in the usage control enforcement process as well as distinguish system states with strong technological protection from those with weaker security guarantees.

Minimality. Requirement S3 demands that the trustworthiness score must be minimal, i.e., that it is limited by the adequacy of the least strong mechanism capability, as well as the policy issuer's opinion of the least trusted component operator that are required for the enforcement of the policy in question. The previously discussed properties of technological and operator propriety ensure that the relevant baseline estimations $t_m(\cdot)$ and $t_o(\cdot)$ are always included in the resulting trustworthiness score for the policy. Since these values are between 0 and 1, and the score is defined as a product (cf. definition 6.8), the degrees of belief in the least strong mechanism capability and the least trusted critical operator are upper bounds for the resulting trustworthiness score. For example, if a usage control component that is part of the policy enforcement process relies on a mechanism $\mathbf{m}$, which provides low security guarantees for the required protection goal $\mathbf{g}$ against the expected attacker $\mathbf{a}$, then the trustworthiness score will be limited by the respective degree of belief $t_m(\mathbf{m}, \mathbf{g}, \mathbf{a})$. A proof sketch for this property is given in appendix D.

Monotony. Requirement S4 demands that a policy P_1, which relies on at least the same usage control components as another policy P_2, must not receive a higher trustworthiness score. This ensures that a policy with "greater" demands, such as additional dependencies to PIPs or PXPs, cannot be rated more favorably under the same circumstances. We can formalize this property as $\forall S \in \mathcal{S} : J_1 \supseteq J_2 \implies t(d, r, P_1, S) \leq t(d, r, P_2, S)$ for any policy deployer d, policy receiver r, and with $J_x \coloneqq ops(P_x, (d, r))$ denoting the usage control operations of both policies. Our trustworthiness score fulfills this requirement mainly because it is constructed such that additional trust dependencies in the usage control operations can only *add* more restrictions in terms of critical stakeholders and/or critical mechanism capabilities to the product specified in definition 6.8. As a result, an extended usage control operation can only lead to a lower (or equal) trustworthiness score. In appendix D we provide a complete proof sketch for this property by laying out all possible consequences of including an additional trust dependency in a usage control operation.

Language agnosticism. Finally, our trustworthiness score should also be independent of policy languages (requirement S5). We achieve this by defining our score based on the abstract concept of usage control operations (cf. definition 6.5). The connection between concrete usage control policies and their trustworthiness score is then made with the function ops. This function identifies all usage control components and dependencies that are relevant for a specific policy. Hence, in order to support a new policy language, we only need to appropriately re-define the function ops. Because of this, our trustworthiness score fulfills the requirement of language agnosticism.

6.3.4 Probabilistic Interpretation

In the previous section we asserted the compliance of our trustworthiness score by showing that it fulfills the necessary requirements. This makes our score useful as a quantitative indicator that can point out problematic system states to policy issuers. However, since it is defined as a value between 0 and 1, we can also ask the question to what extent the trustworthiness score can be interpreted as the actual *probability* of policy enforcement. In the remainder of this section, we describe the necessary assumptions and remaining issues to be solved in order to allow a probabilistic interpretation of the defined score.

Our trustworthiness score is based on deriving a set of statements from a usage control policy, which must all be true for the policy enforcement process to be properly secured. These statements correspond to assessments about component operators and trusted computing mechanisms, such as "*component operator Alice is honest*" and "*the SGX technology can ensure the integrity of data processed on the CPU against malicious platform owners*". As described in the previous sections, we define the (expertly estimated) probabilities that these statements are true using the functions t_o and t_m. Then we multiply together the degrees of belief in the veracity of all statements that are relevant for the analyzed policy (cf. definition 6.8). The result is an aggregated score that fulfills our requirement of identifying untrustworthy component operators and weak technological protection. However, for the resulting value to actually resemble the policy enforcement probability, several additional assumptions are necessary. First, we must assume that the formal model considers and adequately represents *all* influences on the enforcement of usage control policies. While in our case the security properties of the used trusted computing technologies are certainly a major influence on the probability of policy enforcement, there are aspects that our model does not consider. For example, a reliable policy enforcement also presupposes that the software implementing the functionalities of distributed usage control components works correctly and does not contain any security-critical bugs. Furthermore, we do not consider the possibility of random (i.e., non-malicious) failures or human errors in setting up the protection mechanisms at the usage control components. Another issue is that, since our attacker model is focused on malicious usage control participants, we assume components operated by trusted stakeholders to always behave correctly, even though this may not be the case.

Second, in order to get accurate trustworthiness scores, we must also ensure that our model is instantiated with correct information. In terms of the component model, this mainly concerns the proper identification of all usage control components and trust dependencies that are relevant for the enforcement of the analyzed policy. As shown in listing 6.1, we achieve this by directly extracting the referenced components from the policy specification. The relevant trust dependencies between the usage control components have been determined as part of our security analysis in section 3.4. Because of this, we can reasonably expect to identify all relevant usage control components and their dependencies. The remaining parts of the component model (i.e., mechanism, operator, and attestation mapping) can then be easily determined during the operation of the system. The attacker model, however, is more difficult to define correctly. This is mainly because it includes the set of expected attackers and relevant protection goals that must be considered for each component, which depends on the concrete application scenario. Usually, system designers conduct an informal security analysis to determine this information. For our score to deliver accurate results, we must assume that this analysis has been conducted thoroughly and diligently enough for all attackers and protection goals to have been identified. Finally, we also need to accurately instantiate the trust model. This requires that the collected degrees of belief in the underlying trust statements (i.e., the functions t_o and t_m) adequately represent reality. Since we rely on user input and expert estimations to determine these probabilities, this is not necessarily guaranteed. Methods for determining accurate expert estimations in a scientific context have been extensively researched over many decades. Popular methods today include variants of the Delphi technique [Row01], cost estimation approaches [Gan14], and gamified methods such as planning poker [Gre02, Mol08]. Furthermore, expert estimations also play an important role in many probabilistic risk assessment methods [Bed01, pp. 191–217]. Nevertheless, the application of these methods to acquire accurate estimations for t_o and t_m is beyond the scope of this work.

Assuming that the formal model and the baseline trust estimations are correct, the last remaining question concerns the soundness of multiplying together the identified degrees of belief (cf. definition 6.8). While this is an adequate construction to receive a quantitative score that can inform data providers about problematic system states, for it to result in the actual probability of policy enforcement, we additionally need to assume that the multiplied degrees of belief are *independent*. Since usage control participants are usually self-reliant actors, we can reasonably expect the independence of statements about operators, i.e., the degrees of belief t_o. Furthermore, we avoid some obvious dependencies in our construction of the set of critical mechanism capabilities (see definition 6.7) by filtering out similar tuples that only differ in the expected attacker. Nevertheless, the estimations concerning the trusted computing mechanisms t_m generally cannot be considered independent. This is because different technologies rely on common cryptographic primitives, assumptions, and protocols. Similarly, the different protection goals for a single mechanism also have common influences and are not independent. Because of this our score, as the multiplication of these degrees of beliefs, generally does not equal the actual probability of policy enforcement.

To alleviate this issue and achieve a more accurate representation of the enforcement probability, we could extend our trust model by breaking down the collected baseline estimations to a finer granularity. For example, consider that both the TPM- and SGX-based attestation protocols rely on the RSA cryptosystem to generate digital signatures. Hence, the degrees of belief $t_m(\text{TPM}, I_{code}, \cdot)$ and $t_m(\text{SGX}, I_{code}, \cdot)$ both depend on the RSA assumption. For simplicity, we can describe these three events as T, S, and R. To deal with the detected dependency, instead of multiplying $P(T)$ and $P(S)$, we could also collect expert estimations about the likelihood that the RSA assumption holds true, i.e., $P(R)$, as well as the corresponding *conditional* probabilities for the mechanisms, i.e., $P(T \mid R)$ and $P(S \mid R)$. Assuming the conditional independence of T and S given R, the *joint*[1] probability of all three events then results to $P(T, S, R) = P(R) \cdot P(T \mid R) \cdot P(S \mid R)$ [Ste00]. Since this is not the only shared cryptographic primitive between SGX and TPMs, we then need to consider additional dependencies in the same manner, in order to get an accurate estimation for the actual enforcement probability. Bayesian networks [Ste00] provide a graph-based model to represent such conditionally (in)dependent events and calculate the joint probability over a subset of variables. However, since the considered trusted computing mechanisms are large and complicated systems with a multitude of different cryptographic protocols and primitives, breaking down our trust model to this level and modeling individual influences on the trusted computing mechanisms adds a high degree of complexity to the score construction, which in turn affects the applicability of the score. Furthermore, in addition to obvious dependencies such as commonly used cryptographic functions, there are also dependencies that are much more difficult to model appropriately, for example when assessing the different protection goals of a particular trusted computing platform. Because of these reasons, and since constructing our score on the level of trusted computing technologies already achieves our requirements (see section 6.1) and provides useful information for data providers (see our evaluation in section 7.3.3), we leave the further exploration of this approach for future work.

[1] Note that in this particular case the desired marginal probability $P(T, S)$ is equal to the joint probability $P(T, S, R)$, because the RSA assumption is a precondition, i.e., $P(T, S, \overline{R}) = 0$.

6.4 The DataSov Trust Dashboard

As final contribution in this chapter, we describe the integration of the developed trustworthiness score into the DataSov framework. For this, in section 6.4.1, we present our design of a *trust dashboard* that can provide feedback to data providers and policy issuers by visualizing the trustworthiness score for all deployed policies. In section 6.4.2 we then show how the user interface of our dashboard represents degrees of beliefs in a human-readable way. Finally, in section 6.4.3, we present the concrete trust estimations describing the capabilities of trusted computing mechanisms, which we use for our subsequent evaluation in the next chapter.

6.4.1 Dashboard Design

We integrate the calculation and display of our trustworthiness score into the DataSov framework by means of a web-based dashboard, which allows data owners to monitor the sharing of assets and their corresponding policies. The goal of our trust dashboard is to provide policy issuers with information about the adequacy of the current usage control system state to properly enforce their usage rules. We implemented the trust dashboard as a web application using Node.js together with the ReactJS ecosystem. The dashboard's front-end is integrated into the DataSov framework in the same way as the provenance dashboard presented in section 5.1.3. The back-end of the dashboard is realized in Java using the Spark[1] web-server.

In order to calculate our trustworthiness score, the dashboard needs access to four different types of information:

(i) The list of currently deployed policies as input for the function ops.

(ii) The current system state as defined by our formal component model $\mathcal{C}$.

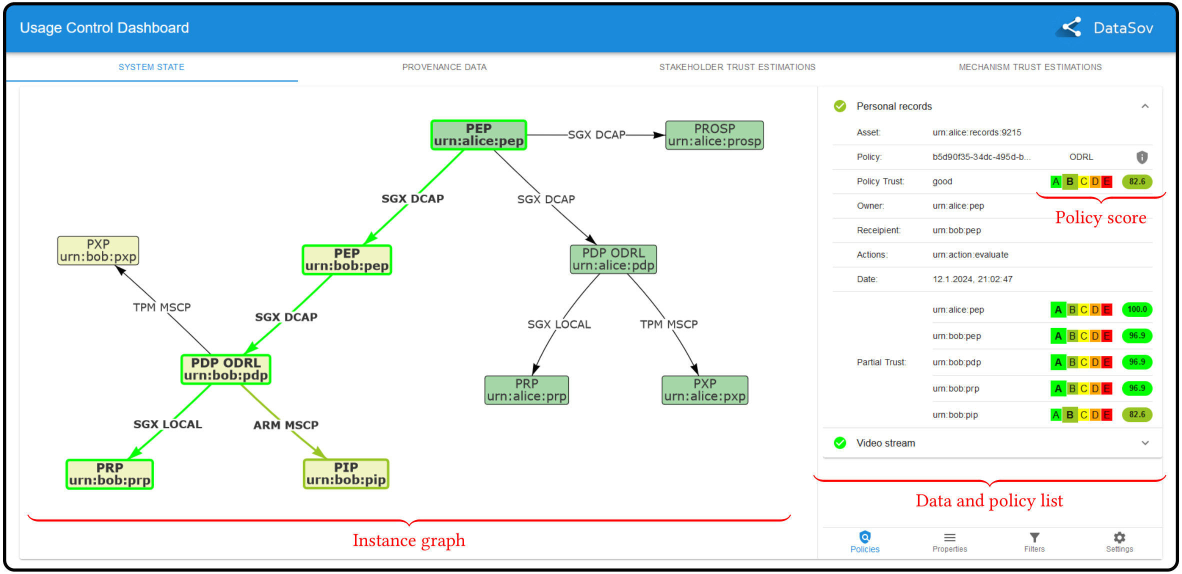
(iii) The formal attacker model $\mathcal{A}$, describing the expected attackers and protection goals of the components.

[1] https://sparkjava.com/ *(accessed on 12/08/2023).*

(iv) The trust model $\mathcal{T}$, which includes the baseline trust estimations about the component operators and trusted computing mechanisms.

The list of deployed policies and the current state of the usage control system is continuously retrieved by the dashboard's back-end server using the DataSov monitor components (see section 5.1.3). The received state updates include the currently active usage control components and their dependencies, as well as the conducted remote attestations. The back-end server then continuously integrates these state updates into our formal system model to determine a complete picture of the usage control system state. Since the attacker and trust models are subjective to the data owner, they are not automatically determined, but can instead be configured individually via the front-end. Based on this information, the back-end server then calculates the trustworthiness scores for all currently active policies as described in the previous section.

Figure 6.3 shows a screenshot of the DataSov trust dashboard. On the left-hand side a visualization of the current instance graph is located, which gives dashboard users an overview of the current system state. This graph displays all active usage control components and their trust dependencies. Note that the dashboard only shows *active* trust dependencies (cf. fig. 3.13), although both directions are being attested and verified. For clarity, the edges PEP → PRP are also omitted in the visualization. The labels that are shown on the edges identify the trusted computing mechanisms and protocols that have been used for the remote attestations. In case of timed-out attestations, the edges are shown in red instead of black. Furthermore, users can also select individual components as a data provider, and then inspect a list of shared data assets and their corresponding policies on the right-hand side of the screen. Near the top we show the calculated trustworthiness score $t(d, r, P, S) \in [0{,}1]$ for the currently selected policy/data deployment in the form of a percentage. For easier visual comprehension, the score is also mapped to a colored grading scale spanning **A** to **E**, which is inspired by the visualization of the European Nutri-Score [Her22]. Table 6.1 shows the definition of the colored grading scale as it is implemented in the DataSov trust dashboard. Note that the defined ranges are chosen as examples and are meant only for illustrative purposes.



Usage Control Dashboard

DataSov

SYSTEM STATE

PROVENANCE DATA

STAKEHOLDER TRUST ESTIMATIONS

MECHANISM TRUST ESTIMATIONS

Personal records

Asset:	urn:alice:records:9215		
Policy:	b5d90f35-34dc-495d-b...	ODRL	
Policy Trust:	good	A B C D E	82.6
Owner:	urn:alice:pep		
Receipient:	urn:bob:pep		
Actions:	urn:action:evaluate		
Date:	12.1.2024, 21:02:47		
	urn:alice:pep	A B C D E	100.0
	urn:bob:pep	A B C D E	96.9
Partial Trust:	urn:bob:pdp	A B C D E	96.9
	urn:bob:prp	A B C D E	96.9
	urn:bob:pip	A B C D E	82.6

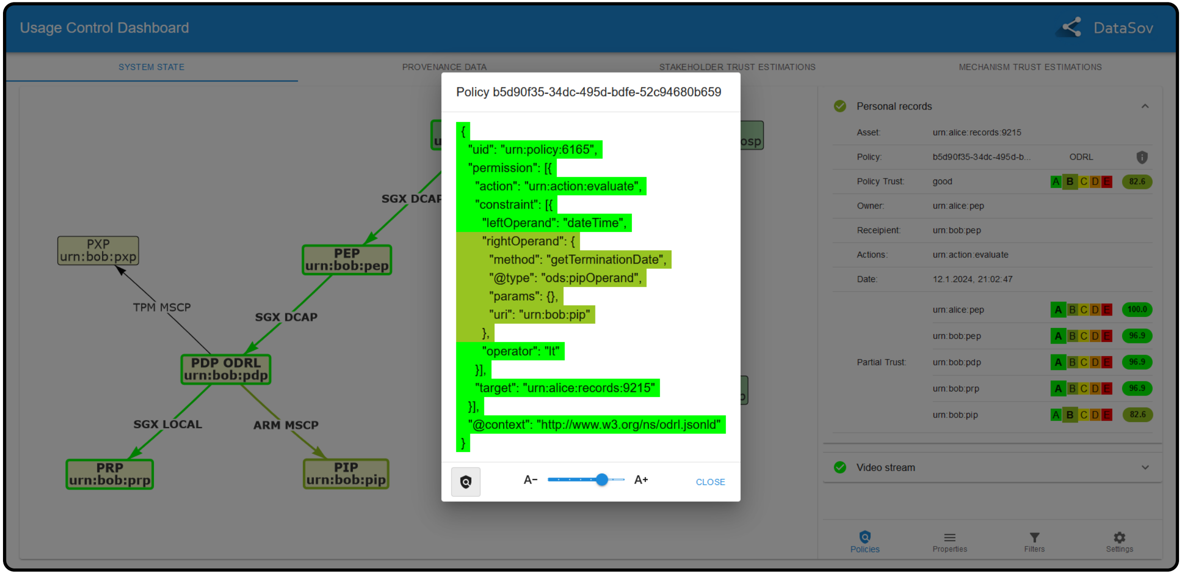
Video stream

Figure 6.3: Screenshot of the DataSov trust dashboard.

Table 6.1: Colored grading scale for the developed trustworthiness score.

Grade	**A**	**B**	**C**	**D**	**E**
Label	excellent	good	adequate	insufficient	poor
Range	over 90%	90% - 80%	80% - 60%	60% - 40%	under 40%
Color	#00FF00	#97C422	#FFFF00	#FFA500	#FF0000

Below the overall trust score for the policy, we also display additional information about the *trust propagation* of the policy. The idea behind this is to give users an intuitive understanding of where the assumptions that influence the trustworthiness score come from. To achieve this, in addition to the trust score for the complete usage control operation $J := ops(P, (d, r))$, we also calculate the scores for *partial* usage control operations. These partial operations are defined for each component $v \in J$ and are comprised of the path from the operation's root (i.e., the policy deployer d) to the respective component: $\{t(d, r, J|_{(d \to r \to \ldots \to v)}, S)\}_{v \in J}$. Showing this allows the user to go over the trust propagation step by step and identify at which usage control component the trustworthiness score deteriorates. For example, in the scenario illustrated in fig. 6.3, we can see that the reliance on the remote PIP noticeably decreases the trustworthiness score of the policy. This is because this component introduces ARM TrustZone as an additional trusted computing technology into the usage control operation, which expands the set of mechanism capabilities that the policy enforcement relies on. Removing the dependency to this PIP will hence decrease the attack surface for this policy. Furthermore, calculating the trust propagation in this manner also allows the DataSov dashboard to illustrate which parts of the deployed usage control policies likely can or cannot be enforced. This is achieved by overlaying the policy text with the partial trust scores of each usage control component that is required to enforce a specific policy section. Figure 6.4 shows a screenshot of the resulting policy visualization for the previously described scenario. Note that the highlighted section in the middle of the policy contains an ODRL operand definition referencing the ARM TrustZone PIP in question.



Policy b5d90f35-34dc-495d-bdfe-52c94680b659

```
{
 "uid": "urn:policy:6165",
 "permission": [{
  "action": "urn:action:evaluate",
  "constraint": [{
   "leftOperand": "dateTime",
   "rightOperand": {
    "method": "getTerminationDate",
    "@type": "ods:pipOperand",
    "params": {},
    "uri": "urn:bob:pip"
   },
   "operator": "lt"
  }],
  "target": "urn:alice:records:9215"
 }],
 "@context": "http://www.w3.org/ns/odrl.jsonld"
}
```

Personal records

Asset:	urn:alice:records:9215			
Policy:	b5d90f35-34dc-495d-b...	ODRL		
Policy Trust:	good		A B C D E	82.6
Owner:	urn:alice:pep			
Receipient:	urn:bob:pep			
Actions:	urn:action:evaluate			
Date:	12.1.2024, 21:02:47			
	urn:alice:pep		A B C D E	100.0
	urn:bob:pep		A B C D E	96.9
Partial Trust:	urn:bob:pdp		A B C D E	96.9
	urn:bob:prp		A B C D E	96.9
	urn:bob:pip		A B C D E	82.6

Video stream

Figure 6.4: Policy visualization in the DataSov trust dashboard.

6.4.2 Representing Degrees of Belief

As described in the previous section, our trust model consists of two functions: the operator trust estimation t_o and the mechanism trust estimation t_m. While the former estimation is specified by the usage control participants themselves, the latter must be decided by experts. In both cases, however, we need to represent the degrees of belief in a way that is comprehensible for humans. For this purpose, Ries and Schreiber introduced the *Human-Trust-Interface (HTI)* [Rie08b]. The HTI visually represents degrees of belief on a two-dimensional scale, as is shown in fig. 6.5. The tuple $(t, c) \in [0,1]^2$ is also called an *opinion* and is comprised of a trust value t together with a certainty value c. While the value t represents the (subjective) level of trust in the veracity of a claim, the certainty c denotes the subject's confidence in that trust estimation. Representing degrees of belief in such a way has the advantage of making the uncertainty that is associated with the trust estimation explicit. This facilitates the expression of trust statements that are based on evidence (high certainty value, e.g., o_1) in contrast to statements that are based on conjecture (low certainty value, e.g., o_2). Furthermore, the authors conducted a user study which found this representation to be intuitively interpretable [Rie08b]. For these reasons, we adopt the HTI as a method for visually representing degrees of belief in the graphical user interface of our trust dashboard.

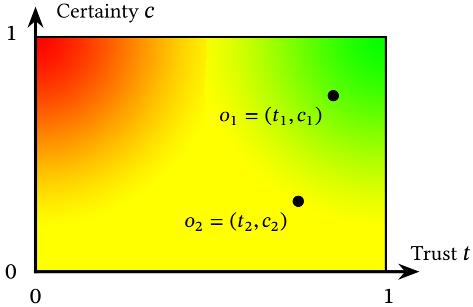


Figure 6.5: Human-readable representation of degrees of belief after Ries and Schreiber. Illustration modified from [Rie08b].

Even though this representation of degrees of belief is useful for visualization purposes, we still need to retrieve values between 0 and 1 in order to calculate our trustworthiness score. For this, the underlying mathematical model of HTI offers the function $\mathbb{E} : [0,1]^2 \rightarrow [0,1]$, which maps the two-dimensional opinion space back to standard probabilities in the Bayesian sense [Rie09]. This function is defined as the trust value weighted by the confidence and an initial expectation value f, i.e., $\mathbb{E}(t, c) \coloneqq c \cdot t + (1 - c) \cdot f$. The initial expectation $f \in [0,1]$ can be seen as a default value that is chosen if the level of trust cannot be confidently estimated. Ries and Heinemann propose to choose f depending on the application scenario and the subject's general attitude, for example pessimistic ($f \approx 0$), neutral ($f \approx 0.5$), or optimistic ($f \approx 1$) [Rie08a]. In addition, it is also possible to directly calculate with opinions in the same way as with probabilities. For this, Ries introduces the logical operators AND, OR, and NOT in the opinion space [Rie11]. These operators are defined to be compliant with standard probabilistic evaluations of propositional terms, e.g., it holds that $\mathbb{E}(o_1 \wedge o_2) = \mathbb{E}(o_1) \cdot \mathbb{E}(o_2)$. Because of this property, we can also use the logical operator AND as defined in [Rie11] to calculate our trustworthiness score directly on the collected baseline trust estimations.

6.4.3 Baseline Trust Estimations

Finally, in preparation of our dashboard evaluation in section 7.3, we need to define concrete values for the baseline trust estimations t_o and t_m. As mentioned before, the operator trust estimation t_o is subjective and depends on the concrete application. Hence, we will define it later specifically for the example scenario presented in section 7.3. The mechanism trust estimation function t_m, on the other hand, describes the capabilities of the deployed trusted computing mechanisms to ensure the necessary protection goals against the expected attackers. As such, we assume this function to be determined by technology experts. It must also be continuously updated to reflect novel attack vectors and mitigations. For the purposes of evaluating our trust dashboard, we define an exemplary estimation describing the capabilities of the three technologies supported by the DataSov framework, i.e., TPMs, Intel SGX, and ARM TrustZone. However, it should be pointed out that this estimation is meant for illustrative purposes only, and we do not claim its completeness. Acquiring reliable estimates for the function t_m would require interviewing multiple independent technology experts, which is beyond the scope of this work. Furthermore, we base our exemplary estimation only on publicly available technical descriptions and specifications, as well as on published attacks that are discussed in the scientific literature. As part of our analysis in chapter 4, we have already pointed out the different capabilities and limitations of the three considered trusted computing technologies with regard to our application of distributed usage control. In the remainder of this section, we give a brief summary of the notable technological differences and motivate our exemplary definition of the trust estimation function t_m.

Trusted Platform Modules. As presented earlier in section 4.2.1, TPMs can protect the confidentiality and integrity of data at rest (i.e., the protection goals C_{store} and I_{store}) by means of cryptographic sealing. The security of the sealing operations is determined by the capability of the TPM to shield the associated private keys from attackers. While the TPM takes care to always retain generated sealing keys on the trusted hardware, in the past this hardware isolation has been successfully circumvented by side-channel attacks. Recently, Moghimi et al. published a timing-based attack that is capable of extracting secret key material from the TPM [Mog20]. However, this attack vector has since been mitigated by the TPM manufacturers with firmware updates for the affected devices [Mog19]. As a result, for our evaluation we presume TPMs to be capable of achieving the protection goals C_{store} and I_{store} against both network and software attackers. In terms of the trust model, this is reflected by high trust and confidence values for $t_m(\text{TPM}, \{C_{store}, I_{store}\}, \{\mathit{network}, \mathit{software}\})$. Privileged attackers, on the other hand, have physical access to the system hardware and hence could intercept critical information directly on the TPM bus [And19, Dew21]. Because of this additional attack vector, we must assume that privileged adversaries can generally bypass TPM-based sealing, albeit with the hurdle of having to mount a dedicated physical attack on the trusted hardware. We reflect this in the trust model by assigning a much lower trust value for privileged adversaries than for network and software attackers. Furthermore, TPMs do not provide isolated execution environments for user applications. As a result, TPM-protected systems can safeguard the confidentiality and integrity of data during processing (protection goals C_{proc} and I_{proc}) only in the sense that we can exclude malicious software being executed on the trusted platform. While this prevents software-based attacks on these two protection goals, we must still assume that processed data will not get intercepted via hardware attacks, e.g., on the unencrypted memory bus. Consequently, we also assign a lower trust rating to these protection goals in our trust model.

In terms of remotely attesting TPM-based systems, our framework relies on the MSCP protocol as described in section 4.2.4. Our security analysis shows that the MSCP protocol can ensure confidentiality and integrity during data transmission (protection goals C_{tra} and I_{tra}) even against privileged attackers. Furthermore, the attestation evidence transmitted during the protocol handshake also protects code integrity (protection goal I_{code}) against all adversaries. Regarding side-channel attacks against code integrity, Han et al. identified a power management flaw in the TPM specification that allows to forge PCR values [Han18]. However, the authors have already submitted suitable firmware patches that effectively mitigate this attack [Han18]. Because of this, we classify all three protection goals concerning remote attestation as fulfilled in our trust estimation.

Finally, TPMs also offer some limited amount of non-volatile (NV) memory, which could be used to protect critical information against malicious deletion (protection goal I_{del}). However, due to the technical limitations of NV memory (see section 4.2.1), usually only cryptographic keys are kept inside the TPM, while encrypted data blobs are outsourced to the file system. Furthermore, storing information directly in the NV memory still cannot protect against malicious deletion by privileged attackers, since they have full access to the system and can reset or physically destroy the TPM. Because of this, for our evaluation we assume that the TPM cannot adequately protect against deletion attacks by software and privileged attackers (i.e., low trust values with high confidence). Based on the presented analysis and arguments, we set the concrete values for the trust estimation function $t_m(\text{TPM}, \mathbf{g}, \mathbf{a})$ as shown in table 6.2.

Table 6.2: Mechanism trust estimations $t_m(\text{TPM}, \mathbf{g}, \mathbf{a})$.

$(t, c) \times 10^{-3}$	**Attacker a**					
	network		*software*		*privileged*	
Goal g	t	c	t	c	t	c
C_{store}, I_{store}	990	900	980	850	450	800
C_{proc}, I_{proc}	800	950	600	900	550	900
C_{tra}, I_{tra}	995	950	990	950	980	900
I_{code}	998	990	995	950	990	900
I_{del}	700	800	300	850	150	850

Software Guard Extensions. Similar to TPMs, Intel SGX processors also provide cryptographic sealing functions that can protect the confidentiality and integrity of outsourced data (see section 4.3.1). In this regard, however, the SGX technology has the advantage that it does not rely on any external hardware modules to perform the sealing. Instead, the critical data never leave the protected CPU unencrypted, which effectively prevents bus snooping attacks by privileged adversaries. To reflect this in our trust model, we classify the capability of SGX to protect the goals C_{store} and I_{store} against privileged attackers with a higher trust value than for TPMs. Furthermore, in contrast to TPMs, the SGX processor allows to run user code inside a completely isolated Trusted Execution Environment (TEE). Because of this, in addition to data at rest, SGX can also cryptographically ensure the confidentiality and integrity of data in use (protection goals C_{proc} and I_{proc}) without any additional assumptions about the host system. However, over the recent years several attacks have been published that aim to break the isolation of executed SGX enclaves. As indicated earlier, attaining expert assessments about this by conducting a comprehensive and independent peer review of published attacks on the SGX technology is beyond the scope of this work. Instead, we give a brief summary of the identified attack classes and assess their impact based on two surveys by Nilsson et al. [Nil20] and Fei et al. [Fei21].

One of the most researched classes of attacks on SGX enclave isolation are *cache attacks* [Mog17, Göt17, Sch17, Bra17b]. The principal idea behind these attacks is to deduce enclave memory contents by priming the processor cache with known data, and then observing the execution times of applications.

Since cache attacks aim to recover enclave memory in plain text, they threaten the protection goal of data confidentiality (protection goal C_{proc}) rather than integrity. However, these attacks usually require the execution of attacker-controlled software on the SGX processor, which limits the expected adversaries to software and/or privileged attackers. A related class of attacks are *memory attacks*, which also target the protection goal C_{proc}. Even though they apply a similar concept as cache attacks, memory attacks work on the level of memory management instead of caches. For example, an attacker can manipulate the (unencrypted) page tables that are managed by the untrusted operating system, in order to recover enclave memory by observing the resulting page faults [Xu15]. There are also more advanced attack vectors on SGX, which do not even require triggering page faults [Van17b, Van17a, Wan17, Kim19]. Nevertheless, since the attacker still needs access to the page tables, the attacker models for these attacks usually assume privileged adversaries such as malicious system owners. Furthermore, SGX processors have also been identified as being vulnerable to various types of *branch prediction attacks* [Lee17b, Evt18, Van18, Kor18, Che19a, Huo20]. These attacks exploit the speculative branch prediction features that most modern processors employ for efficiency. By carefully measuring execution timings, untrusted applications can obtain information about the control flow of enclaves and ultimately deduce enclave memory contents. Hence, these attacks also break the confidentiality of data in use (protection goal C_{proc}) under the assumption that the attacker can execute malicious software on the target machine.

Like TPMs, SGX processors can also be attacked via *side-channel attacks* on the physical hardware. For example, it has been demonstrated that measuring the power consumption of cryptographic operations can leak sensitive key material [Lip21]. While this attack once again only targets data confidentiality (protection goal C_{proc}), there are also side-channel attacks that can break the integrity of processed data (protection goal I_{proc}). This can be achieved by manipulating the processor voltages in order to inject faults into the enclave execution [Qiu19, Mur20], or by provoking random bit flips in the system memory [Jan17]. Finally, purely *software-based attacks* are also possible on SGX enclaves. Most importantly, return-oriented programming has been used to inject malicious operations into enclaves [Lee17a, Yoo22], which breaks both confidentiality and integrity of data in use. However, these attacks require a privileged attacker that is in complete control of the untrusted operating system [Lee17a, Yoo22]. In recent years, there has also been extensive research into viable countermeasures against the identified attacks on SGX hardware. Effective mitigations have been developed against cache attacks [Gru17, Yav22, Cai23], memory attacks [Shi17, Che17, Ahm19, Bra19, Lan22, Con23], branch prediction vulnerabilities [Wei18, Int21, Jin21], as well as software- and hardware-based attacks [Seo17, Kog22]. Because of this, for the evaluation of our trust dashboard, we rate SGX as being generally capable of ensuring the confidentiality and integrity of data in use. However, due to the presented attack vectors on SGX processors, we assign higher trust and certainty values against network attackers than against software and privileged attackers. Furthermore, we assume that the deployed SGX hardware is fully patched and works as specified.

Our framework relies on the Enclave Key Exchange Protocol (EKEP) for the SGX-based attestation of remote system components (see section 4.3.2). Even though Dall et al. identified an attack vector against the implementation of the EPID attestation scheme for SGX enclaves [Dal18], this is not relevant for our system, because EKEP uses the DCAP primitives instead of EPID [Asy21c]. Since the EKEP attestation protocol has been formally verified by Roeder et al. [Roe22], we can associate high trust values with the protection goals C_{tra}, I_{tra}, and I_{code} against all attackers. However, we still need to assume that the quoting enclave used by EKEP in our proof of concept, which is provided by the Asylo framework, is correctly implemented. Finally, SGX does not provide any capabilities regarding the protection of critical data against deletion attacks by software and privileged adversaries. Hence, we must classify the protection goal I_{del} as not fulfilled (i.e., low trust values with high certainty). The concrete values that we choose for the trust estimation function $t_m(\text{SGX}, \mathbf{g}, \mathbf{a})$ are given in table 6.3.

Table 6.3: Mechanism trust estimations $t_m(\text{SGX}, \mathbf{g}, \mathbf{a})$.

$(t, c) \times 10^{-3}$	**Attacker a**					
	network		*software*		*privileged*	
Goal g	t	c	t	c	t	c
C_{store}, I_{store}	997	950	995	900	993	850
C_{proc}, I_{proc}	995	950	992	900	987	850
C_{tra}, I_{tra}	997	980	995	950	993	900
I_{code}	998	950	995	900	990	850
I_{del}	600	800	150	850	050	850

ARM TrustZone. Since both technologies are designed as Trusted Execution Environments, the capabilities of ARM TrustZone concerning the protection of critical data during processing are largely comparable to those of Intel SGX. However, as discussed in section 4.4.1, Trusted Applications running on ARM TrustZone devices have a larger Trusted Computing Base than SGX enclaves, because they also rely on the trusted firmware and the secure world operating system. As a result, we have to assume the correctness of a much larger set of software, which leads to a larger attack surface under TrustZone. To reflect this in our trust model, we choose slightly lower estimates for the protection goals C_{proc} and I_{proc} compared to Intel SGX.

In terms of cryptographic sealing (protection goals C_{store} and I_{store}) and remote attestation (protection goals C_{tra}, I_{tra}, and I_{code}), ARM TrustZone unfortunately does not provide the same level of functionality as Intel SGX. Because of this, we rely on the fTPM module [Raj16] running as a Trusted Application inside the secure world to achieve these protection goals (see sections 4.4.3 and 4.4.4). One advantage of this approach is that we can apply the same methods as previously used for hardware TPMs, such as relying on TPM-managed sealing keys for secure storage and our MSCP protocol for remote attestation. As a result, we can select trust values for these protection goals that are generally comparable to those previously determined for TPMs. In addition, having a firmware-level TPM also prevents bus snooping attacks, and hence even increases the security of sealing operations against privileged attackers compared to hardware TPMs (protection goals C_{store} and I_{store}). On the other hand, since we use a customized trusted boot process that relies on a larger CRTM than when using hardware TPMs (see section 4.4.3), we must rate the protection goal I_{code} with a lower trust value than before. Furthermore, as a software module, fTPM cannot provide the same level of immutability as a discrete TPM chip. Recently, Jacob et al. found several software-based attacks on the fTPM implementation for the AMD SEV processor architecture [Jac23]. Even though these attacks do not target the ARM TrustZone architecture directly, it still proves that software-based attacks on firmware-level TPMs are a concern to be considered. We reflect this uncertainty in our trust model by reducing the confidence values of the protection goals C_{tra}, I_{tra}, and I_{code} compared to both other technologies.

Finally, like TPMs and Intel SGX, ARM TrustZone also does not provide any effective protection mechanisms to secure critical data against malicious deletion by software and privileged attackers (protection goal I_{del}). Based on these considerations, table 6.4 shows the resulting trust and certainty values for the trust estimation function $t_m(\mathrm{TZ}, \mathbf{g}, \mathbf{a})$.

Table 6.4: Mechanism trust estimations $t_m(\mathrm{TZ}, \mathbf{g}, \mathbf{a})$.

$(t,c) \times 10^{-3}$	**Attacker a**					
	network		*software*		*privileged*	
Goal g	t	c	t	c	t	c
C_{store}, I_{store}	990	850	980	750	850	750
C_{proc}, I_{proc}	990	900	980	850	970	800
C_{tra}, I_{tra}	995	900	990	850	980	800
I_{code}	985	850	980	830	970	800
I_{del}	700	750	300	750	100	750

As mentioned earlier, conducting an in-depth security assessment that consists of a peer-reviewed process collecting and aggregating the opinions of multiple independent technology experts is beyond the scope of this work. Hence, the estimated trust values given in tables 6.2 to 6.4 cannot necessarily be considered reliable. Nevertheless, they are still founded in an analysis of the current scientific literature and as such are useful to evaluate the proof of concept of our proposal, as presented in the next chapter.

6.5 Conclusion

In this chapter we presented our research contribution RC7 by developing a trustworthiness score for distributed usage control systems. Our score is designed to represent the adequacy of the current system state to reliably protect the enforcement of a particular usage control policy from a data provider's point of view. To achieve this, we rely on expert estimations regarding the capabilities of trusted computing technologies to ensure the necessary protection goals against the expected attackers. We also consider the degree of trust that data providers might initially show towards other usage control system participants. Our score is constructed using a graph-based formal model that can represent the currently deployed system components, their attacker and trust model, as well as the conducted remote attestations. We validated the definition of our score by showing that it fulfills the previously set requirements, which includes operator and technological propriety, minimality, monotony, and language agnosticism. To provide policy issuers with qualified feedback regarding the adequacy of the current system state with respect to their usage rules, we also integrated the developed score into a web-based trust dashboard that is part of our trustworthy usage control framework. The developed trust dashboard illustrates the current usage control system state, as well as the individual policy scores using a color-coded visual scale. To conclude, the research contributions presented in this chapter add transparency to our distributed usage control and provenance tracking framework, and hence achieve goal 4 of the thesis objective. In the following chapter we validate the effectiveness of the developed trustworthiness score by means of a concrete application scenario as well.

Finally, this chapter also provides several starting points for future research. Since the expressiveness of our score relies mainly on the quality of the underlying expert estimations, it must be ensured that they are collected from a sufficient number of knowledgeable individuals using an adequately implemented process. Hubbard and Seiersen present several suggestions for setting up processes to acquire accurate expert estimations [Hub16, pp. 133–155]. In addition, the design choice to represent degrees of belief with tuples consisting of both trust and certainty values in our dashboard could prove to be advantageous for this process as well, because it provides a simple way to distinguish varying levels of expertise [Rie07]. A related question is if we can leverage real-world data in form of *observations*, in addition to mere (subjective) estimations, to get more accurate results. As motivated earlier, the direct observation of component responses is unfeasible in our case, since we cannot reliably decide if a particular response of a usage control component is malicious or not. However, other possible sources of observations include statistical data about successfully executed attacks on trusted computing platforms. So far we see this kind of knowledge as being implicitly represented by the opinions of well-informed experts, but it may be beneficial to also consider it explicitly in the construction of the score, provided that suitable statistical data are available. Furthermore, our attacker model could be augmented to represent more detailed attributes of adversaries, such as motivation, opportunity, and individual attacker skills. Since these properties are usually not binary in nature, they could be integrated into the proposed model in the form of degrees of belief as well. Finally, the granularity of the collected security estimations used in the construction of our score could be improved by modeling more detailed security properties, such as cryptographic assumptions and individual algorithms. While this would reduce the impact that dependent properties have on the accuracy of the score (see section 6.3.4), it also increases the complexity of the construction.

7 Evaluation and Results

To demonstrate the practicality of our trustworthy usage control and provenance tracking system in a real-world scenario, we conduct a brief evaluation using an exemplary application from the realm of smart manufacturing. In this chapter, we first introduce our concrete evaluation scenario and present the individual usage control policies that are enforced across two domains. Based on this application, we then evaluate the performance of the initial component provisioning, as well as the distributed policy enforcement and provenance tracking mechanisms. Finally we show that our trust dashboard, as presented in the last chapter, is capable of assisting policy issuers with the identification of problematic usage control system states in this scenario.

7.1 Example Scenario: Smart Manufacturing

Smart manufacturing is an umbrella term that summarizes many recent efforts of leveraging the potential of big data and collaborative data processing in production systems [Win21]. One aspect of smart manufacturing, which has been researched over the past few years, concerns the advantages of integrating assistance functions into assembly tables [Tsu20, Len20]. Such intelligent workstations can provide personalized assistance to human workers, which helps with the execution of difficult work steps and increases both the workers' wellbeing and their productivity. Furthermore, smart assembly stations facilitate the automated and continuous collection of data about work processes, as well as the assembled goods themselves, such as work step protocols, sensor data, and even video recordings [Wag21b]. This information can then be harnessed locally for the optimization of manufacturing processes [Man19], but may also be shared with the recipients of assembled parts further down in the supply chain [Won17]. However, collecting and sharing information in a smart manufacturing context entails many data security and privacy concerns, which must be adequately addressed [Man19, Wag21b]. As Birnstill and Beyerer pointed out, usage control is a viable building block to mitigate many of the resulting problems in smart manufacturing use cases [Bir18a]. Because of these findings, we choose our evaluation scenario to consist of a smart assembly table sharing potentially sensitive information with different services, both locally as well as with remote parties such as customers. The goal of this evaluation is to demonstrate that our developed trustworthy usage control framework is capable of performing efficiently in a smart manufacturing context.

7.1.1 Scenario Overview

Figure 7.1 illustrates the exemplary scenario from the realm of smart manufacturing, which we use to evaluate our distributed usage control and provenance tracking framework. For the sake of simplicity, we assume the existence of just two stakeholders: a *producer* of goods and an associated *customer*, who acquires and further processes these commodities. The producer operates a smart *assembly table*, which is capable of continuously collecting process data about the manufactured parts. The resulting *task logs* for each assembled part are the main assets in this scenario. Among other data, the task logs contain automatically generated work step protocols and sensor data captured during the manufacturing process. They may also include video recordings of the conducted work steps, which are taken for documentation and quality management purposes [Wag21b]. The collected task logs are shared with two different services for evaluation purposes. First, the producer operates an *accounting service* in the same application domain as the assembly table. This service evaluates the collected task logs in order to extract statistical information about the work processes, for example the employees' working hours. In addition, the information collected by the assembly table is also of interest to the customer further down in the supply chain. For this, the customer operates an *optimization* service in their own infrastructure, which utilizes the shared task logs to fine tune any subsequent manufacturing processes. However, in order to protect the privacy of the involved workers, the information contained in the task logs must be anonymized before any cross-domain sharing can be allowed.

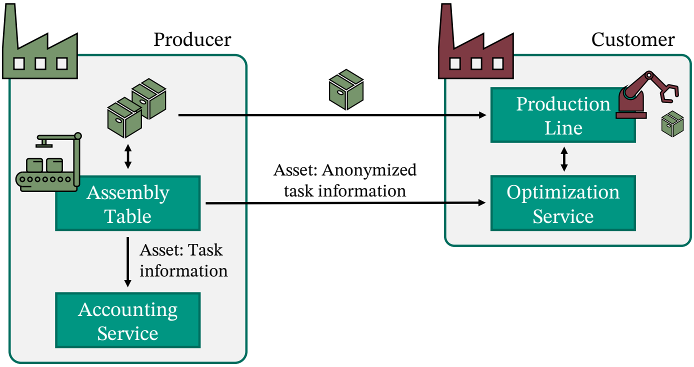


Figure 7.1: Overview of the evaluation scenario.

In this scenario, we leverage our trustworthy usage control framework to restrict the evaluation of the shared data assets to the described use cases. Most importantly, the remote customer must be prevented from using the shared task logs for any other than the advertised purposes. Furthermore, since the assembly table potentially captures personal information about human workers, we also demand that the provenance of the data transmissions and usages must be reliably tracked.

7.1.2 System Deployment

To securely enforce these usage restrictions by technical means, both stakeholders leverage our DataSov distributed usage control framework as presented in chapter 5. Figure 7.2 shows the concrete system deployment of the necessary usage control components, as well as the three data processing applications that we use for our performance evaluation. Note that all three applications integrate a DataSov Policy Enforcement Point (PEP) into their program logic, which intercepts relevant data usage events and enforces the resulting decisions. Furthermore, both the producer and the customer operate dedicated instances of retrieval points (PRPs) and decision points (PDPs) in their respective domains. The producer also operates a Policy Execution Point (PXP) and a Provenance Storage Point (ProSP), which in this scenario is responsible for collecting and retaining the provenance information of shared data assets. Furthermore, the customer operates a Policy Information Point (PIP) providing additional attributes during the enforcement process. To reflect the two distinct usage control domains, we identify each component via a unique URI that is prefixed by `urn:producer` or `urn:customer`, respectively.

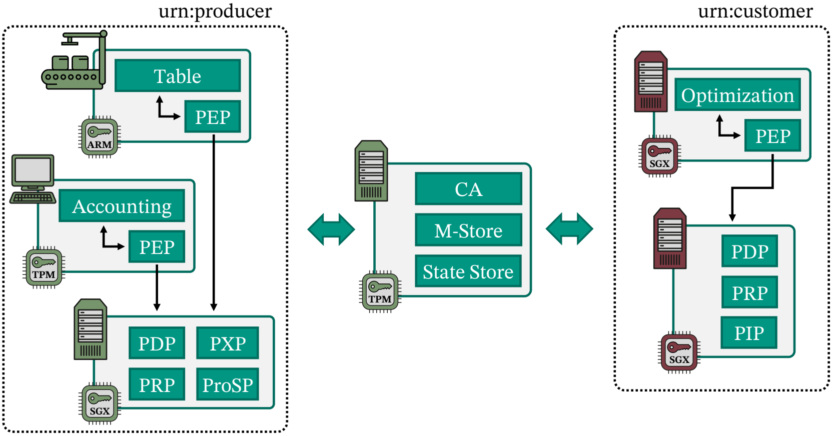


Figure 7.2: Deployment of the systems and usage control components used in the evaluation.

As shown in chapter 5, the DataSov framework relies on our previously developed heterogeneous remote attestation protocol to secure this distributed usage control infrastructure against malicious tampering and component removal. In order to design a representative evaluation scenario, we leverage all three trusted computing technologies that are supported by DataSov. Since assembly tables are often realized as ARM-based embedded systems, we use an fTPM instance executed on a TrustZone device to measure the platform integrity of the assembly table. The accounting service is instead deployed on a server that is protected by a hardware TPM. The customer's optimization service, as well as all individual usage control components, are implemented as separate Intel SGX enclaves. Finally, each DataSov usage control component also requires connection to a Certification Authority (CA) for the initial certificate provisioning, as well as to measurement and state stores for the necessary code and state integrity checks. For the sake of simplicity, and since it does not influence performance, we deploy only one instance of these components on an additional TPM-protected server. In reality, both stakeholders would operate individual CA servers and measurement stores in their own domains.

Note that the resulting system deployment as shown in fig. 7.2 represents an exemplary scenario, intended to showcase and evaluate the application of our distributed usage control framework. To ensure that the performance evaluation of the trustworthy usage control enforcement is not affected by one of the domain-specific applications, for our tests we only *simulate* the concrete application functionalities, i.e., the assistance functions in the assembly table, the accounting service, and the process optimization at the customer. Furthermore, we oversimplify some of the implementation details that do not concern the usage control enforcement process itself. For example, in a production environment the data sharing would not be launched directly from the assembly tables, but instead be managed by an intermediate distribution service.

7.1.3 Usage Control Policies

Finally, we define the concrete usage rules and corresponding ODRL policies that should be enforced by the DataSov framework in our evaluation scenario. First, the assembly table must be restricted to distribute the acquired task information exclusively to the producer's accounting service and the customer's optimization service. Furthermore, all task logs must be anonymized before they may be shared with the customer's domain. Listing 7.1 shows the ODRL policy that enforces these usage restrictions on a specific asset. This policy consists of two permissions for the action `distribute`, each with a constraint on the recipient's URI. The DataSov framework takes care to authenticate all asset recipients based on the permitted URIs via the provisioned component certificates (see section 5.2.2). In addition, the rule that permits the asset distribution to the customer also includes a duty to anonymize the task log by redacting any worker names prior to transmission (lines 19-24). This obligation is defined as a PEP action using the modifyParam method, which is automatically executed by the assembly table's enforcement point before allowing the data to be distributed to the optimization service.

Listing 7.1: DataSov ODRL policy enforced at the assembly table.

```
1  {"@context": "https://gitlab.cc-asp.fraunhofer.de/datasov/core/.../ods.jsonld",
2   "profile": "https://gitlab.cc-asp.fraunhofer.de/datasov/core#ods-ttl",
3   "permission": [
4      {"target": "urn:producer:asset:task:1587",
5       "action": "urn:producer:action:table:distribute",
6       "constraint": [{
7           "leftOperand": {"@type": "ods:ctxOperand", "key": "recipient_uri"},
8           "operator": "eq",
9           "rightOperand": "urn:producer:service:accounting"
10      }]
11     },
12     {"target": "urn:producer:asset:task:1587",
13      "action": "urn:producer:action:table:distribute",
14      "constraint": [{
15          "leftOperand": {"@type": "ods:ctxOperand", "key": "recipient_uri"},
16          "operator": "eq",
17          "rightOperand": "urn:customer:service:optimization"
18      }],
19      "duty": [{
20          "action": {
21              "@type": "ods:pepAction", "method": "modifyParam",
22              "params": {"key": "employee_name", "mask": "[REDACTED]"}
23          }
24      }]
25  }]
26 }
```

Since the producer's accounting service processes data that potentially include personal information about workers, it must be ensured that any evaluation of task information is logged as part of the asset's provenance graph. Listing 7.2 shows the ODRL policy that achieves this requirement. This policy permits the action evaluateTask on the asset under the precondition that the producer's PXP successfully executes the provenance method. The associated provenance object is assembled from the event context and defines the evaluating process, identified by its application URI, as a new activity in the PROV model (lines 12-18). It then registers a new `prov:used` relation between the process and the evaluated asset. Attaching this policy to the assets automatically enforces provenance tracking at the accounting service.

Listing 7.2: DataSov ODRL policy enforced at the producer's accounting service.

```
1  {"@context": "https://gitlab.cc-asp.fraunhofer.de/datasov/core/.../ods.jsonld",
2   "profile": "https://gitlab.cc-asp.fraunhofer.de/datasov/core#ods-ttl",
3   "permission": [
4      {"target": "urn:producer:asset:task:1587",
5       "action": "urn:producer:action:accounting:evaluateTask",
6       "duty": [{
7         "action": {
8           "@type": "ods:pxpAction",
9           "uri": "urn:producer:service:pxp", "method": "provenance",
10          "params" : {
11            "prosps": ["urn:producer:service:prosp"],
12            "provenance": {
13              "@type": "ods:provenance",
14              "activities": [{"@type": "ods:ctxOperand", "key": "app_uri"}],
15              "relations": [
16                [{"@type": "ods:ctxOperand", "key": "app_uri"},
17                  "prov:used", "urn:producer:asset:task:1587"]]
18            }
19          }
20        }
21      }]
22   }]
23 }
```

Finally, listing 7.3 shows the ODRL policy restricting the usage of assets that are shared with the customer's optimization service. While this policy unconditionally permits the evaluation of shared sensor data (lines 4-6), the action `evaluateWorksteps` may only be executed if the producer is explicitly notified about it using the `log` method of the producer's PXP (lines 7-16). Furthermore, the policy prohibits the evaluation of video images unless the depicted workers consented (lines 19-26). To achieve this, the policy uses a PIP operand with the method `hasEmployeeConsented`. Since this method is not part of the DataSov ODRL profile (cf. appendix C), the customer's PIP must explicitly implement it. Because the video images may contain personal information about the workers, the policy also enforces provenance tracking at the producer's ProSP before the data evaluation is permitted (lines 27-42).

Listing 7.3: DataSov ODRL policy enforced at the customer's optimization service.

```
1  {"@context": "https://gitlab.cc-asp.fraunhofer.de/datasov/core/.../ods.jsonld",
2   "profile": "https://gitlab.cc-asp.fraunhofer.de/datasov/core#ods-ttl",
3   "permission": [
4      {"target": "urn:producer:asset:task:1587",
5       "action": "urn:customer:action:optimization:evaluateSensorData"
6      },
7      {"target": "urn:producer:asset:task:1587",
8       "action": "urn:customer:action:optimization:evaluateWorksteps",
9       "duty": [{
10        "action": {
11          "@type": "ods:pxpAction",
12          "uri": "urn:producer:service:pxp", "method": "log",
13          "params": { "message": "Task 1587 is being evaluated by the customer!" }
14        }
15      }]
16     },
17     {"target": "urn:producer:asset:task:1587",
18      "action": "urn:customer:action:optimization:evaluateVideo",
19      "constraint": [{
20        "leftOperand": {
21          "@type": "ods:pipOperand",
22          "uri": "urn:customer:service:pip", "method": "hasEmployeeConsented",
23          "params": {"employee_id": {"@type": "ods:ctxOperand", "key": "employee_id"}}
24        },
25        "operator": "eq", "rightOperand": "true"
26      }],
27      "duty": [{
28        "action": {
29          "@type": "ods:pxpAction",
30          "uri": "urn:producer:service:pxp", "method": "provenance",
31          "params" : {
32            "prosps": ["urn:producer:service:prosp"],
33            "provenance": {
34              "@type": "ods:provenance",
35              "activities": [{"@type": "ods:ctxOperand", "key": "app_uri"}],
36              "relations": [
37                [{"@type": "ods:ctxOperand", "key": "app_uri"},
38                  "prov:used", "urn:producer:asset:task:1587"]]
39            }
40          }
41        }
42      }]
43   }]
44 }
```

Note that the policies described in listings 7.1 to 7.3 are not statically deployed to the respective application services. Instead, these policies are dynamically created for each individual task log as soon as it is generated by the assembly table. Following the sticky policy concept, the DataSov framework then attaches them to the task logs and transmits the resulting policy set to all services with which the assets are being shared (see section 3.2.3). This allows the implementation of flexible use cases where many different types of data assets, all with individual usage rules, should be distributed.

7.2 Performance Evaluation

To evaluate the performance of our trustworthy usage control framework in this scenario, we set up a testbed according to the system deployment shown in fig. 7.2. For the sake of comparability, we used the same trusted computing platforms as for the evaluation of our heterogeneous remote attestation protocol in section 4.5.4. Hence, we implemented the assembly table functionalities on an STM32MP157C-DK2[1] TrustZone prototyping board, using the MSCP protocol together with the normal world fTPM service[2] for remote attestation. The accounting service has been executed on a Thinkpad T480s (i7-8550U processor, 16GB RAM, Ubuntu 22.04) using the Infineon SLB9670[3] hardware TPM as underlying trusted computing technology. The optimization service, as well as all usage control components, have been deployed as SGX enclaves on an HP 250 G8 (i7-1065G7 processor, 8GB RAM, Ubuntu 22.04). The enclaves are executed on the SGX processor in release mode. Finally, all participating system components have been connected to a single network via Gigabit Ethernet. In the remainder of this section, we present the results of our performance evaluation concerning component provisioning, as well as the deployment and enforcement of usage control policies.

[1] https://www.st.com/resource/en/data_brief/stm32mp157c-dk2.pdf *(accessed on 12/21/2023).*

[2] https://github.com/microsoft/ms-tpm-20-ref/tree/main/TPMCmd/Simulator *(accessed on 12/25/2023).*

[3] https://www.infineon.com/cms/de/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/slb-9670vq2.0/ *(accessed on 12/08/2023).*

7.2.1 Component Provisioning

The process of provisioning DataSov components mainly consists of creating a new public/private key pair and then requesting a corresponding component certificate at the local domain CA (see section 3.2.8). In addition to the certificate generation, this process also includes a (uni-directional) remote attestation to ensure that the certified private key of the component will be adequately protected by its TCB. Since the private key and the issued certificates are then sealed and stored locally by the DataSov component, this provisioning step must be executed only once for each component. To evaluate the performance of component provisioning in DataSov, we executed this process a total of 100 times on each of the three deployed systems. Figure 7.3 shows the mean component provisioning times in milliseconds.

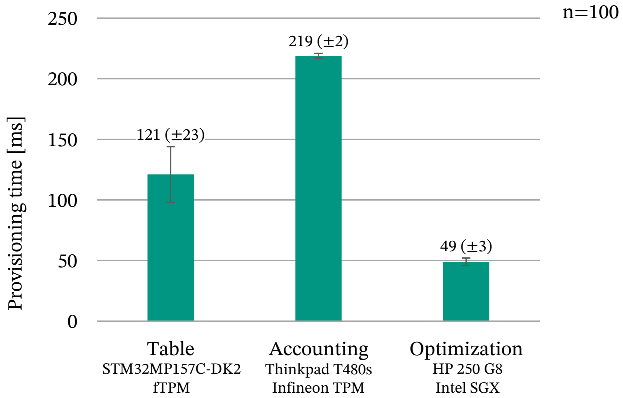


Figure 7.3: Mean provisioning times of DataSov components in milliseconds. The standard deviation is given in brackets.

Unsurprisingly, component provisioning is dominated by the time that is necessary to conduct the required remote attestations (cf. section 4.5.4). Since hardware TPMs are the least efficient technology supported by our framework, provisioning the accounting service takes the longest with almost 220 milliseconds. Besides the remote attestation, this time also includes the generation of the Certificate Signing Request (CSR) at the accounting service and the issuance of a component certificate by the CA. As determined earlier in this thesis, fTPM-based remote attestation on ARM TrustZone devices is more efficient than using hardware TPMs. Consequently, the provisioning of the assembly table is achieved in about 120 milliseconds. Finally, provisioning the SGX enclave of the optimization service only requires about 50 milliseconds. Note that, even though we evaluated the provisioning process on the three domain applications, these results are specific to the underlying trusted computing platforms and not the individual applications. Hence provisioning the usage control components, which are implemented as SGX enclaves, also takes about 50 milliseconds. Given that each component must be provisioned only once, we can conclude that the DataSov provisioning process is fast enough to support realistic use cases even using resource-constrained devices.

7.2.2 Policy Deployment and Enforcement

In addition to the component provisioning, we also evaluated the performance of the complete asset distribution and policy enforcement process in the DataSov framework. To keep the results interpretable, we once again conduct this evaluation separately for all three applications in our exemplary scenario. Furthermore, in our tests we distinguish between the *initial* asset distribution and *subsequent* asset distributions. This is to account for the fact that the latency of policy operations in our distributed usage control framework greatly depends on the number of remote attestations that must be conducted between individual system components. If a new asset is distributed in a freshly launched system, the DataSov framework must first authenticate and remotely attest *all* usage control components that are necessary for the resulting policy deployment and enforcement process. Since the DataSov framework automatically caches the heterogeneously attested connections for a certain period of time (see section 5.2.1), subsequent deployments and enforcements of similar policies (e.g., for new assets) can then re-use the cached connections. Because of this system property, it is not useful to simply determine the average latency for a policy deployment and enforcement process in our evaluation scenario. Instead, we reflect this in our tests by first evaluating the performance of the initial asset distribution in a completely fresh system (i.e., the worst case), before separately testing the subsequent policy deployments and enforcements that make use of cached channels.

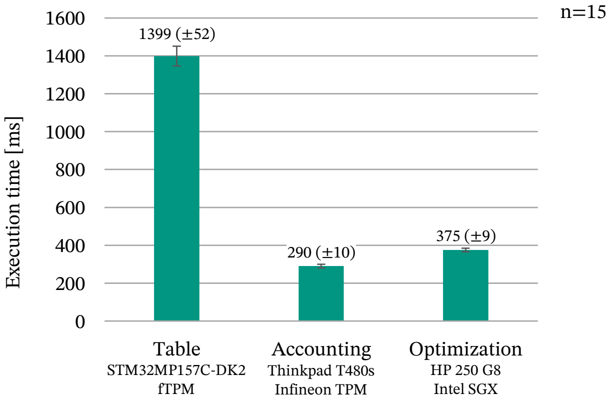


Figure 7.4: Mean initial asset distribution and policy enforcement times in milliseconds. The standard deviation is given in brackets.

Figure 7.4 shows the mean execution times of the three domain applications during the *initial* asset distribution. In a fresh system, the assembly table takes about 1.4 seconds to generate a new asset, deploy the three policies described in section 7.1.3, and then transmit the new asset to both the accounting and the optimization services. This distribution process also triggers the evaluation and enforcement of the policy in listing 7.1 at the assembly table. Furthermore, the assembly table contacts the producer's Provenance Storage Point to track the creation of the new asset. Hence, the resulting total execution time includes the required remote attestations of the PRP, PDP, and ProSP, as well as both of the application services. Also note that contacting the producer's ProSP requires a bi-directional remote attestation instead of a uni-directional one (cf. the identified trust dependencies in section 3.3.3).

The accounting and optimization services then periodically process the received assets, which in turn causes the evaluation of the two policies shown in listings 7.2 and 7.3, respectively. The policy enforcement at the accounting service initially takes about 290 milliseconds, which is again caused mainly by the remote attestations of the required usage control components, i.e., the producer's PRP, PDP, PXP, and ProSP. Similarly, the optimization service must initially attest all local usage control components in the customer's domain (PRP, PDP, and PIP). In addition, since the asset's usage control policy demands provenance tracking at the data source, the customer's services must also remotely attest the producer's PXP and ProSP during the policy evaluation. Together with the enforcement of the resulting usage decision, this takes the optimization service about 375 milliseconds.

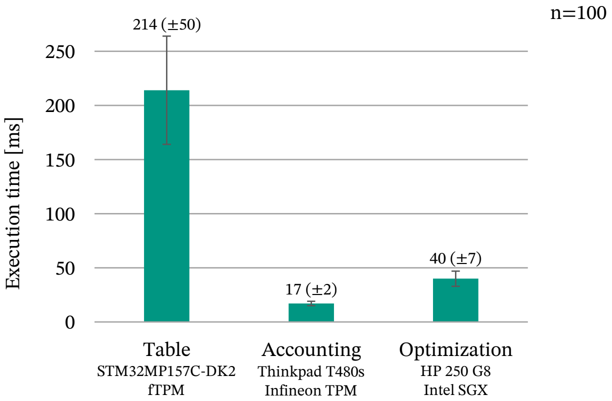


Figure 7.5: Mean subsequent asset distribution and policy enforcement times in milliseconds. The standard deviation is given in brackets.

In contrast to the initial asset distribution, fig. 7.5 shows the mean execution times for all *subsequent* distributions of new data assets. We conduct these tests in the same exact environment as before, except that the remotely attested connections are now cached. As a result, these values show only the time required for asset generation and distribution, as well as the overhead for policy deployment, evaluation, and enforcement in the distributed usage control system. As before, the assembly table has the highest execution time of all three applications with about 214 milliseconds. This is because it is running on the slowest platform and it has to transmit the data assets, including the captured video recordings, to the remote applications. However, compared to the initial asset distribution, the required execution time is significantly reduced.

Similarly, both the accounting and the optimization service can also enforce their policies much quicker than before, with execution times of about 17 and 40 milliseconds, respectively.

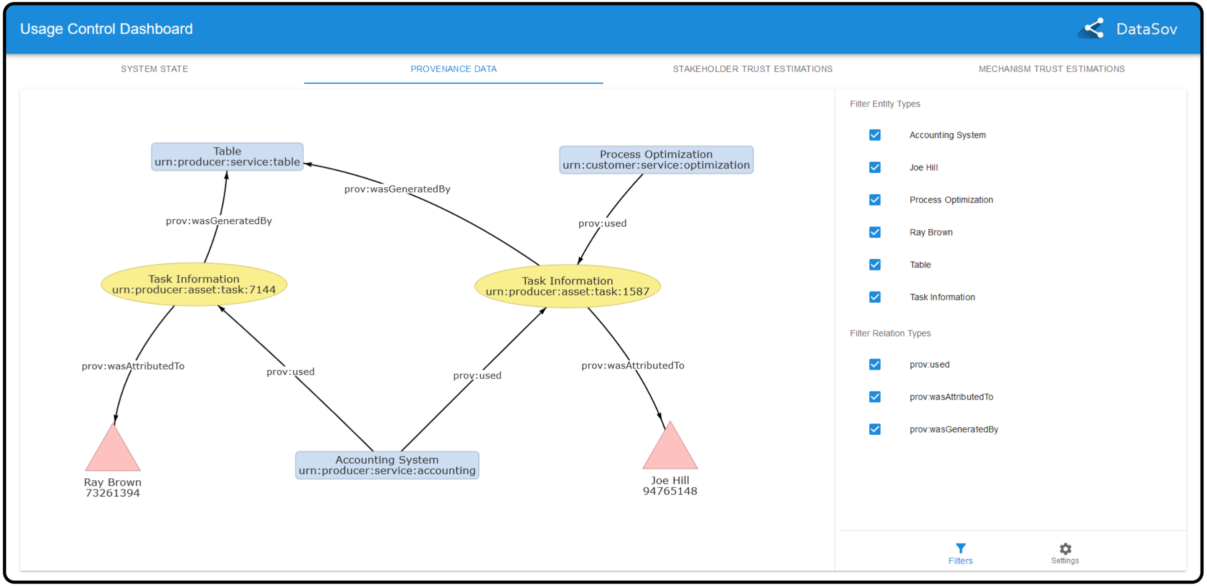
To conclude, these results show that our framework can efficiently disseminate and enforce usage control policies, even when they require support of multiple distributed components, such as remote PIPs, PXPs, and ProSPs. As expected, the number of required remote attestations has a significant impact on the resulting policy enforcement time. Because of this, it is important to carefully configure channel timeouts in the DataSov framework, in order to find a compromise between acceptable performance and frequent re-attestations. Nevertheless, our results show that even on a completely uninitialized distributed usage control system, the policy enforcement times are still usable in practice (cf. fig. 7.4).

7.3 Dashboard Evaluation

We conclude our evaluation by presenting the functionality of the web-based dashboard that is provided by the DataSov framework. More concretely, we show that the DataSov dashboard adequately represents the collected provenance information in our evaluation scenario. Furthermore, we demonstrate that our dashboard can also reveal potentially dangerous usage control system states using the trustworthiness score described in the previous chapter.

7.3.1 Collected Provenance Graphs

The usage control policies presented in listings 7.2 and 7.3 leverage ODRL actions to enforce provenance tracking on all shared data assets. Figure 7.6 shows an excerpt of the resulting provenance graph for two deployed assets, as it is provided by the DataSov dashboard. At the time of the data creation, the assembly table saves the `prov:wasGeneratedBy` relation, as well as the `prov:wasAttributedTo` relation between the asset and the worker, at the local ProSP. This makes it possible to retrospectively identify the data subjects that



Filter Entity Types

☑ Accounting System

☑ Joe Hill

☑ Process Optimization

☑ Ray Brown

☑ Table

☑ Task Information

Filter Relation Types

☑ prov:used

☑ prov:wasAttributedTo

☑ prov:wasGeneratedBy

Figure 7.6: Screenshot of the DataSov provenance dashboard in the example scenario.

are affected by a particular shared asset. Furthermore, the disseminated usage control policies obligate both data processing services to track their individual usages of these assets by logging respective `prov:used` relations. Note that in our example scenario not every asset is used by all services, since the data processing is conditional on the enforced usage restrictions.

In summary, the DataSov framework allows data owners to enforce provenance tracking on shared data assets by issuing usage control policies that include appropriate ODRL obligations. The subsequently collected provenance information is represented according to the PROV data model [Bel13]. Our provenance dashboard then enables the data owners (or an oversight authority) to display and analyze the resulting provenance graphs, in order to understand the usage history of shared data assets. Furthermore, since the DataSov ODRL extension offers a custom RDF description that is referenced in the usage control policies, the underlying provenance model can also be extended with additional relation types or entity subclasses, as required by the specific application scenario.

7.3.2 Configured System Model

Before discussing the trustworthiness scores that our dashboard calculates for the deployed usage control policies, we first have to describe the configuration of the system model that we used in this scenario. As presented previously in section 6.4.1, the majority of the formal system model is automatically determined by the DataSov framework, including the usage control system state, the trust dependencies, the conducted remote attestations, and the deployed usage control policies. However, some of the required information is either subjective or application-specific, and hence must be defined individually by the policy issuers. This includes the operator trust definitions and parts of the attacker model, as well as the protection goals that are relevant for the domain applications. Note that these definitions are always made from the point of view of the dashboard user, which in our evaluation scenario is the producer.

To finalize the formal trust model, we need to define the operator trust estimations for both system participants. As data owner and policy issuer, we can assume the producer to be fully trusted. We reflect this in the trust model by setting $t_o(producer) \coloneqq (t{=}1.0, c{=}1.0)$. The customer, however, is the main antagonist of our usage control enforcement and hence is viewed as principally untrusted. This is reflected by setting $t_o(customer) \coloneqq (t{=}0.1, c{=}0.8)$. Note that the graphical user interface of the DataSov dashboard allows users to easily configure these operator trust estimations using the Human-Trust-Interface (HTI) presented in section 6.4.2. In terms of the attacker model, we only have to decide on the expected adversaries for the customer's components, because the producer is fully trusted anyway. Since the customer operates their own remote distributed usage control components, we must expect malicious system administrators as the most capable adversaries. We can model this by setting the expected attacker to *privileged* for all of the customer's components: $\forall v \in V : o(v) = customer \implies a(v) = privileged.$

Finally, we also need to define the protection goals that are relevant for the various distributed system components. In section 3.3.1 we have already analyzed what protection goals are (generally) necessary for the different usage control components. To briefly summarize, all usage control components must ensure *code integrity* as their main protection goal. PEPs, PDPs, and PRPs must also ensure *data integrity* to preclude manipulations of deployed policies. In addition to data integrity, PIPs and ProSPs must sometimes also protect the *confidentiality* of provided information. Furthermore, a crucial requirement for the reliable enforcement of provenance tracking is that the used ProSPs can adequately protect the collected information against unauthorized deletion. In addition to the general usage control components, we must also define the required protection goals of the new domain applications to properly prepare the trust dashboard for our evaluation scenario. Most importantly, all three applications must once again protect the integrity of the executed code base (I_{code}). Furthermore, since the assembly table is responsible for disseminating the critical data assets, it also must protect data confidentiality and integrity during transmission (C_{tra}, I_{tra}). In addition, the accounting service must be able to protect confidentiality and integrity of the received information during processing (C_{proc}, I_{proc}). This is because the processed data are sensitive and, with the producer being interested in accurate results, must also not be tampered with. In contrast, from the producer's point of view it is sufficient if the optimization service running in the customer's domain only protects the confidentiality of the shared data, since we are not interested in the integrity of the customer's results. Finally, the protection goals C_{store} and I_{store} are of no further relevance for the deployed applications, since they never store any critical data anyway.

Table 7.1: Goal mapping used in the evaluation scenario.

$g : V \to 2^G$	C_{tra}	C_{proc}	C_{store}	I_{tra}	I_{proc}	I_{store}	I_{code}	I_{del}
Table	●	○	○	●	○	○	●	○
Accounting	●	●	○	●	●	○	●	○
Optimization	●	●	○	○	○	○	●	○
PEP	○	○	○	●	●	●	●	○
PDP	○	○	○	●	●	○	●	○
PRP	○	○	○	●	○	●	●	○
PIP	●	○	●	●	○	●	●	○
PXP	○	○	○	●	●	●	●	●
ProSP	○	○	○	●	●	●	●	●

Table 7.1 gives an overview of the resulting goal mapping for the three exemplary data processing applications, as well as all usage control components. Note that we do not require PEPs and PDPs to protect confidentiality, because in our scenario the permitted data usage strategy is known to all participants, and hence the generated usage events are not secret. Consequently, we also do not consider the provenance information stored at the ProSP to be confidential. Since PRPs and PIPs do not process any data, they must reach their protection goals only during data transmission and storage. PDPs and PXPs, on the other hand, must provide secure processing. Finally, the PXP used in our scenario must also protect the integrity of stored data, because it receives log information generated by the optimization service (cf. listing 7.3).

The producer can conveniently define this additional information concerning the formal model via a JSON-based configuration file when setting up the dashboard server. The concrete configuration file that we used for our evaluation in the following section can be found in the DataSov repository.[1]

7.3.3 Resulting Trustworthiness Scores

After establishing the missing parts of the underlying system model in our evaluation scenario, we now show that the DataSov trust dashboard can assist policy issuers in determining the extent to which the distributed usage control system can safeguard the enforcement of their deployed usage rules. To achieve this, we demonstrate that our trustworthiness score identifies and reveals problematic system states in a practical application scenario.

We conduct our demonstration with the usage control policy that is being enforced at the customer's optimization service (see listing 7.3). Figure 7.7 shows the corresponding usage control operation graph that represents the enforcement process of this particular policy in our evaluation scenario. The root of the operation graph is the producer's assembly table, which initially deploys the shared data assets together with their policies at the customer's optimization service. The optimization service then relies on the customer's local PRP and PDP for the evaluation of the received usage control policy. Note that the PEP responsible for enforcing this policy is already included in the code base of the optimization service, and hence is not modeled as an individual vertex in the operation graph. Furthermore, the customer's decision point has a dependency to the local information point, in order to evaluate the ODRL operands specified in the policy (see listing 7.3). Finally, the provenance tracking obligation that is included in the policy triggers the producer's PXP during the enforcement process, which in turn stores the collected provenance information at the ProSP in the producer's domain.

[1] https://gitlab.cc-asp.fraunhofer.de/datasov/ucdashboard/-/blob/master/server/model.json *(accessed on 01/25/2024).*

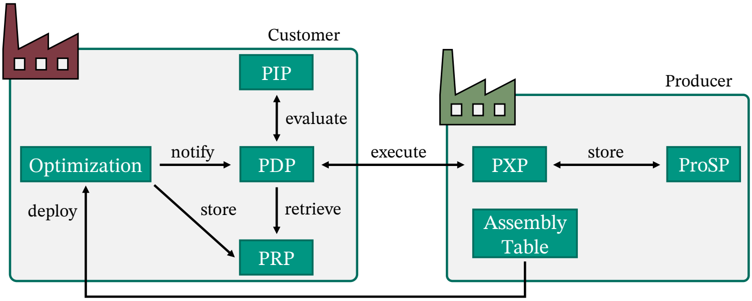
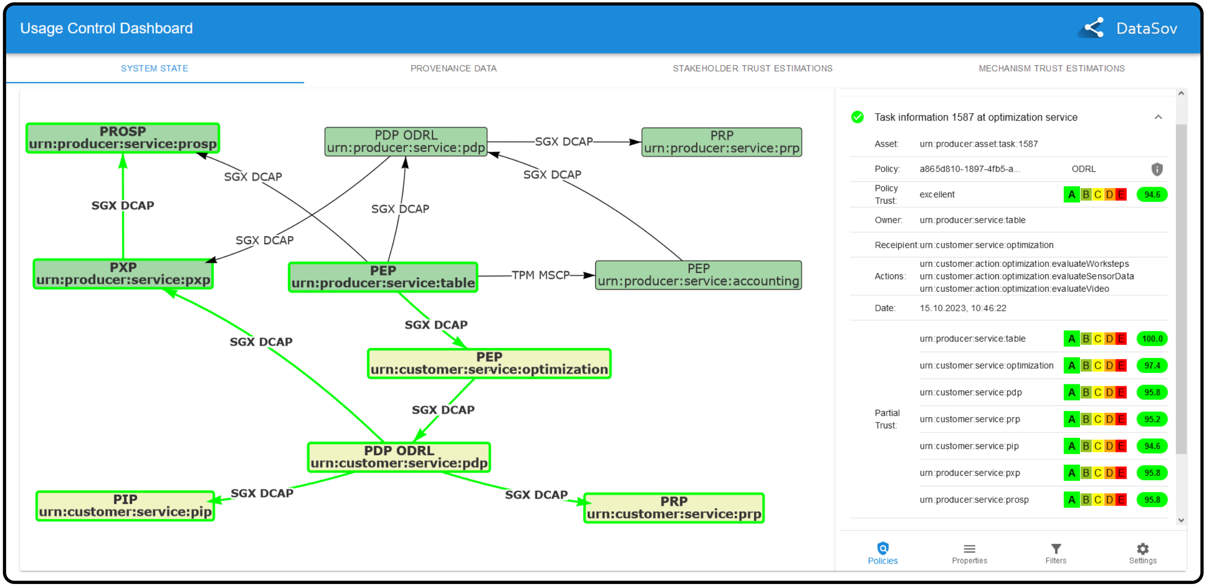


Figure 7.7: Usage control operation graph for the policy enforced at the optimization service in the customer's domain.

Our goal with the usage control operation depicted in fig. 7.7 is to demonstrate that the DataSov trust dashboard can indeed warn a policy issuer (in our case the producer) about potentially dangerous usage control system states. In section 6.1 we have identified *operator propriety* and *technological propriety* as the two major functional requirements of our trustworthiness score. Hence, in the remainder of this section we show that our trust dashboard, and by extension the underlying trustworthiness score, can indeed identify untrusted component operators and point out weakly protected usage control components to the data provider by means of a low policy score in a realistic application setting. Note that for our evaluation we use the baseline trust estimations as specified in section 6.4.3, with the initial expectation value f set to 1.

Evaluation baseline. To establish a baseline for our demonstration, fig. 7.8 depicts the user interface of the producer's trust dashboard during the execution of the previously presented evaluation scenario. In the policy list on the right-hand side of the screen, the asset that has been shared with the optimization service is already selected. The left-hand side shows the automatically generated state graph of the distributed system with the producer's components in green and the customer's components in yellow. The highlighted nodes and edges show the usage control operation (sub-)graph of the currently selected policy (cf. fig. 7.7). As fig. 7.8 shows, all relevant usage control



Task information 1587 at optimization service

Asset: urn:producer:asset:task:1587

Policy: a865d810-1897-4fb5-a... ODRL

Policy Trust: excellent A B C D E 94.6

Owner: urn:producer:service:table

Receipient: urn:customer:service:optimization

Actions: urn:customer:action:optimization:evaluateWorksteps
urn:customer:action:optimization:evaluateSensorData
urn:customer:action:optimization:evaluateVideo

Date: 15.10.2023, 10:46:22

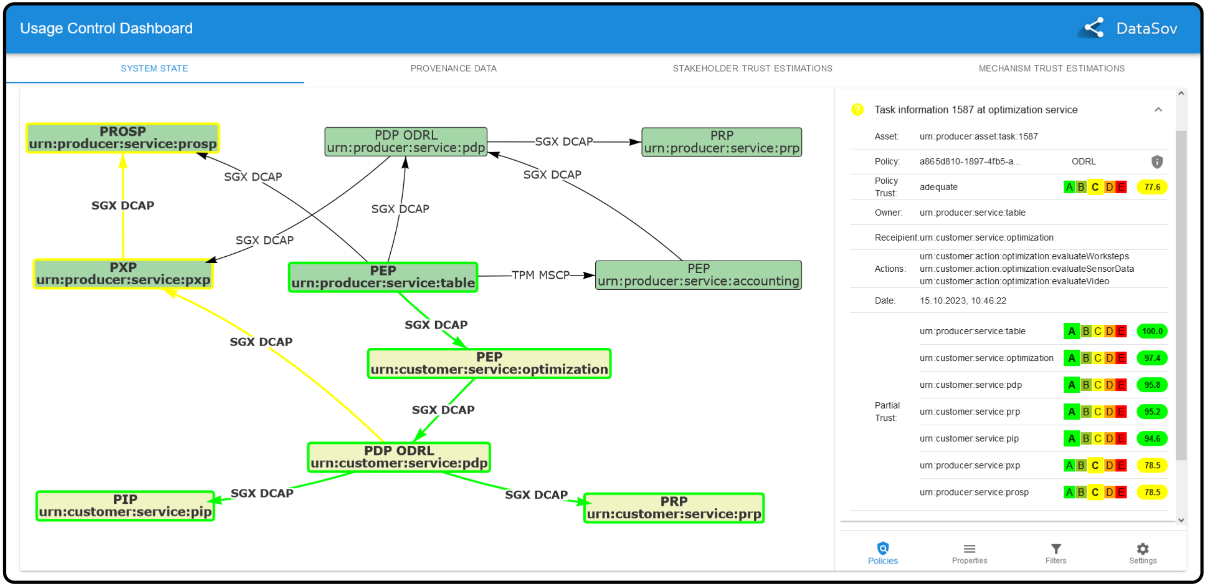
Partial Trust:	urn:producer:service:table	A B C D E	100.0
	urn:customer:service:optimization	A B C D E	97.4
	urn:customer:service:pdp	A B C D E	95.8
	urn:customer:service:prp	A B C D E	95.2
	urn:customer:service:pip	A B C D E	94.6
	urn:producer:service:pxp	A B C D E	95.8
	urn:producer:service:prosp	A B C D E	95.8

Figure 7.8: Screenshot of the DataSov trust dashboard showing the original evaluation scenario. The resulting policy score is 94.6%.

components have been successfully attested. Furthermore, the lower right-hand side of the dashboard summarizes the trust propagation for the partial usage control operations (cf. section 6.4.1). At the root component of the operation (i.e., the assembly table), the partial trustworthiness score initially starts at 100%. This is because the producer is known to be fully trusted. Progressing through the operation graph, however, the resulting partial scores gradually decrease as more and more new security properties are collected from the usage control components in the customer's domain. Since in our evaluation scenario all relevant components are protected by adequate trusted computing technologies that can fulfill the necessary protection goals, the final policy score results to 94.6%. The adequacy of the deployed technologies to enforce the given policy is then pointed out to the user by visualizing the overall policy score at the upper right-hand side of the screen.

Low operator propriety. To demonstrate that our trust dashboard is capable of notifying policy issuers about low operator propriety, we executed a modified version of the smart manufacturing application scenario. Figure 7.9 shows the trust dashboard in the same exact system configuration as before, except in this instance the operator trust of the producer's PXP has been reduced from fully trusted to $t_O(\mathit{producer{:}pxp}) \coloneqq (t{=}0.8, c{=}0.9)$. This might become necessary if a particular subset of components is under the influence of an employee that is not fully trusted. As can be seen in fig. 7.9, modifying the evaluation scenario in such a way reduces the overall policy score from 94.6% to 77.6%. This result shows that the operator trust estimation indeed limits the trustworthiness score even in a realistic use case, according to the minimality principle applied during the score design (see section 6.3.3).

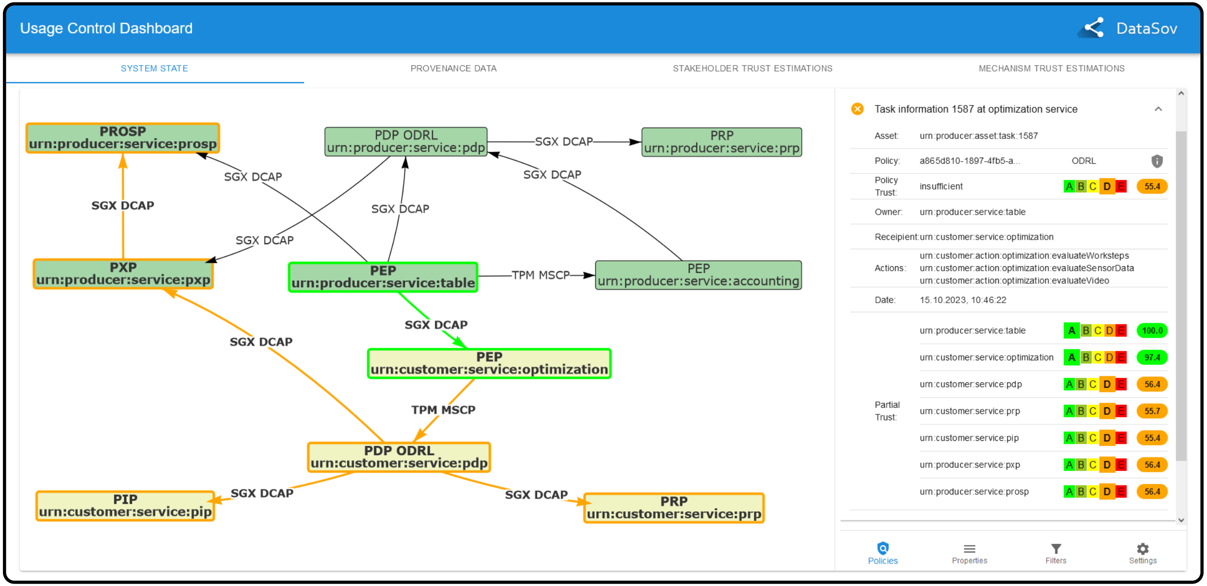
Low technological propriety. For the next evaluation scenario we consider the question of technological propriety. Figure 7.10 shows a screenshot of the trust dashboard with the customer's PDP being executed as a TPM-protected server instead of an Intel SGX enclave. Even though all usage control components are still successfully attested, the dashboard now shows that the partial trustworthiness score sharply decreases at the customer's decision point. This



Task information 1587 at optimization service

Asset:	urn:producer:asset:task:1587		
Policy:	a865d810-1897-4fb5-a...	ODRL	
Policy Trust:	adequate	A B C D E	77.6
Owner:	urn:producer:service:table		
Receipient:	urn:customer:service:optimization		
Actions:	urn:customer:action:optimization:evaluateWorksteps urn:customer:action:optimization:evaluateSensorData urn:customer:action:optimization:evaluateVideo		
Date:	15.10.2023, 10:46:22		
Partial Trust:	urn:producer:service:table	A B C D E	100.0
	urn:customer:service:optimization	A B C D E	97.4
	urn:customer:service:pdp	A B C D E	95.8
	urn:customer:service:prp	A B C D E	95.2
	urn:customer:service:pip	A B C D E	94.6
	urn:producer:service:pxp	A B C D E	78.5
	urn:producer:service:prosp	A B C D E	78.5

Figure 7.9: Screenshot of the DataSov trust dashboard showing the evaluation scenario with reduced operator trust for the producer's PXP. The resulting policy score is 77.6%.



Task information 1587 at optimization service

Asset: urn:producer:asset:task:1587

Policy: a865d810-1897-4fb5-a... ODRL

Policy Trust: insufficient A B C D E 55.4

Owner: urn:producer:service:table

Receipient: urn:customer:service:optimization

Actions: urn:customer:action:optimization:evaluateWorksteps
urn:customer:action:optimization:evaluateSensorData
urn:customer:action:optimization:evaluateVideo

Date: 15.10.2023, 10:46:22

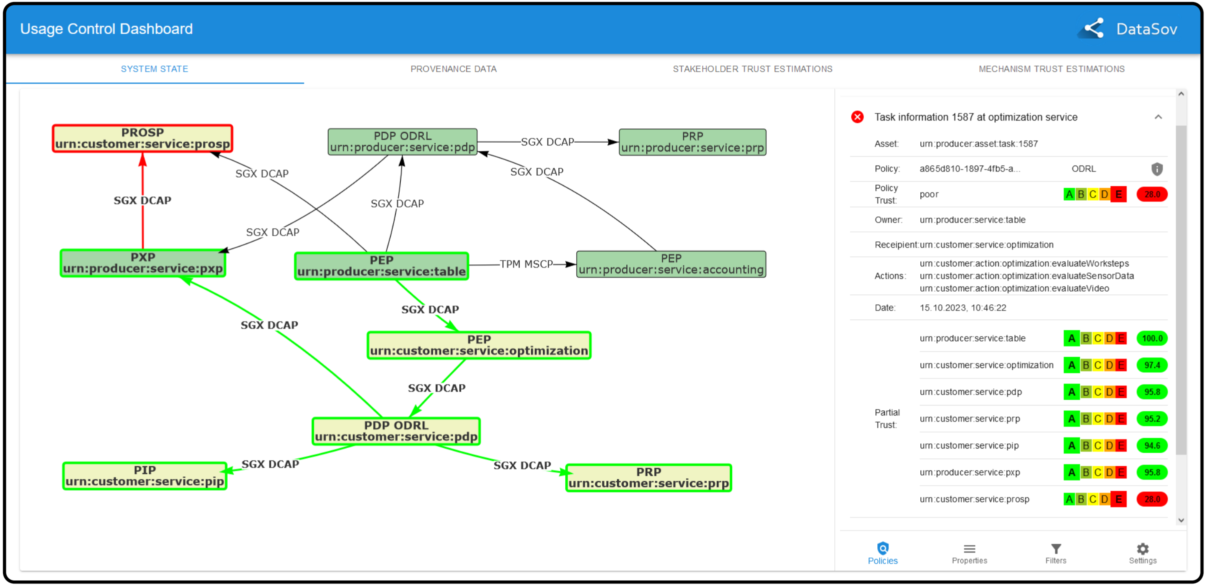
Partial Trust:

Partial Trust:	urn:producer:service:table	A B C D E	100.0
	urn:customer:service:optimization	A B C D E	97.4
	urn:customer:service:pdp	A B C D E	56.4
	urn:customer:service:prp	A B C D E	55.7
	urn:customer:service:pip	A B C D E	55.4
	urn:producer:service:pxp	A B C D E	56.4
	urn:producer:service:prosp	A B C D E	56.4

Figure 7.10: Screenshot of the DataSov trust dashboard showing the evaluation scenario using a TPM-protected PDP in the customer's domain. The resulting policy score is 55.4%.

is because TPMs are not as well-suited as Intel SGX to protect decision points. More concretely, PDPs must fulfill the protection goal of data integrity during processing (see table 7.1). However, TPMs do not provide isolated execution environments for the platform applications, which results in a much larger attack vector on this protection goal than when using Intel SGX. As a result, we must now make additional assumptions to achieve a successful policy enforcement, most notably that the policy evaluation will not be influenced by physical attacks such as intercepting the unencrypted memory bus of the PDP (cf. section 6.4.3). While this may be an acceptable risk for the data provider, depending on the concrete data that should be shared, a trustworthy usage control system must reliably point out this potential weak spot in the enforcement process. Our dashboard achieves this by adding the property $(\text{TPM}, I_{proc}, \textit{privileged}) \in M \times G \times A$ to the set of critical mechanism capabilities, which in turn limits the trustworthiness score in this situation to 55.4%. This result shows that our dashboard can indeed warn data providers about potential security issues with the technological mechanisms that are used to protect the enforcement of their usage control policies. Note that it can still be sensible to operate PDPs as TPM-protected services instead of SGX enclaves, for example if no SGX-capable hardware is available. However, in that case data providers should *consciously* make data sharing decisions based on the feedback that our dashboard provides.

Inadequate provenance tracking. A variant of both operator and technological propriety, which is especially important for provenance tracking, concerns the protection of collected provenance information against unauthorized deletion. Since none of the discussed trusted computing technologies adequately ensure this protection goal against internal adversaries, and deletion attacks are easily executed, policy issuers need to ensure that the used ProSPs are running in a trusted domain (see section 3.4.3). We test the capability of the dashboard to identify this issue by moving the deployed ProSP from the producer's domain to the customer. Figure 7.11 shows the resulting trustworthiness score for our usage control policy in this modified scenario.



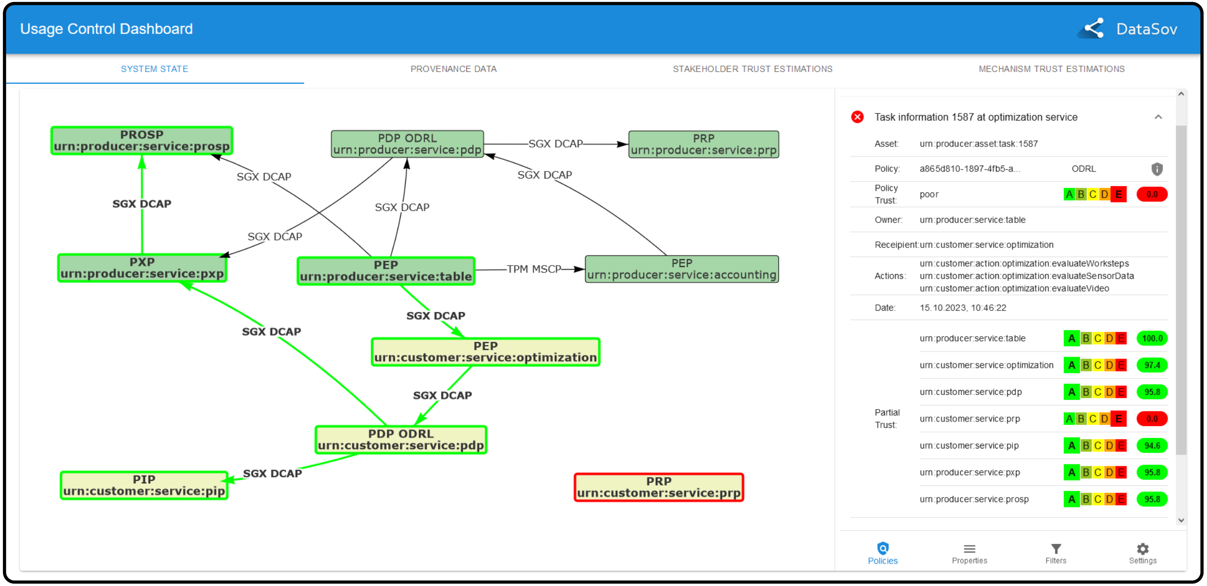
Task information 1587 at optimization service

Asset:	urn:producer:asset:task:1587		
Policy:	a865d810-1897-4fb5-a...	ODRL	
Policy Trust:	poor	A B C D E	28.0
Owner:	urn:producer:service:table		
Receipient:	urn:customer:service:optimization		
Actions:	urn:customer:action:optimization:evaluateWorksteps urn:customer:action:optimization:evaluateSensorData urn:customer:action:optimization:evaluateVideo		
Date:	15.10.2023, 10:46:22		
Partial Trust:	urn:producer:service:table	A B C D E	100.0
	urn:customer:service:optimization	A B C D E	97.4
	urn:customer:service:pdp	A B C D E	95.8
	urn:customer:service:prp	A B C D E	95.2
	urn:customer:service:pip	A B C D E	94.6
	urn:producer:service:pxp	A B C D E	95.8
	urn:customer:service:prosp	A B C D E	28.0

Figure 7.11: Screenshot of the DataSov trust dashboard showing the evaluation scenario using a ProSP in the customer's domain. The resulting policy score is 28.0%.

As we can see in fig. 7.11, the trust dashboard correctly identifies the ProSP as being untrustworthy and consequently marks it with a red border. This is because the component is no longer covered by the high operator trust of the producer, and instead must achieve its security requirements by relying on the deployed trusted computing mechanism. However, the low capability estimations of the SGX technology for the necessary protection goal I_{del} limit the resulting policy score to just 28.0%. This result shows that our dashboard warns policy issuers if the activated usage rules contain provenance tracking obligations that cannot be reliably enforced in the distributed system. The same is true also for PXPs that have to permanently store information relevant for the policy issuer, such as collected log files.

Attestation failures. Finally, for the sake of completeness we also show how our trust dashboard represents failed remote attestations and missing components. Note that this particular representation is *not* based on the previously demonstrated trustworthiness score, since attestation failures and missing usage control components are not considered in the formal model used for our trustworthiness estimation method (see section 6.2). Nevertheless, we can still give data providers comprehensible feedback in these cases. Figure 7.12 shows the trust dashboard with the attestation between the customer's PDP and PRP unsuccessful. In reality, this situation is most likely due to the required PRP not being launched. However, it may also be the case that the establishment of the secure connection between the two components failed due to a deliberate manipulation of the PRP's launch configuration, sealed state, or code base. In both cases our trust dashboard handles the failed attestation by assigning each usage control operation that depends on the missing component a fixed score of 0.0%. We do this because in case of an unresponsive or otherwise unusable component, the proper execution of the policy enforcement process cannot be guaranteed anymore. However, note that in such a scenario the DataSov enforcement point at the optimization service automatically falls back to denying any requested data usages, independently of the trust dashboard. Hence, this is not a situation that requires an active intervention by the policy issuer (security by default).


Usage Control Dashboard
DataSov
SYSTEM STATE
PROVENANCE DATA
STAKEHOLDER TRUST ESTIMATIONS
MECHANISM TRUST ESTIMATIONS
PROSP urn:producer:service:prosp
PDP ODRL urn:producer:service:pdp
PRP urn:producer:service:prp
SGX DCAP
PXP urn:producer:service:pxp
PEP urn:producer:service:table
TPM MSCP
PEP urn:producer:service:accounting
PEP urn:customer:service:optimization
PDP ODRL urn:customer:service:pdp
PIP urn:customer:service:pip
PRP urn:customer:service:prp
Policies
Properties
Filters
Settings

Task information 1587 at optimization service

Asset: urn:producer:asset:task:1587

Policy: a865d810-1897-4fb5-a... ODRL

Policy Trust: poor A B C D E 0.0

Owner: urn:producer:service:table

Receipient: urn:customer:service:optimization

Actions: urn:customer:action:optimization:evaluateWorksteps
urn:customer:action:optimization:evaluateSensorData
urn:customer:action:optimization:evaluateVideo

Date: 15.10.2023, 10:46:22

Partial Trust:			
	urn:producer:service:table	A B C D E	100.0
	urn:customer:service:optimization	A B C D E	97.4
	urn:customer:service:pdp	A B C D E	95.8
	urn:customer:service:prp	A B C D E	0.0
	urn:customer:service:pip	A B C D E	94.6
	urn:producer:service:pxp	A B C D E	95.8
	urn:producer:service:prosp	A B C D E	95.8

Figure 7.12: Screenshot of the DataSov trust dashboard showing the evaluation scenario with a missing attestation to the customer's PRP. The resulting policy score is 0.0%.

7.4 Conclusion

In this chapter we validated our implemented proof of concept by realizing an exemplary application scenario from the realm of smart manufacturing. Our evaluation scenario consists of a smart assembly table disseminating collected work step protocols to remote data processing applications, namely an optimization and an accounting service. Respectively, these three applications are protected by ARM TrustZone, Intel SGX, and a hardware TPM. We also defined ODRL policies for this application scenario, which make use of both customized PXP obligations and PIP operands, as well as the provenance tracking functionalities integrated into the DataSov framework. Based on the defined scenario, we then conducted a performance evaluation of the trustworthy usage control enforcement process (research contribution RC6). Our results show that the initial asset deployment and policy enforcement over freshly attested channels takes about 1.4 seconds at the TrustZone-protected assembly table. At the more powerful accounting and optimization services, the policy enforcement process still takes about 300 to 400 milliseconds. We found these relatively high enforcement times to be caused almost exclusively by the conducted remote attestations. Re-evaluating the policy deployment and enforcement on cached channels showed that the actual usage control overhead results to about 200 milliseconds on the assembly table, and about 20 to 40 milliseconds at the optimization and accounting services. As second contribution in this chapter, we validated that our trust dashboard is capable of identifying problematic usage control system states in this scenario, including low operator trustworthiness and weak technological protection of distributed components (research contribution RC7). This demonstrates that our proposed trustworthiness score fulfills the previously set requirements also in a practical application. However, since in this thesis we focus on the underlying technical aspects and consider questions regarding usability to be out of scope, further research is necessary to validate the adequacy of the visual representations chosen for the dashboard, e.g., by means of user studies.

In summary, our evaluation shows that the developed DataSov framework is capable of implementing trustworthy distributed usage control and provenance tracking in a smart manufacturing use case. The achieved performance of the policy enforcement is sufficient for this purpose, which validates the final goal 5 of the thesis objective. However, our evaluation also shows that the necessary remote attestations cause a relatively high overhead, especially in cases with many contacted usage control components. To alleviate the resulting performance impact, our framework allows to individually determine the frequency of re-attestations (see section 5.2.1). In the future, additional research into "averaging" the conducted attestations, e.g., by randomizing re-attestation periods, may help to avoid sudden latency peaks.

8 Conclusion and Outlook

This chapter concludes the thesis with a brief summary recapitulating the addressed research questions and achieved contributions. We also show that the main objective of this thesis, as well as the initially defined goals, have all been reached. Finally, we identify several starting points for possible future work.

8.1 Summary

Usage control and provenance tracking are two promising technologies to achieve security and transparency in collaborative data processing applications. However, the hitherto unsolved challenge of how to reliably enforce these mechanisms on a technical level – especially in remote and potentially hostile environments – has limited their practical application so far. Achieving this requires a distributed system architecture that is designed to withstand attacks even from malicious component operators. By providing immutable hardware-based trust anchors, trusted computing technologies constitute important building blocks for this goal. However, we must still develop suitable remote attestation protocols that can achieve the necessary requirements both in terms of security and functionality. Finally, such a distributed usage control infrastructure should also give (potential) data providers feedback about the adequacy of the deployed technologies, in order to facilitate informed decisions about data releases and policy definitions.

In the light of these challenges, the main objective of this thesis is to contribute a proof of concept for a *trustworthy* and *distributed* usage control and provenance tracking system, which can be securely operated in *heterogeneous* trusted computing environments. We have achieved this objective by implementing and evaluating the DataSov framework. Furthermore, we have also reached all five goals that have been set out in the introduction.

Goal 1: Our framework provides a joint distributed usage control and provenance tracking infrastructure, which supports flexible and generic application scenarios by being designed in accordance with the widely used XACML reference architecture.

Goal 2: Our framework leverages trusted computing technologies to enforce usage control and provenance tracking on a technical level. A comprehensive security analysis shows that malicious data receivers cannot manipulate the enforcement of usage rules or disturb the tracking of data provenance.

Goal 3: Our framework supports TPMs, Intel SGX, and ARM TrustZone as trusted computing platforms. It also provides technological interoperability by means of a heterogeneous attestation protocol.

Goal 4: Our framework includes a trust dashboard that gives policy issuers a notion of the trustworthiness of the usage control system in its current state.

Goal 5: We demonstrated that our framework is capable of realizing a practical use case from the realm of smart manufacturing.

To achieve these goals, we made several research contributions in three different areas over the course of this thesis. In the remainder of this section, we reiterate the relevant research questions and summarize our contributions.

8.1.1 Usage Control and Provenance Tracking

Research Question 1: *How can we monitor and control the usage of shared data across multiple remote domains? How can shared usage control policies be reliably enforced even in the presence of malicious data receivers?*

To answer this question, we developed a system architecture for *trustworthy* distributed usage control enforcement (research contribution RC1). Our design includes a transitive remote attestation concept that automatically protects the integrity of all relevant system components against malicious tampering. Furthermore, we introduced a certificate-based component authentication scheme to prevent the impersonation of usage control components. To validate the effectiveness of our proposed design, we identified the relevant protection goals, developed a suitable attacker and trust model, and finally conducted a comprehensive security analysis.

Research Question 2: *How can we effectively obligate the collection of provenance information with usage control policies? How can we utilize previously collected provenance information in the specified usage rules?*

We solved this question by conceptually integrating provenance tracking into the distributed usage control enforcement process, as defined by our system design. This allows policy issuers to (i) obligate provenance tracking as part of usage restrictions and (ii) leverage collected provenance information for the specification of new usage rules. To implement this concept in a concrete policy language, we developed an extension of the ODRL information model that adds support for provenance tracking (research contribution RC2). Finally, we also implemented and evaluated a corresponding ODRL decision point that is capable of processing such policies.

8.1.2 Technical Enforcement

Research Question 3: *How can we leverage trusted computing technologies to enforce the correct behavior of usage control and provenance tracking components? Which concrete remote attestation protocols can we use to securely instantiate our distributed system design? Is the performance of those remote attestation protocols sufficient for our purposes?*

To instantiate our conceptual system design with concrete trusted computing technologies, we analyzed the capabilities of TPMs, Intel SGX, and ARM TrustZone. Our analysis revealed that TPM-based attestation protocols currently used in virtual data spaces are vulnerable against nonce-data attacks by malicious administrators. To solve this issue, we designed, analyzed, and implemented a remote attestation protocol that uses a TPM-internal key exchange to establish mutually attested and encrypted communication channels (research contribution RC3). We also implemented a proof of concept for a trusted boot process on ARM TrustZone platforms that allows to conduct integrity measurements of loaded applications both in the normal, as well as in the secure world of the device (research contribution RC4). Finally, we integrated these contributions into our trustworthy distributed usage control framework and validated them using an exemplary application scenario from the realm of smart manufacturing (research contribution RC6). While our evaluation shows that the developed framework fulfills the set expectations, it also revealed a noticeable performance impact that is caused by the conducted remote attestations. We alleviate this issue in our framework by offering cached communication channels, which can intermittently re-attest the components in fixed time intervals.

Research Question 4: *How can we conduct remote attestations in heterogeneous execution environments, i.e., between different trusted computing technologies? Are there significant performance overheads when using a heterogeneous remote attestation protocol compared to single-technology protocols?*

In addition to the previously mentioned contributions in this research area, we also developed and evaluated a heterogeneous remote attestation protocol for our framework (research contribution RC5). Our solution is based on the existing EKEP protocol and can establish mutually attested and encrypted communication channels between TPM-protected software stacks, Intel SGX enclaves, and ARM TrustZone devices. The protocol evaluation shows that heterogeneously conducted attestations do not introduce any significant overhead compared to conventional single-technology approaches.

8.1.3 Trustworthiness Estimation

Research Question 5: *How can the trustworthiness of a distributed usage control system be estimated in relation to the policies that should be enforced, from a particular data provider's point of view?*

To answer this question, in the final part of the thesis we developed a trustworthiness score that represents the ability of a distributed usage control system, which is protected by trusted computing technologies, to successfully enforce a particular usage control policy (research contribution RC7). Our score is based on a formal model that describes the current usage control system state, including the components required for the enforcement of the analyzed policy, the deployed trusted computing technologies, and the conducted remote attestations. We validated our trustworthiness score by showing that it fulfills the previously defined requirements, which include operator and technological propriety, minimality, monotony, and language agnosticism. Finally, we also integrated the described policy score into a trust dashboard that is part of the developed usage control framework. As part of our framework evaluation, we showed that the dashboard indeed identifies problematic system states and issues respective warnings by means of low trustworthiness scores. In addition to this core functionality, the developed dashboard also gives data owners an overview of the currently shared data assets and deployed policies, as well as the collected provenance graphs.

8.2 Future Work

To further improve the applicability of our developed trustworthy usage control and provenance tracking system, additional trusted computing technologies should be integrated into the framework. Complementing the established SGX architecture, Intel recently introduced their Trust Domain Extensions (TDX) as a new Trusted Execution Environment for the Intel platform. Unlike SGX however, TDX provides virtualization-based instead of process-based isolation [Int23b]. Since this increases the size of the protected TCBs compared to SGX, acquiring and validating platform measurements will likely have to be handled differently. Furthermore, AMD's Secure Encrypted Virtualization (SEV) [Kap21] is currently becoming more and more relevant. Since SEV also uses a VM-based architecture, its integration should be conceptually similar to Intel TDX. In addition, integrating the ARM Confidential Computing Architecture (CCA) as successor for the

TrustZone technology on newer processor architectures should be pursued as well. Finally, there may also be a demand for usage control on very resource-constrained devices, which do not have any dedicated trusted computing hardware. For these platforms, the possibilities of DICE-based remote attestation [Tao21] should be investigated. Besides the *conceptual* integration of novel trusted computing hardware into our system, support for new platforms must also be *implemented* in the framework. Since we designed our trustworthy usage control and provenance tracking architecture independently of concrete trusted computing technologies (see chapter 3), our framework can be extended with novel trusted hardware rather easily. This essentially requires implementing a new technology-specific launcher application and sealing layer, as well as adding new assertion providers to the heterogeneous attestation protocol. However, in terms of the technical foundation of our proof of concept implementation, it may be beneficial to use a different trusted computing framework than Asylo for these integrations, since Asylo is very focused on SGX as core technology. One candidate for a more suitable environment is Intel's upcoming SGX/TDX combined attestation infrastructure [Int23c]. Also, there are recent developments towards building a *universal* trusted computing environment [Ott23], which could be a good starting point to develop even more broadly applicable trustworthy usage control systems in the future.

Since the developed framework constitutes a proof of concept implementation, more work must be done before it can be used effectively in productive scenarios. This includes simplifying and automating the deployment and provisioning process of trustworthy usage control and provenance tracking components. We partially considered this in our framework implementation by providing all system components as individual Docker containers. However, so far they still require manual configuration and deployment. In addition, making our framework ready for more complicated use cases than our evaluation scenario also requires additional research into the configuration of remote attestations. As our evaluation showed, the number of conducted attestations significantly impacts the performance of the distributed policy enforcement. This impact is most apparent when using resource-constrained ARM TrustZone devices, but is also noticeable with hardware TPMs and SGX enclaves. Even though our framework can alleviate this issue by offering cached communication channels with intermittent re-attestations, there may be better solutions to avoid sudden latency peaks caused by remote attestations, for example using randomized re-attestation periods. Future research may also investigate the trade-off between security and performance that arises when having to select an appropriate attestation frequency. Finally, since we designed the proof of concept implementation directly according to our previous security analysis (see section 3.4), the framework is always attesting *all* identified trust dependencies between contacted usage control components. Future research could consider minimizing the number of required attestations by dynamically merging components into a single TCB, depending on the current deployment context.

In addition, we have also identified some open research questions regarding the trustworthiness estimation approach presented in chapter 6. For one, in this thesis we focused on the technical aspects of defining a useful trustworthiness score and implementing a corresponding dashboard as a mechanism to provide feedback for data providers. Consequently, our dashboard evaluation in section 7.3 concentrates on demonstrating that the developed score is sound and meets our security requirements. For a future adoption of the proposed method into productive use cases, the adequacy and usability of the included visualization concepts must be evaluated further, e.g., by means of representative user studies. Furthermore, we used exemplary trust values to define the capabilities of trusted computing mechanisms for the evaluation of our score. To apply the constructed policy score productively, these estimations must be acquired from a sufficient number of technology experts using an adequate peer review process. As a starting point for future research on this matter, Hubbard and Seiersen present several suggestions for acquiring accurate expert estimations [Hub16, pp. 133–155]. In addition, we defined the baseline trust estimations underlying our score construction at the level of entire trusted computing mechanisms. While this is an appropriate granularity to use for our application, it also leads to issues with dependent properties, and hence decreases the overall accuracy of the score. To improve on this aspect of our approach, more research should be conducted into modeling the security properties of trustworthy distributed usage control systems.

Bibliography

[Agr07] Agreiter, Berthold; Alam, Muhammad; Breu, Ruth; Hafner, Michael; Pretschner, Alexander; Seifert, Jean-Pierre and Zhang, Xinwen: “A Technical Architecture for Enforcing Usage Control Requirements in Service-Oriented Architectures”. In: *Proceedings of the 2007 ACM Workshop on Secure Web Services.* 2007, pp. 18–25 (cit. on pp. 20, 53, 65).

[Ahm19] Ahmad, Adil; Joe, Byunggill; Xiao, Yuan; Zhang, Yinqian; Shin, Insik and Lee, Byoungyoung: “Obfuscuro: A Commodity Obfuscation Engine on Intel Sgx”. In: *Network and Distributed System Security Symposium.* 2019. url: https://par.nsf.gov/biblio/10134884 (visited on 10/01/2023) (cit. on p. 245).

[Ahn20] Ahn, Jaehwan; Lee, Il-Gu and Kim, Myungchul: “Design and Implementation of Hardware-Based Remote Attestation for a Secure Internet of Things”. In: *Wireless personal communications* 114.1 (2020), pp. 295–327 (cit. on pp. 146, 147).

[Aka22] Akaichi, Ines and Kirrane, Sabrina: “Usage Control Specification, Enforcement, and Robustness: A Survey”. 2022. arXiv: 2203.04800 (cit. on pp. 5, 18, 20, 24).

[Akr16] Akram, Raja Naeem; Markantonakis, Konstantinos; Mayes, Keith; Bonnefoi, Pierre-François; Sauveron, Damien and Chaumette, Serge: “An Efficient, Secure and Trusted Channel Protocol for Avionics Wireless Networks”. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC).* IEEE. 2016, pp. 1–10 (cit. on pp. 110, 113, 114, 116, 123).

[Alh12] Alharbi, Khalid and Lin, Xiaodong: “Pdp: A Privacy-Preserving Data Provenance Scheme”. In: *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp. 500–505 (cit. on p. 31).

[Aln10] Alnemr, Rehab; König, Stefan; Eymann, Torsten and Meinel, Christoph: “Enabling Usage Control through Reputation Objects: A Discussion on e-Commerce and the Internet of Services Environments”. In: *Journal of theoretical and applied electronic commerce research* 5.2 (2010), pp. 59–76 (cit. on p. 51).

[Ama20] Aman, Muhammad Naveed; Basheer, Mohamed Haroon; Dash, Siddhant; Wong, Jun Wen; Xu, Jia; Lim, Hoon Wei and Sikdar, Biplab: “HAtt: Hybrid Remote Attestation for the Internet of Things with High Availability”. In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7220–7233 (cit. on p. 47).

[Amb22] Amblank, Roman: “Both-World Measured Boot Architecture on Arm TrustZone Devices with Firmware-TPMs”. BA thesis. Karlsruher Institut für Technologie (KIT), Apr. 1, 2022. 62 pp. (cit. on p. 138).

[AMD20a] AMD: AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. Jan. 2020. url: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf (visited on 03/30/2023) (cit. on p. 41).

[AMD20b] AMD: Secure Encrypted Virtualization API Version 0.24. Revision 3.24. Apr. 2020. url: https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf (visited on 03/30/2023) (cit. on p. 41).

[And19] Andzakovic, Denis: Extracting BitLocker Keys from a TPM. Pulse Security. Mar. 13, 2019. url: https://pulsesecurity.co.nz/articles/TPM-sniffing (visited on 09/29/2023) (cit. on p. 241).

[Arm08] Armknecht, Frederik; Gasmi, Yacine; Sadeghi, Ahmad-Reza; Stewin, Patrick; Unger, Martin; Ramunno, Gianluca and Vernizzi, Davide: "An Efficient Implementation of Trusted Channels Based on OpenSSL". In: *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing*. 2008, pp. 41–50 (cit. on pp. 111, 113, 116).

[ARM20] ARM: TrustZone for Armv8-A. Version 1.0. Jan. 8, 2020. url: https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Learn%20the%20Architecture/TrustZone%20for%20Armv8-A.pdf (visited on 03/31/2023) (cit. on p. 42).

[ARM21a] ARM: Interaction between Measured Boot and an fTPM (PoC). 2021. url: https://trustedfirmware-a.readthedocs.io/en/latest/design_documents/measured_boot_poc.html (visited on 07/03/2023) (cit. on pp. 137, 140).

[ARM21b] ARM: Trusted Board Boot. 2021. url: https://trustedfirmware-a.readthedocs.io/en/latest/design/trusted-board-boot.html (visited on 07/08/2023) (cit. on pp. 43, 138, 139).

[Art15] Arthur, Will; Challener, David and Goldman, Kenneth: A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security. Springer Nature, 2015 (cit. on pp. 34, 167).

[Asg11] Asghar, Muhammad Rizwan; Ion, Mihaela; Russello, Giovanni and Crispo, Bruno: "Securing Data Provenance in the Cloud". In: *International Workshop on Open Problems in Network Security*. Springer, 2011, pp. 145–160 (cit. on p. 31).

[Asy21a] Asylo: Enclave Key Exchange Protocol (EKEP). 2021. url: https://asylo.dev/docs/concepts/ekep.html (visited on 06/29/2023) (cit. on pp. 131, 132, 149, 151–153, 185).

[Asy21b] Asylo: gRPC Authn and Authz. 2021. url: https://asylo.dev/docs/reference/grpc_auth.html (visited on 08/27/2023) (cit. on p. 189).

[Asy21c] Asylo: Remote Attestation. 2021. url: https://asylo.dev/docs/concepts/remote_attestation.html (visited on 01/25/2024) (cit. on pp. 153, 245).

[Aub17] Aublin, Pierre-Louis; Kelbert, Florian; O'Keffe, D; Muthukumaran, Divya; Priebe, Christian; Lind, Joshua; Krahn, Robert; Fetzer, Christof; Eyers, David and Pietzuch, Peter: TaLoS: Secure and Transparent TLS Termination inside SGX Enclaves. 2017. url: http://www.doc.ic.ac.uk/research/technicalreports/2017/DTRS17-5.pdf (visited on 06/30/2023). preprint (cit. on pp. 130, 132).

[Azi14] Aziz, NorazahAbd; Udzir, Nur Izura and Mahmod, Ramlan: "Extending TLS with Mutual Attestation for Platform Integrity Assurance." In: *Journal of Communication* 9.1 (2014), pp. 63–72 (cit. on pp. 111, 113, 116).

[Bai10] Bai, Guangdong; Gu, Liang; Feng, Tao; Guo, Yao and Chen, Xiangqun: "Context-Aware Usage Control for Android". In: *International Conference on Security and Privacy in Communication Systems.* Springer. 2010, pp. 326–343 (cit. on pp. 21, 23).

[Bal13] Baldini, Gianmarco; Kounelis, Ioannis; Fovino, Igor Nai and Neisse, Ricardo: "A Framework for Privacy Protection and Usage Control of Personal Data in a Smart City Scenario." In: *CRITIS.* Springer, 2013, pp. 212–217 (cit. on pp. 51, 214).

[Ban21] Banks, Alexander Sprogø; Kisiel, Marek and Korsholm, Philip: "Remote Attestation: A Literature Review". 2021. arXiv: 2105.02466 (cit. on p. 105).

[Bar18a] Barker, Elaine; Chen, Lily; Roginsky, Allen; Vassilev, Apostol and Davis, Richard: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. NIST Special Publication (SP) 800-56A Rev. 3. National Institute of Standards and Technology, Apr. 16, 2018. doi: 10.6028/NIST.SP.800-56Ar3 (cit. on pp. 110, 111).

[Bar18b] Bartusch, Felix; Hanussek, Maximilian and Krüger, Jens: "Automatic Generation of Provenance Metadata during Execution of Scientific Workflows." In: *IWSG*. 2018 (cit. on p. 28).

[Bat19] Bates, Adam and Hassan, Wajih Ul: "Can Data Provenance Put an End to the Data Breach?" In: *IEEE Security & Privacy* 17.4 (2019), pp. 88–93 (cit. on p. 28).

[Bed01] Bedford, Tim; Cooke, Roger et al.: Probabilistic Risk Analysis: Foundations and Methods. Cambridge University Press, 2001 (cit. on p. 232).

[Bel13] Belhajjame, Khalid; B'Far, Reza; Cheney, James; Coppens, Sam; Cresswell, Stephen; Gil, Yolanda; Groth, Paul; Klyne, Graham; Lebo, Timothy and McCusker, Jim: PROV-DM: The PROV Data Model. PROV-DM: The PROV Data Model. Apr. 30, 2013. url: https://www.w3.org/TR/prov-dm/ (visited on 03/19/2023) (cit. on pp. 29, 178, 179, 203, 266).

[Bey16] Beyerer, Jürgen and Geisler, Jürgen: "A Framework for a Uniform Quantitative Description of Risk with Respect to Safety and Security". In: *European Journal for Security Research* 1.2 (2016), pp. 135–150 (cit. on p. 212).

[Bie13] Bier, Christoph: "How Usage Control and Provenance Tracking Get Together-a Data Protection Perspective". In: *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 13–17 (cit. on pp. 9, 31).

[Bie21] Bier, Philipp Christoph Sebastian: Umsetzung des datenschutzrechtlichen Auskunftsanspruchs auf Grundlage von Usage-Control und Data-Provenance-Technologien. KIT Scientific Publishing, 2021 (cit. on pp. 6, 9, 28, 31, 60, 70, 96, 177).

[Bir16] Birnstill, Pascal: Privacy-Respecting Smart Video Surveillance Based on Usage Control Enforcement. Vol. 25. KIT Scientific Publishing, 2016 (cit. on pp. 20–22).

[Bir18a] Birnstill, Pascal and Beyerer, Jürgen: “Building Blocks for Identity Management and Protection for Smart Environments and Interactive Assistance Systems”. In: *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*. 2018, pp. 292–296 (cit. on p. 252).

[Bir18b] Birrell, Eleanor; Gjerdrum, Anders; van Renesse, Robbert; Johansen, Håvard; Johansen, Dag and Schneider, Fred B.: “SGX Enforcement of Use-Based Privacy”. In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 2018, pp. 155–167 (cit. on p. 54).

[Bra08] Bratus, Sergey; D’Cunha, Nihal; Sparks, Evan and Smith, Sean W: “TOCTOU, Traps, and Trusted Computing”. In: *International Conference on Trusted Computing*. Springer. 2008, pp. 14–32 (cit. on p. 93).

[Bra17a] Brandenburger, Marcus; Cachin, Christian; Lorenz, Matthias and Kapitza, Rüdiger: “Rollback and Forking Detection for Trusted Execution Environments Using Lightweight Collective Memory”. In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2017, pp. 157–168 (cit. on p. 128).

[Bra17b] Brasser, Ferdinand; Müller, Urs; Dmitrienko, Alexandra; Kostiainen, Kari; Capkun, Srdjan and Sadeghi, Ahmad-Reza: “Software Grand Exposure: SGX Cache Attacks Are Practical”. In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. 2017 (cit. on p. 243).

[Bra19] Brasser, Ferdinand; Capkun, Srdjan; Dmitrienko, Alexandra; Frassetto, Tommaso; Kostiainen, Kari and Sadeghi, Ahmad-Reza: “DR.SGX: Automated and Adjustable Side-Channel Protection for SGX Using Data Location Randomization”. In: *Proceedings of the 35th Annual Computer Security Applications Conference*. San Juan Puerto Rico USA: ACM, Dec. 9, 2019, pp. 788–800. doi: 10.1145/3359789.3359809 (cit. on p. 245).

[Bre18] Brenner, Stefan; Behlendorf, Michael and Kapitza, Rüdiger: "Trusted Execution, and the Impact of Security on Performance". In: *Proceedings of the 3rd Workshop on System Software for Trusted Execution.* 2018, pp. 28–33 (cit. on p. 39).

[Bri10] Brickell, Ernie and Li, Jiangtao: "Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation". In: *2010 IEEE Second International Conference on Social Computing.* IEEE. 2010, pp. 768–775 (cit. on pp. 46, 129).

[Bro22] Brost, Gerd: IDSCP2 Overview. 2022. url: https://github.com/industrial-data-space/idscp2-jvm/wiki/IDSCP2-Overview (visited on 06/30/2023) (cit. on pp. 54, 112–114, 116, 123, 149, 351).

[Bru18] Bruckner, Fabian; Nagel, Ralf; Krüger, Dominik; Wenzel, Sven and Otto, Boris: "Eine Programmiersprache zur souveränen Datenverarbeitung". In: D•A•CH Security 2018. syssec, 2018, pp. 35–46. url: https://www.syssec.at/de/veranstaltungen/dachsecurity2018/papers/DACH_Security_2018_Paper_12A1.pdf (visited on 11/02/2023) (cit. on p. 98).

[Bru21] Bruckner, Fabian and Howar, Falk: "Utilizing Remote Evaluation for Providing Data Sovereignty in Data-sharing Ecosystems". In: Hawaii International Conference on System Sciences. 2021. doi: 10.24251/HICSS.2021.842. url: http://hdl.handle.net/10125/71463 (visited on 11/02/2023) (cit. on p. 98).

[Bub14] Bubeck, Alexander; Gruhler, Matthias; Reiser, Ulrich and Weißhardt, Florian: "Vom fahrerlosen Transportsystem zur intelligenten mobilen Automatisierungsplattform". In: *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung· Technologien· Migration* (2014), pp. 221–233 (cit. on p. 2).

[Bun06] Buneman, Peter; Chapman, Adriane and Cheney, James: "Provenance Management in Curated Databases". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data.* 2006, pp. 539–550 (cit. on p. 28).

[Bun10] Buneman, Peter and Davidson, Susan B.: “Data Provenance – The Foundation of Data Quality”. In: *Workshop: Issues and Opportunities for Improving the Quality and Use of Data within the DoD, Arlington, USA*. 2010, pp. 26–28 (cit. on p. 5).

[Bur17] Burger, Ansgar; Lang, Andreas and Müller, Yannis: “Mögliche Veränderungen von System-Architekturen im Bereich der Produktion”. In: *Industrie 4.0: Wie cyber-physische Systeme die Arbeitswelt verändern* (2017), pp. 57–68 (cit. on p. 1).

[Büt17] Büttner, Karl-Heinz and Brück, Ulrich: “Use Case Industrie 4.0 - Fertigung im Siemens Elektronikwerk Amberg”. In: *Handbuch Industrie 4.0 Bd. 4: Allgemeine Grundlagen* (2017), pp. 45–70 (cit. on p. 1).

[Cai23] Cai, Wenjing; Zhu, Ziyuan; Liu, Yuxin; Zhang, Yusha and Cheng, Xu: “Detecting and Mitigating Cache Side Channel Threats on Intel SGX”. In: *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD).* IEEE, 2023, pp. 972–977 (cit. on p. 245).

[Car18] Carvalho, L.; Belhajjame, Khalid and Medeiros, C.: “A PROV-compliant Approach to Script-to-Workflow Process”. In: *The Sem. Web J* (2018) (cit. on p. 28).

[Che09a] Cheney, James; Chiticariu, Laura and Tan, Wang-Chiew: “Provenance in Databases: Why, How, and Where”. In: *Foundations and Trends® in Databases* 1.4 (2009), pp. 379–474 (cit. on p. 28).

[Che09b] Cheng, Song; Bing, Liu; Yang, Xin; Yixian, Yang; Li, Zhongxian and Han, Yin: “A Security-Enhanced Remote Platform Integrity Attestation Scheme”. In: *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2009, pp. 1–4 (cit. on pp. 111, 113, 114, 116).

[Che17] Chen, Sanchuan; Zhang, Xiaokuan; Reiter, Michael K. and Zhang, Yinqian: “Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjà Vu”. In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ASIA CCS ’17. ACM, Apr. 2, 2017, pp. 7–18. doi: 10.1145/3052973.3053007 (cit. on p. 245).

[Che19a] Chen, Guoxing; Chen, Sanchuan; Xiao, Yuan; Zhang, Yinqian; Lin, Zhiqiang and Lai, Ten H.: “Sgxpectre: Stealing Intel Secrets from Sgx Enclaves via Speculative Execution”. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 142–157 (cit. on p. 244).

[Che19b] Chen, Guoxing; Zhang, Yinqian and Lai, Ten-Hwang: “Opera: Open Remote Attestation for Intel’s Secure Enclaves”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2317–2331 (cit. on pp. 130, 132).

[Che20] Chen, Guoxing and Zhang, Yinqian: “MAGE: Mutual Attestation for a Group of Enclaves without Trusted Third Parties”. 2020. arXiv: 2008.09501 (cit. on p. 72).

[Che22] Cheang, Kevin; Rasmussen, Cameron; Lee, Dayeol; Kohlbrenner, David W.; Asanović, Krste and Seshia, Sanjit A.: “Verifying RISC-V Physical Memory Protection”. 2022. arXiv: 2211.02179 (cit. on p. 43).

[Che23] Chen, Lily; Moody, Dustin; Regenscheid, Andrew; Robinson, Angela and Randall, Karen: Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. NIST Special Publication (SP) 800-186. National Institute of Standards and Technology, Feb. 3, 2023. doi: 10.6028/NIST.SP.800-186 (cit. on p. 156).

[Cho15] Cho, Jin-Hee; Chan, Kevin and Adali, Sibel: “A Survey on Trust Modeling”. In: *ACM Computing Surveys (CSUR)* 48.2 (2015), pp. 1–40 (cit. on pp. 211, 212).

[Cir20] Cirillo, Flavio; Cheng, Bin; Porcellana, Raffaele; Russo, Marco; Solmaz, Gürkan; Sakamoto, Hisashi and Romano, Simon Pietro: "Intentkeeper: Intent-oriented Data Usage Control for Federated Data Analytics". In: *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE. 2020, pp. 204–215 (cit. on p. 53).

[Cok11] Coker, George; Guttman, Joshua; Loscocco, Peter; Herzog, Amy; Millen, Jonathan; O'Hanlon, Brian; Ramsdell, John; Segall, Ariel; Sheehy, Justin and Sniffen, Brian: "Principles of Remote Attestation". In: *International Journal of Information Security* 10.2 (2011), pp. 63–81 (cit. on pp. 110, 116).

[Col10] Colombo, Maurizio; Lazouski, Aliaksandr; Martinelli, Fabio and Mori, Paolo: "A Proposal on Enhancing XACML with Continuous Usage Control Features". In: *Grids, P2P and Services Computing*. Springer, 2010, pp. 133–146 (cit. on p. 25).

[Con23] Constable, Scott; Van Bulck, Jo; Cheng, Xiang; Xiao, Yuan; Xing, Cedric; Alexandrovich, Ilya; Kim, Taesoo; Piessens, Frank; Vij, Mona and Silberstein, Mark: "AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 4051–4068 (cit. on p. 245).

[Cos16a] Costan, Victor and Devadas, Srinivas: "Intel SGX Explained". In: *Cryptology ePrint Archive* (2016) (cit. on pp. 38, 39, 126).

[Cos16b] Costan, Victor; Lebedev, Ilia A. and Devadas, Srinivas: "Sanctum: Minimal Hardware Extensions for Strong Software Isolation." In: *USENIX Security Symposium*. 2016, pp. 857–874 (cit. on p. 44).

[Cui19] Cui, Pinchen; Dixon, Julie; Guin, Ujjwal and Dimase, Daniel: "A Blockchain-Based Framework for Supply Chain Provenance". In: *IEEE Access* 7 (2019), pp. 157113–157125 (cit. on p. 52).

[Dal18] Dall, Fergus; De Micheli, Gabrielle; Eisenbarth, Thomas; Genkin, Daniel; Heninger, Nadia; Moghimi, Ahmad and Yarom, Yuval: “Cachequote: Efficiently Recovering Long-Term Secrets of SGX EPID via Cache Attacks”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.2 (2018), pp. 171–191. doi: tches.v2018.i2.171-191 (cit. on p. 245).

[Dem11] Demsky, Brian: “Cross-Application Data Provenance and Policy Enforcement”. In: *ACM Transactions on Information and System Security (TISSEC)* 14.1 (2011), pp. 1–22 (cit. on p. 55).

[Dew21] Dewaele, Thomas and Oberson, Julien: TPM Sniffing. Sec Team Blog. Nov. 15, 2021. url: https://blog.scrt.ch/2021/11/15/tpm-sniffing/ (visited on 09/29/2023) (cit. on p. 241).

[Di 18] Di Cerbo, Francesco; Martinelli, Fabio; Matteucci, Ilaria and Mori, Paolo: “Towards a Declarative Approach to Stateful and Stateless Usage Control for Data Protection.” In: *WEBIST*. 2018, pp. 308–315 (cit. on p. 25).

[Djo20] Djoko, Judicael Briand: “Towards Practical Access Control and Usage Control on the Cloud Using Trusted Hardware”. PhD thesis. University of Pittsburgh, 2020 (cit. on pp. 8, 54).

[Dui22] Duisberg, Alexander: “Legal Aspects of IDS: Data Sovereignty - What Does It Imply?” In: *Designing Data Spaces* (2022), p. 61 (cit. on p. 5).

[Dus20] Dushku, Edlira; Rabbani, Md Masoom; Conti, Mauro; Mancini, Luigi V and Ranise, Silvio: “SARA: Secure Asynchronous Remote Attestation for IoT Systems”. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3123–3136 (cit. on p. 47).

[El 15] El Kateb, Donia; ElRakaiby, Yehia; Mouelhi, Tejeddine; Rubab, Iram and Le Traon, Yves: “Towards a Full Support of Obligations in XACML”. In: *Risks and Security of Internet and Systems: 9th International Conference, CRiSIS 2014, Trento, Italy, August 27-29, 2014, Revised Selected Papers 9*. Springer, 2015, pp. 213–221 (cit. on p. 25).

[Evt18] Evtyushkin, Dmitry; Riley, Ryan; Abu-Ghazaleh, Nael Cse; Ece and Ponomarev, Dmitry: "BranchScope: A New Side-Channel Attack on Directional Branch Predictor". In: *ACM SIGPLAN Notices* 53.2 (Nov. 30, 2018), pp. 693–707. doi: 10.1145/3296957.3173204 (cit. on p. 244).

[Fei21] Fei, Shufan; Yan, Zheng; Ding, Wenxiu and Xie, Haomeng: "Security Vulnerabilities of SGX and Countermeasures: A Survey". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–36 (cit. on pp. 39, 243).

[Fen21] Feng, Erhu; Lu, Xu; Du, Dong; Yang, Bicheng; Jiang, Xueqiang; Xia, Yubin; Zang, Binyu and Chen, Haibo: "Scalable Memory Protection in the PENGLAI Enclave." In: *OSDI*. 2021, pp. 275–294 (cit. on p. 44).

[Gal12] Galloway, Brendan and Hancke, Gerhard P.: "Introduction to Industrial Control Networks". In: *IEEE Communications surveys & tutorials* 15.2 (2012), pp. 860–880 (cit. on p. 1).

[Gan14] Gandomani, Taghi Javdani; Wei, Koh Tieng and Binhamid, Abdulelah Khaled: "A Case Study Research on Software Cost Estimation Using Experts' Estimates, Wideband Delphi, and Planning Poker Technique". In: *International Journal of Software Engineering and its applications* 8.11 (2014), pp. 173–182 (cit. on p. 232).

[Gao20] Gao, Yuanzhao; Chen, Xingyuan and Du, Xuehui: "A Big Data Provenance Model for Data Security Supervision Based on PROV-DM Model". In: *IEEE Access* 8 (2020), pp. 38742–38752 (cit. on p. 30).

[Gas07] Gasmi, Yacine; Sadeghi, Ahmad-Reza; Stewin, Patrick; Unger, Martin and Asokan, N: "Beyond Secure Channels". In: *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing*. 2007, pp. 30–40 (cit. on pp. 111, 113, 116).

[Geh12] Gehani, Ashish and Tariq, Dawood: “SPADE: Support for Provenance Auditing in Distributed Environments”. In: *Middleware 2012: ACM/IFIP/USENIX 13th International Middleware Conference, Montreal, QC, Canada, December 3-7, 2012. Proceedings 13*. Springer, 2012, pp. 101–120 (cit. on p. 31).

[Gha17] Ghali, Cesar; Stubblefield, Adam; Knapp, Ed; Li, Jiangtao; Schmidt, Benedikt and Boeuf, Julien: Application Layer Transport Security. Dec. 2017. url: https://cloud.google.com/docs/security/encryption-in-transit/application-layer-transport-security (visited on 07/01/2023) (cit. on p. 131).

[Gla10] Glavic, Boris: “Perm: Efficient Provenance Support for Relational Databases”. PhD thesis. University of Zurich, 2010 (cit. on p. 31).

[Glo10] Global Platform: TEE Client API Specification v1.0. July 2010. url: https://globalplatform.org/wp-content/uploads/2010/07/TEE_Client_API_Specification-V1.0.pdf (visited on 02/06/2024) (cit. on p. 133).

[Glo21] Global Platform: TEE Internal Core API Specification v1.3.1. July 2021. url: https://globalplatform.org/wp-content/uploads/2021/03/GPD_TEE_Internal_Core_API_Specification_v1.3.1_PublicRelease_CC.pdf (visited on 12/14/2023) (cit. on pp. 133, 157).

[Gol06] Goldman, Kenneth; Perez, Ronald and Sailer, Reiner: “Linking Remote Attestation to Secure Tunnel Endpoints”. In: *Proceedings of the First ACM Workshop on Scalable Trusted Computing*. 2006, pp. 21–24 (cit. on pp. 111, 113, 116).

[Göt17] Götzfried, Johannes; Eckert, Moritz; Schinzel, Sebastian and Müller, Tilo: “Cache Attacks on Intel SGX”. In: *Proceedings of the 10th European Workshop on Systems Security*. EuroSys ’17: Twelfth EuroSys Conference 2017. Belgrade Serbia: ACM, Apr. 23, 2017, pp. 1–6. doi: 10.1145/3065913.3065915 (cit. on p. 243).

[Gra22] Gramine: Gramine: Attestation and Secret Provisioning. 2022. url: https://gramine.readthedocs.io/en/stable/attestation.html (visited on 08/11/2022) (cit. on pp. 131, 132).

[Gre02] Grenning, James: "Planning Poker or How to Avoid Analysis Paralysis While Release Planning". In: *Hawthorn Woods: Renaissance Software Consulting* 3 (2002), pp. 22–23 (cit. on p. 232).

[Gre11] Greveler, Ulrich; Justus, Benjamin and Loehr, Dennis: "Mutual Remote Attestation: Enabling System Cloning for TPM Based Platforms". In: *International Workshop on Security and Trust Management.* Springer. 2011, pp. 193–206 (cit. on pp. 110, 113, 114, 116, 123).

[Gro13] Groth, Paul and Moreau, Luc: PROV-Overview. PROV-Overview. Apr. 30, 2013. url: https://www.w3.org/TR/prov-overview/ (visited on 03/25/2023) (cit. on pp. 28–30).

[Gru17] Gruss, Daniel; Lettner, Julian; Schuster, Felix; Ohrimenko, Olya; Haller, Istvan and Costa, Manuel: "Strong and Efficient Cache Side-Channel Protection Using Hardware Transactional Memory". In: *26th USENIX Security Symposium (USENIX Security 17).* 2017, pp. 217–233 (cit. on p. 245).

[Gün17] Günthner, Willibald; Klenk, Eva and Tenerowicz-Wirth, Peter: "Adaptive Logistiksysteme als Wegbereiter der Industrie 4.0". In: *Handbuch Industrie 4.0 Bd. 4: Allgemeine Grundlagen* (2017), pp. 99–125 (cit. on p. 2).

[Han18] Han, Seunghun; Shin, Wook; Park, Jun-Hyeok and Kim, HyoungChun: "A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping". In: *27th USENIX Security Symposium (USENIX Security 18).* 2018, pp. 1229–1246 (cit. on p. 242).

[Has17] Hasan, Omar: A Survey of Privacy Preserving Reputation Systems. LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, Nov. 15, 2017 (cit. on pp. 51, 213).

[Her17] Herschel, Melanie; Diestelkämper, Ralf and Ben Lahmar, Houssem: "A Survey on Provenance: What for? What Form? What From?" In: *The VLDB Journal* 26 (2017), pp. 881–906 (cit. on pp. 5, 28).

[Her22] Hercberg, Serge; Touvier, Mathilde and Salas-Salvado, Jordi: "The Nutri-Score Nutrition Label: A Public Health Tool Based on Rigorous Scientific Evidence Aiming to Improve the Nutritional Status of the Population". In: *International Journal for Vitamin and Nutrition Research* 92.3-4 (July 2022), pp. 147–157. doi: 10.1024/0300-9831/a000722 (cit. on p. 235).

[Hil07a] Hilty, Manuel; Pretschner, Alexander and Basin, David: "Verteilte Nutzungskontrolle". In: *digma.* 2007 (cit. on p. 19).

[Hil07b] Hilty, Manuel; Pretschner, Alexander; Basin, David; Schaefer, Christian and Walter, Thomas: "A Policy Language for Distributed Usage Control". In: *European Symposium on Research in Computer Security.* Springer. 2007, pp. 531–546 (cit. on pp. 20, 25).

[Hu15] Hu, Vincent C.; Kuhn, D. Richard; Ferraiolo, David F. and Voas, Jeffrey: "Attribute-Based Access Control". In: *Computer* 48.2 (2015), pp. 85–88 (cit. on p. 17).

[Hu20] Hu, Rui; Yan, Zheng; Ding, Wenxiu and Yang, Laurence T.: "A Survey on Data Provenance in IoT". In: *World Wide Web* 23 (2020), pp. 1441–1463 (cit. on p. 31).

[Hub16] Hubbard, Douglas W and Seiersen, Richard: How to Measure Anything in Cybersecurity Risk. John Wiley & Sons, 2016 (cit. on pp. 212, 248, 287).

[Hub22] Huber, Monika; Wessel, Sascha; Brost, Gerd and Menz, Nadja: "Building Trust in Data Spaces". In: *Designing Data Spaces* (2022), p. 147 (cit. on pp. 7, 50).

[Huo20] Huo, Tianlin; Meng, Xiaoni; Wang, Wenhao; Hao, Chunliang; Zhao, Pei; Zhai, Jian and Li, Mingshu: “Bluethunder: A 2-Level Directional Predictor Based Side-Channel Attack against Sgx”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 321–347 (cit. on p. 244).

[Ian18a] Ianella, Renato; Steidl, Michael; Myles, Stuart and Rodríguez-Doncel, Víctor: ODRL Vocabulary & Expression 2.2. ODRL Vocabulary & Expression 2.2. Feb. 15, 2018. url: https://www.w3.org/TR/odrl-vocab/ (visited on 08/09/2023) (cit. on p. 193).

[Ian18b] Ianella, Renato and Villata, Serena: ODRL Information Model 2.2. ODRL Information Model 2.2. Feb. 15, 2018. url: https://www.w3.org/TR/odrl-model/ (visited on 03/12/2023) (cit. on pp. 12, 25, 66, 193–195, 199, 203).

[Int21] Intel Corporation: Speculative Execution Side Channel Mitigations. May 26, 2021. url: https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/speculative-execution-side-channel-mitigations.html (visited on 10/01/2023) (cit. on p. 245).

[Int23a] Intel Corporation: Intel Software Guard Extensions SDK Developer Reference. Revision 2.19. 2023. url: https://download.01.org/intel-sgx/sgx-linux/2.19/docs/Intel_SGX_Developer_Reference_Linux_2.19_Open_Source.pdf (visited on 03/29/2023) (cit. on pp. 130, 132).

[Int23b] Intel Corporation: Intel Trust Domain Extensions. Feb. 2023. url: https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/documentation.html (visited on 11/06/2023) (cit. on pp. 40, 285).

[Int23c] Intel Corporation: Put Zero Trust Within Reach and Get Public Cloud Flexibility with Private Cloud Security. Intel. Sept. 19, 2023. url: https://www.intel.com/content/www/us/en/content-details/788131/put-zero-trust-within-reach-and-get-public-cloud-flexibility-with-private-cloud-security.html (visited on 11/05/2023) (cit. on pp. 148, 149, 286).

[Jac23] Jacob, Hans Niklas; Werling, Christian; Buhren, Robert and Seifert, Jean-Pierre: faulTPM: Exposing AMD fTPMs' Deepest Secrets. May 2, 2023. arXiv: 2304.14717 [cs]. preprint (cit. on p. 247).

[Jan17] Jang, Yeongjin; Lee, Jaehyuk; Lee, Sangho and Kim, Taesoo: "SGX-Bomb: Locking Down the Processor via Rowhammer Attack". In: *Proceedings of the 2nd Workshop on System Software for Trusted Execution.* SOSP '17: ACM SIGOPS 26th Symposium on Operating Systems Principles. Shanghai China: ACM, Oct. 28, 2017, pp. 1–6. doi: 10.1145/3152701.3152709 (cit. on p. 244).

[Jar19] Jarke, Matthias; Otto, Boris and Ram, Sudha: "Data Sovereignty and Data Space Ecosystems". In: *Business & Information Systems Engineering* 61.5 (2019). url: https://link.springer.com/article/10.1007/s12599-019-00614-2 (cit. on pp. 2, 4).

[Jav18] Javaid, Uzair; Aman, Muhammad Naveed and Sikdar, Biplab: "Blockpro: Blockchain Based Data Provenance and Integrity for Secure Iot Environments". In: *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems.* 2018, pp. 13–18 (cit. on p. 52).

[Jim22] Jimale, Mohamud Ahmed; Z'aba, Muhammad Reza; Kiah, Miss Laiha Binti Mat; Idris, Mohd Yamani Idna; Jamil, Norziana; Mohamad, Moesfa Soeheila and Rohmad, Mohd Saufy: "Authenticated Encryption Schemes: A Systematic Review". In: *IEEE Access* 10 (2022), pp. 14739–14766 (cit. on p. 119).

[Jin21] Jinhui, Yuan; Hongwei, Zhou and Laisun, Zhang: "RSGX: Defeating SGX Side Channel Attack with Return Oriented Programming". In: *2021 IEEE International Conference on Artificial*

Intelligence and Computer Applications (ICAICA). IEEE, 2021, pp. 1094–1098 (cit. on p. 245).

[Jøs07] Jøsang, Audun; Ismail, Roslan and Boyd, Colin: “A Survey of Trust and Reputation Systems for Online Service Provision”. In: *Decision support systems* 43.2 (2007), pp. 618–644 (cit. on p. 211).

[Jos17] Jost, Jana; Kirks, Thomas; Mättig, Benedikt; Sinsel, Alexander and Trapp, Thies Uwe: “Der Mensch in der Industrie – Innovative Unterstützung durch Augmented Reality”. In: *Handbuch Industrie 4.0 Bd. 1: Produktion* (2017), pp. 153–174 (cit. on p. 2).

[Jun14] Jung, Christian; Eitel, Andreas and Schwarz, Reinhard: “Enhancing Cloud Security with Context-aware Usage Control Policies.” In: *GI-Jahrestagung* 211 (2014), p. 50 (cit. on pp. 20–22, 26, 59, 63, 75, 89).

[Jun17] Jungbluth, Volker: “Intelligente, vernetzte Lagersysteme für die Industrie 4.0: Beispiel Shuttle-Technologien”. In: *Handbuch Industrie 4.0 Bd. 3: Logistik* (2017), pp. 139–149 (cit. on p. 1).

[Jun22] Jung, Christian and Dörr, Jörg: “Data Usage Control”. In: *Designing Data Spaces.* Springer, 2022, pp. 129–146 (cit. on pp. 26, 59, 63, 89).

[Kaa20] Kaaniche, Nesrine; Belguith, Sana; Laurent, Maryline; Gehani, Ashish; Russello, Giovanni et al.: “Prov-Trust: Towards a Trustworthy SGX-based Data Provenance System”. In: *Proceedings of the 17th International Joint Conference on E-Business and Telecommunications-Volume 3: SECRYPT.* ScitePress. 2020, pp. 225–237 (cit. on p. 55).

[Kan22] Kanal, Martin: Der Eclipse Dataspace Connector (EDC) – Architektur und Nutzen des Frameworks. Business Software und IT-Blog - Wir gestalten digitale Wertschöpfung. Aug. 23, 2022. url: https://blog.doubleslash.de/der-eclipse-dataspace-connector-edc-architektur-und-nutzen-des-frameworks/ (visited on 02/28/2023) (cit. on p. 3).

[Kap17] Kaplan, David: "Protecting VM Register State with SEV-ES". In: *White paper* (Feb. 17, 2017). url: https://www.amd.com/system/files/TechDocs/Protecting%20VM%20Register%20State%20with%20SEV-ES.pdf (visited on 03/30/2023) (cit. on p. 41).

[Kap21] Kaplan, David; Powell, Jeremy and Woller, Tom: "AMD Memory Encryption". In: *White paper* (Oct. 18, 2021). url: https://www.amd.com/system/files/TechDocs/memory-encryption-white-paper.pdf (visited on 03/30/2023) (cit. on pp. 41, 285).

[Keb18] Kebede, Milen G.; Sileno, Giovanni and Van Engers, Tom: "A Critical Reflection on ODRL". In: *International Workshop on AI Approaches to the Complexity of Legal Systems*. Springer, 2018, pp. 48–61 (cit. on p. 193).

[Kel13] Kelbert, Florian and Pretschner, Alexander: "Data Usage Control Enforcement in Distributed Systems". In: *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*. 2013, pp. 71–82 (cit. on pp. 5, 18, 20, 22, 23).

[Kel14] Kelbert, Florian and Pretschner, Alexander: "Decentralized Distributed Data Usage Control". In: *International Conference on Cryptology and Network Security*. Springer. 2014, pp. 353–369 (cit. on pp. 23, 24).

[Kel15] Kelbert, Florian and Pretschner, Alexander: "A Fully Decentralized Data Usage Control Enforcement Infrastructure". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2015, pp. 409–430 (cit. on pp. 23, 24).

[Kel18] Kelbert, Florian and Pretschner, Alexander: "Data Usage Control for Distributed Systems". In: *ACM Transactions on Privacy and Security (TOPS)* 21.3 (2018), pp. 1–32 (cit. on pp. 20–22).

[Kim19] Kim, Deokjin; Jang, Daehee; Park, Minjoon; Jeong, Yunjong; Kim, Jonghwan; Choi, Seokjin and Kang, Brent Byunghoon: "SGX-LEGO: Fine-grained SGX Controlled-Channel Attack and Its Countermeasure". In: *computers & security* 82 (2019), pp. 118–139 (cit. on p. 244).

[Kin21] King, Gordon and Wang, Hans: "HTTPA: HTTPS Attestable Protocol". 2021. arXiv: 2110.07954 (cit. on pp. 131, 132).

[Kir22] Kirstein, Fabian and Bohlen, Vincent: "IDS as a Foundation for Open Data Ecosystems". In: *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*. Springer International Publishing Cham, 2022, pp. 225–240 (cit. on p. 3).

[Kna18] Knauth, Thomas; Steiner, Michael; Chakrabarti, Somnath; Lei, Li; Xing, Cedric and Vij, Mona: "Integrating Remote Attestation with Transport Layer Security". 2018. arXiv: 1801.05863 (cit. on pp. 45, 130, 132).

[Kog22] Kogler, Andreas; Gruss, Daniel and Schwarz, Michael: "Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks". In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 4147–4164 (cit. on p. 245).

[Kon14] Kong, Joonho; Koushanfar, Farinaz; Pendyala, Praveen K; Sadeghi, Ahmad-Reza and Wachsmann, Christian: "PUFatt: Embedded Platform Attestation Based on Novel Processor-Based PUFs". In: *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2014, pp. 1–6 (cit. on p. 47).

[Kor18] Koruyeh, Esmaeil Mohammadian; Khasawneh, Khaled N.; Song, Chengyu and Abu-Ghazaleh, Nael: "Spectre Returns! Speculation Attacks Using the Return Stack Buffer". In: *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. 2018 (cit. on p. 244).

[Krä22] Krämer, Jan; Stüdlein, Nadine and Zierke, Oliver: "Sharing Needs Caring: Experimental Insights on the Optimal Design of B2B Data Sharing Platforms". In: *Data as a Common Good* (2022), p. 66 (cit. on p. 4).

[Kum12] Kumari, Prachi and Pretschner, Alexander: "Deriving Implementation-Level Policies for Usage Control Enforcement". In: *Proceedings of the Second ACM Conference on Data and Application Security and Privacy.* 2012, pp. 83–94 (cit. on p. 25).

[Kun06] Kuntze, Nicolai and Schmidt, Andreas U.: "Transitive Trust in Mobile Scenarios". In: *Emerging Trends in Information and Communication Security: International Conference, ETRICS 2006, Freiburg, Germany, June 6-9, 2006. Proceedings.* Springer, 2006, pp. 73–85 (cit. on p. 58).

[Kyl07] Kyle, David and Brustoloni, José Carlos: "Uclinux: A Linux Security Module for Trusted-Computing-Based Usage Controls Enforcement". In: *Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing.* 2007, pp. 63–70 (cit. on pp. 53, 65).

[Lan14] Lan, Anna; Han, Zhen; Zhang, Dawei; Jiang, Yichen; Liu, Tianhua and Li, Meihong: "An Anonymous Remote Attestation Protocol to Prevent Masquerading Attack". In: *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops.* IEEE. 2014, pp. 590–595. doi: 10.1109/UIC-ATC-ScalCom.2014.30 (cit. on pp. 112, 116).

[Lan16] Langley, Adam; Hamburg, Mike and Turner, Sean: Elliptic Curves for Security. Request for Comments RFC 7748. Internet Engineering Task Force, Jan. 2016. 22 pp. doi: 10.17487/RFC7748. url: https://datatracker.ietf.org/doc/rfc7748 (visited on 01/22/2024) (cit. on p. 156).

[Lan22] Lang, Fan; Wang, Wei; Meng, Lingjia; Lin, Jingqiang; Wang, Qiongxiao and Lu, Linli: "MoLE: Mitigation of Side-channel Attacks against SGX via Dynamic Data Location Escape". In: *Proceedings of the 38th Annual Computer Security Applications Conference.* ACSAC: Annual Computer Security Applications Conference. Austin TX USA: ACM, Dec. 5, 2022, pp. 978–988. doi: 10.1145/3564625.3568002 (cit. on p. 245).

[Lau22] Lauf, Florian; Scheider, Simon; Bartsch, Jan; Herrmann, Philipp; Radic, Marija; Rebbert, Marcel; Nemat, André T.; Schlueter Langdon, Christoph; Konrad, Ralf and Sunyaev, Ali: “Linking Data Sovereignty and Data Economy: Arising Areas of Tension”. In: 17th International Conference on Wirtschaftsinformatik. 2022 (cit. on p. 4).

[Laz10] Lazouski, Aliaksandr; Martinelli, Fabio and Mori, Paolo: “Usage Control in Computer Security: A Survey”. In: *Computer Science Review* 4.2 (2010), pp. 81–99 (cit. on pp. 5, 17).

[Laz14] Lazouski, Aliaksandr; Mancini, Gaetano; Martinelli, Fabio and Mori, Paolo: “Architecture, Workflows, and Prototype for Stateful Data Usage Control in Cloud”. In: *2014 IEEE Security and Privacy Workshops*. IEEE. 2014, pp. 23–30 (cit. on p. 21).

[Lee17a] Lee, Jaehyuk; Jang, Jinsoo; Jang, Yeongjin; Kwak, Nohyun; Choi, Yeseul; Choi, Changho; Kim, Taesoo; Peinado, Marcus and Kang, Brent ByungHoon: “Hacking in Darkness: Return-oriented Programming against Secure Enclaves”. In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 523–539 (cit. on pp. 244, 245).

[Lee17b] Lee, Sangho; Shih, Ming-Wei; Gera, Prasun; Kim, Taesoo; Kim, Hyesoon and Peinado, Marcus: “Inferring Fine-Grained Control Flow inside SGX Enclaves with Branch Shadowing”. In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 557–574 (cit. on p. 244).

[Lee20] Lee, Dayeol; Kohlbrenner, David; Shinde, Shweta; Asanović, Krste and Song, Dawn: “Keystone: An Open Framework for Architecting Trusted Execution Environments”. In: *Proceedings of the Fifteenth European Conference on Computer Systems*. 2020, pp. 1–16 (cit. on p. 44).

[Len20] Lengenfelder, Christian; Frese, Christian; Zube, Angelika; Voit, Michael and Beyerer, Jürgen: “A Cooperative HCI Assembly Station with Dynamic Projections”. In: *ISR 2020; 52th*

International Symposium on Robotics. VDE, 2020, pp. 1–8 (cit. on pp. 2, 251).

[Les16] Leslie-Hurd, Rebekah: "Sealing and Attestation in Intel Software Guard Extensions". Jan. 8, 2016. url: https://rwc.iacr.org/2016/Slides/Sealing%20and%20Attestation%20in%20SGX.pdf (visited on 03/29/2023) (cit. on p. 40).

[Li10] Li, Yanlin; McCune, Jonathan M. and Perrig, Adrian: "SBAP: Software-based Attestation for Peripherals". In: *Trust and Trustworthy Computing: Third International Conference, TRUST 2010, Berlin, Germany, June 21-23, 2010. Proceedings 3.* Springer, 2010, pp. 16–29 (cit. on p. 47).

[Li19] Li, Wenhao; Xia, Yubin and Chen, Haibo: "Research on ARM Trustzone". In: *GetMobile: Mobile Computing and Communications* 22.3 (2019), pp. 17–22 (cit. on p. 42).

[Li22] Li, Xupeng; Li, Xuheng; Dall, Christoffer; Gu, Ronghui; Nieh, Jason; Sait, Yousuf and Stockwell, Gareth: "Design and Verification of the Arm Confidential Compute Architecture". In: *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22).* 2022, pp. 465–484 (cit. on p. 43).

[Lia17] Liang, Xueping; Shetty, Sachin; Tosh, Deepak; Kamhoua, Charles; Kwiat, Kevin and Njilla, Laurent: "Provchain: A Blockchain-Based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability". In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID).* IEEE, 2017, pp. 468–477 (cit. on pp. 31, 52).

[Lin21] Ling, Zhen; Yan, Huaiyu; Shao, Xinhui; Luo, Junzhou; Xu, Yiling; Pearson, Bryan and Fu, Xinwen: "Secure Boot, Trusted Boot and Remote Attestation for ARM TrustZone-based IoT Nodes". In: *Journal of Systems Architecture* 119 (2021), p. 102240 (cit. on p. 147).

[Lip21] Lipp, Moritz; Kogler, Andreas; Oswald, David; Schwarz, Michael; Easdon, Catherine; Canella, Claudio and Gruss, Daniel: "PLATYPUS: Software-based Power Side-Channel Attacks on X86". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 355–371 (cit. on p. 244).

[Lyl10] Lyle, John and Martin, Andrew: "Trusted Computing and Provenance: Better Together". In: *Proceedings of the 2nd Conference on Theory and Practice of Provenance*. 2010 (cit. on p. 55).

[Man19] Mannhardt, Felix; Petersen, Sobah Abbas and Oliveira, Manuel Fradinho: "A Trust and Privacy Framework for Smart Manufacturing Environments". In: *Journal of Ambient Intelligence and Smart Environments* 11.3 (2019), pp. 201–219 (cit. on pp. 4, 251, 252).

[Mar19] Marra, Antonio La; Martinelli, Fabio; Mori, Paolo and Saracino, Andrea: "A Distributed Usage Control Framework for Industrial Internet of Things". In: *Security and Privacy Trends in the Industrial Internet of Things*. Springer, 2019, pp. 115–135 (cit. on p. 20).

[Mat17] Matetic, Sinisa; Ahmed, Mansoor; Kostiainen, Kari; Dhar, Aritra; Sommer, David; Gervais, Arthur; Juels, Ari and Capkun, Srdjan: "ROTE: Rollback Protection for Trusted Execution". In: *Proceedings of the 26th USENIX Conference on Security Symposium*. Usenix Association, 2017, pp. 1289–1306 (cit. on pp. 94, 95, 128, 183).

[Mei13] Meier, Simon; Schmidt, Benedikt; Cremers, Cas and Basin, David: "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer, 2013, pp. 696–701 (cit. on p. 119).

[Mén22] Ménétrey, Jämes; Pasin, Marcelo; Felber, Pascal and Schiavoni, Valerio: “Watz: A Trusted WebAssembly Runtime Environment with Remote Attestation for TrustZone”. 2022. arXiv: 2206.08722 (cit. on p. 147).

[Mey21] Meyer Zum Felde, Hendrik; Morbitzer, Mathias and Schütte, Julian: “Securing Remote Policy Enforcement by a Multi-Enclave Based Attestation Architecture”. In: *2021 IEEE 19th International Conference on Embedded and Ubiquitous Computing (EUC).* IEEE. 2021, pp. 102–108 (cit. on pp. 8, 54).

[Mio19] Miorandi, Daniele; Rizzardi, Alessandra; Sicari, Sabrina and Coen-Porisini, Alberto: “Sticky Policies: A Survey”. In: *IEEE Transactions on Knowledge and Data Engineering* 32.12 (2019), pp. 2481–2499 (cit. on pp. 63, 64, 98).

[Mof18] Mofrad, Saeid; Zhang, Fengwei; Lu, Shiyong and Shi, Weidong: “A Comparison Study of Intel SGX and AMD Memory Encryption Technology”. In: *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy.* 2018, pp. 1–8 (cit. on p. 41).

[Mog17] Moghimi, Ahmad; Irazoqui, Gorka and Eisenbarth, Thomas: “CacheZoom: How SGX Amplifies the Power of Cache Attacks”. In: *Cryptographic Hardware and Embedded Systems – CHES 2017.* Ed. by Fischer, Wieland and Homma, Naofumi. Vol. 10529. Cham: Springer International Publishing, 2017, pp. 69–90. doi: 10.1007/978-3-319-66787-4_4 (cit. on p. 243).

[Mog19] Moghimi, Daniel; Sunar, Berk; Eisenbarth, Thomas and Heninger, Nadia: TPM-FAIL Website. 2019. url: https://tpm.fail (visited on 09/29/2023) (cit. on p. 241).

[Mog20] Moghimi, Daniel; Sunar, Berk; Eisenbarth, Thomas and Heninger, Nadia: “TPM-FAIL: TPM Meets Timing and Lattice Attacks”. In: *29th USENIX Security Symposium (USENIX Security 20).* 2020, pp. 2057–2073 (cit. on p. 241).

[Mol08] Moløkken-Østvold, Kjetil; Haugen, Nils Christian and Benestad, Hans Christian: “Using Planning Poker for Combining Expert Estimates in Software Projects”. In: *Journal of Systems and Software* 81.12 (2008), pp. 2106–2117 (cit. on p. 232).

[Mun20] Munoz-Arcentales, Andres; López-Pernas, Sonsoles; Pozo, Alejandro; Alonso, Álvaro; Salvachúa, Joaquín and Huecas, Gabriel: “Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE”. In: *Sustainability* 12.9 (2020), p. 3885 (cit. on p. 21).

[Mur20] Murdock, Kit; Oswald, David; Garcia, Flavio D; Van Bulck, Jo; Gruss, Daniel and Piessens, Frank: “Plundervolt: Software-based Fault Injection Attacks against Intel SGX”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 1466–1482 (cit. on p. 244).

[Nag22] Nagel, Lars and Lycklama, Douwe: “How to Build, Run, and Govern Data Spaces”. In: *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*. Springer International Publishing Cham, 2022, pp. 17–28 (cit. on p. 2).

[Nei11a] Neisse, Ricardo; Holling, Dominik and Pretschner, Alexander: “Implementing Trust in Cloud Infrastructures”. In: *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2011, pp. 524–533 (cit. on p. 53).

[Nei11b] Neisse, Ricardo; Pretschner, Alexander and Di Giacomo, Valentina: “A Trustworthy Usage Control Enforcement Framework”. In: *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE. 2011, pp. 230–235 (cit. on pp. 7, 53, 65).

[Nei15] Neisse, Ricardo; Steri, Gary; Fovino, Igor Nai and Baldini, Gianmarco: “SecKit: A Model-Based Security Toolkit for the Internet of Things”. In: *computers & security* 54 (2015), pp. 60–76 (cit. on pp. 51, 212, 214).

[Nei17] Neisse, Ricardo; Steri, Gary and Nai-Fovino, Igor: “A Blockchain-Based Approach for Data Accountability and Provenance Tracking”. In: *Proceedings of the 12th International Conference on Availability, Reliability and Security.* 2017, pp. 1–10 (cit. on pp. 31, 52).

[Nil20] Nilsson, Alexander; Bideh, Pegah Nikbakht and Brorsson, Joakim: “A Survey of Published Attacks on Intel SGX”. 2020. arXiv: 2006.13598 (cit. on p. 243).

[OAS13] OASIS Standard: eXtensible Access Control Markup Language (XACML) Version 3.0. Jan. 22, 2013. url: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf (visited on 03/16/2023) (cit. on pp. 20, 25).

[Oli19] Oliveira, Marcelo Iury S.; Barros Lima, Glória de Fátima and Farias Lóscio, Bernadette: “Investigations into Data Ecosystems: A Systematic Mapping Study”. In: *Knowledge and Information Systems* 61 (2019), pp. 589–630 (cit. on p. 2).

[OP-19a] OP-TEE: Secure Storage. 2019. url: https://optee.readthedocs.io/en/latest/architecture/secure_storage.html (visited on 07/03/2023) (cit. on pp. 133, 134, 158).

[OP-19b] OP-TEE: Trusted Applications. 2019. url: https://optee.readthedocs.io/en/latest/architecture/trusted_applications.html (visited on 07/04/2023) (cit. on pp. 134, 142).

[Opr21] Opriel, Sebastian; Skubowius, Emanuel and Lamberjohann, Marvin: “How Usage Control Fosters Willingness to Share Sensitive Data in Inter-Organizational Processes of Supply Chains”. In: *International Scientific Symposium on Logistics.* Vol. 91. 2021 (cit. on p. 5).

[Opr22] Opriel, Sebastian and Schmelting, Jürgen: “Datensouveränität”. In: *Silicon Economy: Wie digitale Plattformen industrielle Wertschöpfungsnetzwerke global verändern.* Springer, 2022, pp. 41–54 (cit. on pp. 3–5).

[Ott18] Otto, Boris; Ten Hompel, Michael and Wrobel, Stefan: "Industrial Data Space: Referenzarchitektur für die Digitalisierung der Wirtschaft". In: *Digitalisierung: Schlüsseltechnologien für Wirtschaft und Gesellschaft* (2018), pp. 113–133 (cit. on p. 2).

[Ott19] Otto, Boris; Steinbuß, Sebastian; Teuscher, Andreas and Lohmann, Steffen: IDS Reference Architecture Model (Version 3.0). International Data Spaces Association, Apr. 1, 2019. url: http://doi.org/10.5281/zenodo.5105529 (visited on 02/28/2023) (cit. on pp. 3, 7, 54).

[Ott22a] Otto, Boris: "A Federated Infrastructure for European Data Spaces". In: *Communications of the ACM* 65.4 (2022), pp. 44–45 (cit. on p. 3).

[Ott22b] Otto, Boris: "The Evolution of Data Spaces". In: *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*. Springer International Publishing Cham, 2022, pp. 3–15 (cit. on p. 2).

[Ott23] Ott, Simon; Kamhuber, Monika; Pecholt, Joana and Wessel, Sascha: "Universal Remote Attestation for Cloud and Edge Platforms". In: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. Benevento Italy: ACM, Aug. 29, 2023, pp. 1–11. doi: 10.1145/3600160.3600171 (cit. on pp. 148, 149, 286).

[Par02] Park, Jaehong and Sandhu, Ravi: "Towards Usage Control Models: Beyond Traditional Access Control". In: *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*. 2002, pp. 57–64 (cit. on pp. 5, 17).

[Par04] Park, Jaehong and Sandhu, Ravi: "The UCONABC Usage Control Model". In: *ACM transactions on information and system security (TISSEC)* 7.1 (2004), pp. 128–174 (cit. on pp. 17, 18).

[Par11] Parno, Bryan; Lorch, Jacob R.; Douceur, John R.; Mickens, James and McCune, Jonathan M.: "Memoir: Practical State Continuity for Protected Modules". In: *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 379–394 (cit. on p. 108).

[Pat10] Patni, Harshal Kamlesh; Sahoo, Satya S.; Henson, Cory Andrew and Sheth, Amit P.: "Provenance Aware Linked Sensor Data". In: *2nd Workshop on Trust and Privacy on the Social and Semantic Web*. 2010 (cit. on p. 31).

[Pet22] Pettenpohl, Heinrich; Spiekermann, Markus and Both, Jan Ruben: "International Data Spaces in a Nutshell". In: *Designing Data Spaces* (2022), p. 29 (cit. on pp. 2, 3).

[Pin19] Pinto, Sandro and Santos, Nuno: "Demystifying Arm Trustzone: A Comprehensive Survey". In: *ACM computing surveys (CSUR)* 51.6 (2019), pp. 1–36 (cit. on pp. 41, 42).

[Pre06] Pretschner, Alexander; Hilty, Manuel and Basin, David: "Distributed Usage Control". In: *Communications of the ACM* 49.9 (2006), pp. 39–44 (cit. on pp. 5, 18).

[Pre07] Pretschner, Alexander; Massacci, Fabio and Hilty, Manuel: "Usage Control in Service-Oriented Architectures". In: *International Conference on Trust, Privacy and Security in Digital Business*. Springer. 2007, pp. 83–93 (cit. on p. 20).

[Pre08] Pretschner, Alexander; Hilty, Manuel; Basin, David; Schaefer, Christian and Walter, Thomas: "Mechanisms for Usage Control". In: *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*. 2008, pp. 240–244 (cit. on p. 20).

[Pre09a] Pretschner, Alexander: "An Overview of Distributed Usage Control". In: *2nd Conf. Knowledge Engineering: Principles and Techniques*. 2009 (cit. on p. 19).

[Pre09b] Pretschner, Alexander; Rüesch, Judith; Schaefer, Christian and Walter, Thomas: "Formal Analyses of Usage Control Policies". In: *2009 International Conference on Availability, Reliability and Security*. IEEE. 2009, pp. 98–105 (cit. on p. 20).

[Pre11] Pretschner, Alexander; Lovat, Enrico and Büchler, Matthias: “Representation-Independent Data Usage Control”. In: *Data Privacy Management and Autonomous Spontaneus Security*. Springer, 2011, pp. 122–140 (cit. on pp. 20, 64).

[Qiu19] Qiu, Pengfei; Wang, Dongsheng; Lyu, Yongqiang and Qu, Gang: “VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults”. In: *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2019, pp. 1–6 (cit. on p. 244).

[Qur21] Qureshi, Mahmood Azhar and Munir, Arslan: “PUF-RAKE: A PUF-based Robust and Lightweight Authentication and Key Establishment Protocol”. In: *IEEE Transactions on Dependable and Secure Computing* 19.4 (2021), pp. 2457–2475 (cit. on p. 47).

[Raj16] Raj, Himanshu; Saroiu, Stefan; Wolman, Alec; Aigner, Ronald; Cox, Jeremiah; England, Paul; Fenner, Chris; Kinshumann, Kinshuman; Loeser, Jork; Mattoon, Dennis et al.: “fTPM: A Software-Only Implementation of a TPM Chip”. In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 841–856 (cit. on pp. 135, 145, 163–165, 246).

[Ram18] Ramachandran, Aravind and Kantarcioglu, Murat: “Smartprovenance: A Distributed, Blockchain Based Dataprovenance System”. In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 35–42 (cit. on p. 52).

[Rao22] Rao, Anil: Rising to the Challenge - Data Security with Intel Confidential Computing. Jan. 20, 2022. url: https://community.intel.com/t5/Blogs/Products-and-Solutions/Security/Rising-to-the-Challenge-Data-Security-with-Intel-Confidential/post/1353141 (visited on 11/06/2023) (cit. on p. 40).

[Res18] Rescorla, Eric: The Transport Layer Security (TLS) Protocol Version 1.3. Aug. 2018. url: https://datatracker.ietf.org/doc/html/rfc8446.html (visited on 06/23/2023) (cit. on p. 122).

[Rie07] Ries, Sebastian: “Certain Trust: A Trust Model for Users and Agents”. In: *Proceedings of the 2007 ACM Symposium on Applied Computing*. 2007, pp. 1599–1604 (cit. on p. 249).

[Rie08a] Ries, Sebastian and Heinemann, Andreas: “Analyzing the Robustness of CertainTrust”. In: *Trust Management II*. Ed. by Karabulut, Yücel; Mitchell, John; Herrmann, Peter and Jensen, Christian Damsgaard. Vol. 263. Boston, MA: Springer US, 2008, pp. 51–67. doi: 10.1007/978-0-387-09428-1_4 (cit. on p. 240).

[Rie08b] Ries, Sebastian and Schreiber, Daniel: “Evaluating User Representations for the Trustworthiness of Interaction Partners”. In: *ReColl Workshop at IUI*. Vol. 8. 2008. url: https://fileserver.tk.informatik.tu-darmstadt.de/Publications/2008/ReColl(final).pdf (visited on 09/26/2023) (cit. on p. 239).

[Rie09] Ries, Sebastian: “Extending Bayesian Trust Models Regarding Context-Dependence and User Friendly Representation”. In: *Proceedings of the 2009 ACM Symposium on Applied Computing*. 2009, pp. 1294–1301 (cit. on p. 240).

[Rie11] Ries, Sebastian; Habib, Sheikh Mahbub; Mühlhäuser, Max and Varadharajan, Vijay: “Certainlogic: A Logic for Modeling Trust and Uncertainty”. In: *International Conference on Trust and Trustworthy Computing*. Springer. 2011, pp. 254–261 (cit. on p. 240).

[Riz19] Rizos, Athanasios; Bastos, Daniel; Saracino, Andrea and Martinelli, Fabio: “Distributed UCON in CoAP and MQTT Protocols”. In: *Computer Security*. Springer, 2019, pp. 35–52 (cit. on p. 21).

[Roe22] Roeder, Tom: A Formal Analysis of EKEP. 2022. url: https://github.com/google/ekep-analysis/blob/main/README.md (visited on 06/29/2023) (cit. on pp. 131, 149, 161, 245, 364–366).

[Row01] Rowe, Gene and Wright, George: "Expert Opinions in Forecasting: The Role of the Delphi Technique". In: *Principles of forecasting: A handbook for researchers and practitioners* (2001), pp. 125–144 (cit. on p. 232).

[Sah19] Sahoo, Satya S.; Valdez, Joshua; Kim, Matthew; Rueschman, Michael and Redline, Susan: "ProvCaRe: Characterizing Scientific Reproducibility of Biomedical Research Studies Using Semantic Provenance Metadata". In: *International journal of medical informatics* 121 (2019), pp. 10–18 (cit. on p. 28).

[Sai04] Sailer, Reiner; Zhang, Xiaolan; Jaeger, Trent and Van Doorn, Leendert: "Design and Implementation of a TCG-based Integrity Measurement Architecture." In: *USENIX Security Symposium*. Vol. 13. 2004, pp. 223–238 (cit. on pp. 110, 116).

[Sca18] Scarlata, Vinnie; Johnson, Simon; Beaney, James and Zmijewski, Piotr: "Supporting Third Party Attestation for Intel SGX with Intel Data Center Attestation Primitives". In: *White paper* (2018). url: https://www.intel.com/content/dam/develop/external/us/en/documents/intel-sgx-support-for-third-party-attestation-801017.pdf (visited on 03/29/2023) (cit. on p. 130).

[Sch11] Schulz, Steffen; Sadeghi, Ahmad-Reza and Wachsmann, Christian: "Short Paper: Lightweight Remote Attestation Using Physical Functions". In: *Proceedings of the Fourth ACM Conference on Wireless Network Security*. 2011, pp. 109–114 (cit. on p. 47).

[Sch16] Schear, Nabil; Cable, Patrick T.; Moyer, Thomas M.; Richard, Bryan and Rudd, Robert: "Bootstrapping and Maintaining Trust in the Cloud". In: *Proceedings of the 32Nd Annual Conference on Computer Security Applications*. 2016, pp. 65–77 (cit. on p. 73).

[Sch17] Schwarz, Michael; Weiser, Samuel; Gruss, Daniel; Maurice, Clémentine and Mangard, Stefan: "Malware Guard Extension:

Using SGX to Conceal Cache Attacks". In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer. 2017, pp. 3–24 (cit. on p. 243).

[Sch18a] Schütte, Julian; Brost, Gerd and Wessel, Sascha: "Der Trusted Connector im Industrial Data Space". 2018. arXiv: 1804.09442 (cit. on pp. 3, 7, 27).

[Sch18b] Schütte, Julian and Brost, Gerd Stefan: "LUCON: Data Flow Control for Message-Based IoT Systems". In: *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 289–299 (cit. on p. 27).

[Sch22] Schneider, Moritz; Masti, Ramya Jayaram; Shinde, Shweta; Capkun, Srdjan and Perez, Ronald: "Sok: Hardware-supported Trusted Execution Environments". 2022. arXiv: 2205.12742 (cit. on pp. 8, 33, 38).

[Seg16] Segall, Ariel: Trusted Platform Modules: Why, When and How to Use Them. IET, 2016 (cit. on pp. 108, 114, 121, 183).

[Seo17] Seo, Jaebaek; Lee, Byoungyoung; Kim, Seong Min; Shih, Ming-Wei; Shin, Insik; Han, Dongsu and Kim, Taesoo: "SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs." In: *NDSS*. 2017. url: https://gts3.org/assets/papers/2017/seo:sgx-shield.pdf (visited on 10/01/2023) (cit. on p. 245).

[Ses04] Seshadri, Arvind; Perrig, Adrian; Van Doorn, Leendert and Khosla, Pradeep: "SWATT: Software-based Attestation for Embedded Devices". In: *IEEE Symposium on Security and Privacy*. IEEE, 2004, pp. 272–282 (cit. on p. 47).

[She17] Shepherd, Carlton; Akram, Raja Naeem and Markantonakis, Konstantinos: "Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security*. 2017, pp. 1–10 (cit. on pp. 146, 147).

[She21] Shepherd, Carlton; Markantonakis, Konstantinos and Jaloyan, Georges-Axel: "LIRA-V: Lightweight Remote Attestation for Constrained RISC-V Devices". In: *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 221–227 (cit. on p. 44).

[Shi17] Shih, Ming-Wei; Lee, Sangho; Kim, Taesoo and Peinado, Marcus: "T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs." In: *NDSS*. 2017. url: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_07-2_Shih_paper.pdf (visited on 10/01/2023) (cit. on p. 245).

[Sig19] Sigwart, Marten; Borkowski, Michael; Peise, Marco; Schulte, Stefan and Tai, Stefan: "Blockchain-Based Data Provenance for the Internet of Things". In: *Proceedings of the 9th International Conference on the Internet of Things*. 2019, pp. 1–8 (cit. on p. 52).

[Son22] Song, Liantao; Ding, Yan; Dong, Pan; Guo, Yong and Wang, Chuang: "TZ-IMA: Supporting Integrity Measurement for Applications with ARM TrustZone". In: *International Conference on Information and Communications Security*. Springer, 2022, pp. 342–358 (cit. on p. 141).

[Sou19] Souza, Renan; Azevedo, Leonardo; Lourenço, Vítor; Soares, Elton; Thiago, Raphael; Brandao, Rafael; Civitarese, Daniel; Brazil, Emilio; Moreno, Marcio and Valduriez, Patrick: "Provenance Data in the Machine Learning Lifecycle in Computational Science and Engineering". In: *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 2019, pp. 1–10 (cit. on p. 28).

[Ste00] Stephenson, Todd Andrew: An Introduction to Bayesian Network Theory and Usage. Idiap, 2000. url: https://infoscience.epfl.ch/record/82584 (visited on 09/23/2023) (cit. on p. 233).

[Ste19] Steinbuss, Sebastian: Framework for the IDS Certification Scheme. Whitepaper Version 2. International Data Spaces Association, 2019. url: https://internationaldataspaces.org/

wp-content/uploads/dlm_uploads/IDSA-White-Paper-certification-scheme-V.2.pdf (visited on 04/09/2023) (cit. on p. 50).

[Ste21] Steinbuss, Sebastian: Usage Control in the International Data Spaces. International Data Spaces Association, 2021. url: https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-Paper-Usage-Control-in-the-IDS-V3..pdf (visited on 08/11/2022) (cit. on pp. 5, 6, 26, 28).

[Str14] Strackx, Raoul; Jacobs, Bart and Piessens, Frank: "ICE: A Passive, High-Speed, State-Continuity Scheme". In: *Proceedings of the 30th Annual Computer Security Applications Conference.* 2014, pp. 106–115 (cit. on pp. 109, 128).

[Str16] Strackx, Raoul and Piessens, Frank: "Ariadne: A Minimal Approach to State Continuity". In: *USENIX Security.* Vol. 16. 2016 (cit. on pp. 109, 128).

[Stu06] Stumpf, Frederic; Tafreschi, Omid; Röder, Patrick; Eckert, Claudia et al.: "A Robust Integrity Reporting Protocol for Remote Attestation". In: *Proceedings of the Workshop on Advances in Trusted Computing (WATC).* 2006, p. 65 (cit. on pp. 110, 113, 114, 116, 123).

[Stu08] Stumpf, Frederic; Fuchs, Andreas; Katzenbeisser, Stefan and Eckert, Claudia: "Improving the Scalability of Platform Attestation". In: *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing.* 2008, pp. 1–10 (cit. on pp. 110, 113, 114, 116, 123).

[Tah15] Taha, Mohammad M. Bany; Chaisiri, Sivadon and Ko, Ryan KL: "Trusted Tamper-Evident Data Provenance". In: *2015 IEEE Trustcom/Bigdatase/Ispa.* Vol. 1. IEEE, 2015, pp. 646–653 (cit. on p. 55).

[Tan19] Tank, Darshan; Aggarwal, Akshai and Chaubey, Nirbhay: "Virtualization Vulnerabilities, Security Issues, and Solutions: A Critical Study and Comparison". In: *International Journal of Information Technology* (2019), pp. 1–16 (cit. on p. 32).

[Tao21] Tao, Zhe; Rastogi, Aseem; Gupta, Naman; Vaswani, Kapil and Thakur, Aditya V.: "DICE*: A Formally Verified Implementation of DICE Measured Boot." In: *USENIX Security Symposium.* 2021, pp. 1091–1107 (cit. on p. 286).

[Tar22] Tardieu, Hubert: "Role of Gaia-X in the European Data Space Ecosystem". In: *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage.* Springer International Publishing Cham, 2022, pp. 41–59 (cit. on p. 3).

[Ten14] Ten Hompel, Michael and Henke, Michael: "Logistik 4.0". In: *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung· Technologien· Migration.* Springer, 2014, pp. 615–624 (cit. on p. 1).

[Tho17] Thoben, Klaus-Dieter; Wiesner, Stefan and Wuest, Thorsten: "Industrie 4.0 and Smart Manufacturing - A Review of Research Issues and Application Examples". In: *International journal of automation technology* 11.1 (2017), pp. 4–16 (cit. on p. 2).

[Tre17] Trenkle, Andreas and Furmans, Kai: "Der Mensch als Teil von Industrie 4.0: Interaktionsmechanismen bei autonomen Materialflusssystemen". In: *Handbuch Industrie 4.0 Bd. 3: Logistik* (2017), pp. 45–59 (cit. on p. 2).

[Tru16a] Truong, Nguyen B.; Cao, Quyet H.; Um, Tai-Won and Lee, Gyu Myoung: "Leverage a Trust Service Platform for Data Usage Control in Smart City". In: *2016 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2016, pp. 1–7 (cit. on pp. 51, 214).

[Tru16b] Trusted Computing Group: TCG EFI Protocol Specification. Revision 00.13. Mar. 30, 2016. url: https://trustedcomputinggroup.org/wp-content/uploads/EFI-Protocol-Specification-rev13-160330final.pdf (visited on 07/09/2023) (cit. on pp. 140, 141).

[Tru19a] Trusted Computing Group: TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0. Revision 0.36. Sept. 3, 2019. url: https://trustedcomputinggroup.org/wp-content/uploads/TNC_TAP_Information_Model_v1.00_r0.36-FINAL.pdf (visited on 03/28/2023) (cit. on p. 109).

[Tru19b] Trusted Computing Group: Trusted Platform Module Library Part 1: Architecture. Revision 01.59. Nov. 8, 2019. url: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf (visited on 03/28/2023) (cit. on pp. 32–35, 119).

[Tru19c] Trusted Computing Group: Trusted Platform Module Library Part 2: Structures. Revision 01.59. Nov. 8, 2019. url: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part2_Structures_pub.pdf (visited on 03/28/2023) (cit. on pp. 106, 117–119).

[Tru19d] Trusted Computing Group: Trusted Platform Module Library Part 3: Commands. Revision 01.59. Nov. 8, 2019. url: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part3_Commands_pub.pdf (visited on 03/28/2023) (cit. on pp. 106, 117, 118, 166).

[Tru20] Trusted Computing Group: TCG PC Client Platform TPM Profile Specification for TPM 2.0. Revision 14. Sept. 4, 2020. url: https://trustedcomputinggroup.org/wp-content/uploads/PC-Client-Specific-Platform-TPM-Profile-for-TPM-2p0-v1p05p_r14_pub.pdf (visited on 07/09/2023) (cit. on pp. 123, 156).

[Tru21a] Trusted Computing Group: TCG PC Client Platform Firmware Profile Specification. Revision 23. May 7, 2021. url: https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClient_PFP_r1p05_v23_pub.pdf (visited on 07/10/2023) (cit. on p. 141).

[Tru21b] Trusted Computing Group: TPM 2.0 Keys for Device Identity and Attestation. Revision 12. Oct. 8, 2021. url: https://trustedcomputinggroup.org/wp-content/uploads/TPM-2p0-Keys-for-Device-Identity-and-Attestation_v1_r12_pub10082021.pdf (visited on 06/26/2023) (cit. on p. 117).

[Tsu20] Tsutsumi, Daisuke; Gyulai, Dávid; Takács, Emma; Bergmann, Júlia; Nonaka, Youichi and Fujita, Kikuo: "Personalized Work Instruction System for Revitalizing Human-Machine Interaction". In: *Procedia CIRP* 93 (2020), pp. 1145–1150 (cit. on p. 251).

[Ujc18] Ujcich, Benjamin E.; Bates, Adam and Sanders, William H.: "A Provenance Model for the European Union General Data Protection Regulation". In: *International Provenance and Annotation Workshop*. Springer, 2018, pp. 45–57 (cit. on pp. 6, 28, 30).

[Usl22] Usländer, Thomas and Teuscher, Andreas: "Industrial Data Spaces". In: *Designing Data Spaces* (2022), p. 313 (cit. on p. 4).

[Van17a] Van Bulck, Jo; Piessens, Frank and Strackx, Raoul: "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control". In: *Proceedings of the 2nd Workshop on System Software for Trusted Execution*. SOSP '17: ACM SIGOPS 26th Symposium on Operating Systems Principles. Shanghai China: ACM, Oct. 28, 2017, pp. 1–6. doi: 10.1145/3152701.3152706 (cit. on p. 244).

[Van17b] Van Bulck, Jo; Weichbrodt, Nico; Kapitza, Rüdiger; Piessens, Frank and Strackx, Raoul: "Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution". In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 1041–1056 (cit. on p. 244).

[Van18] Van Bulck, Jo; Minkin, Marina; Weisse, Ofir; Genkin, Daniel; Kasikci, Baris; Piessens, Frank; Silberstein, Mark; Wenisch, Thomas F; Yarom, Yuval and Strackx, Raoul: "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient out-of-Order Execution”. In: *27th USENIX Security Symposium (USENIX Security 18).* 2018, pp. 991–1008 (cit. on p. 244).

[Wag18a] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Distributed Usage Control Enforcement through Trusted Platform Modules and SGX Enclaves”. In: *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018, Indianapolis, IN, USA, June 13-15, 2018.* Association for Computing Machinery (ACM), 2018, pp. 85–91. doi: 10.1145/3205977.3205990 (cit. on p. 103).

[Wag18b] Wagner, Steffen: “Implicit Remote Attestation for Microkernel-Based Embedded Systems”. Technische Universität München, 2018 (cit. on pp. 32, 33, 37).

[Wag19a] Wagner, Paul Georg: “Towards a Formal Model for Quantifying Trust in Distributed Usage Control Systems”. In: *Proceedings of the 2019 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.* Vol. 45. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2019, pp. 113–131 (cit. on p. 210).

[Wag19b] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Challenges of Using Trusted Computing for Collaborative Data Processing”. In: *Security and Trust Management.* Ed. by Mauw, Sjouke and Conti, Mauro. Cham: Springer International Publishing, 2019, pp. 107–123. doi: 10.1007/978-3-030-31511-5_7 (cit. on pp. 7, 49).

[Wag20] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Establishing Secure Communication Channels Using Remote Attestation with TPM 2.0”. In: *Security and Trust Management.* Ed. by Markantonakis, Kostantinos and Petrocchi, Marinella. Cham: Springer International Publishing, 2020, pp. 73–89. doi: 10.1007/978-3-030-59817-4_5 (cit. on pp. 7, 103, 105).

[Wag21a] Wagner, Paul Georg: “Classifying Usage Control and Data Provenance Architectures”. In: *Proceedings of the 2020 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.* Vol. 51. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2021, pp. 135–154 (cit. on p. 49).

[Wag21b] Wagner, Paul Georg; Lengenfelder, Christian; Holzbach, Gerrit; Becker, Maximilian; Birnstill, Pascal; Voit, Michael; Bejhad, Ali; Samorei, Tim and Beyerer, Jürgen: “Secure and Privacy-Respecting Documentation for Interactive Manufacturing and Quality Assurance”. In: *Applied Sciences* 11.16 (2021), Art.–Nr.: 7339. doi: 10.3390/app11167339 (cit. on pp. 4, 251, 252).

[Wag22a] Wagner, Paul Georg: “Conceptualization of a Trust Dashboard for Distributed Usage Control Systems”. In: *Proceedings of the 2021 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.* Vol. 54. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2022, pp. 169–188 (cit. on p. 210).

[Wag22b] Wagner, Paul Georg and Beyerer, Jürgen: “Quantifying Trustworthiness in Decentralized Trusted Applications”. In: *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (Sat-CPS’22).* Association for Computing Machinery (ACM), 2022, pp. 67–76. doi: 10.1145/3510547.3517930 (cit. on pp. 210, 214, 220).

[Wag22c] Wagner, Paul Georg and Beyerer, Jürgen: “Towards Heterogeneous Remote Attestation Protocols”. In: *Proceedings of the 19th International Conference on Security and Cryptography, SECRYPT 2022, Lisbon, Portugal, 11th - 13th July 2022.* Vol. 1. SciTePress, 2022, pp. 586–591. doi: 10.5220/0011289000003283 (cit. on p. 103).

[Wal22] Walther, Robert; Weinhold, Carsten and Roitzsch, Michael: “RATLS: Integrating Transport Layer Security with Remote

Attestation”. In: *Applied Cryptography and Network Security Workshops: ACNS 2022 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, SiMLA, Rome, Italy, June 20–23, 2022, Proceedings.* Springer, 2022, pp. 361–379 (cit. on pp. 112, 113, 116).

[Wan17] Wang, Wenhao; Chen, Guoxing; Pan, Xiaorui; Zhang, Yinqian; Wang, XiaoFeng; Bindschaedler, Vincent; Tang, Haixu and Gunter, Carl A.: “Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* CCS ’17. ACM, Oct. 30, 2017, pp. 2421–2434. doi: 10.1145/3133956.3134038 (cit. on p. 244).

[Wan20] Wang, Ziwang; Zhuang, Yi and Yan, Zujia: “TZ-MRAS: A Remote Attestation Scheme for the Mobile Terminal Based on ARM TrustZone”. In: *Security and Communication Networks* 2020 (2020) (cit. on pp. 46, 48, 146, 147).

[Wat17] Waterman, Andrew and Asanović, Krste: The RISC-V Instruction Set Manual Volume II: Privileged Architecture. Version 1.10. May 7, 2017. url: https://riscv.org/wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf (visited on 04/01/2023) (cit. on p. 43).

[Wei18] Weisse, Ofir; Bulck, Jo Van; Minkin, Marina; Genkin, Daniel; Kasikci, Baris; Piessens, Frank; Silberstein, Mark; Strackx, Raoul; Wenisch, Thomas F and Yarom, Yuval: Foreshadow-NG: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution. Revision 1.0. Aug. 14, 2018. url: https://foreshadowattack.eu/foreshadow-NG.pdf (visited on 10/01/2023) (cit. on p. 245).

[Wei19] Weiser, Samuel; Werner, Mario; Brasser, Ferdinand; Malenko, Maja; Mangard, Stefan and Sadeghi, Ahmad-Reza: “Timber-V: Tag-isolated Memory Bringing Fine-Grained Enclaves to RISC-V”. In: *NDSS.* 2019 (cit. on p. 44).

[Wer22] Werder, Karl; Ramesh, Balasubramaniam and Zhang, Rongen: “Establishing Data Provenance for Responsible Artificial Intelligence Systems”. In: *ACM Transactions on Management Information Systems (TMIS)* 13.2 (2022), pp. 1–23 (cit. on p. 28).

[Win21] Winkler, Dietmar; Korobeinykov, Alexander; Novák, Petr; Lüder, Arndt and Biffl, Stefan: “Big Data Needs and Challenges in Smart Manufacturing: An Industry-Academia Survey”. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2021, pp. 1–8 (cit. on pp. 1, 251).

[Won17] Wong, Kok-Seng and Kim, Myung Ho: “Privacy Protection for Data-Driven Smart Manufacturing Systems”. In: *International Journal of Web Services Research (IJWSR)* 14.3 (2017), pp. 17–32 (cit. on p. 252).

[Wüc12] Wüchner, Tobias and Pretschner, Alexander: “Data Loss Prevention Based on Data-Driven Usage Control”. In: *2012 IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, 2012, pp. 151–160 (cit. on p. 25).

[Xu15] Xu, Yuanzhong; Cui, Weidong and Peinado, Marcus: “Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems”. In: *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 640–656 (cit. on p. 244).

[Yan09] Yang, Li and Cemerlic, Alma: “Integrating Dirichlet Reputation into Usage Control”. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. 2009, pp. 1–4 (cit. on pp. 51, 212, 214).

[Yan20] Yan, Zheng; Govindaraju, Venu; Zheng, Qinghua and Wang, Yan: “IEEE Access Special Section Editorial: Trusted Computing”. In: *IEEE Access* 8 (2020), pp. 25722–25726 (cit. on p. 7).

[Yav22] Yavuz, Tuba; Fowze, Farhaan; Hernandez, Grant; Bai, Ken Yihang; Butler, Kevin RB and Tian, Dave Jing: “ENCIDER: Detecting Timing and Cache Side Channels in SGX Enclaves and Cryptographic APIs”. In: *IEEE Transactions on Dependable and Secure Computing* 20.2 (2022), pp. 1577–1595 (cit. on p. 245).

[Yoo22] Yoon, HanJae and Lee, ManHee: “SGXDump: A Repeatable Code-Reuse Attack for Extracting SGX Enclave Memory”. In: *Applied Sciences* 12.15 (2022), p. 7655 (cit. on pp. 244, 245).

[Zaf17] Zafar, Faheem; Khan, Abid; Suhail, Saba; Ahmed, Idrees; Hameed, Khizar; Khan, Hayat Mohammad; Jabeen, Farhana and Anjum, Adeel: “Trustworthy Data: A Survey, Taxonomy and Future Trends of Secure Provenance Schemes”. In: *Journal of network and computer applications* 94 (2017), pp. 50–68 (cit. on pp. 5, 31).

[Zha08] Zhang, Xinwen; Seifert, Jean-Pierre and Sandhu, Ravi: “Security Enforcement Model for Distributed Usage Control”. In: *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (Sutc 2008).* IEEE. 2008, pp. 10–18 (cit. on pp. 7, 53, 65).

[Zho10] Zhou, Lingli and Zhang, Zhenfeng: “Trusted Channels with Password-Based Authentication and TPM-based Attestation”. In: *2010 International Conference on Communications and Mobile Computing*. Vol. 1. IEEE. 2010, pp. 223–227 (cit. on pp. 112–114, 116).

Own Publications

This section includes a complete list of all own publications. Relating to the research contributions presented in this dissertation, the publications [9] and [11] deal with challenges and security requirements of distributed usage control systems. The publications [7], [10], and [15] contain previous work concerning remote attestation protocols and trusted computing technologies. Finally, the publications [8], [13], and [14] provide preparatory work regarding the trustworthiness estimation of distributed usage control systems. The remaining publications only superficially pertain to the thesis at hand.

[1] Pickhardt, Rene; Gottron, Thomas; Körner, Martin; Wagner, Paul Georg; Speicher, Till and Staab, Steffen: “A Generalized Language Model as the Combination of Skipped N-Grams and Modified Kneser-Ney Smoothing”. 2014. arXiv: 1404.3377.

[2] Wagner, Paul Georg; Birnstill, Pascal; Krempel, Erik; Bretthauer, Sebastian and Beyerer, Jürgen: “Privacy-Dashcam – Datenschutzfreundliche Dashcams durch Erzwingen externer Anonymisierung”. In: *Lecture Notes in Informatics (LNI).* Informatik 2016. Klagenfurt, Austria: Gesellschaft für Informatik e.V. (GI), 2016, pp. 427–440.

[3] Wagner, Paul; Birnstill, Pascal; Krempel, Erik; Bretthauer, Sebastian and Beyerer, Jürgen: “Privacy Dashcam – Towards Lawful Use of Dashcams Through Enforcement of External Anonymization”. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Ed. by Garcia-Alfaro, Joaquin; Navarro-Arribas, Guillermo; Hartenstein, Hannes and Herrera-Joancomartí, Jordi. Cham: Springer International Publishing, 2017, pp. 183–201. doi: 10.1007/978-3-319-67816-0_11.

[4] Wagner, Paul Georg; Birnstill, Pascal; Krempel, Erik and Bretthauer, Sebastian: “Auf dem Weg zu datenschutzfreundlichen Dashcams”. In: *Datenschutz und Datensicherheit* 41.3 (2017), pp. 159–164.

[5] Birnstill, Pascal; Bier, Christoph; Wagner, Paul and Beyerer, Jürgen: “Generic Semantics Specification and Processing for Inter-System Information Flow Tracking”. In: *Computer and Network Security Essentials.* Ed. by Daimi, Kevin. Cham: Springer International Publishing, 2018, pp. 445–460. doi: 10.1007/978-3-319-58424-9_25.

[6] Birnstill, Pascal; Krempel, Erik; Wagner, Paul Georg and Beyerer, Jürgen: “Identity Management and Protection Motivated by the General Data Protection Regulation of the European Union - A Conceptual Framework Based on State-of-the-Art Software Technologies”. In: *Technologies* 6.4 (2018), pp. 115/1–14. doi: 10.3390/technologies6040115.

[7] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Distributed Usage Control Enforcement through Trusted Platform Modules and SGX Enclaves”. In: *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, SACMAT 2018, Indianapolis, IN, USA, June 13-15, 2018.* Association for Computing Machinery (ACM), 2018, pp. 85–91. doi: 10.1145/3205977.3205990.

[8] Wagner, Paul Georg: “Towards a Formal Model for Quantifying Trust in Distributed Usage Control Systems”. In: *Proceedings of the 2019 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.* Vol. 45. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2019, pp. 113–131.

[9] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Challenges of Using Trusted Computing for Collaborative Data Processing”. In: *Security and Trust Management.* Ed. by Mauw, Sjouke and Conti, Mauro. Cham: Springer International Publishing, 2019, pp. 107–123. doi: 10.1007/978-3-030-31511-5_7.

[10] Wagner, Paul Georg; Birnstill, Pascal and Beyerer, Jürgen: “Establishing Secure Communication Channels Using Remote Attestation with TPM 2.0”. In: *Security and Trust Management.* Ed. by

Markantonakis, Kostantinos and Petrocchi, Marinella. Cham: Springer International Publishing, 2020, pp. 73–89. doi: 10.1007/978-3-030-59817-4_5.

[11] Wagner, Paul Georg: "Classifying Usage Control and Data Provenance Architectures". In: *Proceedings of the 2020 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. Vol. 51. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2021, pp. 135–154.

[12] Wagner, Paul Georg; Lengenfelder, Christian; Holzbach, Gerrit; Becker, Maximilian; Birnstill, Pascal; Voit, Michael; Bejhad, Ali; Samorei, Tim and Beyerer, Jürgen: "Secure and Privacy-Respecting Documentation for Interactive Manufacturing and Quality Assurance". In: *Applied Sciences* 11.16 (2021), Art.–Nr.: 7339. doi: 10.3390/app11167339.

[13] Wagner, Paul Georg: "Conceptualization of a Trust Dashboard for Distributed Usage Control Systems". In: *Proceedings of the 2021 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. Vol. 54. Karlsruher Schriften Zur Anthropomatik. KIT Scientific Publishing, 2022, pp. 169–188.

[14] Wagner, Paul Georg and Beyerer, Jürgen: "Quantifying Trustworthiness in Decentralized Trusted Applications". In: *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (Sat-CPS'22)*. Association for Computing Machinery (ACM), 2022, pp. 67–76. doi: 10.1145/3510547.3517930.

[15] Wagner, Paul Georg and Beyerer, Jürgen: "Towards Heterogeneous Remote Attestation Protocols". In: *Proceedings of the 19th International Conference on Security and Cryptography, SECRYPT 2022, Lisbon, Portugal, 11th - 13th July 2022*. Vol. 1. SciTePress, 2022, pp. 586–591. doi: 10.5220/0011289000003283.

Supervised Student Theses

This section includes a complete list of all student theses supervised during the course of this dissertation. Of those, [2] and [5] deal with conducting integrity measurements on ARM TrustZone platforms. Furthermore, [4] is concerned with the application of distributed usage control in cloud environments, while [3], [7], and [10] examine remote attestation techniques. The remaining works only superficially pertain to the dissertation at hand.

[1] Samorei, Tim Philipp: “SCrypt – SmartCard-Emulation und PKCS11-Schnittstelle für NFC-fähige Smartphones”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2019.

[2] Hartstern, Daniel: “RAffT: A Remote Attestation Library for Firmware-level Trusted Platform Modules”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2020.

[3] Naseband, Clemens: “TPM-basierte Remote Attestation eines Videodokumentationssystems über NFC”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2020.

[4] Schuler, Nicolas: “Verteilte Nutzungskontrolle und Provenance Tracking am Beispiel von Cloud-Technologien”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2020.

[5] Amblank, Roman: “Both-World Measured Boot Architecture on Arm TrustZone Devices with Firmware-TPMs”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2021.

[6] Riekert, Thomas: “Datenschutzgerechte Dokumentation kritischer Arbeitsschritte mit Intel SGX”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2022.

[7] Heine, Jonas: “Absicherung von erklärbarer künstlicher Intelligenz durch TPM-basierte Attestierung”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2023.

[8] Kara, Kerem: “Integrity Measurement for CI/CD Build Processes Based on Trusted Platform Modules”. Bachelor’s Thesis. Karlsruhe Institute of Technology (KIT), 2023.

[9] Saeed, Haris: “Development of a Trusted Continuous Integration Infrastructure Based on Trusted Platform Module (TPM)”. Master’s Thesis. University of Applied Sciences Karlsruhe (HKA), 2023.

[10] Samorei, Tim Philipp: “Integration von Remote Attestation in DDS und ROS2 mittels Trusted Platform Modules”. Master’s Thesis. Karlsruhe Institute of Technology (KIT), 2024.

List of Figures

List of Tables

Acronyms

ABAC	Attribute-based Access Control
ACL	Access Control List
AES	Advanced Encryption Standard
AK	Attestation Key
ALTS	Application Layer Transport Security
API	Application Programming Interface
CA	Certification Authority
CCA	Confidential Computing Architecture
CoT	Chain of Trust
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurement
CSR	Certificate Signing Request
DCAP	Data Center Attestation Primitives
DHKE	Diffie-Hellman Key Exchange
DICE	Device Identifier Composition Engine
DRM	Digital Rights Management

DSL	Domain Specific Language
DUC	Distributed Usage Control
ECA	Event-Condition-Action
ECDH	Elliptic Curve Diffie-Hellman
EKEP	Enclave Key Exchange Protocol
eMMC	Embedded Multi Media Card
EPID	Enhanced Privacy ID
fTPM	Firmware-Level Trusted Platform Module
HMAC	Hash-based Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
IAS	Intel Attestation Service
ID	Identifier
IDS	International Data Space
IDSCP	International Data Space Communication Protocol
IMA	Integrity Measurement Architecture
IoT	Internet of Things
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
LUCON	Logic-Based Usage Control

MSCP	Mutually-Attested Secure Communication Protocol
M-Store	Measurement Store
NIST	National Institute of Standards and Technology
NV Storage	Non-volatile Storage
OASIS	Organization for the Advancement of Structured Information Standards
ODRL	Open Digital Rights Language
OS	Operating System
OSL	Obligation Specification Language
PAP	Policy Administration Point
PCR	Platform Configuration Register
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKI	Public Key Infrastructure
PMP	Policy Management Point
ProDP	Provenance Dissemination Point
ProSP	Provenance Storage Point
PRP	Policy Retrieval Point
PUF	Physically Unclonable Function
PXP	Policy Execution Point
RAT	Remote Attestation

RBP	Rollback Protection/Prevention
RDF	Resource Description Framework
REE	Rich Execution Environment
RPC	Remote Procedure Call
RPMB	Replay Protected Memory Block
RSA	Rivest-Shamir-Adleman
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SDK	Software Development Kit
SEV	Secure Encrypted Virtualization
SGX	Software Guard Extensions
SHA	Secure Hash Algorithm
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SRK	Storage Root Key
TA	Trusted Application
TBB	Trusted Board Boot
TCB	Trusted Computing Base
TCG	Trusted Computing Group
TDX	Trust Domain Extensions
TEE	Trusted Execution Environment

TLS	Transport Layer Security
TOCTOU	Time-of-Check Time-of-Use
TPM	Trusted Platform Module
TTP	Trusted Third Party
UC	Usage Control
UEFI	Unified Extensible Firmware Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UUID	Universally Unique Identifier
VM	Virtual Machine
W3C	World Wide Web Consortium
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language

A The IDSCP Handshake

Table A.1 summarizes the remote attestation messages of an IDSCP handshake between two TPM-protected platforms. A description of the complete IDSCP protocol is given in [Bro22]. The implementation of the remote attestation drivers for IDSCP are available on Github[1] under the Apache license.

Table A.1: The IDSCP remote attestation protocol.

	TLS handshake
$A \leftrightarrow B$:	Establish a TLS channel with certificates $cert^A$ and $cert^B$
	Initiation phase
$A \rightarrow B$:	Non-predictable nonce N^A
$A \leftarrow B$:	Non-predictable nonce N^B
	Attestation phase
A :	$(quoted^A, quoteSig^A) \leftarrow$ TPM2_Quote(ak^A, SHA1($N^B \parallel cert^B$))
$A \rightarrow B$:	$PCR^A, (quoted^A, quoteSig^A), akCert^A$
B :	$(quoted^B, quoteSig^B) \leftarrow$ TPM2_Quote(ak^B, SHA1($N^A \parallel cert^A$))
$A \leftarrow B$:	$PCR^B, (quoted^B, quoteSig^B), akCert^B$
	Verification phase
A :	Verify $(quoted^B, quoteSig^B)$ is valid under $akCert^B$
A :	Verify $quoted^B$ contains expected PCR^B and SHA1($N^A \parallel cert^A$)
B :	Verify $(quoted^A, quoteSig^A)$ is valid under $akCert^A$
B :	Verify $quoted^A$ contains expected PCR^A and SHA1($N^B \parallel cert^B$)

[1] https://github.com/industrial-data-space/idscp2-rat-drivers *(accessed on 12/08/2023).*

B Formal Protocol Verification

This chapter includes the formal models that we used to verify the attestation protocols discussed in this thesis.

B.1 The IDSCP Protocol

Listing B.1 shows our formal model of the IDSCP protocol for the Tamarin theorem prover. We model the IDSCP protocol handshake with a dedicated Diffie-Hellman key exchange between two endpoints Alice and Bob. The key exchange is performed by an (abstract) underlying TLS stack to achieve perfect forward secrecy. Our modeled theory proves that IDSCP is vulnerable against nonce-data attacks if an attestation endpoint for the standard TCG remote attestation protocol is present on one of the trusted platforms. We model this quoting oracle with the `TSS_Quote_TPMB` rule.

Listing B.1: Formal model of the IDSCP attestation protocol in Tamarin.

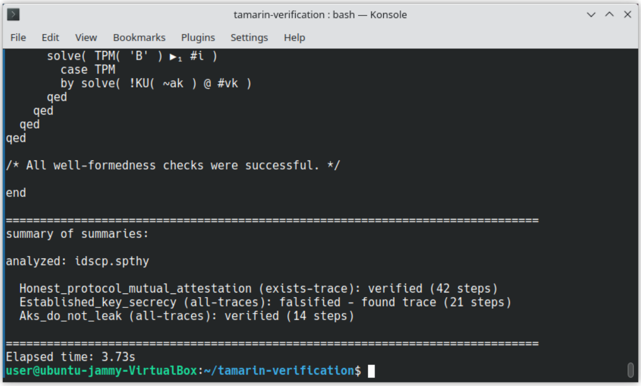
```
theory IDSCP
begin

builtins: hashing, signing, diffie-hellman, symmetric-encryption

/* Our goal is to verify the security of the key exchange protocol under the assumption
 * that the trusted platform is secure. This means that we assume the used attestation
 * public keys to be known by Alice and Bob. Furthermore we assume that the attacker
 * is unable to compromise an established attestation key or forge quote signatures. We
 * model these assumptions by creating the attestation keys once for each platform.
 *
 * However, IDSCP is vulnerable against internal attackers who have knowledge about the
 * platform secrets that are not bound to a TPM (in our case the TLS key). We model the
 * administrator of Bob's system as an internal attacker by leaking Bob's TLS key.
 */
```

```
// Define two TPMs (Alice and Bob)
rule TPM:
  []  -->  [TPM('A'), TPM('B')]

// Create a unique attestation key for each TPM
rule Get_ak:
  [ Fr(~ak), TPM(X) ]
  --[
    OnlyOnceFor(<'Get_ak', X>)
    , IsAK(X, ~ak)
  ]->
  [ Ak(X, ~ak), !AkCert(X, pk(~ak)) ]

// Create a TLS signature key for each system
rule Get_tls:
  [ Fr(~tls), TPM(X) ]
  --[
    OnlyOnceFor(<'Get_tls', X>)
  ]->
  [ !Tls(X, ~tls), !TlsCert(X, pk(~tls)) ]

// The TLS public keys are known to any attacker
rule Tls_Certs_Are_Public:
  [ !TlsCert('A', tlsCertA), !TlsCert('B', tlsCertB) ]
  -->
  [ Out(tlsCertA), Out(tlsCertB) ]

/* Since we need at least one honest player, we assume that the internal attacker knows
 * Bob's TLS signature key. This models an internal attacker who knows Bob's secrets
 * (for example the administrator of the system) and wants to intercept the messages
 * between Alice and the (trusted) Bob on the attested secure channel. For this purpose,
 * rule BobIsAnInternalAttacker leaks Bob's TLS signature key.
 */
rule BobIsAnInternalAttacker:
  [ !Tls('B', tlsB) ] --> [ Out(tlsB) ]

/* Create quotes for Bob's trusted software stack (TSS). This rule models that there is
 * some valid software on Bob's platform that acts as an attestation endpoint. It takes
 * some qualifying data (e.g. a nonce) and asks the TPM to create a signed quote.
 * Activating this rule will break the security of IDSCP!
 */
rule TSS_Quote_TPMB:
  [ In(qualifyingData)
    , Ak('B', ak)
  ]-->
  [ Out(sign(<qualifyingData, 'PCRB'>, ak)) ]

// Step 1: Ephemeral DH key exchange during TLS handshake
rule tlsDHKE_A1:
  [ Fr(~a)                                      // Choose fresh DH private key for Alice
    , !Tls('A', tlsA)                           // Lookup Alice's TLS signature key
  ]--[
    OnlyOnceFor('tlsDHKE_A1')
  ]->
  [ Out(<'DHKE1', 'g'^~a, sign('g'^~a, tlsA)>)
    , DHKE_A(~a)                               // Store Alice's DH private key
  ]

rule tlsDHKE_B:
  [ In(<'DHKE1', dhPubA, sigA>)
    , !TlsCert('A', tlsCertA)                  // Lookup Alice's TLS certificate
    , Fr(~b)                                   // Choose fresh DH private key for Bob
    , !Tls('B', tlsB)                          // Lookup Bob's TLS signature key
  ]--[
    OnlyOnceFor('tlsDHKE_B')
    , Neq(dhPubA, 'g')                         // Alice's public DH key must not be 'g'
    , Eq(verify(sigA, dhPubA, tlsCertA), true) // Verify the DHKE signature
    , EstablishedKey('B', dhPubA^~b)           // Take note of Bob's established key
  ]->
  [ Out(<'DHKE2', 'g'^~b, sign('g'^~b, tlsB)>)
    , Tls_Finish('B')                          // Bob finished the TLS handshake and
                                               // used tlsCertA for verification.
                                               // This must later be included the quote.
    , !Tls_Ephemeral_Key_B(dhPubA^~b)
  ]

rule tlsDHKE_A2:
  [ In(<'DHKE2', dhPubB, sigB>)
    , DHKE_A(~a)
    , !TlsCert('B', tlsCertB)                  // Lookup Bob's TLS certificate
  ]--[
    OnlyOnceFor('tlsDHKE_A2')
    , Neq(dhPubB, 'g')                         // Bob's public DH key must not be 'g'
    , Eq(verify(sigB, dhPubB, tlsCertB), true) // Verify the DHKE signature
    , EstablishedKey('A', dhPubB^~a)           // Take note of Alice's established key
  ]->
  [ Tls_Finish('A')                            // Alice finished the TLS handshake and
                                               // used tlsCertB for verification.
                                               // This must later be included the quote.
    , !Tls_Ephemeral_Key_A(dhPubB^~a)
  ]

/* So far only the authenticated Diffie-Hellman key exchange during the TLS handshake
 * has been conducted. Now the remote attestation protocol that binds the TLS public
 * keys to the TPM state starts.
 */

// Step 2: IdscpRaVerifier messages
rule IdscpRaVerifier_A:
  [ Tls_Finish('A')
    , !Tls_Ephemeral_Key_A(key)
    , Fr(~challengeA)                                     // Choose nonce for Alice
  ]--[
    OnlyOnceFor('IdscpRaVerifier_A')
    , IdscpRaVerifier('A', 'B', ~challengeA)
  ]->
  [ Out(senc(<'IdscpRaVerifier', 'A', ~challengeA>, key)) // Send attestation request
    , Alices_Nonce(~challengeA)
  ]

rule IdscpRaVerifier_B:
  [ Tls_Finish('B')
    , !Tls_Ephemeral_Key_B(key)
    , In(senc(<'IdscpRaVerifier', A, challengeA>, key))   // Receive attestation request
    , Fr(~challengeB)                                     // Choose nonce for Bob
  ]--[
    OnlyOnceFor('IdscpRaVerifier_B')
    , IdscpRaVerifier('B', 'A', ~challengeB)
  ]->
  [ Out(senc(<'IdscpRaVerifier', 'B', ~challengeB>, key)) // Send attestation request
    , Bobs_Nonce(~challengeB)
    , Bobs_Received_Nonce(challengeA)
  ]

rule IdscpRaProver_A:
let quoteA = sign(<h(challengeB, tlsCertB), 'PCRA'>, akA) in
  [ !Tls_Ephemeral_Key_A(key)
    , In(senc(<'IdscpRaVerifier', B, challengeB>, key)) // Receive attestation request
    , !TlsCert(B, tlsCertB)                             // Load Bob's TLS certificate
    , Ak('A', akA), !AkCert('A', akCertA)               // Load Alice's attestation key
  ]--[
    OnlyOnceFor('IdscpRaProver_A')
    , IdscpRaProver('A', B, quoteA)
  ]->
  [ Out(senc(<'IdscpRaProver', 'A', quoteA>, key)) ]    // Send quote to Bob

rule IdscpRaProver_B:
let quoteB = sign(<h(challengeA, tlsCertA), 'PCRB'>, akB) in
  [ !Tls_Ephemeral_Key_B(key)
    , In(senc(<'IdscpRaProver', A, quoteA>, key))  // Receive quote from Alice
    , Bobs_Received_Nonce(challengeA)              // Load the previously received nonce
    , !TlsCert(A, tlsCertA)                        // Load Alice's TLS certificate
    , Ak('B', akB), !AkCert('B', akCertB)          // Load Bob's attestation key
  ]--[
    OnlyOnceFor('IdscpRaProver_B')
    , IdscpRaProver('B', A, quoteB)
  ]->
  [ Out(senc(<'IdscpRaProver', 'B', quoteB>, key)) // Send quote to Alice
    , Bobs_Received_Quote(quoteA)                  // Store quote for later verification
  ]

rule IdscpRaResult_A:
  [ !Tls_Ephemeral_Key_A(key)
```

```
    , In(senc(<'IdscpRaProver', B, quoteB>, key))    // Receive quote from Bob
    , Alices_Nonce(challengeA)                       // Retrieve own stored nonce
    , !Tls('A', tlsA)                                // Lookup own TLS key
    , !AkCert(B, akCertB)                            // Get Bob's attestation certificate
  ]--[
    OnlyOnceFor('IdscpRaResult_A')
    , Eq(verify(quoteB,                              // Verify the quote, including the
                <h(challengeA, pk(tlsA)), 'PCRB'>,   // nonce and the TLS public key
                akCertB), true)
    , Attested('A', B, challengeA, akCertB)          // Take note that Alice attested Bob
  ]->
  [ Out(senc(<'IdscpRaResult', 'A'>, key)) ]

rule IdscpRaResult_B:
  [ !Tls_Ephemeral_Key_B(key)
    , In(senc(<'IdscpRaResult', A>, key))
    , Bobs_Nonce(challengeB)                       // Retrieve own stored nonce
    , Bobs_Received_Quote(quoteA)                  // Retrieve stored quote from Alice
    , !Tls('B', tlsB)                              // Lookup own TLS key
    , !AkCert(A, akCertA)                          // Get Alice's attestation certificate
  ]--[
    OnlyOnceFor('IdscpRaResult_B')
    , Eq(verify(quoteA,                              // Verify the quote, including the
                <h(challengeB, pk(tlsB)), 'PCRA'>,   // nonce and the TLS public key
                akCertA), true)
    , Attested('B', A, challengeB, akCertA)          // Take note that Bob attested Alice
  ]->[]

restriction Equality:
  All x y #i. Eq(x,y) @i ==> x = y
restriction InEquality:
  All x y #i. Neq(x,y) @i ==> not(x = y)
restriction OnlyOnceFor:
  All X #i #j. OnlyOnceFor(X)@#i & OnlyOnceFor(X)@#j ==> #i = #j

/* Prove that Alice and Bob can mutually attest to one another and establish a shared
 * secret. Proving this lemma makes sure that the model can be fully executed.
 */
lemma Honest_protocol_mutual_attestation:
  exists-trace
    Ex akA akB challengeA challengeB tlsCertA tlsCertB key #i #j #k #l #m #n #o #p.
      IdscpRaVerifier('A', 'B', challengeA) @ #i
      & IdscpRaVerifier('B', 'A', challengeB) @ #j
      & IdscpRaProver('A', 'B', sign(<h(challengeB, tlsCertB), 'PCRA'>, akA)) @ #k
      & IdscpRaProver('B', 'A', sign(<h(challengeA, tlsCertA), 'PCRB'>, akB)) @ #l
      & Attested('A', 'B', challengeA, pk(akB) ) @ #m
      & Attested('B', 'A', challengeB, pk(akA) ) @ #n
      & EstablishedKey('A', key ) @ #o
      & EstablishedKey('B', key ) @ #p

/* Prove that an attacker cannot establish a shared secret with honest Alice and still
```

```
 * successfully complete the attestation. An internal attacker with knowledge of Bob's
 * TLS signature key (cf. rule BobIsAnInternalAttacker) can establish a DH secret with
 * Alice during the TLS handshake. However, that attacker still has to present a valid
 * quote from Bob's system that is signed with his attestation key. If the rule
 * TSS_Quote_TPMB is activated this lemma is falsified, because the internal attacker
 * can use the system of honest Bob to get such a quote and break the security of IDSCP.
 */
lemma Established_key_secrecy:
  /* It cannot be that */
  not(
    Ex key challengeA akCertB #i #j #k #l.
      /* Alice and Bob mutually attested one another, */
      IdscpRaVerifier('A', 'B', challengeA) @ #i
      & Attested('A', 'B', challengeA, akCertB ) @ #j
      /* they established a shared key, */
      & EstablishedKey('A', key) @ #k
      /* and the adversary knows the key */
      & K(key) @ #l
  )

/* Sanity check: Prove that the attacker does not learn the secret attestation keys.
 * In our model we assume that the trusted platforms are secure and no attestation keys
 * leak. Otherwise the protocol would be trivially insecure, since an attacker could
 * just forge a quote.
 */
lemma Aks_do_not_leak:
  not(
    Ex akA #i #j.
      IsAK('A', akA) @ #i
      & K(akA) @ #j
  ) & not(
    Ex akB #i #j.
      IsAK('B', akB) @ #i
      & K(akB) @ #j
  )

end
```

Figure B.1 shows the execution of this model using the Tamarin theorem prover. As expected, Tamarin falsifies the lemma Established_key_secrecy by finding the nonce-data attack on the IDSCP handshake.



```
      solve( TPM( 'B' ) ▶₁ #i )
        case TPM
        by solve( !KU( ~ak ) @ #vk )
      qed
    qed
  qed
qed

/* All well-formedness checks were successful. */

end

==============================================================================
summary of summaries:

analyzed: idscp.spthy

  Honest_protocol_mutual_attestation (exists-trace): verified (42 steps)
  Established_key_secrecy (all-traces): falsified - found trace (21 steps)
  Aks_do_not_leak (all-traces): verified (14 steps)

==============================================================================
Elapsed time: 3.73s
user@ubuntu-jammy-VirtualBox:~/tamarin-verification$ 
```

Figure B.1: Proof of the nonce-data attack on the IDSCP protocol using Tamarin.

B.2 The MSCP Protocol

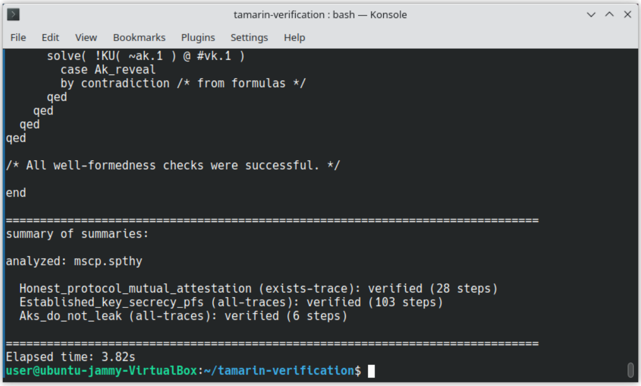
Listing B.2 shows our formal model of the MSCP protocol for the Tamarin theorem prover. We modeled the MSCP handshake using a dedicated Diffie-Hellman key exchange between two endpoints Alice and Bob. Note that in the formalization we make no distinction between the MSCP variants with TPM-internal and TPM-external key establishment. Instead, we model our quoting structure to include `creationData`, which is signed by the attestation key and can either represent the PCR 16, or the output of `TPM2_CertifyCreation`. In both cases, the used `creationData` is cryptographically bound to the ephemeral keys. Like in the formalization of IDSCP, we also include the `TSS_Quote_TPM` rules to model quoting oracles that an attacker can leverage for nonce-data attacks.

Listing B.2: Formal model of the MSCP attestation protocol in Tamarin.

```
theory MSCP
begin

builtins: hashing, signing, diffie-hellman, symmetric-encryption

/* Our goal is to verify the security of the key exchange protocol under the assumption
 * that the trusted platform is secure. This means that we assume the used attestation
 * public keys to be known by Alice and Bob. Furthermore we assume that the attacker
 * is unable to compromise an established attestation key or forge quote signatures. We
 * model these assumptions by creating the attestation keys once for each platform.
 */

// Define two TPMs (Alice and Bob)
rule TPM:
  [] --> [TPM('A'), TPM('B')]

// Create a unique attestation key for each TPM
rule Get_ak:
  [ Fr(~ak), TPM(X) ]
  --[
    OnlyOnceFor(<'Get_ak', X>)
    , IsAK(X, ~ak)
  ]->
  [ !Ak(X, ~ak), !AkCert(X, pk(~ak)) ]

/* Create quotes for Alice's and Bob's trusted software stack (TSS). This rule models
 * that there is some valid software on both platforms that acts as an attestation
 * endpoint. It takes some qualifying data (e.g. a nonce) and asks the TPM to create a
 * signed quote.
 */
rule TSS_Quote_TPMA:
  [ In(qualifyingData)
    , Fr(~creationData) // Creation data for Alice's ephemeral key. Not attacker-chosen.
    , !Ak('A', ak)
  ]-->
  [ Out(sign(<qualifyingData, 'PCRA', ~creationData>, ak)) ]
rule TSS_Quote_TPMB:
  [ In(qualifyingData)
    , Fr(~creationData) // Creation data for Bob's ephemeral key. Not attacker-chosen.
    , !Ak('B', ak)
  ]-->
  [ Out(sign(<qualifyingData, 'PCRB', ~creationData>, ak)) ]

// Initiation phase
rule Init_A:
  [ Fr(~nonceA) ]                     // Choose fresh nonce for Alice
  --[
    OnlyOnceFor('Init_A')
    , Initiation('A', 'B', ~nonceA)   // Take note of Alice's init request
  ]->
  [ Out(<'Init_A', ~nonceA>)          // Send the nonce to Bob
    , !Alices_Nonce(~nonceA)
    , Init_A('B')
  ]

rule Init_B:
  [ In(<'Init_A', nonceA>)            // Receive nonce from Alice
    , Fr(~nonceB)                     // Choose fresh nonce for Bob
  ]
  --[ OnlyOnceFor('Init_B')
      , Initiation('B', 'A', ~nonceB) // Take note of Bob's init request
  ]->
  [ Out(<'Init_B', ~nonceB>)           // Send the nonce to Alice
    , !Bobs_Nonce(~nonceB)
    , !Bobs_Received_Nonce(nonceA)
    , Init_B('A')
  ]

// Attestation phase
rule Attestation_A:
let quoteA = sign(<nonceB, 'PCRA', 'g'^~a>, akA) in
/* This quoting information attests to both the platform state and the ephemeral key.
 * In case of MSCP with internal key establishment, this is created by
 * TPM2_CertifyCreation. The creation data includes the name of the public key. In case
 * of MSCP with external key establishment, this is created by TPM2_Quote. The ephemeral
 * public key is then included by extending it into PCR 16.
 */
  [ Init_A(B)
    , In(<'Init_B', nonceB>)                     // Receive nonce from Bob
    , Fr(~a)                                     // Create a new DH private key for Alice
    , !Ak('A', akA)                              // Load Alice's attestation key
  ]--[
    OnlyOnceFor('Attestation_A')
    , AttestationResponse('A', B, quoteA)        // Take note of Alice's quote
  ]->
  [ Out(<'Attestation_A', quoteA, 'g'^~a>)       // Send quote and DH public key to Bob
    , Attestation_A(B, ~a)
  ]

rule Attestation_B:
let quoteB = sign(<nonceA, 'PCRB', 'g'^~b>, akB) in

  [ Init_B(A)
    , In(<'Attestation_A', quoteA, dhPubA>)   // Receive quote and public key from Alice
    , Fr(~b)                                  // Create a new DH private key for Bob
    , !Ak('B', akB)                           // Load Bob's attestation key
    , !Bobs_Received_Nonce(nonceA)            // Load the nonce that Bob received earlier
  ]--[
    OnlyOnceFor('Attestation_B')
    , Neq(dhPubA, 'g')                        // Alice's public DH key must not be 'g'
    , AttestationResponse('B', A, quoteB)     // Take note of Bob's quote
  ]->
  [ Out(<'Attestation_B', quoteB, 'g'^~b>)    // Send quote and public key to Alice
    , !Bobs_Received_Attestation(<quoteA, dhPubA>)
    , Attestation_B(A, ~b)
  ]

// Verification phase
rule Verification_A:
  [ Attestation_A(B, a)
```

```
    , In(<'Attestation_B', quoteB, dhPubB>)  // Receive quote and public key from Bob
    , !Alices_Nonce(nonceA)                  // Load Alice's nonce
    , !AkCert('B', akCertB)                  // Load Bob's attestation certificate
  ]--[
    OnlyOnceFor('Verification_A')
    , Eq(verify(quoteB,                      // Verify the quote, including the nonce
                <nonceA, 'PCRB', dhPubB>,    // and the DH public key
                akCertB), true)
    , Attested('A', B, nonceA, akCertB)      // Take note that Alice attested Bob
    , EstablishedKey('A', dhPubB^a)          // Generate shared key
  ]->
  [ Out(<'Verification_A'>) ]

rule Verification_B:
  [ Attestation_B(A, b)
    , In(<'Verification_A'>)
    , !Bobs_Nonce(nonceB)                  // Load Bob's nonce
    , !Bobs_Received_Attestation(          // Load the quote and DH key received earlier
                       <quoteA, dhPubA>)
    , !AkCert('A', akCertA)                // Load Alice's attestation certificate
  ]--[
    OnlyOnceFor('Verification_B')
    , Eq(verify(quoteA,                       // Verify the quote, including the nonce
                <nonceB, 'PCRA', dhPubA>,     // and the DH public key
                akCertA), true)
    , Attested('B', A, nonceB, akCertA)       // Take note that Alice attested Bob
    , EstablishedKey('B', dhPubA^b)           // Generate shared key
  ]->[]

// Reveal the attestation keys
rule Ak_reveal:
  [ !Ak(X, ak) ] --[ RevealAk(X) ]-> [ Out(ak) ]

restriction Equality:
  All x y #i. Eq(x,y) @i ==> x = y
restriction InEquality:
  All x y #i. Neq(x,y) @i ==> not(x = y)
restriction OnlyOnceFor:
  All X #i #j. OnlyOnceFor(X)@#i & OnlyOnceFor(X)@#j ==> #i = #j

 /* Prove that Alice and Bob can mutually attest to one another and establish a shared
 * secret. Proving this lemma makes sure that the model can be fully executed.
 */
lemma Honest_protocol_mutual_attestation:
  exists-trace
  Ex akA akB nonceA nonceB dhPubA dhPubB key #i #j #k #l #m #n #o #p.
    Initiation('A', 'B', nonceA) @ #i
    & Initiation('B', 'A', nonceB) @ #j
    & AttestationResponse('A', 'B', sign(<nonceB, 'PCRA', dhPubA>, akA)) @ #k
    & AttestationResponse('B', 'A', sign(<nonceA, 'PCRB', dhPubB>, akB)) @ #l
    & Attested('B', 'A', nonceB, pk(akA)) @ #m
    & Attested('A', 'B', nonceA, pk(akB)) @ #n
    & EstablishedKey('B', key) @ #o
    & EstablishedKey('A', key) @ #p

/* Prove that an attacker cannot retrieve the shared secret between Alice and Bob and
 * still successfully complete the attestation, unless the attestation keys are revealed
 * before the attestation is accepted (forward secrecy). Unlike IDSCP, this lemma holds
 * true even if the TSS_Quote_TPM rules are active (i.e. internal attackers have access
 * to the platform quote oracles).
 */
lemma Established_key_secrecy_pfs:
  /* For all MSCP handshakes, */
  (All key nonceA akCertB #i #j #k #l.
      /* where Alice attests Bob, */
      Initiation('A', 'B', nonceA) @ #i
      & Attested('A', 'B', nonceA, akCertB) @ #j
      /* and a shared key is established, */
      & EstablishedKey('A', key) @ #k
      /* which is known by the attacker, */
      & K(key) @ #l
    /* the attacker must already have known Bob's attestation key, */
    ==> Ex #z. RevealAk('B') @ #z
               /* even before Alice accepted the attestation */
               & #z < #j
  ) &
  /* And the same also the other way around */
  (All key nonceB akCertA #i #j #k #l.
      Initiation('B', 'A', nonceB) @ #i
      & Attested('B', 'A', nonceB, akCertA) @ #j
      & EstablishedKey('B', key) @ #k
      & K(key) @ #l
    ==> Ex #z. RevealAk('A') @ #z & #z < #j
   )

/* Sanity check: Prove that the attacker does not learn the secret attestation keys.
 * In our model we assume that the trusted platforms are secure and no attestation keys
 * leak. Otherwise the protocol would be trivially insecure, since an attacker could
 * just forge a quote.
 */
lemma Aks_do_not_leak:
  All akA #i #j.
      IsAK('A', akA) @ #i
      & K(akA) @ #j
      ==> Ex #z. RevealAk('A') @ #z
  & All akB #i #j.
      IsAK('B', akB) @ #i
      & K(akB) @ #j
      ==> Ex #z. RevealAk('B') @ #z

end
```

Figure B.2 shows that Tamarin verifies the MSCP protocol handshake in under 4 seconds. The lemma `Honest_protocol_mutual_attestation` checks the correct execution of the mutual attestation between Alice and Bob. This verifies our protocol requirements R1 (authentication), R2 (mutual attestation), and R3 (replay protection). The lemma `Established_key_secrecy_pfs` proves that any attacker capable of intercepting the shared secret between Alice and Bob on the attested channel must have known the long-term attestation keys even before the attestation took place. This verifies the final two requirements R4 (secure channels) and R5 (perfect forward secrecy). Together with the inclusion of the `TSS_Quote_TPM` oracle rules, this proves the security of MSCP even against nonce-data attacks.



```
      solve( !KU( ~ak.1 ) @ #vk.1 )
        case Ak_reveal
        by contradiction /* from formulas */
      qed
    qed
  qed
qed

/* All well-formedness checks were successful. */

end

==============================================================================
summary of summaries:

analyzed: mscp.spthy

  Honest_protocol_mutual_attestation (exists-trace): verified (28 steps)
  Established_key_secrecy_pfs (all-traces): verified (103 steps)
  Aks_do_not_leak (all-traces): verified (6 steps)

==============================================================================
Elapsed time: 3.82s
user@ubuntu-jammy-VirtualBox:~/tamarin-verification$ 
```

Figure B.2: Verification of the MSCP protocol using Tamarin.

B.3 The EKEP Protocol

The EKEP attestation protocol has already been formally verified by Roeder et al. using the ProVerif theorem prover [Roe22]. However, as discussed in section 4.5.3, when using EKEP to conduct TPM-based remote attestations we need to ensure that the ephemeral Diffie-Hellman keys are properly bound to the attested identity. Otherwise the resulting protocol may be vulnerable to nonce-data attacks. To better represent this issue in the formal model, we slightly modified the existing formalization from [Roe22]. Listing B.3 shows the relevant excerpts[1] compared to the existing base model of EKEP.

Listing B.3: Excerpt of the ProVerif formal model for the EKEP attestation protocol, modified to include TPM-based attestation. The original model has been published in [Roe22].

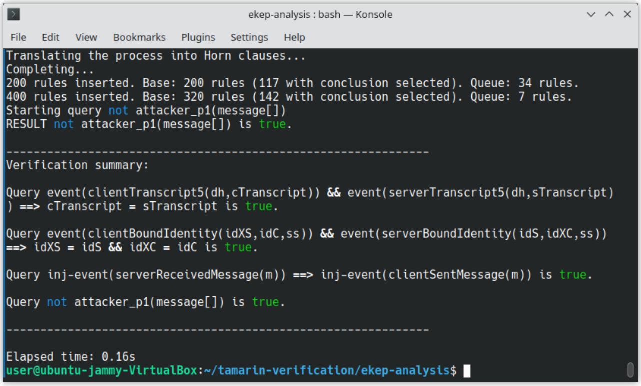
```
[...]

(** Implements the server side of EKEP. *)
let Server(serverName: Name, tpmPrivKey: SigningKey, sgxPubKey: VerifyingKey) =
  (* Receive Client Precommit. *)
  in(c, clientPrecommit: Challenge);
  let transcript0 = CreateTranscript0(clientPrecommit) in

  (* Send Server Precommit. *)
  new serverPrecommit: Challenge;
  out(c, serverPrecommit);
  let transcript1 = ExtendToTranscript1(transcript0, serverPrecommit) in

[...]

  (* Compute Server Id *)
  new serverPrivateKey: Z;
  let serverPublicKey = exp(g, serverPrivateKey) in

  (* Create a message for TPM to sign. Unlike with SGX, this message is directly signed
        by the TPM and does not leave the TCB over the channel. *)
  let serverTpmMessage = BuildServerTpmMessage(
      serverName, kServerId, serverPublicKey, transcript2) in
  let serverId = signServerTpm(tpmPrivKey, serverTpmMessage) in

  (* Send Server Id. *)
  out(c, serverId);
  let transcript3 = ExtendToTranscript3(transcript2, serverId) in

[...]

(* SgxAttestationForClient provides SGX signatures for clients. *)
let SgxAttestationForClient(clientSgxMacKey: HmacKey, sgxPrivKey: SigningKey) =
  in(c, (message: ClientSgxMessage, tag: HmacAuthTag));
  let kind = ClientSgxMessage_kind(message) in
  if kind = kClientId then
  if hmacClientSgxVerify(clientSgxMacKey, message, tag) then
  out(c, signClientSgx(sgxPrivKey, message)).

```

[1] The complete (modified) formal model is available at https://gitlab.cc-asp.fraunhofer.de/datasov/ekep/-/blob/main/ekep_tpm.pv *(accessed on 12/08/2023)*.

```
(* SgxAttestationForOther provides SGX signatures for the adversary. *)
let SgxAttestationForOther(sgxPrivKey: SigningKey) =
  in(c, (name: Name, publicKey: G, transcript: bitstring));
  out(c, sign(sgxPrivKey, (name, kOtherId, publicKey, transcript))).

(* TpmAttestationForOther provides TPM signatures for the adversary. *)
let TpmAttestationForOther(tpmPrivKey: SigningKey) =
  in(c, (name: Name, transcript: bitstring));
  new publicKey: G;
  out(c, sign(tpmPrivKey, (name, kServerId, publicKey, transcript))).

process
  new clientSgxMacKey: HmacKey;
  (* The server is not a TEE and does not have a secret MAC key *)

  new serverName: Name;
  new clientName: Name;
  out(c, clientName);
  out(c, serverName);

  new sgxPrivKey: SigningKey;
  let sgxPubKey = pubKey(sgxPrivKey) in
  out(c, sgxPubKey);

  (* This is the server's long term secret *)
  new tpmPrivKey: SigningKey;
  let tpmPubKey = pubKey(tpmPrivKey) in
  out(c, tpmPubKey);

  new message: bitstring;

  ( (! Server(serverName, tpmPrivKey, sgxPubKey) )
  | (! Client(clientName, clientSgxMacKey, sgxPubKey, tpmPubKey, message) )
  | (! SgxAttestationForClient(clientSgxMacKey, sgxPrivKey) )
  | (! SgxAttestationForOther(sgxPrivKey) )
  | (! TpmAttestationForOther(tpmPrivKey) )
  (* Expose the client and server's long-term identity keys in phase 1 to test perfect forward secrecy in the protocol. *)
  | ( phase 1; out(c, clientSgxMacKey); out(c, tpmPrivKey) )
  )
```

The original EKEP formalization in [Roe22] represents attestation identities using HMAC keys that must be kept secret by the endpoint enclaves. In order to receive attestation evidence that is signed by the long-term attestation keys, each endpoint first needs to prove their own identity to the quoting enclave by providing a signature under the respective HMAC key. While this model works for TEEs such as Intel SGX, it does not quite fit the TPM-based attestation mechanism. When conducting TPM-based attestations, the report is usually signed directly with the attestation key by the TPM, instead of first being authenticated at a quoting enclave. To better represent this in the model, we modified the formalization to sign TPM assertions directly with the long-term attestation key `tpmPrivKey` (cf. lines 20ff.).

The second necessary modification deals with the assumptions about the attackers' abilities to request attestation evidence from the trusted platforms. When using SGX, the assumption is that an attacker can get a signed attestation report that authenticates any attacker-chosen ephemeral public key. However, the SGX attestation report cannot include forged enclave identities, since these are authenticated using the secret HMAC keys. This models the existence of other enclaves on the SGX platform and is represented in the formalization by the `SgxAttestationForOther` subprocess in lines 40ff. However, this representation is not completely accurate anymore when using TPMs instead of TEEs. In that case we need to assume that attackers *can* get valid attestation evidence (i.e., a quote) from the TPM-protected platform, for example through secondary attestation endpoints as part of a nonce-data attack. A secure TPM-based attestation protocol must then take care to prevent the inclusion of attacker-chosen ephemeral public keys into the quote, which could otherwise be used to impersonate the trusted platform. For example, when using MSCP the ephemeral keys are generated inside the TPM and are hence out of reach for the attacker. To represent this difference in the formal model, we include an additional `TpmAttestationForOther` subprocess (cf. lines 45ff.). This subprocess gives attackers the option to retrieve TPM-based attestation evidence for the correct platform identity (in this case `kServerId`). However, the attacker is not allowed to arbitrarily choose the ephemeral public key that should be included in the attestation report. By including this new subprocess in the formal model, we can now verify the security of MSCP endpoints in EKEP also against nonce-data attacks.


ekep-analysis : bash — Konsole
File Edit View Bookmarks Plugins Settings Help

```
Translating the process into Horn clauses...
Completing...
200 rules inserted. Base: 200 rules (117 with conclusion selected). Queue: 34 rules.
400 rules inserted. Base: 320 rules (142 with conclusion selected). Queue: 7 rules.
Starting query not attacker_p1(message[])
RESULT not attacker_p1(message[]) is true.

--------------------------------------------------------------
Verification summary:

Query event(clientTranscript5(dh,cTranscript)) && event(serverTranscript5(dh,sTranscript)
) ==> cTranscript = sTranscript is true.

Query event(clientBoundIdentity(idXS,idC,ss)) && event(serverBoundIdentity(idS,idXC,ss))
==> idXS = idS && idXC = idC is true.

Query inj-event(serverReceivedMessage(m)) ==> inj-event(clientSentMessage(m)) is true.

Query not attacker_p1(message[]) is true.

--------------------------------------------------------------

Elapsed time: 0.16s
user@ubuntu-jammy-VirtualBox:~/tamarin-verification/ekep-analysis$
```

Figure B.3: Verification of the modified EKEP protocol including TPM-based heterogeneous attestations using ProVerif.

Figure B.3 shows that ProVerif validates the security properties of EKEP also with our modified formalization. This proves the soundness of our approach to achieve heterogeneous remote attestation between TPMs and SGX enclaves by including an MSCP attestation generator into the existing EKEP design.

C The DataSov ODRL Profile

This chapter includes the definition of our custom ODRL profile for the DataSov framework. The DataSov ODRL profile adds support for direct PIP lookups and PXP execute demands, which is a prerequisite for trustworthy distributed usage control. Furthermore, the profile includes new vocabulary definitions for representing provenance information in usage control policies.

Listing C.1 shows the complete DataSov ODRL profile definition, which is extending the core ODRL information model with the additional concepts that have been described in section 5.3. The DataSov ODRL profile is defined under the `ods:` prefix and is specified in the Turtle RDF language.

Listing C.1: The DataSov ODRL profile in Turtle.

```
@base <https://gitlab.cc-asp.fraunhofer.de/datasov/core#> .
@prefix ods: <https://gitlab.cc-asp.fraunhofer.de/datasov/core#> .
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix profile: <http://www.w3.org/ns/dx/prof/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix role: <http://www.w3.org/ns/dx/prof/role/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .

ods: a owl:Ontology, profile:Profile ;
  profile:isProfileOf <http://www.w3.org/ns/odrl/2/core> ;
  profile:hasResource ods:ods-ttl ;
  rdfs:label "ODRL DataSov Profile Ontology"@en ;
  owl:versionInfo "1.0" ;
  dct:creator "Paul Wagner" ;
  dct:description "The ODRL DataSov Profile. Adds required concepts for trustworthy
        usage control and provenance tracking."@en ;
  rdfs:comment "This is the RDF ontology for the ODRL DataSov Profile Version 1.0."@en ;
  dct:conformsTo <https://www.w3.org/TR/odrl-model/> .
```

```
ods:ods-ttl a profile:ResourceDescriptor ;
  profile:hasRole role:vocabulary ;
  profile:hasArtifact <https://gitlab.cc-asp.fraunhofer.de/datasov/core/-/tree/master/pdp/profile/ods.ttl> ;
  dct:title "Turtle Vocabulary for the ODRL DataSov Profile."@en ;
  dct:format <https://www.iana.org/assignments/media-types/text/turtle> ;
  dct:conformsTo <https://www.w3.org/TR/turtle/> .

## SKOS Collections for ODRL DataSov Vocabulary.
<https://gitlab.cc-asp.fraunhofer.de/datasov/core#> a skos:Collection ;
  skos:prefLabel "ODRL DataSov Profile v1.0"@en ;
  skos:scopeNote "The terms of ODRL DataSov Profile"@en ;
  skos:member ods:ctxOperand ;
  skos:member ods:ctxKey ;
  skos:member ods:pipOperand ;
  skos:member ods:pipMethod ;
  skos:member ods:pxpMethod ;
  skos:member ods:pepMethod ;
  skos:member ods:uri ;
  skos:member ods:pxpAction ;
  skos:member ods:pepAction ;
  skos:member ods:params ;
  skos:member ods:provenance ;
  skos:member ods:entities ;
  skos:member ods:activities ;
  skos:member ods:agents ;
  skos:member ods:relations ;
  skos:member ods:match ;
  skos:member ods:nmatch ;
  skos:member ods:getTimePipMethod ;
  skos:member ods:storeValuesPipMethod ;
  skos:member ods:getValuePipMethod ;
  skos:member ods:incCounterPipMethod ;
  skos:member ods:decCounterPipMethod ;
  skos:member ods:resetCounterPipMethod ;
  skos:member ods:provenancePipMethod ;
  skos:member ods:logPxpMethod ;
  skos:member ods:provenancePxpMethod ;
  skos:member ods:modifyParamPepMethod ;
  skos:member ods:deployPolicyPepMethod ;
  skos:member ods:revokePolicyPepMethod .

## Concepts for context operands
ods:ctxOperand a odrl:LeftOperand, odrl:RightOperand, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Context operand"@en ;
  skos:definition "A left or right operand that is being evaluated on the event context."@en .

ods:ctxKey a rdf:Property, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Context operand key"@en ;
  skos:definition "The lookup key for a context operand."@en ;
  rdfs:domain ods:ctxOperand .

## Concepts for external information sources
ods:pipOperand a odrl:LeftOperand, odrl:RightOperand, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "External operand"@en ;
  skos:definition "A left or right operand that is being evaluated using an external information source."@en .

ods:pipMethod a rdf:Property, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Policy information point method"@en ;
  skos:definition "An unambiguous identifier for a method implemented by a policy information point."@en ;
  rdfs:domain ods:pipOperand .

ods:uri a rdf:Property, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Component uri"@en ;
  skos:definition "An uri unambiguously representing a component identity."@en ;
  rdfs:domain ods:pipOperand, ods:pxpAction .

## Concepts for actions
ods:pxpAction a odrl:Action, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "PXP action"@en ;
  skos:definition "An action definition that should be executed at an external execution point."@en .

ods:pxpMethod a rdf:Property, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Policy execution point method"@en ;
  skos:definition "An unambiguous identifier for a method implemented by a policy execution point."@en ;
  rdfs:domain ods:pxpAction .

ods:pepAction a odrl:Action, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "PEP action"@en ;
  skos:definition "An action definition that should be executed at the local enforcement point."@en .

ods:pepMethod a rdf:Property, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Policy enforcement point method"@en ;
  skos:definition "An unambiguous identifier for a method implemented by a policy enforcement point."@en ;
  rdfs:domain ods:pepAction .
ods:params a rdf:Property, skos:Concept ;
```

```
  rdfs:isDefinedBy ods: ;
  rdfs:label "Operand parameters"@en ;
  skos:definition "A key-value map of operand parameters."@en ;
  rdfs:domain ods:pipOperand, ods:pxpAction, ods:pepAction .

## Concepts for provenance
ods:provenance a rdfs:Class, owl:Class, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Provenance information"@en ;
  skos:definition "A representation of provenance data in the PROV model."@en .

ods:entities a rdf:Property, rdf:List, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Provenance entities"@en ;
  skos:definition "A list of provenance entities. Each entity is represented by a unique uri."@en ;
  rdfs:domain ods:provenance ;
  rdfs:range prov:Entity .

ods:activities a rdf:Property, rdf:List, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Provenance activities"@en ;
  skos:definition "A list of provenance activities. Each activity is represented by a unique uri."@en ;
  rdfs:domain ods:provenance ;
  rdfs:range prov:Activity .

ods:agents a rdf:Property, rdf:List, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Provenance agents"@en ;
  skos:definition "A list of provenance agents. Each agent is represented by a unique uri."@en ;
  rdfs:domain ods:provenance ;
  rdfs:range prov:Agent .

ods:relations a rdf:Property, rdf:List, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Provenance relations"@en ;
  skos:definition "A list of provenance relations. Each relation is represented as a triple [from_uri, relation_type, to_uri]."@en ;
  rdfs:domain ods:provenance .

ods:match a odrl:Operator, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Match"@en ;
  skos:definition "Indicating that a given regular expression matches a constraint operand."@en .

ods:nmatch a odrl:Operator, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Not match"@en ;
  skos:definition "Indicating that a given regular expression does not match a constraint operand."@en .

## DataSov methods
ods:getTimePipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for time retrieval"@en ;
  skos:definition "An information point method that returns the current time of day."@en ;
  skos:note "Parameters: datatype, format, timezone. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:storeValuesPipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for value storage"@en ;
  skos:definition "An information point method that stores a set of parameters."@en .

ods:getValuePipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for value retrieval"@en ;
  skos:definition "An information point method that returns a previously stored parameter value."@en ;
  skos:note "Parameters: name. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:incCounterPipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for counter increases"@en ;
  skos:definition "An information point method that increases the value of a named counter."@en ;
  skos:note "Parameters: name, diff. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:decCounterPipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for counter decreases"@en ;
  skos:definition "An information point method that decreases the value of a named counter."@en ;
  skos:note "Parameters: name, diff. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:resetCounterPipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for counter resets"@en ;
  skos:definition "An information point method that resets a named counter to 0."@en ;
  skos:note "Parameters: name. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:provenancePipMethod a ods:pipMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pip method for provenance retrieval"@en ;
  skos:definition "An information point method that returns provenance information."@en ;
  skos:note "Parameters: prosps, request. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:logPxpMethod a ods:pxpMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pxp method for logging"@en ;
  skos:definition "An execution point method that logs a message."@en ;
  skos:note "Parameters: message, level. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:provenancePxpMethod a ods:pxpMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pxp method for provenance storage"@en ;
  skos:definition "An execution point method that stores provenance information."@en ;
  skos:note "Parameters: prosps, provenance. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:modifyParamPepMethod a ods:pepMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pep method for parameter modification"@en ;
  skos:definition "An enforcement point method that modifies an event parameter."@en ;
  skos:note "Parameters: key, mask. String parameters may also have: match "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:deployPolicyPepMethod a ods:pepMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pep method for policy deployment"@en ;
  skos:definition "An enforcement point method that deploys a new policy."@en ;
  skos:note "Parameters: policy. "@en ;
  skos:scopeNote "Non-Normative"@en .

ods:revokePolicyPepMethod a ods:pepMethod, owl:NamedIndividual, skos:Concept ;
  rdfs:isDefinedBy ods: ;
  rdfs:label "Pep method for policy revocation"@en ;
  skos:definition "An enforcement point method that revokes a policy."@en .

## Declaration of annotation properties to keep the ontology within OWL DL
  skos:member rdf:type owl:AnnotationProperty .
  skos:scopeNote rdf:type owl:AnnotationProperty .
  skos:prefLabel rdf:type owl:AnnotationProperty .
  skos:definition rdf:type owl:AnnotationProperty .
  skos:example rdf:type owl:AnnotationProperty .
  skos:note rdf:type owl:AnnotationProperty .
  dct:format rdf:type owl:AnnotationProperty .
  dct:title rdf:type owl:AnnotationProperty .
  dct:conformsTo rdf:type owl:AnnotationProperty .
  dct:creator rdf:type owl:AnnotationProperty .
  dct:description rdf:type owl:AnnotationProperty .
  skos:Collection a owl:Class .
```

Listing C.2 shows the JSON-LD context definition that can be used to parse ODRL policies in the DataSov framework. It extends the core ODRL context definition with the vocabulary specified in the DataSov ODRL profile.

Listing C.2: JSON-LD context definition for the DataSov ODRL profile.

```
{"@context": ["http://www.w3.org/ns/odrl.jsonld",
  { "ods": "https://gitlab.cc-asp.fraunhofer.de/datasov/core#",
    "ods:ctxOperand": {
      "@context": { "key": "ods:ctxKey" }
    },
    "ods:pipOperand": {
      "@context": {
        "uri": "@id",
        "method": "ods:pipMethod",
        "params": {
          "@id": "ods:params",
          "@container": "@index"
        }
      }
    },
    "ods:pxpAction": {
      "@context": {
        "uri": "@id",
        "method": "ods:pxpMethod",
        "params": {
          "@id": "ods:params",
          "@container": "@index"
        }
      }
    },
    "ods:pepAction": {
      "@context": {
        "method": "ods:pepMethod",
        "params": {
          "@id": "ods:params",
          "@container": "@index"
        }
      }
    },
    "ods:provenance": {
      "@context": {
        "entities": "ods:entities",
        "activities": "ods:activities",
        "agents": "ods:agents",
        "relations": "ods:relations"
      }
    }
}]}
```

D Proof Sketches

This chapter contains proof sketches for two of the requirements that our trustworthiness score should fulfill.

Proposition D.1 formalizes the requirement S3 as described in section 6.1.

Proposition D.1 (Minimality). *Let $S \in \mathcal{S}$ be the state of a distributed usage control system with instance graph $I = (V, E)$ and trust model $\mathcal{T} = (t_o, t_m)$. Let $P \in \mathcal{P}$ be a usage control policy and $J \coloneqq ops(P, (d, r))$ its usage control operation for policy deployer/receiver $(d, r) \in V^2$. Let $\hat{s} \in O$ be the least trusted component operator and $\hat{c} \in M \times G \times A$ the least strong mechanism capability that is required for the enforcement of usage control operation J. For the trustworthiness score t then follows*

$$t(d, r, P, S) \leq \min\left(t_o(\hat{s}), t_m(\hat{c})\right).$$

Proof.

$$\begin{aligned}
t(d, r, P, S) &\overset{Def.}{=} \max_{T \in PT} \left(\prod_{s \in T} t_o(s) \cdot \prod_{c \in Caps_S(J|T)} t_m(c) \right) \\
&= \prod_{s \in T_{max}} t_o(s) \cdot \prod_{c \in Caps_S(J|T_{max})} t_m(c) \\
&\overset{\hat{s} \in T_{max}}{=} t_o(\hat{s}) \cdot \underbrace{\prod_{s \in T_{max} \setminus \{\hat{s}\}} t_o(s) \cdot \prod_{c \in Caps_S(J|T_{max})} t_m(c)}_{\leq 1} \\
&\leq t_o(\hat{s}). \qquad \text{(i)}
\end{aligned}$$

Since $\hat{c} \in Cap_S(J \mid T_{max})$, extracting $t_m(\hat{c})$ analogously yields

$$t(d,r,P,S) \leq t_m(\hat{c}). \tag{ii}$$

With (i) and (ii) it follows that $t(d,r,P,S) \leq \min\left(t_o(\hat{s}), t_m(\hat{c})\right)$. ■

Proposition D.2 formalizes the requirement S4 as described in section 6.1.

Proposition D.2 (Monotony). *Let $S \in \mathcal{S}$ be the state of a distributed usage control system with component model $\mathcal{C} = ((V,E), att, m, o)$, attacker model $\mathcal{A} = (a, g)$, and trust model $\mathcal{T} = (t_o, t_m)$. Let $P_1, P_2 \in \mathcal{P}$ be two policies with usage control operations $J_1 := ops(P_1, (d,r))$ and $J_2 := ops(P_2, (d,r))$ for policy deployer/receiver $(d,r) \in V^2$. For the trustworthiness score t then follows*

$$J_1 \supseteq J_2 \implies t(d,r,P_1,S) \leq t(d,r,P_2,S).$$

Proof. *For convenience, we abbreviate the trustworthiness scores $t(d,r,P_1,S)$ and $t(d,r,P_2,S)$ as t_1 and t_2. We also denote the sets of trusted operators for which t_1 and t_2 get maximal as $\hat{T}_1 \in PT_1$ and $\hat{T}_2 \in PT_2$.*

If $J_1 = J_2$, then trivially follows $t_1 = t_2$.

Without loss of generality, let $J_1 = J_2 \cup \{(u,v)\}$ for exactly one trust dependency $(u,v) \in E \setminus J_2$. Due to the construction of our score as given in definition 6.8, the component $v \in J_1$ must now be covered either by operator trust (i.e., $o(v) \in \hat{T}_1$) or by mechanism trust (i.e., $o(v) \notin \hat{T}_1$). We can distinguish three cases concerning the component operator $o(v)$.

***Case 1:** $o(v) \in \hat{T}_1$ and $o(v) \in \hat{T}_2$.*
In this case, the component $v \in J_1$ is covered by operator trust in the extended usage control operation graph, i.e., $o(v) \in \hat{T}_1$. Since $o(v)$ has already been a trusted operator in J_2, i.e., $o(v) \in \hat{T}_2$, and both graphs differ only in the edge (u,v), it must follow that $\hat{T}_1 = \hat{T}_2$ and subsequently $t_1 = t_2$.

***Case 2:** $o(v) \in \hat{T}_1$ and $o(v) \notin \hat{T}_2$.*
If component v is covered by operator trust in J_1, but $o(v)$ is not already a trusted operator in J_2, we must distinguish two more subcases.

Subcase 1: If $o(v) \notin o(J_2)$, then the added component v introduces a completely new stakeholder into J_1. In this case, with $\hat{T}_1 \setminus \hat{T}_2 = \{o(v)\}$ it follows from the construction of the score that

$$t_1 = \underbrace{t_o(o(v))}_{\leq 1} \cdot t_2 \leq t_2.$$

Subcase 2: If $o(v) \in o(J_2)$, then $\hat{T}_1$ is an operator combination that fully covers the critical stakeholders of J_2, i.e., $\hat{T}_1 \in PT_2$. Since t_2 is maximized over PT_2 (cf. definition 6.8), and both graphs differ only in the edge (u, v), it must follow that $t_1 \leq t_2$.

Case 3: *$o(v) \notin \hat{T}_1$ and $o(v) \notin \hat{T}_2$.*
The last possibility is for component v to be covered by mechanism trust, i.e., $o(v) \notin \hat{T}_1$. This introduces a (possibly empty) set of additional critical mechanism capabilities into the score construction. Even though the capabilities introduced by component v can supersede weaker capabilities that are already present in J_2 (cf. line 8 of the algorithm in definition 6.7), the well-formedness of the trust estimation function t_m (i.e., stronger attackers lead to smaller trust values for the same mechanism and protection goal) ensures that the resulting capability trust can only decrease. Since the new mechanism capabilities are considered for all operator combinations $T_1 \in PT_1$ with $o(v) \notin T_1$ (cf. definition 6.8), it must follow that $t_1 \leq t_2$.

Note that if $o(v)$ is already a trusted operator for J_2, then it must also be one for J_1, i.e., $o(v) \in \hat{T}_2 \implies o(v) \in \hat{T}_1$. This is because adding a new trust dependency cannot remove any critical stakeholders (cf. definition 6.6) and all mechanism capabilities of components operated by stakeholder $o(v)$ in J_2 are present in J_1 as well. Hence, the three identified cases are exhaustive. ■

Karlsruher Schriftenreihe zur Anthropomatik (ISSN 1863-6489)

Band 1 Jürgen Geisler
Leistung des Menschen am Bildschirmarbeitsplatz.
ISBN 3-86644-070-7

Band 2 Elisabeth Peinsipp-Byma
Leistungserhöhung durch Assistenz in interaktiven Systemen zur Szenenanalyse. 2007
ISBN 978-3-86644-149-1

Band 3 Jürgen Geisler, Jürgen Beyerer (Hrsg.)
Mensch-Maschine-Systeme.
ISBN 978-3-86644-457-7

Band 4 Jürgen Beyerer, Marco Huber (Hrsg.)
Proceedings of the 2009 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-86644-469-0

Band 5 Thomas Usländer
Service-oriented design of environmental information systems.
ISBN 978-3-86644-499-7

Giulio Milighetti
Band 6 **Multisensorielle diskret-kontinuierliche Überwachung und Regelung humanoider Roboter.**
ISBN 978-3-86644-568-0

Jürgen Beyerer, Marco Huber (Hrsg.)
Band 7 **Proceedings of the 2010 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.**
ISBN 978-3-86644-609-0

Eduardo Monari
Band 8 **Dynamische Sensorselektion zur auftragsorientierten Objektverfolgung in Kameranetzwerken.**
ISBN 978-3-86644-729-5

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 9 Thomas Bader
Multimodale Interaktion in Multi-Display-Umgebungen.
ISBN 3-86644-760-8

Band 10 Christian Frese
Planung kooperativer Fahrmanöver für kognitive Automobile.
ISBN 978-3-86644-798-1

Band 11 Jürgen Beyerer, Alexey Pak (Hrsg.)
Proceedings of the 2011 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-86644-855-1

Band 12 Miriam Schleipen
Adaptivität und Interoperabilität von Manufacturing Execution Systemen (MES).
ISBN 978-3-86644-955-8

Band 13 Jürgen Beyerer, Alexey Pak (Hrsg.)
Proceedings of the 2012 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-86644-988-6

Band 14 Hauke-Hendrik Vagts
Privatheit und Datenschutz in der intelligenten Überwachung: Ein datenschutzgewährendes System, entworfen nach dem „Privacy by Design“ Prinzip.
ISBN 978-3-7315-0041-4

Band 15 Christian Kühnert
Data-driven Methods for Fault Localization in Process Technology. 2013
ISBN 978-3-7315-0098-8

Band 16 Alexander Bauer
Probabilistische Szenenmodelle für die Luftbildauswertung.
ISBN 978-3-7315-0167-1

Band 17 Jürgen Beyerer, Alexey Pak (Hrsg.)
Proceedings of the 2013 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-0212-8

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 18 Michael Teutsch
Moving Object Detection and Segmentation for Remote Aerial Video Surveillance.
ISBN 978-3-7315-0320-0

Band 19 Marco Huber
**Nonlinear Gaussian Filtering:
Theory, Algorithms, and Applications.**
ISBN 978-3-7315-0338-5

Band 20 Jürgen Beyerer, Alexey Pak (Hrsg.)
Proceedings of the 2014 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-0401-6

Band 21 Todor Dimitrov
Permanente Optimierung dynamischer Probleme der Fertigungssteuerung unter Einbeziehung von Benutzerinteraktionen.
ISBN 978-3-7315-0426-9

Band 22 Benjamin Kühn
Interessengetriebene audiovisuelle Szenenexploration.
ISBN 978-3-7315-0457-3

Band 23 Yvonne Fischer
Wissensbasierte probabilistische Modellierung für die Situationsanalyse am Beispiel der maritimen Überwachung.
ISBN 978-3-7315-0460-3

Band 24 Jürgen Beyerer, Alexey Pak (Hrsg.)
Proceedings of the 2015 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-0519-8

Band 25 Pascal Birnstill
Privacy-Respecting Smart Video Surveillance Based on Usage Control Enforcement.
ISBN 978-3-7315-0538-9

Band 26 Philipp Woock
Umgebungskartenschätzung aus Sidescan-Sonardaten für ein autonomes Unterwasserfahrzeug.
ISBN 978-3-7315-0541-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 27 Janko Petereit
Adaptive State × Time Lattices: A Contribution to Mobile Robot Motion Planning in Unstructured Dynamic Environments.
ISBN 978-3-7315-0580-8

Band 28 Erik Ludwig Krempel
Steigerung der Akzeptanz von intelligenter Videoüberwachung in öffentlichen Räumen.
ISBN 978-3-7315-0598-3

Band 29 Jürgen Moßgraber
Ein Rahmenwerk für die Architektur von Frühwarnsystemen. 2017
ISBN 978-3-7315-0638-6

Band 30 Andrey Belkin
World Modeling for Intelligent Autonomous Systems.
ISBN 978-3-7315-0641-6

Band 31 Chettapong Janya-Anurak
Framework for Analysis and Identification of Nonlinear Distributed Parameter Systems using Bayesian Uncertainty Quantification based on Generalized Polynomial Chaos.
ISBN 978-3-7315-0642-3

Band 32 David Münch
Begriffliche Situationsanalyse aus Videodaten bei unvollständiger und fehlerhafter Information.
ISBN 978-3-7315-0644-7

Band 33 Jürgen Beyerer, Alexey Pak (Eds.)
Proceedings of the 2016 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-0678-2

Band 34 Jürgen Beyerer, Alexey Pak and Miro Taphanel (Eds.)
Proceedings of the 2017 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-0779-6

Band 35 Michael Grinberg
Feature-Based Probabilistic Data Association for Video-Based Multi-Object Tracking.
ISBN 978-3-7315-0781-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 36 Christian Herrmann
Video-to-Video Face Recognition for
Low-Quality Surveillance Data.
ISBN 978-3-7315-0799-4

Band 37 Chengchao Qu
Facial Texture Super-Resolution
by Fitting 3D Face Models.
ISBN 978-3-7315-0828-1

Band 38 Miriam Ruf
Geometrie und Topologie von Trajektorienoptimierung
für vollautomatisches Fahren.
ISBN 978-3-7315-0832-8

Band 39 Angelika Zube
Bewegungsregelung mobiler Manipulatoren für die
Mensch-Roboter-Interaktion mittels kartesischer
modellprädiktiver Regelung.
ISBN 978-3-7315-0855-7

Band 40 Jürgen Beyerer and Miro Taphanel (Eds.)
Proceedings of the 2018 Joint Workshop of
Fraunhofer IOSB and Institute for Anthropomatics,
Vision and Fusion Laboratory.
ISBN 978-3-7315-0936-3

Band 41 Marco Thomas Gewohn
Ein methodischer Beitrag zur hybriden Regelung
der Produktionsqualität in der Fahrzeugmontage.
ISBN 978-3-7315-0893-9

Band 42 Tianyi Guan
Predictive energy-efficient motion trajectory
optimization of electric vehicles.
ISBN 978-3-7315-0978-3

Band 43 Jürgen Metzler
Robuste Detektion, Verfolgung und Wiedererkennung von
Personen in Videodaten mit niedriger Auflösung.
ISBN 978-3-7315-0968-4

Band 44 Sebastian Bullinger
Image-Based 3D Reconstruction of Dynamic Objects Using
Instance-Aware Multibody Structure from Motion.
ISBN 978-3-7315-1012-3

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 45 Jürgen Beyerer, Tim Zander (Eds.)
Proceedings of the 2019 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-1028-4

Band 46 Stefan Becker
Dynamic Switching State Systems for Visual Tracking.
ISBN 978-3-7315-1038-3

Band 47 Jennifer Sander
Ansätze zur lokalen Bayes'schen Fusion von Informationsbeiträgen heterogener Quellen.
ISBN 978-3-7315-1062-8

Band 48 Philipp Christoph Sebastian Bier
Umsetzung des datenschutzrechtlichen Auskunftsanspruchs auf Grundlage von Usage-Control und Data-Provenance-Technologien.
ISBN 978-3-7315-1082-6

Band 49 Thomas Emter
Integrierte Multi-Sensor-Fusion für die simultane Lokalisierung und Kartenerstellung für mobile Robotersysteme.
ISBN 978-3-7315-1074-1

Band 50 Patrick Dunau
Tracking von Menschen und menschlichen Zuständen.
ISBN 978-3-7315-1086-4

Band 51 Jürgen Beyerer, Tim Zander (Eds.)
Proceedings of the 2020 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-1091-8

Band 52 Lars Wilko Sommer
Deep Learning based Vehicle Detection in Aerial Imagery.
ISBN 978-3-7315-1113-7

Band 53 Jan Hendrik Hammer
Interaktionstechniken für mobile Augmented-Reality-Anwendungen basierend auf Blick- und Handbewegungen.
ISBN 978-3-7315-1169-4

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 54 Jürgen Beyerer, Tim Zander (Eds.)
**Proceedings of the 2021 Joint Workshop of
Fraunhofer IOSB and Institute for Anthropomatics,
Vision and Fusion Laboratory.**
ISBN 978-3-7315-1171-7

Band 55 Ronny Hug
Probabilistic Parametric Curves for Sequence Modeling.
ISBN 978-3-7315-1198-4

Band 56 Florian Patzer
**Automatisierte, minimalinvasive Sicherheitsanalyse und
Vorfallreaktion für industrielle Systeme.**
ISBN 978-3-7315-1207-3

Band 57 Achim Christian Kuwertz
**Adaptive Umweltmodellierung für kognitive Systeme in
offener Welt durch dynamische Konzepte und quantitative
Modellbewertung.**
ISBN 978-3-7315-1219-6

Band 58 Julius Pfrommer
Distributed Planning for Self-Organizing Production Systems.
ISBN 978-3-7315-1253-0

Band 59 Ankush Meshram
Self-learning Anomaly Detection in Industrial Production.
ISBN 978-3-7315-1257-8

Band 60 Patrick Philipp
**Über die Formalisierung und Analyse medizinischer Prozesse
im Kontext von Expertenwissen und künstlicher Intelligenz.**
ISBN 978-3-7315-1289-9

Band 61 Mathias Anneken
Anomaliedetektion in räumlich-zeitlichen Datensätzen.
ISBN 978-3-7315-1300-1

Band 62 Jürgen Beyerer, Tim Zander (Eds.)
**Proceedings of the 2022 Joint Workshop of
Fraunhofer IOSB and Institute for Anthropomatics,
Vision and Fusion Laboratory.**
ISBN 978-3-7315-1304-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 63 Fabian Dürr
Multimodal Panoptic Segmentation of 3D Point Clouds.
ISBN 978-3-7315-1314-8

Band 64 Jutta Hild
Nutzung von Blickbewegungen für die Mensch-Computer-Interaktion mit dynamischen Bildinhalten am Beispiel der Videobildauswertung.
ISBN 978-3-7315-1330-8

Band 65 Jürgen Beyerer, Tim Zander (Eds.)
Proceedings of the 2023 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory.
ISBN 978-3-7315-1351-3

Band 66 Tobias Michael Kalb
Principles of Catastrophic Forgetting for Continual Semantic Segmentation in Automated Driving.
ISBN 978-3-7315-1373-5

Band 67 Arno Appenzeller
Datensouveränität für Betroffene über persönliche medizinische Daten durch technische Umsetzung einer datenschutzgerechten Forschungsplattform.
ISBN 978-3-7315-1377-3

Band 68 Paul Georg Wagner
Trustworthy Distributed Usage Control Enforcement in Heterogeneous Trusted Computing Environments.
ISBN 978-3-7315-1390-2

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Lehrstuhl für Interaktive Echtzeitsysteme
Karlsruher Institut für Technologie

Fraunhofer-Institut für Optronik, Systemtechnik
und Bildauswertung IOSB Karlsruhe

Distributed usage control is a data protection technology that allows to regulate the use of sensitive data even after they have been shared with third parties. However, existing usage control solutions are susceptible to malicious manipulation by dishonest data receivers. The work at hand addresses this issue by investigating the use of trusted computing technologies to achieve a trustworthy usage control enforcement process. The core of the proposed solution consists of an XACML-based system architecture, extended by a concept for the automated remote attestation and authentication of distributed usage control components. System integrity is ensured by means of several trusted computing mechanisms based on TPMs, Intel SGX, and ARM TrustZone, which protect against malicious tampering and removal of usage control components. To ensure technological interoperability, a heterogeneous attestation protocol is developed and implemented as well. Finally, the resulting trustworthy usage control framework is evaluated using an application scenario from the realm of smart manufacturing.

ISSN 1863-6489
ISBN 978-3-7315-1390-2

ISBN 978-3-7315-1390-2
9 783731 513902