UNIVERSITÀ
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF INDUSTRIAL ENGINEERING
FONDAZIONE BRUNO KESSLER
Doctorate Program in Industrial Innovation

Extensible Model and Policy Engine for Usage Control and Policy-Based Governance

Industrial Applications

Ali Hariri

Advisor

Prof. Bruno Crispo

Università degli Studi di Trento

Co-Advisor

Prof. Theo Dimitrakos

German Research Center, Huawei Technologies Düsseldorf GmbH

January 2024

Dedication

To my beloved parents for their unwavering support, endless encouragement and confident belief throughout my journey.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Prof. Bruno Crispo and Prof. Theo Dimitrakos for their profound expertise and guidance throughout this research.

Secondly, a special thank you to Prof. Marco Calautti for his invaluable contributions to this work.

Thirdly, I would like to extend my heartfelt appreciation to my esteemed colleagues at the University of Trento and Huawei Munich Research Centre for their insightful discussions and valuable feedback, which greatly enriched this study.

Lastly, but certainly not least, I am forever indebted to my beloved family and treasured friends for their unwavering support and companionship that eased this challenging journey.

Abstract

The main focus of this thesis is applied research targeting industrial applications of Usage Control (UCON) and policy-based governance. Nonetheless, we also tackle an associated core problem to address the diverse requirements of the targeted application domains. The core research problem is three-fold. (1) UCON enacts usage control in a fixed lifecycle of three temporal phases: `pre`, `ongoing` and `post`. However, emerging security paradigms require custom and finer-grained lifecycles with phases and transitions tailored for the application domain. For example, data hub applications entail data-oriented usage control throughout the different stages of the data lifecycle (e.g., `collection`, `retention`, `processing` and `destruction`). Therefore, policy systems must enable custom lifecycles to accommodate a wide variety of applications. (2) Although UCON allows attribute values to change and updates usage decisions accordingly, it does not specify a mechanism to govern attribute values. This becomes necessary in decentralised environments where attributes are collected from external parties that are not necessarily trusted. For this reason, policy systems must incorporate a mechanism to govern attributes, prepare them for policy evaluation and ensure their trustworthiness. (3) Due to its widespread adoption, UCON has been extended and adapted for diverse purposes, leading to a proliferation of frameworks. While these variations added significant contributions in their respective fields, they lack comprehensiveness and generality. Therefore, a unified solution is needed to encompass the existing variations of UCON as well as future applications. By addressing these core problems, we aim to leverage policy-based governance in the following four industrial applications: (1) Industrial/International Data Spaces (IDS), (2) data hubs, (3) smart vehicles, and (4) credential transformation.

To address these challenges and fulfil our applied research goals, we present six contributions in this thesis. (1) We propose UCON+: an extensible model that extends beyond traditional access and usage control providing a comprehensive framework for policy-based governance. UCON+ builds on the same foundations of UCON, making it an attribute-based model that incorporates continuous monitoring and policy re-evaluation.

However, it only defines general structures and common functions, and outlines extensible behaviour to be implemented by concrete extensions. Specifically, UCON+ allows concrete extensions to govern attribute values and updates, and to specify custom lifecycles tailored for their respective requirements. (2) We introduce a general-purpose policy engine that implements the UCON+ model. The engine conserves an Attribute-Based Access Control (ABAC) baseline using a standard policy language. The policy engine also introduces another type of policies used to govern attribute values, and to define and drive custom lifecycles. Thus, different extensions of UCON+ can be realised within the same policy engine using policies, eliminating the need for reimplementation. The policy engine leverages a modular architecture with an optimised implementation. (3) We demonstrate the use of the policy engine in a cloud service that provides an IDS for contract-based data exchange. We specifically used the policy engine and designed a custom lifecycle to govern and drive the contract negotiation between the data provider and data consumer using policies. We also used the policy engine to govern data usage based on the negotiated data sharing agreement. (4) We also showcase the policy engine in a data hub setting, where we leveraged it to track and govern data objects throughout their lifecycles. We designed a lifecycle that captures the different stages of the data lifecycle based on the General Data Protection Regulation (GDPR). We show how data usage is controlled at each stage of the lifecycle using policies. (5) We present a dynamic identity management and usage control framework for smart vehicles using the policy engine. We specifically introduce a policy-based Security Token Service (STS) that issues contextualised capabilities that specify what subjects are allowed to do within the vehicle. The STS also manages the capabilities throughout their lifecycles and revokes them if the corresponding policies are violated, while also taking safety measures into consideration. (6) Finally, we describe an application of the policy engine for policy-based credential transformation. Specifically, we introduce a policy-based credential bridge that exchanges, aggregates or maps credentials between different domains or regulatory frameworks. The bridge uses policies that specify how to transform or issue credentials according to the requirements of each domain.

Keywords

[Policy-Based Governance, Usage Control, Industrial Data Spaces, Data Hubs, Smart Vehicles, Credential Transformation, Self-Sovereign Identity]

Contents

Dedication iii

Acknowledgement v

Abstract vii

List of Contents ix

List of Acronyms xiii

List of Tables xvii

List of Figures xix

1 Introduction 1

1.1 Context . . . 1

1.2 Challenges and Objectives . . . 2

1.2.1 Core Research Problems . . . 3

1.2.2 Applied Research Goals . . . 5

1.3 Research Contributions . . . 8

1.3.1 Extensible Model for Policy-Based Governance . . . 8

1.3.2 Policy Engine . . . 10

1.3.3 Industrial Impact . . . 10

1.3.4 List of Publications . . . 12

1.4 Thesis Structure . . . 13

2 State of the Art 15

2.1 Introduction . . . 15

2.2 Access Control Models . . . 15

2.2.1 Discretionary Access Control (DAC) . . . 15
2.2.2 Mandatory Access Control (MAC) . . . 16
2.2.3 Role-Based Access Control (RBAC) . . . 16
2.2.4 Attribute-Based Access Control (ABAC) . . . 17
2.3 The Usage Control (UCON) Model . . . 17
2.4 Policy Languages . . . 22
2.4.1 Open Digital Rights Language (ODRL) . . . 23
2.4.2 Open Policy Agent (OPA) Rego . . . 24
2.4.3 eXtensible Access Control Markup Language (XACML) . . . 24
2.4.4 Abbreviated Language For Authorisation (ALFA) . . . 26
2.5 Related Work . . . 27
2.5.1 Trusted Context Transformation . . . 27
2.5.2 Industrial/International Data Spaces (IDS) . . . 28
2.5.3 Cloud Data Hubs . . . 29
2.5.4 Smart Vehicles . . . 30
2.5.5 Credential Transformation . . . 31
2.5.6 Summary . . . 33

3 UCON+: An Extensible Model for Policy-Based Governance 37
3.1 Introduction . . . 37
3.2 Evaluation Session Components . . . 38
3.2.1 Evaluation Context . . . 39
3.2.2 Evaluation Lifecycle . . . 39
3.2.3 Access Control Policy . . . 41
3.2.4 Context Transformation Policy . . . 42
3.2.5 Lifecycle Transition Policy . . . 43
3.3 Evaluation Session Execution . . . 43
3.4 Model Coverage . . . 47
3.4.1 Constructing ABAC Using UCON+ . . . 47
3.4.2 Constructing UCON Using UCON+ . . . 48

4 UCON+ Policy Engine 51
4.1 Introduction . . . 51
4.2 Policy Language . . . 52
4.2.1 Access Control Policies . . . 52
4.2.2 Context Transformation Policies . . . 54

4.2.3 Lifecycle Transition Policies . . . 55
4.3 Architecture . . . 56
4.3.1 Components . . . 58
4.3.2 Components Interactions . . . 60
4.4 Experimental Performance Analysis . . . 62
4.4.1 Overhead of Access Control Policy Evaluation . . . 64
4.4.2 Overhead of Context Transformation and Lifecycle Transition Policies 65

5 Industrial Applications of the Policy Engine 69
5.1 Introduction . . . 69
5.2 Industrial/International Data Spaces (IDS) . . . 69
5.2.1 Contract Negotiation . . . 71
5.2.2 Contract Enforcement . . . 74
5.2.3 Architecture . . . 77
5.3 Cloud Data Hub . . . 78
5.3.1 Data Lifecycle . . . 80
5.3.2 Data Governance Policies . . . 83
5.3.3 Architecture . . . 86
5.4 Smart Vehicles . . . 89
5.4.1 SIUV Security Token Service (STS) . . . 90
5.4.2 SIUV Architecture . . . 92
5.4.3 Use-cases . . . 95
5.5 Credential Bridge . . . 101
5.5.1 Types of Credential Transformation . . . 101
5.5.2 Transformation Lifecycle . . . 103
5.5.3 Architecture . . . 104
5.5.4 Use-cases . . . 105

6 Conclusion 111
6.1 Summary . . . 111
6.1.1 Challenges and Objectives . . . 111
6.1.2 Contributions . . . 112
6.2 Future Directions . . . 113

Bibliography 115

List of Acronyms

AA Attribute Authority

ABAC Attribute-Based Access Control

ADP Administration and Delegation Profile

AI Artificial Intelligence

ALFA Abbreviated Language For Authorisation

API Application Programming Interface

AQ Attribute Queue

CoAP Constrained Application Protocol

DAC Discretionary Access Control

DFA Deterministic Finite Automaton

DID Decentralised Identifier

DLT Distributed Ledger Technology

DRM Digital Rights Management

EMR Electronic Medical Record

ESSIF European Self-Sovereign Identity Framework

EU European Union

EUDIW European Union Digital Identity Wallet

GDPR General Data Protection Regulation

HIPAA Health Insurance Portability and Accountability Act

IAM Identity and Access Management

IdP Identity Provider

IDS Industrial/International Data Spaces

IoT Internet of Things

IoV Internet of Vehicles

MAC Mandatory Access Control

MQTT Message Queuing Telemetry Transport

OASIS Organization for the Advancement of Structured Information Standards

ODRL Open Digital Rights Language

OIDC OpenID Connect

OM Obligation Manager

OOP Object-Oriented Programming

OPA Open Policy Agent

PAP Policy Administration Point

PDP Policy Decision Point

PEP Policy Enforcement Point

PIP Policy Information Point

Pub/Sub Publish/Subscribe

RBAC Role-Based Access Control

SAML Security Assertion Markup Language

SM Session Manager

SOA Service-Oriented Architecture

SPAP Session PAP

SPDP Session PDP

SSI Self-Sovereign Identity

STS Security Token Service

TM Trust Management

UCON Usage Control

UCS Usage Control System

VC Verifiable Credential

VDR Verifiable Data Registry

VP Verifiable Presentation

W3C World Wide Web Consortium

XACML eXtensible Access Control Markup Language

List of Tables

2.1 Summary of overlooked gaps by related work 34

4.1 Average time to parse and evaluate ALFA policies in UCON+ engine compared to XACML in Balana 64
4.2 Average time to evaluate already-parsed ALFA policies 65
4.3 Time to evaluate a hard-coded and a policy-based UCON session 67

List of Figures

2.1 UCON coverage 18
2.2 UCON evaluation session 18
2.3 UCON state model proposed by Zhang et al. [106] 20
2.4 UCON extended state model proposed by Katt et al .[52] 20

3.1 The inputs and output of the access control policy 41
3.2 The inputs and output of the context transformation policy 42
3.3 Example of a Deterministic Finite Automaton (DFA) 43
3.4 The inputs and outputs of the lifecycle transition policy 43
3.5 A valid step in the evaluation session 46
3.6 A valid execution of the evaluation session 46
3.7 An activity diagram illustrating the execution of the session 47
3.8 DFA illustrating the lifecycle of an ABAC evaluation 47
3.9 DFA illustrating the lifecycle of a UCON evaluation session 49

4.1 DFA illustrating the lifecycle of a UCON evaluation session 56
4.2 Architecture of the UCON+ policy engine 58
4.3 Attribute update flow 61
4.4 Flow of an evaluation session 63
4.5 Average time to parse and evaluate ALFA policies in UCON+ engine compared to XACML in Balana 65
4.6 Average time to evaluate ALFA policies in the UCON+ policy engine . . . 66
4.7 Time to evaluate a hard-coded and a policy-based UCON session 67

5.1 Technical components of an IDS 70
5.2 DFA illustrating the evaluation lifecycle of contract negotiation 71
5.3 DFA illustrating the evaluation lifecycle of governing data usage on the consumer side 75

5.4 Logical architecture of integrating UCON+ in IDS 79
5.5 Actors of the cloud data hub use-case 80
5.6 DFA illustrating the data lifecycle used in the cloud data hub application . 80
5.7 Logical architecture of the integration of UCON+ policy engine in the data hub 87
5.8 SIUV STS 91
5.9 DFA illustrating the lifecycle of an STS evaluation session 92
5.10 SIUV architecture 94
5.11 DFA illustrating the evaluation lifecycle of credential transformation 103
5.12 Architecture of the Credential Bridge 106

Chapter 1

Introduction

This introductory chapter sets the context of the thesis, describes the tackled research problems, highlights the corresponding contributions, and outlines the structure of the rest of the thesis.

1.1 Context

The ubiquity of the Internet and digital technologies have pervaded various aspects of human life including communication, transportation, healthcare, infrastructure and business. This resulted in an unprecedented surge and use of digital resources such as data, software and hardware, creating both opportunities and challenges. On the one hand, technologies such as the Internet of Things (IoT), Artificial Intelligence (AI) and big data analytics have provided automation as well as new knowledge and unique insights that were previously unattainable. On the other hand, the pervasive nature of such technologies has sparked significant privacy and security concerns fuelled by several incidents such as the Mirai botnet [5], Pegasus spyware [20] and Cambridge Analytica scandal [29]. As digital resources continue to be a valuable commodity, it is crucial to address the ensuing concerns and ensure the privacy, security and interests of individuals, organisations and governments.

Access control is one of the security mechanisms that have been traditionally used to protect digital resources. It specifies the *rights* that can be exercised by *subjects* on *objects* (i.e., resources), and selectively restricts access to resources accordingly. Access control evolved from coarse-grained models such as Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role-Based Access Control (RBAC) to finer-grained models such as Attribute-Based Access Control (ABAC). Nonetheless, access

control assumes that access rights do not change while resources are being used. This means that access rights are evaluated only when access is requested, and no further control is provided once access is granted. As technologies spread over heterogeneous and dynamic environments where changes are frequent, a more dynamic approach has become prevalent. For this reason, Park and Sandhu introduced Usage Control (UCON) [76] as a generalisation of access control that encompasses preceding models and extends them with attribute mutability and continuous monitoring. The novelties of UCON enable revoking access to resources when changes occur and the security policy is no longer satisfied. In addition, UCON goes beyond access control to cover Digital Rights Management (DRM) and Trust Management (TM) creating a comprehensive and unified framework for the three different but closely related areas. UCON gained a lot of attention in the research community and was adapted and adopted to enable dynamic and long-lasting authorisations in different areas such as cloud, grid computing and IoT.

In this thesis, we argue that despite the advancements and adaptations of UCON, it still lacks an approach for handling the recent challenges brought by emerging security paradigms such as data sovereignty. These paradigms go beyond conventional access and usage control (i.e., access rights management) to incorporate broader policy-based governance that specifies how an entity (e.g., subject, object, process, etc.) should be generally handled in a specific context. We also highlight the lack of a unified framework for the various variations of UCON that address slightly different requirements. To address these challenges, we propose an extensible model for policy-based governance that captures the common behaviours and general structures of policy-based solutions, while allowing further extensions and refinement as required by each application domain. This abstract and extensible nature allows the model to encompass not only conventional access and usage control but also broader policy-based governance applications. More importantly, we target several industrial applications of UCON and policy-based governance. We specifically introduce an optimised general-purpose policy engine that implements the novelties of the proposed model. We then demonstrate the use of the policy engine in several prototypes corresponding to the targeted application domains.

1.2 Challenges and Objectives

The main purpose of this thesis is applied research that targets different industrial applications of UCON and policy-based governance. Nonetheless, we also tackle a core problem to methodically address the diverse requirements of the targeted application domains. In this section, we outline the research challenges and objectives while categorising them as

core research problems and *applied* research goals. We specifically describe the need for broader policy-based governance that goes beyond conventional access and usage control, and highlight the limitations of UCON in meeting such requirements. We also point out the need for a unified solution that encompasses and surpasses the various variations of UCON that address different requirements. Subsequently, we discuss the applications of UCON and policy-based governance in several target domains.

1.2.1 Core Research Problems

Access and usage control are authorisation-oriented approaches concerned with deciding whether a specific subject is allowed to exercise a specific right on a specific object (i.e., resource). Put differently, they can be described as an answer to the following yes/no question: *is subject s allowed to exercise right r on object o?* To effectively address our applied research objectives, we recognise the need to tackle a core problem, that is to transcend this conventional definition of access and usage control. This is because emerging security paradigms (e.g., data sovereignty) require a broader policy-based governance that answers a more general question of *how shall a specific entity (e.g., subject, resource, credential, etc.) be handled?* In this context, we identify four main challenges as follows.

Custom Evaluation Lifecycles

UCON extends ABAC with attribute mutability, continuous monitoring as well as the ability to revoke authorisations. It specifically introduced the concept of a long-lived authorisation that spans over a period of time, during which policies are *re-evaluated* whenever a change occurs, and the authorisation decision is updated accordingly. We designate such long-lived authorisations as *Evaluation Sessions*[1]. UCON evaluation sessions are enacted through a fixed *lifecycle* of three temporal phases named `pre`, `ongoing` and `post`. The `pre` phase indicates that resource usage was requested but has not been granted yet. The `ongoing` phase denotes that usage has been granted and is in progress The `post` phase implies that usage has ended or been revoked. A different set of policies or rules is applied at each of these phases to respectively determine whether usage can be granted, whether usage can be maintained, and the actions that must be taken after usage ends.

We argue that this lifecycle lacks enough expressiveness and comprehensiveness for the following reasons:

[1]The term "session" here does not refer to a user or web session as commonly used. Instead, it denotes a specific long-lived process of policy evaluations in response to a specific request.

(1) UCON's lifecycle is fixed, coarse-grained and limited to three temporal phases only. In addition, transitions between UCON's phases depend on the evaluation decision only. In contrast, new security paradigms necessitate finer-grained lifecycles with phases and transitions based on a variety of contextual information other than the evaluation decision. For instance, evaluation sessions in a data hub scenario must be based on the different stages of the *data lifecycle* and transitions between phases may depend on contextual information related to the data [55, 86]. The General Data Protection Regulation (GDPR) [78] specifies a data lifecycle that involves the following four phases: `collection`, `retention`, `processing` and `destruction`. Likewise, dynamic identity management applications entail evaluation sessions with lifecycles based on credential lifecycle stages such as the ones presented in [49] (i.e., `issue`, `use` and `revoke`).

(2) UCON revokes and terminates authorisations when the security policy is no longer satisfied. Consequently, the lifecycle transitions from the `ongoing` phase to the `post` phase. However, some use-cases need to temporarily withhold an authorisation rather than completely revoking it. For example, changes in user behaviour or trust level scores may not necessarily entail a termination of authorisation, but rather withholding it to allow the user to improve their score [35].

(3) UCON's lifecycle enacts continuous monitoring in the `ongoing` phase only (while usage is in progress), while new paradigms require continuous monitoring before authorisation and/or after revoking it. For instance, data usage applications require continuous monitoring before granting authorisations to process data (e.g., while data is being collected) [85]. Similarly, safety critical systems (e.g., smart vehicles) require monitoring after revoking authorisations to ensure that safety measures are taken [44].

Therefore, policy systems need to enable customising and fine tuning the evaluation lifecycle to overcome these limitations and accommodate a wide variety of applications.

Trusted Context Transformation

UCON introduced that concept of attribute mutability allowing the evaluation context to change throughout the evaluation session. UCON reacts to such contextual changes by re-evaluating the security policy and updating the decision accordingly. However, existing UCON frameworks do not include a mechanism to govern attribute updates and the evaluation context. They rather rely on trusting attribute sources. Data sovereignty

applications and decentralised deployments, however, entail collecting information from external parties that are not necessarily trusted [8]. Therefore, the policy engine must be able to govern attribute values based on predefined rules to ensure the trustworthiness and the correctness of the evaluation context. We designate this ability as "*Trusted Context Transformation*". It allows the policy engine to govern changes in the evaluation context and set attribute values before and after policy evaluation.

Unified Solution

UCON has gained a lot of popularity in the last decade and many works have enhanced, adapted and leveraged it for various purposes and use-cases. Many UCON frameworks have been introduced with different behaviours and slight variations. For instance, several researchers proposed different state models of UCON evaluation sessions [13, 52, 106]. Similarly, different policy languages have been proposed and used to express UCON policies [23, 106, 42, 95]. While these works added significant contributions in their respective fields, they lack comprehensiveness and generality as they are contingent on the targeted application domains. Consequently, applying any of these solutions in alternative application domains requires rebuilding the whole system. Therefore, a unified solution is needed to encompass the existing variations of UCON as well as future applications.

Model-Policy Decoupling

Existing UCON frameworks do not distinguish between policy authoring and the logic and behaviour of the underlying model. Consequently, policy authors need to understand the details of the model such as the lifecycle and how it transitions between phases in order to write usage policies. This results in a tight coupling between usage policies and the UCON framework, making policy authoring more error-prone. Therefore, a unified solution must decouple the logic and behaviour of the policy framework from the use-cases and usage policies. Such decoupling enables specialists to design models or frameworks for a class of use-cases having common behaviour (e.g., lifecycle), allowing policy authors to focus only on writing access control policies without the need to understand the details of model.

1.2.2 Applied Research Goals

As mentioned above, the purpose of this thesis is the industrial application of UCON and policy-based governance. Our objective is to design and implement an optimised

and general-purpose policy-engine, and utilize it in different application domains. The main focus is data sovereignty and security, but other applications are also considered including identity and credential management as well as smart vehicles. In this regard, we specifically describe four applications that are targeted as part of this research.

Industrial/International Data Spaces (IDS)

In recent years, the notion of "data sovereignty" has emerged as a new paradigm that enables legal entities (e.g., individuals, organisations, etc.) to govern the use of their data [26, 47]. It has gained a lot of attention and has been identified as a key objective in the European Union (EU) digital strategy [25, 63]. In light of the growing focus on data sovereignty, several technological initiatives have emerged to enable secure and sovereign data exchange [47, 80, 56]. Among these, the Industrial/International Data Spaces (IDS) stands out as an ecosystem for standardised and contract-based data sharing across organisational boundaries. It allows parties to negotiate a legally binding data sharing agreement that specifies the rules and conditions for sharing and using data as well as the obligations of each party. This enables data owners to govern the use of their data, fostering trust, transparency and sovereignty between data providers and consumers [75, 79]. The negotiation process involves an exchange of offers and counteroffers until a mutual agreement is reached [48]. However, the IDS protocol does not specify how to automate the negotiation process. As part of our applied research goals, we tackle this problem by leveraging our policy engine to drive the contract negotiation process based on policies that specify offers and counteroffers. Moreover, the IDS architecture identifies UCON as a key technology to govern and control data usage based on policies derived from data sharing agreements [75]. Accordingly, we also aim to use our policy engine to control and govern data usage and sharing in a data space as a service.

Cloud Data Hubs

With the exponential growth of data volume and complexity, data hubs emerged as centralised repositories that provide efficient storage, management and processing of large volumes of data from various sources. Data governance and usage control are imperative in such ecosystems to safeguard sensitive data, protect the privacy and interests of data owners, and ensure compliance with regulations. Therefore, data objects must be tracked and governed within the data hub throughout their lifecycles from collection to retention to deletion. In addition, data usage policies and rules must be tailored for and enforced at each stage of the data lifecycle [3, 36]. However, there is no global and comprehensive

data lifecycle model, as this depends on several factors, including data types, use-cases or regulations, thereby requiring customising the lifecycle for each application [99]. In this thesis, we aim to address these requirements within a data hub setting using our policy engine, and demonstrate data usage control throughout the data lifecycle specified by the GDPR.

Smart Vehicles

The exponential growth of the Internet of Vehicles (IoV) innovations, such as autonomous driving, driver assistance, vehicle connectivity, infotainment and shared mobility has recently gained a notable attention in the automotive industry. Such services and innovations are supported by the integration of smart sensors as well as the shift from electromechanical to interconnected software-based systems in modern vehicles. In addition, the automotive industry has witnessed an increased adoption of Service-Oriented Architectures (SOAs) in smart vehicles due to their flexibility [15, 30]. This, however, comes at the expense of the simple security of the statically predefined functions of traditional vehicular architectures. Accordingly, several automotive security and safety challenges have been identified by researchers. This includes dynamic identity management as well as continuous monitoring and usage control of resources. These aspects are crucial because vehicles are transforming into digital systems where applications are provided by third party providers. Besides, vehicles are real-time mobile systems whose environmental conditions change continuously as they move [88, 12, 16]. One of our objectives in this thesis is to address these challenges by introducing a policy-based vehicular Identity and Access Management (IAM) that provides dynamic and stateful identity management as well as continuous usage control.

Credential Transformation

Self-Sovereign Identity (SSI) has emerged as a paradigm that enables entities to control their digital identities and credentials [68, 104]. It eliminates centralised authorities and leverages decentralised technologies such as Decentralised Identifiers (DIDs) [101] and Verifiable Credentials (VCs) [100]. The EU actively promotes SSI as part of its digital strategy [25] through initiatives such as European Self-Sovereign Identity Framework (ESSIF) and European Union Digital Identity Wallet (EUDIW) [32, 24, 25]. Similarly, digital sovereignty ecosystems such as Gaia-X and IDS incorporate SSI as a core component in their architectures [31, 75]. This contributes to the inevitable rise of SSI, which introduced new challenges as organisations have invested in legacy credentials and

identity systems. Therefore, SSI needs to coexist with legacy systems and interoperate with them, which entails credential transformation between different systems. Gaia-X specifically highlights this as a requirement in their architecture [34]. More importantly, different SSI systems may be governed by various regulatory frameworks, making credential transformation imperative for recognition of credentials in different frameworks. In this regard, credential transformation must be dynamic and context-dependent as different domains and countries have different requirements, standards, and regulations. For instance, a university degree VC must be transformed to be recognised by another university in a different country. The transformation must adapt to the specific regulations of the destination university, which may require additional information such as transcripts or accreditation credentials of the university that issued the degree. The transformation must also consider the mapping between the educational systems of both the source and destination universities including details such as credit hours and grading scales. Another objective of this thesis is to tackle these challenges using a policy-based credential bridge that maps, aggregates or transforms credentials based on predefined policy rules.

1.3 Research Contributions

This section provides an overview of the contributions of this thesis, aimed at addressing the challenges discussed in Section 1.2. Our contributions are two-fold: (1) We propose a new model for policy-based governance to solve the core research problems discussed in Section 1.2.1; (2) We introduce a general-purpose policy engine that implements the novelties of the proposed model and fulfils the applied research goals outlined in Section 1.2.2. In addition to the contributions, we describe prototypes that leverage the policy engine, showcasing the industrial impact of this research in the application domains outlined in Section 1.2.2. We also outline the corresponding list of publications.

1.3.1 Extensible Model for Policy-Based Governance

We described in Section 1.2.1 the core problems we aim to address in this thesis. We specifically highlighted the need for custom evaluation lifecycles and trusted context transformation as well as a comprehensive and unified solution that covers existing variations of UCON as well as future applications. To solve these problems, we propose UCON+: an extensible model that extends beyond traditional access and usage control providing a comprehensive framework for policy-based governance. UCON+ builds on the same foundations of UCON, so it is also based on attributes and incorporates session-based moni-

toring and policy re-evaluation. However, it only defines general structures and common behaviour, with detailed aspects left to be specified by concrete extensions tailored for specific requirements. To provide a more intuitive understanding of the extensibility of the model, we draw an analogy to an abstract class in Object-Oriented Programming (OOP). The same way an abstract class defines common properties and abstract function signatures, UCON+ provides a general specification. Also, the same way subclasses extend the abstract class with concrete implementations, extensions of UCON+ extend it with further specifications tailored for their requirements.

UCON+ is concerned with modelling the evaluation session and its enactment. It defines the evaluation session as a series of access control policy evaluations adhering to a lifecycle during which the context (i.e., attribute values) changes. UCON+ models the lifecycle as a Deterministic Finite Automaton (DFA), leaving the specification of its states and transitions to concrete extensions. This allows different extensions of UCON+ to customise the lifecycle for their requirements. Moreover, context transformation and lifecycle transitions in UCON+ are rule-driven such that conditional rules govern attribute values and specify when to fire lifecycle transitions. Concrete extensions of UCON+ can then specify different rules resulting in a distinct behaviour tailored for the corresponding requirements. UCON+ also does not assume a specific enforcement system or policy language. Instead, it provides an *abstract* definition of policies, enabling existing policy systems and languages to be easily used in concrete extensions of the model.

The abstract and rule-driven nature of UCON+ solves the problems described in Section 1.2.1 as they enable customising the evaluation lifecycle and governing attribute updates to provide a trusted context transformation. Therefore, UCON+ can be used as a unified solution that can be extended and tailored for different use-cases, where extensions only have to specify the rules that govern the context and drive the lifecycle. Moreover, the model decouples policy authoring from the specification of concrete extensions. Specifically, specialists can use the model to design concrete extensions that are tailored for a group or class of use-cases, focusing only on the behaviour of the extension (e.g., the lifecycle). On the other hand, policy authors can focus only on writing access control policies for each phase of the lifecycle without the need to know how the concrete extension works or how transitions between phases occur. The novelties and versatility of UCON+ become clearer in Section 1.3.2 where we introduce and describe a policy engine that implements the model.

Clearly, UCON+ introduces a high level of abstraction that may cause confusion and ambiguity. For this reason, we provide rigorous description of the model along with auxiliary notions and annotations to clearly communicate the concepts and avoid unambiguous

understanding. However, we highlight that we do *not* provide a formal specification as this is out of the scope of this thesis, which is focused on applied research. The formal model will rather be provided in another ongoing and complementary project and parts of that work have already been published in [94].

1.3.2 Policy Engine

To realise the goals outlined in Section 1.2.2, we introduce a general-purpose policy engine that implements the novelties of the UCON+ model. The engine integrates an ABAC baseline that uses the Abbreviated Language For Authorisation (ALFA) policy language. ALFA is a profile of eXtensible Access Control Markup Language (XACML) that preserves the same semantics with a significantly simpler and more human readable syntax [72]. It is currently in the process of standardisation by the Organization for the Advancement of Structured Information Standards (OASIS) [89]. The engine also introduces another type of policies, designated as *extension* policies, used to govern contextual changes and drive the evaluation lifecycle based on conditional rules as specified by the model. Therefore, the engine implements the common behaviour and general structures of the UCON+ model, and leaves the extensible behaviour to be specified in extension policies. This allows different concrete extensions of UCON+ to be realised and enacted using the same policy engine without the need for reimplementation. The policy engine also leverages ALFA to specify extension policies by exploiting ALFA obligations as demonstrated Chapter 4. The policy engine leverages a modular architecture based on the XACML architecture [73]. It consists of a core implemented in C++ and optimised for device environments, in addition to a wrapper implemented in Go and optimised for cloud applications. As we show in Chapter 5, the policy engine was integrated in several prototypes, showcasing its general-purpose nature as well as the extensible nature of the UCON+ model.

1.3.3 Industrial Impact

We demonstrate the value and impact of the UCON+ model and policy engine in four industrial applications.

Industrial/International Data Spaces (IDS)

We demonstrate the UCON+ policy engine in Boot-X: a cloud-based implementation of IDS enabling contract-based data exchange. This is highlighted in the Boot-X open-source

architecture, which shows the UCON+ policy engine as part of its stack [2]. We specifically used the policy engine to govern contract negotiation between the data provider and consumer using policies. We designed a lifecycle that captures the different stages of the negotiation process. Policies were used to decide whether to accept or reject offers, or to propose counter offers throughout the negotiation process. Moreover, the policy engine was used to govern data usage and enforce policies according to the negotiated data sharing agreement. We discuss the designed lifecycle as well as the policies used to govern data usage and contract negotiation. We also describe how the policy engine is integrated in the cloud service, and how it is used on both the data provider and data consumer sides.

Cloud Data Hubs

We showcase the use of the UCON+ policy engine to provide data-oriented usage control in a cloud data hub used to store and process personal data. This addresses the challenges and objectives in Section 1.2.2 as the engine is used to track and govern data objects throughout their lifecycles. We specifically constructed a concrete extension of UCON+ with an evaluation lifecycle that captures the different stages of the data lifecycle. The lifecycle was inspired by the GDPR and consists of four data-related phases: `collection`, `retention`, `processing` and `destruction`. We describe how the engine governs data access and usage at each phase of the lifecycle using policies. We also show how policies take data subject rights into consideration such as the right to be forgotten or to restrict processing. In addition, we present the architecture of the data hub and describe how the policy engine is integrated in the architecture and how policies are enforced.

Smart Vehicles

To address the challenges and objectives described in Section 1.2.2, we introduce a vehicular IAM system that provides dynamic and stateful credential management as well as continuous monitoring and usage control. It incorporates a policy-based Security Token Service (STS) that issues and manages contextualised capabilities in the form of VCs. Specifically, the STS collects and verifies external identity VCs (e.g., driving license) issued by Identity Providers (IdPs) and exchanges them with internal VCs that determine the capabilities of the corresponding subjects within the vehicle (e.g., driving capability). The policy-based nature of the STS allows it to dynamically specify capabilities of a subject in a particular context based on the subject's VCs as well as environmental

[2] `https://boot-x.eu/open-source-architecture/`

conditions. Moreover, session-based monitoring and policy re-evaluation enable the STS to manage capabilities throughout their lifecycle and to revoke them if necessary when the conditions of the vehicle change. We leveraged the extensibility of UCON+ to specify the lifecycle of capabilities from their issuance to their revocation or expiry. We also show how STS policies can specify safety measures to be taken upon or before revoking capabilities to ensure graceful revocation. In addition to the STS, the IAM consists of several instances of the policy engine used to protect resources within the vehicle and enforce access and usage control. The capabilities issued by the STS are used in the access and usage control policies of these instances to make authorisation decisions. Finally, we describe three use-cases that demonstrate the introduced IAM.

Credential Transformation

We present a policy-based credential bridge that tackles the challenges and goals discussed in Section 1.2.2. The credential bridge leverages the UCON+ policy engine to transform credentials across different domains or regulatory frameworks according to policies that specify the requirements of each domain. The bridge uses policy-based decision making to determine how to transform credentials, what claims to include, and whether it needs to fetch additional credentials in the process. For instance, to equate a university degree credential in another country, bridge policies may specify that the accreditation credential of the foreign university must be fetched and verified. The policy-based decision making of the bridge enables adaptive and seamless interoperability between different frameworks, and facilitates the transition towards SSI. We describe in this thesis how UCON+ was leveraged to design and implement the credential bridge. We present two use-cases of the credential bridge one involving equating university credentials and one involves transforming credentials to SSI in Boot-X as illustrated in its open-source architecture[2]. We also discuss the evaluation lifecycle as well as the policies used by the bridge.

1.3.4 List of Publications

First-Authored Publications

1. Ali Hariri, Subhajit Bandopadhyay, Athanasios Rizos, Theo Dimitrakos, Bruno Crispo, and Muttukrishnan Rajarajan. SIUV: A Smart Car Identity Management and Usage Control System Based on Verifiable Credentials. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 36–50. Springer, 2021. Published. URL: `https://doi.org/10.1007/978-3-030-78120-0_3`

2. Ali Hariri, Amjad Ibrahim, Theo Dimitrakos, and Bruno Crispo. WiP: Metamodel for Continuous Authorisation and Usage Control. In *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*, pages 43–48, 2022. Published. URL: `https://doi.org/10.1145/3532105.3535039`

Co-Authored Publications

1. Ali Hariri, Amjad Ibrahim, Bithin Alangot, Subhajit Bandopadhyay, Alessandro Rosetti, Antonio La Marra, Hussein Joumaa, and Theo Dimitrakos. UCON+: Comprehensive Model, Architecture and Implementation for Usage Control and Continuous Authorization. In *Collaborative Approaches for Cyber Security in Cyber-Physical Systems*, pages 209–226. Springer, 2023. Published. URL: `https://doi.org/10.1007/978-3-031-16088-2_10`

2. Subhajit Bandopadhyay, Theo Dimitrakos, Yair Diaz, Ali Hariri, Tezcan Dilshener, Antonio La Marra, and Alessandro Rosetti. DataPAL: Data Protection and Authorization Lifecycle framework. In *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–8. IEEE, 2021. Published. URL: `https://ieeexplore.ieee.org/abstract/document/9566212`

1.4 Thesis Structure

The rest of this thesis is organised as follows: Chapter 2 describes the state of the art literature as well as an analysis of the identified research gaps; In Chapter 3, we propose a new model for policy-based governance that addresses the core research problems; We introduce a general-purpose and optimised policy engine that implements the novelties of the proposed model in Chapter 4, where we also describe the architecture and implementation of the engine; Chapter 5 discusses industrial applications of the policy engine and elaborates on prototypes that target different application domains. Finally, we draw conclusions and identify future directions in Chapter 6.

Chapter 2

State of the Art

This chapter describes the state of the art and outlines related work.

2.1 Introduction

In this chapter, we provide a comprehensive overview of the current state of the art covering both the historical developments as well as the latest advancements. Given that the objectives of this thesis involve addressing challenges and applications of Usage Control (UCON) and policy-based governance, we begin by describing the evolution of access and usage control models. Specifically, we briefly outline traditional access control models then discuss the UCON model highlighting its novelties and advancements. Subsequently, we present an overview of the different policy languages used to express usage control policies. Finally, we provide an overview of related work in the field and identify the corresponding gaps.

2.2 Access Control Models

This section provides a brief overview of traditional access control models that preceded UCON.

2.2.1 Discretionary Access Control (DAC)

Discretionary Access Control (DAC) is an access control model that allows owners to manage access rights to their files (e.g., file, directory or object). Thus, resource owners

have the *discretion* to issue policies that determine which users or groups are granted access to the resource and what level of access they are given (i.e., read, write, execute). The simplicity of DAC makes it ideal for small-scale systems such as personal computers and operating systems. Different variations of DAC were proposed to address different subjects like administrative delegation or access revocation. Nonetheless, DAC cannot cope with the complex requirements of larger systems and organisations [11, 91, 74].

2.2.2 Mandatory Access Control (MAC)

MAC manages access rights and determines access decisions based on security labels that are assigned to subjects and resources. Security labels refer to the sensitivity of resources and the security clearances of the subjects. Access is granted only if the label of the subject (i.e., clearance) is equivalent to or higher than the label of the resource (i.e., sensitivity). For instance, a subject with a `confidential` clearance is allowed to access resources with `public` sensitivity, but denied to access resources with label `top secret`. This allows the dynamic creation and destruction of entities (i.e. subjects and objects) since the control is based on the levels not on the explicit identities of entities. In contrast, security levels cannot be created or destroyed dynamically but rather need administrative actions. Therefore, MAC access rights are managed by a central administration in contrast to DAC, where access decisions are largely at the discretion of the resource owner. MAC is particularly effective in organisations with complex hierarchy of security clearances and a need to enforce strict separation of duties [11, 92, 74].

2.2.3 Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) was introduced to address the administration complexity of the aforementioned models by decoupling subjects and access rights and associating permissions with roles. Thus, roles can be considered as the core building blocks of RBAC access policies as they connect subjects to access permissions. Roles represent job functions, authorities or responsibilities and are assigned to subjects. For instance, role “manager” has read access to file “sensitive.txt”, thus all users that are assigned the role of manager will have a read right on this file. This makes access control more efficient because roles are typically more stable and do not change frequently, unlike subjects that may change often. For example, the permissions granted to a manager role in an organisation are preserved for extended time spans, while managers may frequently join and leave. Therefore, RBAC can be considered as a set of relationships between roles and subjects, and between roles and permissions. Additionally, some variations may

also establish relationships between roles themselves, which would support concepts like inheritance, separation of duty and delegation of authority [28, 93].

2.2.4 Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC) enables finer-grained access control by making authorisation decisions based on attributes rather than roles or levels. Attributes refer to different characteristics of the subject requesting access (e.g., name, role), the object to be accessed (e.g., file type, data type) and the access environment (e.g., time, location). This adds more flexibility and agility over preceding models as attribute values can change frequently and authorisation decisions can be made based on a variety of conditions and situations. This is useful for highly dynamic environments such as cloud-based systems or distributed networks where other access control models may be too rigid or limited. Some ABAC specifications also introduce obligations and advice, where obligations are actions that must be fulfilled while advice are recommended actions to be taken with the authorisation decision [21].

2.3 The Usage Control (UCON) Model

The UCON model [76] was proposed as a convergence of access control, Trust Management (TM) and Digital Rights Management (DRM) as shown in Figure 2.1. UCON is based on attributes and integrates Authorisations (A), oBligations (B) and Conditions (C), which compose security policies. Authorisations are functional predicates over subject and object attributes, and must be satisfied to grant access rights; Obligations are mandatory actions that must be fulfilled to permit access; and Conditions are environmental or situational requirements that must be met to grant access rights. Unlike preceding models, UCON assumes that attribute values may change during usage of resources, and specifies that policies must be re-evaluated whenever attribute values change. This adds continuity of control as UCON monitors attribute values and revokes authorisations if the security policy is no longer satisfied [76, 60].

UCON adds a temporal aspect to authorisations as they span over a period of time during which resource usage is continuously monitored and regulated. We designate the lifetime of a single authorisation, from its creation to termination, as the *Evaluation Session*. The temporal state of UCON authorisations is captured in three main phases designated as `pre`, `ongoing` and `post` as shown in Figure 2.2. UCON decision predicates (i.e., policy rules) are classified by these phases such that: `pre` predicates are evaluated when

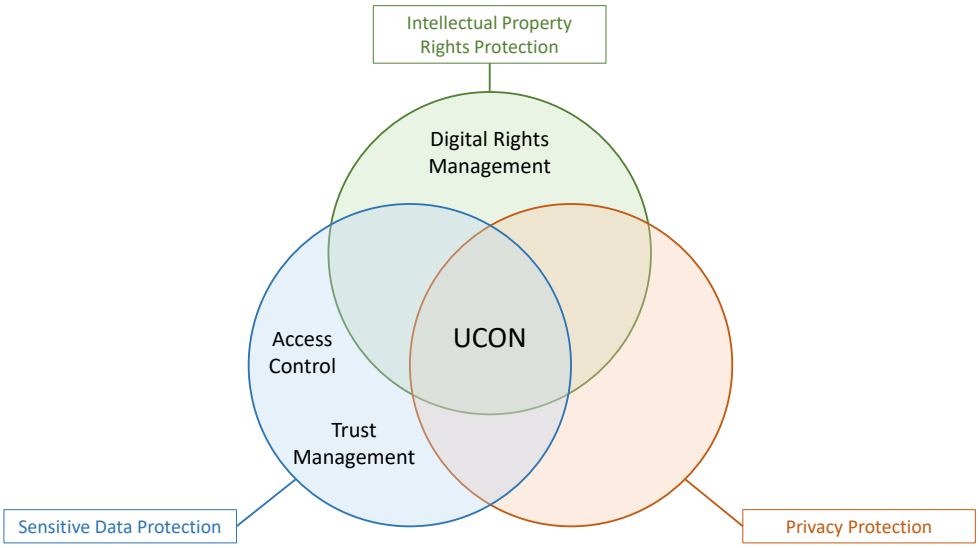

Figure 2.1: UCON coverage

resource is requested but has not been granted yet; `ongoing` predicates are re-evaluated whenever an attribute value changes while usage is in progress; and `post` predicates are evaluated upon the end or revocation of usage [76, 60].

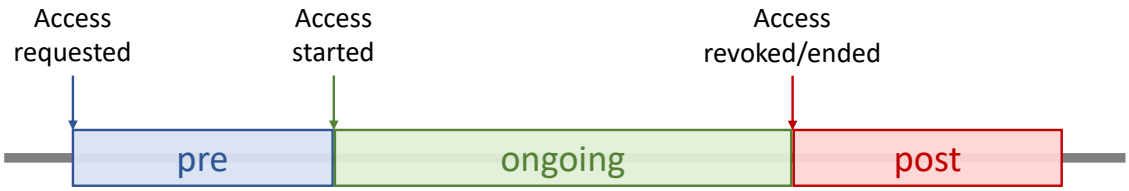

Figure 2.2: UCON evaluation session

The novelties of UCON make it an excellent baseline for long-lived authorisations and dynamic environments where continuous monitoring and future controls are required. For this reason, UCON has been identified as a key technology for data sovereignty ecosystems [31, 75]. For example, UCON can revoke data usage when the data owner withdraws their consent or changes the allowed usage purposes. Similarly, UCON can specify and enforce obligations such as *anonymise data before usage* or *delete data after three months of retention*. UCON may also define compensating actions to be taken upon violating regulations and data sharing agreements.

The UCON model has gained a lot of popularity in the last decade and many works have enhanced, adapted and leveraged it for various purposes and use-cases. For instance, Park et al. [77] defined a taxonomy for attribute management in the UCON model. They classified attributes as immutable and mutable, where immutable attributes require administrative actions to be changed while mutable attributes change as result of usage itself. Their taxonomy also defines subcategories under immutable and mutable attributes, but this is out of the scope of this thesis. Park et al. assume that environmental attributes do not change during usage, and that subject and object attribute updates occur only as a side-effect of usage (i.e., as part of the enforcement of the security policy or due to actions performed by subjects).

Zhang et al. [106] provided a logical specification and formalised the UCON model using an extension of Lamport's Temporal Logic for Actions (TLA). They define UCON policies as temporal logic formulas that are built from predicates about subject, object and environment attributes, as well as actions performed by the system or by subjects. They also defined the state model of UCON such that a single usage process transitions among the following six states: *initial*, *requesting*, *denied*, *accessing*, *revoked* and *end*, as shown in Figure 2.3. The *initial* state indicates that the usage request has not been generated yet. The *requesting* state means that the request has been generated and is waiting for the decision. The process transitions to the *denied* state if the system denies the request as per the security policy. Although the *denied* state may include attribute updates, it is considered a terminal state that terminates the usage process. If the system permits access, then the usage process transits into the *accessing* state, which implies that the subject is accessing the object. The *revoked* state is reached when the system revokes access due to a change in attribute values while subject is still accessing the resource. Finally, the usage process goes to the *end* state when the subject finishes usage.

Katt et al. [52] argued that this state model does not show the decision recheck state that occurs due to changes while usage is in progress. For this reason, they extended the state model by introducing the *ongoing check* state such that the system moves to this state when a change occurs while usage is in progress. Decision predicates are then re-evaluated and the system goes back to the *accessing* state if the access is still granted. Otherwise, the system transitions to the *revoked* state if access is revoked. The extended state model is shown in Figure 2.4 They also extended obligations to cover actions to be taken by the policy engine itself in contrast to the original UCON model, which assumes that obligations must be fulfilled by the requesting subject. For example, the security policy may define that a notification must be sent or a file must be deleted after the end of the usage session. In addition, their extended model distinguishes between trusted and

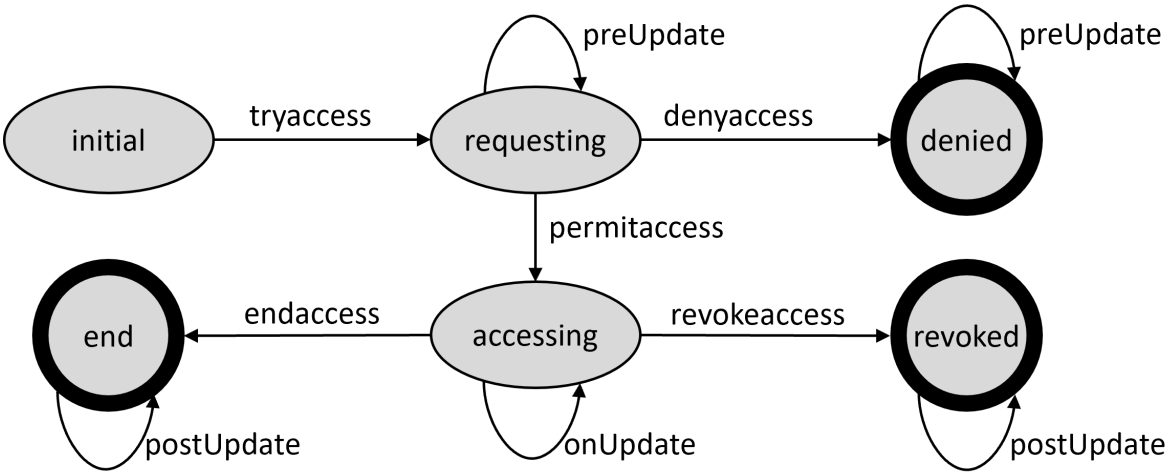

Figure 2.3: UCON state model proposed by Zhang et al. [106]

untrusted obligations, such that trusted obligations are system actions that do not need fulfilment check, while untrusted obligations must be checked to ensure their fulfilment.

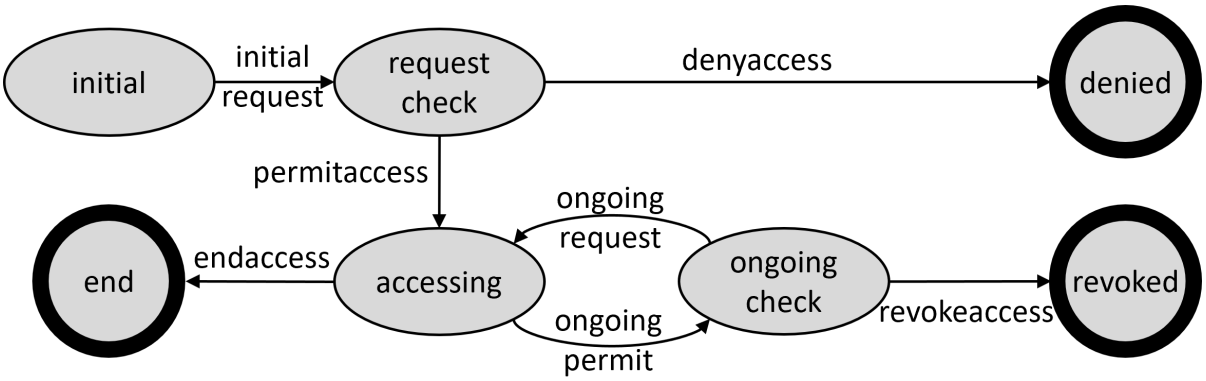

Figure 2.4: UCON extended state model proposed by Katt et al .[52]

Blower and Kotonya [13] adapted the UCON model to provide continuous authorisation decisions in real-time for IoT devices. Their model creates an authorisation session when a request is created, and the session transition between three main states: *initializing*, *accessing* and *advising*. UCON enacts `post` rules only if access was originally granted and then denied later. In contrast, Blower and Kotonya's model always enacts the advising state before terminating an evaluation session, even if access was denied in the initialising state. Their model is optimised for IoT devices with limited computing resources.

Baldi et al. [9] integrated the UCON model with the Apache Hadoop software to

provide an effective and efficient framework for safeguarding datasets in big data environments. Their framework enables continuous monitoring of usage of datasets and revokes the corresponding authorisation if the policy is violated. More importantly, the main novelty of their work is organising authorisation sessions in a tree hierarchy that captures dependencies between them. For instance, if an authorisation to access an API results in another authorisation to access a database, then the database authorisation is dependent on the API one. Accordingly, if one authorisation associated with a specific node in the hierarchy becomes invalid due to attribute updates, all authorisation within the corresponding subtree are also revoked. This allows the context of one authorisation to be propagated to all authorisations that depend on it.

Lazouski et al. [61, 23] introduced an enforcement framework for the UCON model while preserving a full ABAC baseline. They specified a policy language that captures UCON's three temporal states (i.e., `pre`, `ongoing` and `post`). They also devised an architecture that enables session-based monitoring of attributes as well as policy re-evaluation. Their framework was then leveraged in many use-cases in various application domains.

For instance, Carniani et al. [19] introduced an advanced authorisation service that utilises the aforementioned UCON framework to regulate the usage of Cloud resources, with a specific focus on Infrastructure as a Service (IaaS) offerings. Their proposed authorisation service tackles the challenge of prolonged resource usage by enabling the definition and continuous enforcement of usage control policies throughout the duration of resource access. Accordingly, the service can halt resource usage if the corresponding policy is no longer satisfied. To demonstrate their solution, they integrated the authorisation service with OpenNebula, a widely used software for running Cloud services.

Similarly, Lazouski et al. [62] leveraged the aforementioned UCON framework to regulate the usage of data once it has been downloaded onto mobile devices, specifically when it has been copied outside the original domain of the data producer. Their solution focuses on enforcing a usage control policy that is embedded within the data by the data producer. As a result, their solution surpasses traditional access control capabilities by having the ability to interrupt ongoing data access in Android devices if the policy is no longer met.

Likewise, La Marra et al. [58] leveraged the UCON framework to implement usage control functionalities in IoT systems. Their work focuses on a Smart Home environment and introduces two policies for energy saving and physical safety. Their solution also utilizes the CHORD-based Apache Cassandra Distributed Hash Table for distributed communication and data storage.

La Marra et al. [57, 65] integrated UCON with Open-Source MANagement and Or-

chestration (MANO) component of the ETSI Network Function Virtualisation (NFV) framework. MANO is used for orchestration and lifecycle management of physical and/or software resources that support infrastructure virtualization such as computing, networking and storage. The integration of UCON enables MANO to revoke resources in case a security-relevant change occurs (e.g., the integrity status of the virtual machines or the reputation of the user). This allows MANO to use policies to define and enforce countermeasures and to react to changes such as lifecycle events. However, UCON cannot capture the context of MANO lifecycle, nor can it participate in the management of the lifecycle. Moreover, the MANO lifecycle is reduced to UCON's `ongoing` phase only, thus UCON cannot apply fine-grained control at each stage of the MANO lifecycle.

Rizos et al. [84] also leveraged UCON to manage usage in Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) messages. They proposed a security enhancement of both CoAP and MQTT protocols by adding strong authorisation capabilities provided by UCON. The authors implemented UCON on top of CoAP and MQTT protocols to provide dynamic access control to the data shared using these protocols. By enforcing UCON policies, the system can ensure that only authorised entities have access to the data. Their integration enhances interoperability between the protocols by allowing attribute values to be shared via UCON. This enables seamless communication and attribute sharing among different protocols, improving the overall efficiency and effectiveness of Internet of Things (IoT) systems. The integration does not impact how the protocols work, allowing any CoAP and/or MQTT application to support it.

These works show that many various adaptations of the UCON model have been proposed for different purposes and to address a variety of requirements. This proves the need for a generalised model for policy-based governance to cover these various requirements as well as future ones.

2.4 Policy Languages

In this section, we outline the prominent policy languages that are typically used to express usage control rules and conditions. We specifically examine five relevant policy languages and describe their functionalities, syntax, semantics and expressive power.

2.4.1 Open Digital Rights Language (ODRL)

The Open Digital Rights Language (ODRL) [46, 45] is a World Wide Web Consortium (W3C) standard policy language designed to express rights and conditions for using digital content. It can be expressed either in XML or JSON, making it easily machine-readable and interoperable across different systems. ODRL consists of a complex and extensible structure, which allows the definition of a wide variety of semantics and profiles to suit specific requirements including usage control. To this end, we provide a brief and incomplete overview of ODRL focusing only on the constructs that are relevant to the scope of this thesis.

ODRL rules consists of three main constructs: permissions, prohibitions and duties. Permissions specify the actions that are allowed to be exercised on digital assets (e.g., read, write, modify, delete, copy, distribute, etc.). Prohibitions express limitations on the exercise of permissions such as limiting the number of times a specific action can be exercised or the type of devices on which assets can be used. Prohibitions may also be used to specify actions that are not allowed to be performed on the assets. Duties are obligations that must be fulfilled by subjects upon using the corresponding digital assets. This includes conditions that must be met or actions that must be executed before or after using the digital asset. For instance, a duty may specify that the subject must inform the data owner upon using their data. ODRL also incorporates the notion of constraints, which specify conditions for refining the applicability of rules, providing additional granularity.

Due to its expressive power and flexibility, ODRL has been used in several frameworks as baseline language for expressing usage control policies [53]. For instance, the Industrial/International Data Spaces (IDS) reference architecture [75] also leverages ODRL as policy a language for expressing data usage control rules in data spaces and data space connectors. Likewise, Munoz-Arcentales et al. [70] used ODRL for defining data usage control policies in industrial data spaces. They showed how data usage policies can be specified using ODRL permissions, prohibitions and duties. They also demonstrate how to specify consequences or compensating actions for violating obligations. Similarly, Esteves et al. [33] extended the access control language of the Solid protocol[1] with the functionalities of ODRL. Their extension aims to address the need for legal compliance, auditing personal data processing, specifying rights and obligations, as well as evaluating the fulfilment of obligations. By leveraging ODRL, their extension enables a seamless integration with existing RDF-based systems within the Solid architecture, offering en-

[1] https://solid.github.io/web-access-control-spec/

hanced functionality for expressing consent and data handling policies while ensuring legal compliance with the General Data Protection Regulation (GDPR) and other regulations.

2.4.2 Open Policy Agent (OPA) Rego

OPA Rego [1] is a declarative policy language specifically designed for writing policies in the OPA policy engine, which provides a unified policy enforcement mechanism that can be integrated into different systems and services. It was originally designed for Kubernetes, but its flexibility makes it effective in diverse applications beyond container orchestration. Rego follows a declarative syntax, allowing policy authors to express their own semantics rather than being limited to predefined ones. Rego policies are expressed as a collection of rules, each rule defining a condition and a corresponding action. Conditions are logical expressions that refine the applicability of rules and determine when they must be enforced, while actions specify the measures to be taken when the corresponding conditions are satisfied. Rego has been gaining popularity recently and has been adopted in various use-cases including access and usage control as well as compliance and governance [102, 71, 82].

2.4.3 eXtensible Access Control Markup Language (XACML)

The eXtensible Access Control Markup Language (XACML) [73] is the OASIS standard policy language to express ABAC policies. XACML is widely used in the field of access control due to its flexibility, expressiveness and rich semantics. It allows expressing complex policies using a wide variety of logical operators, functions and data types. It also supports obligations, which specify actions or conditions to be fulfilled as part of the access control decision.

XACML syntax is XML-based and consists of policy sets, policies and rules, which are defined in a hierarchical tree structure enabling more flexible and modular policy management. A policy set is a container for multiple policies used to group related policies together, while a policy is a collection of rules that specify access control conditions and decisions. Each rule defines a condition that needs to be satisfied for the rule to apply, and an *effect* (i.e., `permit`, `deny`) that indicates the access control decision that should be made if the rule matches. Rule conditions are complex logical expressions over attributes allowing for more complex conditions to be defined by combining multiple attribute comparisons and logical operators. Policies and rules may include obligation expressions that specify additional actions to be taken upon the access control decision.

XACML relies on target expressions and combining algorithms to resolve conflicts in policies and rules, and to select the applicable ones for making access control decisions. Target expressions consist of attribute-based match expressions, and define the conditions under which a policy or rule is applicable. Thus, when an access request is received, the target expressions are evaluated to determine which policies or rules should be considered for evaluation. In cases where multiple policies or rules are applicable, combining algorithms come into play. Combining algorithms are defined at different levels, such as policy sets and policies, and they determine how the results of individual rules or policies are combined to make an overall access control decision. XACML provides several combining algorithms such *firstApplicable*, *denyOverrides* or *denyUnlessPermit*. These algorithms define the precedence and rules for combining the results and handling conflicts when multiple policies or rules are applicable. For instance, the *denyUnlessPermit* combining algorithm specifies that a `permit` decision must have priority over a `deny` decision when multiple policies or rules apply.

Administration and Delegation Profile (ADP)

The Administration and Delegation Profile (ADP) [83] is an extension to the XACML standard that defines a set of guidelines and mechanisms for managing access control policies. It addresses the administrative aspects of policy creation, modification and distribution, as well as the delegation of authority to different entities within an organisation. ADP specifically enables administrators and resources owners to specify who is allowed to write access control policies about their resources. It also allows them to delegate the authority of managing certain resources in specific circumstances. For instance, an administrator may delegate the authority of policy management to another employee during a vacation.

ADP is realised by allowing administrators or resources owner to express policies about access control policies. These policies are designated as *administrative policies* and they specify the rules and conditions for issuing access control policies for specific resources and in specific contexts. This forms a tree of policies with leaf nodes being the access control policies while the rest of nodes are administrative policies. Top level policies in the tree are designated as roots of trust and they enable defining trust anchors for policy evaluation. Therefore, an access control policy will be enforced only if is authorised by a branch of administrative policies that resolves to a root of a trust.

U-XACML

Colombo et al. introduced U-XACML [23, 61]: an extension of XACML that incorporates UCON's temporal state (i.e., `pre`, `ongoing` and `post`). They specifically extended XACML rules to include three condition expressions instead of one, each referring to one of UCON's three temporal states. Likewise, they added a temporal state to obligation expressions to specify when each obligations must be enforced throughout the usage control session. They also added an expression for attribute updates allowing policies to change attribute values throughout the usage session as specified by the UCON model.

Although U-XACML realises the novelties of the UCON model, it deviates from the XACML standard making it incompatible with existing XACML policy engines. In addition, U-XACML restricts the principles of UCON to the condition expression level only, preventing the upper elements of XACML (e.g., policy sets, policies) from being classified by UCON's temporal states. Conversely, we argue that the XACML standard can be used to express UCON policies without any extensions or modifications. This can be achieved by expressing UCON's temporal state as an attribute, and using it in the target expressions of rules, policies and policy set to refine their applicability to the corresponding temporal state.

2.4.4 Abbreviated Language For Authorisation (ALFA)

The Abbreviated Language For Authorisation (ALFA) [72] is a XACML profile, currently in the standardisation process by OASIS, designed to offer a syntactic alternative for writing XACML policies. It maps directly to XACML without adding any new semantics while providing a simplified and more user-friendly syntax. ALFA's syntax makes it much less verbose than XACML, more human readable, and shorter in size allowing faster parsing and evaluation.

ALFA incorporates the same concepts and elements as XACML and conforms with the same hierarchical structure of policy sets, policies and rules. Like XACML, ALFA relies on combining algorithms to resolve conflicts between sibling rules or policies. It also allows the use of functions, such as regular expression, string concatenation and others, which helps to further refine applicability of policies and rules. A non-inclusive example of ALFA syntax, structure and elements is shown in Listing 2.1.

ALFA has gained popularity due to its simplicity and ease of use. For this reason, we use ALFA as a baseline language for our policy engine presented in Chapter 4. We also show examples of using ALFA to express policies in the policy-based solutions introduced in this thesis.

Listing 2.1: An example ALFA policy

```
policyset example-policyset {
    target clause Attributes.department == "sales"
    apply denyUnlessPermit
    policy example-policy{
        target clause Attributes.resource == "file.pdf"
        apply firstApplicable
        rule manager {
            target clause Attributes.action == "write"
            condition Attributes.subject.role == "manager"
            permit
        }
        rule employee{
            target clause Attributes.action == "read"
            condition Attributes.subject.role == "employee"
            permit
            on permit { obligation notify { email = "manager@email.com" } }
        }
        rule default { deny }
}}
```

2.5 Related Work

In this section, we provide an overview of related work, shedding light on the recent advancements that address similar challenges, and highlighting limitations of these works.

2.5.1 Trusted Context Transformation

Salim et al. [90] presented an administrative model for managing attributes in UCON. Their model considers all attributes to be mutable and defines authorities that manage attributes. In addition, it enables attribute authorities to delegate their rights of attribute administration to other actors. However, their model relies on trusting the attribute authorities and does not provide a rule-based resolution of attribute values. Moreover, it does not address the issues of customisable authorisation lifecycle.

A risk-aware data usage control model was proposed by Martinelli et al. [64]. They exploited the Analytic Hierarchy Process (AHP) to aggregate multiple attributes in a single aggregated value. Specifically, they associate each attribute value with a level of risk in a relation to a specific operation (i.e., performing an action on a protected resource).

If multiple attributes are involved, they use the AHP to calculate the overall risk of the requested operation and make the usage decision accordingly. Therefore, usage policies can be simplified to use a single risk-level attribute that aggregates all other attributes. By associating attributes with risk levels, their model provides a solution for the policy engine to dynamically manage attributes. However, dynamic obligation management and assignment remains overlooked. In addition, their model adheres to UCON's fixed lifecycle, which restricts customisation.

2.5.2 Industrial/International Data Spaces (IDS)

In [107], Zrenner et al. analysed the requirements of data sovereignty and usage control policies in automotive data spaces. Accordingly, they proposed several different architectures for implementing usage control for data sovereignty. The architectures differ in the advantages and disadvantages they provide to data consumers and data providers. Zenner et al. address the challenges of leveraging usage control for data sovereignty at an architectural level. In contrast, we address such challenges at a model level, and we enable applications other than data sovereignty. Therefore, the two works can be considered complementary.

Similarly, Munoz-Arcentales et al. [69] designed an architecture for data usage control in shared data spaces. The proposed architecture is based on the XACML [73] reference architecture and the IDS [75] reference architecture. Their architecture enables data owners and data providers to define policies and monitor the use of their data by data consumers, which ensure data sovereignty. Their work also addresses the challenges of data usage control for data sovereignty at architectural level, and as such it can be considered complementary to our work.

Jung et al. [50] proposed a solution for addressing data sovereignty challenges through the use of data usage control and data provenance. They introduced relevant concepts and technologies for realising usage control within the IDS. Their solution allows the formulation of data usage restrictions as policies and converting them into machine-readable policies that can be enforced by systems. They also presented a general architecture for enforcing data usage control data space connectors. However, their solution does not consider the data lifecycle and preserves UCON's fixed lifecycle. In addition, it does not provide a mechanism to dynamically resolve attribute values and manage obligations.

2.5.3 Cloud Data Hubs

Simonet et al. [98] designed a programming model for data lifecycle management. Their model enables expressing the different events to be monitored and the operations to be executed at each stage of the data lifecycle. The model is then used to develop data lifecycle management applications. Although the main objective of their work is not data usage control, their model can be used for data sovereignty and data usage control by expressing usage policies as events and operations. Nonetheless, a dedicated policy engine offers a more robust and flexible solution for supporting complex data usage control requirements as well as compliance with regulations.

Cirillo et al. [22] introduced and intent-oriented data usage control system for federated data analytics. The system enables data consumers and data provider to specify data usage intents and data usage policies, respectively. The system then manages and controls the data processing flows according to the defined intents and policies. Therefore, policy enforcement is combined with service orchestration to make proactive data usage control decisions. Their framework addresses the challenges of data usage control in shared data spaces based on usage purpose. However, it does not address the challenges we describe in this thesis as it does not provide lifecycle management or trusted context transformation for data usage sessions.

In [10], we leveraged UCON to handle data privacy and data subject rights throughout the data lifecycle. The proposed framework allows data subjects to intervene and revoke the consent for using their data. The framework also ensures compliance with regulations by enabling policy administration and delegation. Nonetheless, the framework does not address the session management issues described in this thesis. For instance, it uses UCON's traditional lifecycle (i.e., `pre`, `ongoing` and `post`), which limits the expressibility of the framework and its ability to capture specific contexts related to the data lifecycle stages.

A policy-based data usage control for smart cities was proposed by Cao et al. [18]. The model is designed to address the diversity of obligations and data usage requirements that data owners impose on the use of their data. It provides a framework for defining usage perspectives, usage restrictions, and data usage obligations. It also allows for the specification and enforcement of policies that govern the usage of shared data. The model is designed to enhance transparency and traceability of data usage, allowing data providers to exercise control over the usage of their data by other actors. Nonetheless, it does not enact usage control throughout the data lifecycle and does not address the issues of dynamic attribute resolution and obligation management.

Appenzeller et al. [6] introduced a set of tools and workflow that allow patients to maintain control over the usage of their data. They also present an implementation of a continuous digital consent enforcement workflow, where patients can define a detailed declaration of consent for their medical data. Researchers can then request data through a dedicated interface that enforces the patient's consent. Their solution allows combining data from different doctors to improve medical diagnosis and care-giving as data becomes more valuable when multiple doctors contribute, while addressing the consequent privacy concerns and potential legal issues. Although their work does not consider policy-based systems, it can be considered complementary to our contributions as they provide solutions for enforcing data subject rights.

Pretschner et al. [81] extended the UCON model to support the tracking of data flow between different representations. Their extended model explicitly distinguishes between data and the representation of data. It recognises that data can exist in multiple representations, such as multiple copies of a file or multiple clones of an object. For example, an image can exist as a network packet, a Java object, a window pixmap, a database record, or a file. The data flow model is designed to capture and track these different representations of data across system layers, including the operating system, runtime system, window manager, and database management system. By incorporating the data flow model into the usage control model, usage control policies can address not just one single representation of data, but all representations. Their extended model does not enact data usage control throughout the data lifecycle and does not provide trusted context transformation. However, it can be considered as complementary work to our contributions as we only focus on the usage control of single data items.

2.5.4 Smart Vehicles

Hamad et al. [38] introduced a policy-based authentication and authorisation framework for vehicular communications. They used a trust management model as the core of their framework. When an application needs to connect to a remote entity, it makes a request to the trust management system. The system evaluates relevant policies and makes a decision based on the context. The framework introduces a performance overhead over network connections, but this only affects new connections as the policies are only evaluated during the setup phase. This means that the framework does not enforce continuous control and does not take contextual changes into consideration once a connection is established. Their work also does not consider managing contextualised credentials and capabilities within the vehicle.

Kim et al. [54] developed a decentralised access control framework by integrating an ABAC module in AUTOSAR [7] platform. They focused on protecting Electronic Control Unit (ECU) diagnostic interfaces from unauthorised access. Accordingly, diagnostic Controller Area Network (CAN) messages are considered as subjects and their properties are used as subject attributes, while ECU properties as environmental attributes. Their framework lacks context awareness and continuity of control as it only works with a predefined set of attributes and enforces atomic access control.

Likewise, Gupta and Sandhu [37] proposed an extended and layered access-control-oriented architecture for smart vehicles. They also provided an authorisation framework that secures the dynamic intra- and inter-communications between vehicles and vehicle components. They discussed and described how different access control models can be enforced at the different layers of the architecture. Their framework, however, lacks continuity of control as well as contextualised capability management.

Ammar et al. [4] developed an end-to-end RBAC mechanism that protects vulnerable On-Board Diagnostics-II (OBD-II) ports. These ports impose a serious security risk as they can be used to install software on ECUs and to access the internal vehicle network. Thus, access to vehicle's resources through the vulnerable OBD-II ports is regulated by the proposed mechanism according to RBAC policies. The mechanism is AUTOSAR-compliant and can be installed on existing vehicles without any modifications. Nonetheless, their work overlooks the need for continuous monitoring as well as context-aware credential management.

Rumez et al. [87] introduced a distributed ABAC tailored for Service-Oriented Architecture (SOA) automotive architectures. Their framework enforces access control for domain controllers and their associated ECUs. The distributed nature of their approach enables controlling individual access privileges based on authenticated subjects and dynamic attribute values associated with subjects and environmental conditions, such as vehicle state or current location. While their approach enables contextualised capabilities, it still lacks continuous monitoring and policy re-evaluation that updates the capabilities when the vehicle's situation changes.

2.5.5 Credential Transformation

MATTR [66], a Self-Sovereign Identity (SSI) solutions provider, have introduced an extension to integrate OpenID Connect (OIDC) with their SSI platform. The extension enables developers that are familiar with legacy identity systems to easily incorporate SSI in their existing systems. Although their extension bridges OIDC with SSI, it remains

limited to the MATTR platform only, and it does not consider other standards (e.g., Security Assertion Markup Language (SAML)).

The OIDC standard introduced the Self-Issued OpenID Provider model [105], which enables end-users to control and release their own identity information without direct involvement of claims issuers. This model is based on the OIDC protocol and provides a secure and user-centric approach to identity management. It gives users more control over their identity information and enhances their privacy. This model enables the principles of SSI, but it does not provide a solution to integrate and operate different identity systems and credential technologies.

Similarly, the OIDC standard also introduce an extension for Verifiable Presentations (VPs) [103]. The extension allows end-users to present VPs containing one or more of their credentials to verifiers using a digital wallet. Verifiers specify their requirements and constraints for the VP, such as the types of credentials needed and any specific attributes or claims required. On the other hand, the wallet ensures that the VP meets the requirements specified by the verifier. This extension provides a standardised protocol for requesting and presenting VPs, enabling interoperability and compatibility with existing systems. This extension is limited to providing VPs only, so it does not enable credential transformation between different technologies and formats.

Siddiqui et al. [97] argued that the adoption of SSI is low because the needs of service providers and credential issuers have been overlooked. To address this problem, they proposed a Credentials as a Service (CaaS) solution that enables service providers to run their credential issuers on public cloud, and issue SSI credentials accordingly. They leverage Trusted Execution Environments (TEEs) to protect user privacy and confidentiality as well as the issuer functionality. While this solution facilitates the adoption of SSI, it does not provide a credential transformation and interoperability mechanisms.

Hong and Kim [43] presented a blockchain-based SSI solution that complies with OAuth 2.0. The solution combines SSI with OAuth 2.0 by designing novel authentication and authorisation processes based on OAuth, while also ensuring the decentralisation and integrity of user and client information through the use of blockchain technology. This provides user-centric authentication and authorisation controlled through the user's own device, giving users control over their own digital identities and personal data, while also ensuring ease of integration with existing systems. Their solution allows SSI to be ported with OAuth-based systems. However, it does not address the challenges described in this thesis as it does not dynamically transform credentials between different formats and technologies.

Likewise, Lagutin et al. [59] introduced a solution for enabling the integration of

Decentralised Identifiers (DIDs) and Verifiable Credentials (VCs) with the OAuth framework for constrained IoT devices. By extending the ACE-OAuth flow, the authors introduced a method for delegating the processing of DIDs and VCs to the OAuth Authorisation Server. This extension significantly expands the number of devices capable of utilising DIDs and VCs, thereby enhancing the privacy, security, and usability of IoT devices. In addition, the authors address the challenge of local authentication, which is not covered by ACE-OAuth, by proposing a solution for it. Their solution integrates the SSI principles with OAuth enabling easy and deployment and adoption of SSI across various organisations, especially for IoT devices. However, their solution does not provide dynamic credential transformation that allows organisations with different legacy identity systems to operate.

Jung et al. [51] also proposed an OID-Based Identifier Framework to address the interoperability of heterogeneous identifiers in the context of the IoT. The framework utilises the concept of meta-resolution and metadata repository to enable a unified identification service. It allows devices belonging to different identifier infrastructures to transparently interact with each other, even if they use different credential formats. The framework supports the interoperability of different credential formats by providing a metadata repository that stores the metadata of different credential formats. This metadata can be used to resolve the credential format and retrieve the identification and communication information of the target peer device. Their framework, however, does not provide a dynamic solution for transforming and interoperating identifiers as applications using the framework must understand the format and contents of the resolved metadata.

A credential conversion service for SAML credentials was introduced by Cánovas et al. [17]. It converts external non-SAML credentials into SAML assertions according to a predefined conversion policy. The proposed service can issue different types of SAML assertions based on the presented user credentials. To express the syntax and semantics of the credential conversion service, the authors introduced extensions to SAML. Their solution is similar to the credential transformation bridge we propose in this thesis as it converts credentials dynamically based on a predefined policy. However, their credential conversion service is focused on SAML only, while our credential bridge is generalised and considers several credential formats and technologies.

2.5.6 Summary

Although these works provide valuable contributions in their respective fields, they still overlook the challenges tackled in this thesis. To this end, we outline a summary of the

overlooked gaps in Table 2.1

Table 2.1: Summary of overlooked gaps by related work

Scope	**Gap**
IDS	IDS is a relatively new field that identifies UCON and policy-based governance as key technologies. Several works have tackled the integration of UCON and IDS. However, they only address architectural challenges and use UCON's traditional lifecycle. None of the discussed works considers contract negotiation or custom lifecycles to govern data sharing and enforce agreements.
Data Hubs	The discussed works in this scope fall in two categories. Some address problems that are complementary to our work while overlooking the challenges we tackle in this thesis. Others address similar challenges to our work, but only partially. For example, several solutions for data lifecycle management were proposed, but they did not address data usage control. In contrast, other solutions leveraged usage control to govern data, but did not consider the data lifecycle.
Smart Vehicles	Fine-grained access control has been considered by many researchers in the automotive field. However, none of the discussed works consider usage control with continuous monitoring and policy re-evaluation. More importantly, they do not consider managing the lifecycle of contextualised capabilities within the vehicle.
Credential Transformation	There are various solutions to interoperate different credential technologies and formats between different jurisdictions. However, dynamic credential transformations that can tailor the decision making for individual cases remain absent. This is important as credential transformations may require collecting additional credentials and information to complement the process.

Trusted Context Transformation	Among the discussed related work, there are solutions proposed to govern attribute values. However, these solutions depend on their respective applications, so they cannot be applied on other use-cases. Therefore, a general solution is needed as identified by this thesis.
Custom Lifecycles	Most of the discussed works use UCON with its traditional temporal lifecycle, even though their respective application domains require a more fine-grained lifecycle. Some approaches have modified the UCON lifecycle or proposed the use of external lifecycle management tools. However, they only focus on fixed lifecycles specific to their respective requirements, and do not provide a general solution that allows tailoring the lifecycle for different application domains.
Unified Solution	Evidently, many adaptations of UCON and policy systems have been proposed to address various requirements. While their contributions are valuable, they lack comprehensiveness. Therefore, a unified solution is required to encompass these variations as well as future applications. The unified solution must also decouple the logic and behaviour of the policy system from its applications and the corresponding policy authoring.

Chapter 3

UCON+: An Extensible Model for Policy-Based Governance

This chapter describes the proposed extensible model for policy-based governance.

3.1 Introduction

In this chapter, we introduce UCON+: an extensible model for policy-based governance that goes beyond conventional access and usage control. UCON+ is based on Usage Control (UCON), so it is also an attribute-based model that allows attribute mutability, incorporates session-based monitoring, and re-evaluates the security policy. However, UCON+ specifies an abstract foundation only, allowing the specification of concrete extensions and tailoring them for specific requirements. To illustrate this, we draw an analogy between UCON+ and an abstract class in Object-Oriented Programming (OOP), as the abstract class defines common attributes and method signatures and its subclasses extend it with concrete implementations. Similarly, UCON+ defines general structures and common behaviour, and concrete extensions specify the extensible behaviour.

The purpose of UCON+ is to model the components and execution of the evaluation session. It defines the evaluation session as a sequence of policy evaluations during which the context (i.e., attribute values) changes between evaluations. This sequence of evaluations adheres to a lifecycle, which is the set of phases that the evaluation session passes through during its lifetime. UCON+ does not specify a predefined or fixed lifecycle. It only outlines the overall structure and components of a lifecycle, and leaves the specific lifecycle definition to concrete extensions. Moreover, UCON+ states that contextual

changes and lifecycle transitions must be governed and driven by conditional rules that are specified by concrete extensions of the model. Therefore, distinct concrete extensions of UCON+ are constructed by specifying such rules and defining a specific lifecycle. To this end, we identify three types of policies used in UCON+: (1) Access Control policies are Attribute-Based Access Control (ABAC) policies that specify rules over attributes that evaluate into a decision (e.g., `permit`, `deny`) as well as obligations; (2) Context Transformation policies specify rules for governing attribute values and updates, and setting the context; (3) Lifecycle Transition policies specify rules that drive the lifecycle and fire transitions between phases based on predefined conditions. UCON+ does not assume a specific enforcement system or policy language, and only defines general structures for these policies, enabling existing policy languages to be easily used as long as they adhere to these structures.

Accordingly, the evaluation session in UCON+ is composed by the following five components: (1) an evaluation context (2) a lifecycle; (3) a context transformation policy; (4) a lifecycle transition policy; (5) access control policies. The execution of the session consists of recurring executions of the three policies until the session terminates.

The rest of this chapter describes the components of an evaluation session and how it is executed. It provides a rigorous description along with annotations based on widely understood notions of set theory to clearly communicate the concepts and avoid unambiguous understanding that may be caused by the high level of abstraction introduced by the model. We highlight that this is *not* a formal specification of the model as formalisation is out of the scope of this thesis which is focused on applied research. We also note that there is a different work-in-progress that is focused on the formal model [94].

3.2 Evaluation Session Components

In this section, we describe the components that form the UCON+ evaluation session. We start by introducing the following basic notions that will be used to define the key concepts of UCON+. We specifically consider disjoint, infinitely countable sets Attr, Val, Dec, Evt of *attributes*, *values*, *decisions*, and *events*, respectively. Attr and Val are the sets of all possible attributes and all possible values respectively. Each attribute $a \in \mathsf{Attr}$ is associated with a (possibly infinite) set $\mathsf{dom}(a) \subseteq \mathsf{Val}$, called the *domain of* a, such that $\forall a \in Attr$, $dom(a) = \{v \in Val \mid v \text{ is a possible value of } a\}$. UCON+ extends the classical two-valued decisions (i.e., `permit` or `deny`) and allows custom defined decisions, so Dec include all possible access policy decisions. We also identify a distinguished element of Dec, denoted $\perp$, which we call the *null decision*, representing the initial decision at the

beginning of the session. Evt includes all possible events that invoke phase transitions (e.g., from `collection` to `retention`) in all possible lifecycles. In the elements described below, we consider a *schema* defined as a triple (A, D, E), where $A \subseteq \mathsf{Attr}$, $D \subseteq \mathsf{Dec}$ with $\perp \in D$, and $E \subseteq \mathsf{Evt}$ are all finite. Intuitively, a schema defines the sets of attributes, decisions and events that are needed in a particular extension model of UCON+.

3.2.1 Evaluation Context

UCON+ is based on attributes, which are characteristics about the subject, object or environment, such as a role, data type or time. Accordingly, the context of a UCON+ evaluation session is the assignment of values to all relevant attributes at a particular instant during the lifetime of the session as defined in Definition 3.2.1.

Definition 3.2.1 (Context) *Considering a schema $\mathcal{S} = (A, D, E)$, an* evaluation context over the schema *S is a function $\mu : A \to \mathsf{Val}$ such that for each $a \in A$, $\mu(a) \in \mathsf{dom}(a)$. That is, μ associates each attribute in A to one of the possible values in its domain. We use $\mathcal{C}[A]$ to denote the set of all contexts over A.*

We identify a special type of attributes designated as *internal session attributes*, which describe the session and are solely managed by it. For example, the lifecycle phase, access control decision and obligation status (e.g., fulfilled, pending) are internal session attributes, and are part of the evaluation context. These attributes are part of the evaluation context and visible to all policies used in the session, but their values are unique for each individual session (i.e., each session has its own snapshot of these attributes). The rest of attributes are designated as external attributes and may be managed by the policy engine itself (e.g., time attribute) or by external authorities such as an Identity Provider (IdP), database, sensors, etc.

All attributes are considered mutable and their values may change. This means that the evaluation context undergoes continuous transformations throughout the session lifetime triggering policy re-evaluations. Attribute updates may be invoked by external authorities and third parties, or by the policy engine itself. Nonetheless, the context transformation policy specifies rules that govern such contextual changes and resolve attribute values, ensuring a *trusted context transformation* as described in Section 3.2.4.

3.2.2 Evaluation Lifecycle

The lifecycle of an evaluation session is a series of phases that defines a workflow for enacting the session. Each phase is a particular step in the course of development of

the evaluation session, describing its state at a specific period of its lifetime. As such, a phase must be a characteristic of the evaluation session that refines and influences the selection of policies and rules to be enforced at that stage of the session. This allows enforcing different rules based on how the session progresses throughout its lifetime. For example, the `ongoing` phase in UCON indicates that usage has started and is in progress, so `ongoing` policies are used to check whether usage should still be maintained.

We argue that the number of phases of the evaluation lifecycle must be finite and the transitions between them must be deterministic to ensure predictable behaviour and consistent policy-based governance. Specifically, if two evaluation sessions are subjected to the exact same policies, conditions and attribute values, then both sessions must follow the exact same path in their lifecycles. For this reason, we model the lifecycle as a Deterministic Finite Automaton (DFA) such that DFA states represent phases, and transitions describe how the lifecycle progresses throughout the session lifetime. This also enables customising the lifecycle in each extension of UCON+ by specifying distinct DFAs. Accordingly, we use a common definition of DFA [27, 96] to describe the lifecycle as defined in Definition 3.2.2.

Definition 3.2.2 (Lifecycle) *Considering a schema* $\mathcal{S} = (A, D, E)$*, a* lifecycle *is a* DFA $\mathcal{W} = (Q, F, E, q_{init}, \delta)$*, where*

- Q *is the set of* states *(or* phases*) of* $\mathcal{W}$*,*
- $F \subseteq Q$ *is the set of* final states *of* $\mathcal{W}$*,*
- E *is the set of* labels *of* $\mathcal{W}$*,*
- $q_{init} \in Q \setminus F$ *is the* initial state *of* $\mathcal{W}$*, and*
- $\delta : (Q \setminus F) \times E \to Q$ *is the* transition function *of* $\mathcal{W}$*, and it determines the next state to transition to given a non-final state in* Q *and a label in* E*.* ∎

Transitions between phases are conditional based on the evaluation context. For instance, the data usage lifecycle moves from the `collection` phase to the `retention` phase only if all `collection` obligations are fulfilled and the data collection process is completed. Similarly, the lifecycle of a UCON authorisation transitions from the `pre` phase to the `ongoing` phase if the authorisation decision is `permit` and resource usage has started. The lifecycle transition policy specifies such rules and conditions that determine when to fire events (i.e., the labels of the DFA transitions) that change the phase. This allows the lifecycle transition policy to dynamically drive the lifecycle as the session evolves and the context changes as described in Section 3.2.5.

3.2.3 Access Control Policy

A UCON+ access control policy consists of a set of attribute-based rules that evaluate into a decision and a set of obligations given a specific context. Thus, the access control policy is a function that takes the context as an input and returns a decision as well as a set of obligations as illustrated in Figure 3.1. Obligations are considered functions that set or modify attribute values. For instance, an obligation to delete data is treated as updating a Boolean attribute named `delete` to `true`.

Internal session attributes (e.g., phase) are passed to the access control policy within the evaluation context, which enables refining policy or rule applicability. However, the access control policy is *agnostic* to the evaluation lifecycle as well as the context transformation and lifecycle transition policies. Namely, the access control policy treats internal session attributes as normal attributes and uses them, but it is not concerned with how these attributes are managed or how they change. This decouples the access control logic from the rest of the session execution logic, enabling existing ABAC systems to be easily extended with the novelties of UCON+. Therefore, existing ABAC policy languages can be used in UCON+ as long as they conform with Definition 3.2.3, which describes the access control policy considering a schema $\mathcal{S} = (A, D, E)$.

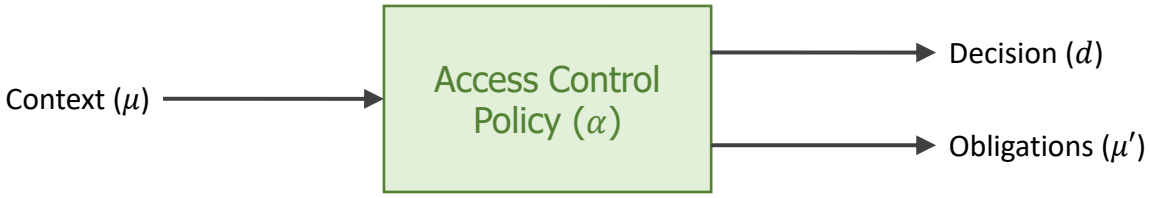

Figure 3.1: The inputs and output of the access control policy

Definition 3.2.3 (Access Policy) *Considering a schema* $\mathcal{S} = (A, D, E)$*, an* access policy over $\mathcal{S}$ *is a pair* $\alpha = (\mathsf{dec}, \mathsf{obl})$ *of functions* $\mathsf{dec} : \mathcal{C}[A] \to D$ *and* $\mathsf{obl} : \mathcal{C}[A] \to \mathcal{C}[A]$*, called the* decision function *and the* obligation function *of* α*, respectively.* ∎

Intuitively, an access control policy $\alpha = (\mathsf{dec}, \mathsf{obl})$ over $\mathcal{S}$ specifies the following: given some context $\mu : A \to \mathsf{Val}$, the decision function dec should be evaluated over μ to obtain a decision $d = \mathsf{dec}(\mu) \in D$ (e.g., `permit`); moreover, the obligation function should be evaluated over μ, to provide a set of possible actions one has to take, i.e., some new values $\mu' = \mathsf{obl}(\mu)$ for the attributes (e.g., delete data).

3.2.4 Context Transformation Policy

As mentioned before, the access control policy is evaluated against the context. Thus, the context must be prepared and enriched before evaluating the access control policy in order to ensure that all required attributes and values are present. Moreover, attribute values may be updated while the access control policy is being evaluated and enforced. We refer to such attribute updates as the *external context*. The external context must be integrated with the evaluation context before the next evaluation of the access control policy. These contextual changes and enrichments are governed by the context transformation policy.

The context transformation policy specifies rules for resolving attribute updates (i.e., the external context) and preparing the evaluation context for the next access control policy evaluation. For example, a context transformation rule may specify that if the `data-type` attribute is `clinical-trials`, then the `data-category` attribute must be set to `public-benefit`[1]. Therefore, the context transformation policy takes the current evaluation context as well as the external context as input and produces a resolved context as illustrated in Figure 3.2. In Definition 3.2.4 we define the context transformation policy considering a schema $\mathcal{S} = (A, D, E)$.

Figure 3.2: The inputs and output of the context transformation policy

Definition 3.2.4 (Context Transformation Policy) *Considering a schema* $\mathcal{S} = (A, D, E)$*, a* context transformation policy over $\mathcal{S}$ *is a function* $\lambda : \mathcal{C}[A]^2 \to \mathcal{C}[A]$. ∎

Intuitively, a context transformation policy λ over $\mathcal{S}$ specifies the following: given some context $\mu : A \to \mathsf{Val}$, and another context $\nu : A \to \mathsf{Val}$ representing possible values from external attribute updates, the context transformation λ is in charge of resolving possible conflicts between μ and ν, providing a new assignment of values to the attributes $\mu' = \lambda(\mu, \nu)$.

[1]In GDPR, public benefit data refers to information processed for significant public interests, allowing for necessary processing exceptions while safeguarding data subjects rights.

3.2.5 Lifecycle Transition Policy

As described in Section 3.2.2, the evaluation lifecycle is a DFA, which transitions from one phase to another based on the event (i.e., DFA label) fed to the transition function. Figure 3.3 shows an example of a DFA that transitions from the `retention` phase to `destruction` phase when the `delete-data` event is fired. The lifecycle transition policy drives the lifecycle by specifying which lifecycle events to fire based on predefined rules and conditions. For instance, the policy may specify that if the current phase is `retention` and the decision is `deny`, then the `delete-data` event must be fired, causing a transition to the `destruction` phase. Therefore, the lifecycle transition policy takes the context and decision as input, and outputs a DFA event as illustrated in Figure 3.4. Definition 3.2.5 describes the lifecycle transition policy considering a schema $\mathcal{S} = (A, D, E)$.

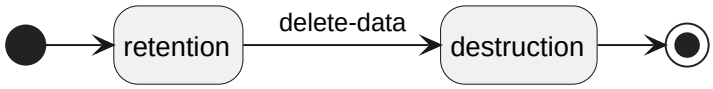

Figure 3.3: Example of a DFA

Figure 3.4: The inputs and outputs of the lifecycle transition policy

Definition 3.2.5 (Lifecycle Transition Policy) *Considering a schema* $\mathcal{S} = (A, D, E)$, *a* lifecycle transition policy over $\mathcal{S}$ *is a function* $\gamma : \mathcal{C}[A] \times D \to E$. ∎

Intuitively, a lifecycle transition policy γ over $\mathcal{S}$ specifies the following: given some context $\mu : A \to \mathsf{Val}$, as well as some decision $d \in D$, the lifecycle transition policy γ specifies which event $e = \gamma(\mu, d)$ should take place (e.g., start data transfer).

3.3 Evaluation Session Execution

In this section, we describe how an evaluation session is formed and executed by the components introduced in Section 3.2. We also provide the corresponding definitions and

explain the constraints and conditions of a valid execution of the session.

A UCON+ evaluation session is a sequence of steps that adheres to the lifecycle. Each step is an access control evaluation followed by context transformation then lifecycle transition evaluations. Different access control rules may apply at each phase of the lifecycle (e.g., the security rules of data `collection` are different than the rules of data `retention`). For this reason, access control policies are classified by lifecycle phases (i.e., one policy for each phase) ensuring that the appropriate access control policies are evaluated when the phase changes within a specific step of the sequence.

The sequence starts with an initial context μ_0 as well as a *null* decision and *null* obligations. This means that the very first access control evaluation (i.e., in the `init` phase during the *initialisation* of the session) takes a non-actionable decision and obligations that do not require enforcement [2]. For this reason, we introduce the concept of partial access control policy defined as follows: considering a schema $\mathcal{S} = (A, D, E)$, an access policy $\alpha = (\mathsf{dec}, \mathsf{obl})$ over $\mathcal{S}$ is *partial* if there is $\mu \in \mathcal{C}[A]$ such that $\mathsf{dec}(\mu) = \bot$; that is, for certain attribute values, the policy provides a null decision. We use $\alpha^{\mathcal{S},\emptyset}$ (or simply $\alpha^{\emptyset}$ when $\mathcal{S}$ is clear from the context) to denote the (partial) access policy $(\mathsf{dec}, \mathsf{obl})$ such that $\mathsf{dec}(\mu) = \bot$ and $\mathsf{obl}(\mu) = \mu$, for all $\mu \in \mathcal{C}[A]$. That is, $\alpha^{\emptyset}$ evaluates into a null decision and null obligations that do not require enforcement[3].

Accordingly, a UCON+ session combines the above elements and consists of a lifecycle, a context transformation policy, a lifecycle transition policy, and a set of access policies, one for each lifecycle phase with the access policy $\alpha^{\emptyset}$ of the initial phase being partial, as defined in Definition 3.3.1.

Definition 3.3.1 (Session) *Consider a schema $\mathcal{S} = (A, D, E)$. A* UCON+ session *over $\mathcal{S}$ is a quadruple $\mathcal{I} = (\mathcal{W}, \lambda, \gamma, \mathcal{A})$, where:*

- *$\mathcal{W} = (Q, F, E, q_{init}, \delta)$ is a lifecycle as defined in Definition 3.2.2;*
- *λ is a context transformation policy over $\mathcal{S}$ as defined in Definition 3.2.4;*
- *γ is a lifecycle transition policy over $\mathcal{S}$ as defined in Definition 3.2.5;*
- *$\mathcal{A} = \{\alpha_q\}_{q \in Q \setminus F}$ is a set of access control policies over $\mathcal{S}$ as defined in Definition 3.2.3, one for each (non-final) phase of $\mathcal{W}$, where $\alpha_{q_{init}} = \alpha^{\emptyset}$ is the only partial*

[2]Note that the null decision and null obligations are different than the complete absence of decision and obligations, so the policy still evaluates into a decision and obligations, but their values are null, which does not require enforcement.

[3]An alternative specification to this is to check if the session is in the initialisation phase and skip access control. However, we decided to use the partial policy to keep all steps in the sequence uniform.

access policy in $\mathcal{A}$. ∎

At each step of the sequence, the session has a specific configuration that defines the contents of the attributes as well as the lifecycle phase as defined in Definition 3.3.2.

Definition 3.3.2 (Configuration) *Consider a schema* $\mathcal{S} = (A, D, E)$*, and a UCON+ session* $\mathcal{I} = (\mathcal{W}, \lambda, \gamma, \mathcal{A})$ *over* $\mathcal{S}$*, with* Q *the set of phases of* $\mathcal{W}$*. A* configuration *of* $\mathcal{I}$ *is a pair* (q, μ) *of a phase* $q \in Q$ *and a context* μ *over the attributes of* $\mathcal{S}$*.* ∎

A valid step in the sequence is a transition from one configuration (q_1, μ_1) to another (q_2, μ_2). When transitioning, access control is performed first, followed by context transformation then lifecycle transition as shown in Figure 3.5. Specifically, a valid step starts with access control evaluation where a new decision d and a new context μ' are obtained. The decision is taken via the decision function of the access control policy α_{q_1} of the current phase q_1 over the context μ_1, such that $d = \mathsf{dec}_{q_1}(\mu_1)$. The new context is obtained according to the obligation function obl_{q_1} of α_{q_1}, when evaluated over the context μ_1, such that $\mu' = \mathsf{obl}_{q_1}(\mu_1)$. Then, according to the new context μ' and the external context (i.e., external attribute updates) ν, the context transformation policy λ solves possible conflicts and provides a final context $\mu_2 = \lambda(\mu', \nu)$. Finally, the lifecycle transition policy γ selects the event $e = \gamma(\mu_2, d)$ that brings the lifecycle to the next phase $q_2 = \delta(q_1, e)$. Accordingly, a valid step is defined in Definition 3.3.3 as follows.

Definition 3.3.3 (Valid Step) *Consider a schema* $\mathcal{S} = (A, D, E)$*, and a UCON+ session* $\mathcal{I} = (\mathcal{W}, \lambda, \gamma, \mathcal{A})$ *over* $\mathcal{S}$*, a* valid step of $\mathcal{I}$ *w.r.t. some (external) context* $\nu \in \mathcal{C}[A]$ *is an expression of the form*

$$(q_1, \mu_1) \overset{\nu}{\Rightarrow} (q_2, \mu_2),$$

where (q_1, μ_1) *and* (q_2, μ_2) *are configurations of* $\mathcal{I}$ *such that:*

- *first the decision* d *and obligations* μ' *are taken via the access control policy* $\alpha_{q_1} = (\mathsf{dec}_{q_1}, \mathsf{obl}_{q_1})$ *such that* $d = \mathsf{dec}_{q_1}(\mu_1)$ *and* $\mu' = \mathsf{obl}_{q_1}(\mu_1)$*;*
- *then a resolved context* μ_2 *is obtained via the context transformation policy* λ *such that* $\mu_2 = \lambda(\mu', \nu)$*;*
- *subsequently an event* e *is obtained via the lifecycle transition policy* γ *such that* $e = \gamma(\mu_2, d)$*;*
- *and finally the next phase* q_2 *is obtained via the transition function* δ *of the DFA such that* $q_2 = \delta(q_1, e)$*.* ∎

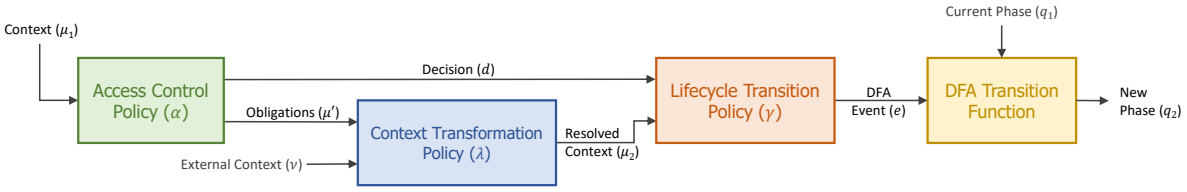

Figure 3.5: A valid step in the evaluation session

Therefore, a valid execution of a UCON+ session $\mathcal{I}$ is a sequence of valid steps that follow the lifecycle, and comply with the policies as depicted in Figure 3.6 and defined in Definition 3.3.4.

Definition 3.3.4 (Valid Execution of a UCON+ Session) *Considering a schema $\mathcal{S} = (A, D, E)$, let μ be a context over A, and let $\mathcal{I} = (\mathcal{W}, \lambda, \gamma, \mathcal{A})$ be a UCON+ session over $\mathcal{S}$, with $\mathcal{W} = (Q, F, E, q_{init}, \delta)$. A* valid execution ρ of $\mathcal{I}$ starting from μ *is a (possibly infinite) sequence*

$$(q_0, \mu_0) \overset{\nu_0}{\Rightarrow} (q_1, \mu_1) \overset{\nu_1}{\Rightarrow} (q_2, \mu_2) \overset{\nu_2}{\Rightarrow} \cdots$$

of valid steps of $\mathcal{I}$ such that:

1. $(q_0, \mu_0) = (q_{init}, \mu)$, *and*
2. *if ρ is finite, its last configuration has a phase in F (i.e., a final phase of $\mathcal{W}$) at which the session terminates.* ∎

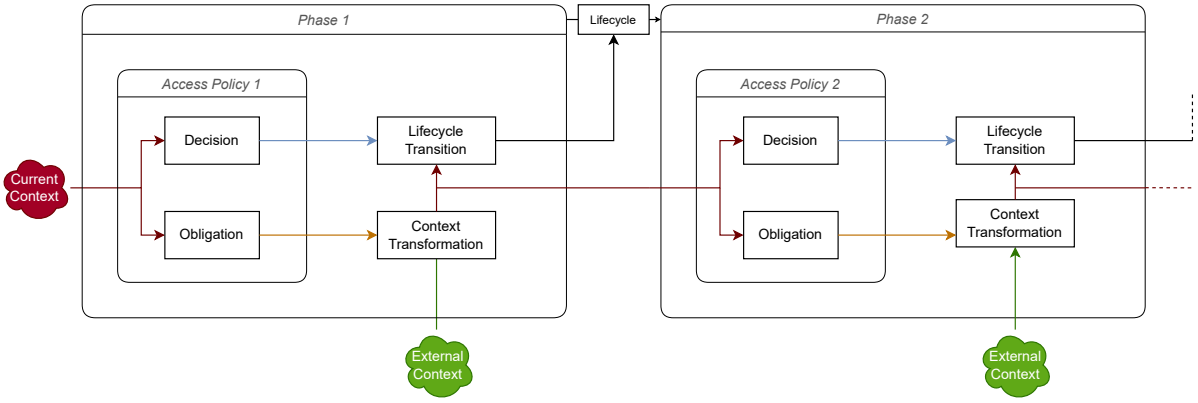

Figure 3.6: A valid execution of the evaluation session

In conclusion, a valid execution of an evaluation session is a loop of valid steps that exits when the lifecycle reaches a final phase as illustrated in the activity diagram in Figure 3.7.

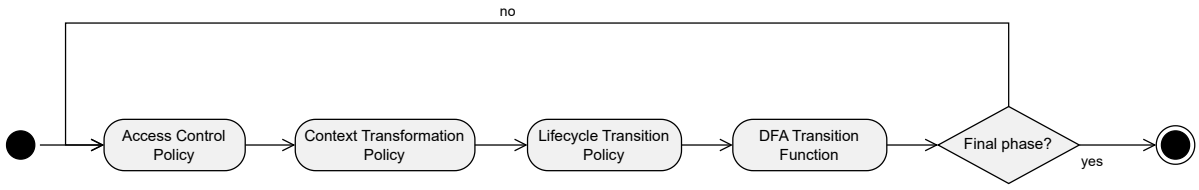

Figure 3.7: An activity diagram illustrating the execution of the session

3.4 Model Coverage

UCON+ is an extensible model for policy-based governance that can be adapted to various requirements and application domains. In Chapters 5, we demonstrate the novelties of UCON+ in several application domains by constructing extensions tailored for different requirements including data-oriented usage control as well as credential transformation. In this section, we further highlight the extensibility of UCON+ by showing how it can be used to construct ABAC and UCON.

3.4.1 Constructing ABAC Using UCON+

ABAC does not incorporate session-based monitoring and policy re-evaluation. Thus, contextual changes are not relevant and do not need to be governed by a context transformation policy. Therefore, only the lifecycle transition policy must be specified to cover ABAC. Park and Sandhu [76] demonstrated that ABAC is a subset of UCON and can be realised by evaluating `pre` rules only. Accordingly, the ABAC lifecycle consists of a single phase only, the `pre` phase, and all access control rules are implicitly classified under it. The ABAC lifecycle is illustrated in the DFA in Figure 3.8.

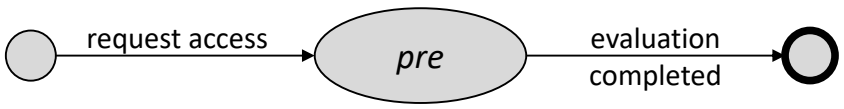

Figure 3.8: DFA illustrating the lifecycle of an ABAC evaluation

To enact this lifecycle, the lifecycle transition policy must specify two rules. The first rule specifies that if the evaluation session is in the `init` phase, then the `request access` event must be fired, triggering a transition to the `pre` phase. The second rule specifies that if the session is in the `pre` phase, then the `evaluation completed` event must be fired,

triggering a transition to the `exit` phase and ending the evaluation session. Therefore, an ABAC evaluation is enacted in the following steps:

(1) the partial access control policy of the `init` phase is evaluated providing a *null* decision and *null* obligations;

(2) the context transformation policy is evaluated next, but it does not specify any rules because contextual changes are irrelevant in ABAC, so the context remains unchanged;

(3) the lifecycle transition policy is then evaluated firing a transition to the `pre` phase;

(4) the evaluation context is then passed to the access control policy of the `pre` phase, and the policy is evaluated providing an ABAC decision and obligations;

(5) the context transformation policy is evaluated again without changing the context;

(6) finally, the lifecycle transition policy is evaluated again and it fires a transition to the `exit` phase terminating the session.

3.4.2 Constructing UCON Using UCON+

UCON does not incorporate rule-based governance of contextual changes, so the context transformation policy is irrelevant here as well, and only the lifecycle transition policy needs to be defined. UCON's lifecycle consists of three temporal phases called `pre`, `ongoing` and `post`. As illustrated in the DFA in Figure 3.9, the lifecycle starts with the `pre` phase when usage is requested. Then, it either terminates if usage is denied or moves to the `ongoing` phase when usage starts. The lifecycle remains in the `ongoing` phase as long as usage is in progress, and transitions to the `post` phase when usage ends or is revoked. Finally, it terminates after passing through the `post` phase.

To enact this lifecycle, the lifecycle transition policy of UCON must specify the following rules:

- if the current phase is `pre` and the decision is `permit`, fire the `start usage` event (this leads to a transition to the `ongoing` phase);
- otherwise if the current phase is `pre` and the decision is `deny`, fire the `exit` event (this ends the session);
- if the current phase is `ongoing` and the decision is `permit`, fire the `in progress` event (to stay in the `ongoing` phase);

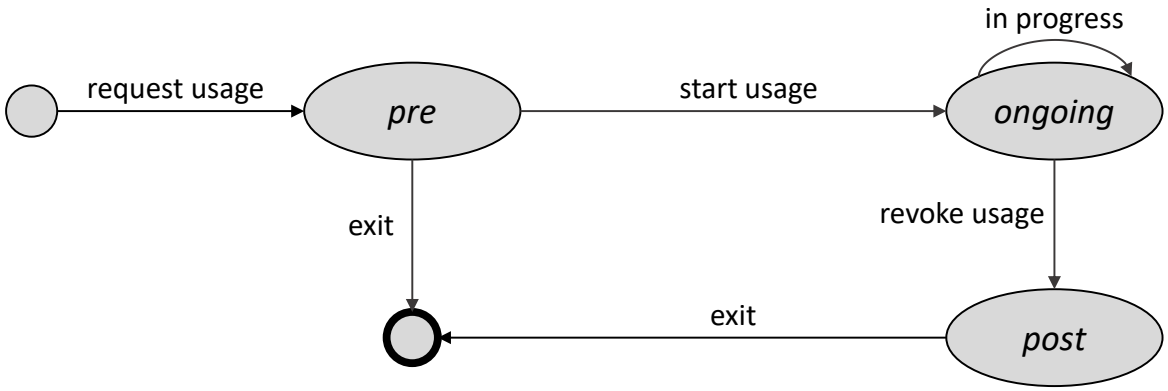

Figure 3.9: DFA illustrating the lifecycle of a UCON evaluation session

- otherwise if the current phase is `ongoing` and decision is `deny`, fire the `revoke usage` event (this causes a transition to the `post` phase);
- finally, if the current phase is `post`, fire the `exit` event (to end the session).

Chapter 4

UCON+ Policy Engine

This chapter introduces a policy engine implemented to realise the proposed model as well as the applied research objectives of this thesis. It specifically describes the policy language and the architecture of the policy engine, and provides an analysis of experimental evaluation results.

4.1 Introduction

To realise the UCON+ model, we introduce a policy engine that embodies its proposed concepts. The engine specifically implements the common behaviour and general structures defined by UCON+. Moreover, it enables the use of policies to specify the extensible behaviour of the model, namely the context transformation and lifecycle transition. This allows the engine to realise different concrete extensions of UCON+ without the need for reimplementation as extensible behaviour is defined in policies. Therefore, the policy engine can be adapted to address different requirements, and to realise our applied research objectives. The engine conserves a full Attribute-Based Access Control (ABAC) baseline, supports auxiliary evaluators and leverages a modular architecture. It consists of core implementation optimised for device environments using C++, as well as a wrapper implemented in Go and optimised for cloud and micro-service environments. This chapter provides an overview of the design and implementation of the policy engine, outlining the used policy language, describing the architecture and interactions among its components, and presenting an experimental performance analysis.

4.2 Policy Language

In this section, we outline the policy language used in the UCON+ policy engine to express access control policies as well as context transformation and lifecycle transition policies. We also demonstrate the use of the policy language with example policies. However, we note that the provided examples are not comprehensive for brevity and are intended solely for demonstration purposes.

4.2.1 Access Control Policies

The policy engine uses Abbreviated Language For Authorisation (ALFA) as a baseline policy language for expressing access control policies. This decision was motivated by the following considerations:

- As outlined in Section 2.4.4, ALFA is a profile of eXtensible Access Control Markup Language (XACML) (a standard and widely adopted policy language), sharing identical semantics and features.
- ALFA's syntax is significantly simpler and less verbose than XACML's syntax, which makes it more human readable and shorter in size allowing faster parsing and evaluation.
- Like XACML, ALFA is highly expressive and flexible, which enables expressing various and complex access control policies.
- ALFA enables fine-grained decision-making, allowing for more precise control over resources.
- ALFA conforms with the definition of access control policies provided by the UCON+ model in Section 3.2.3, as it evaluates into a decision and obligations given a specific context.

As highlighted in Section 3.3, access control policies are classified by the phases of the evaluation lifecycle such that distinct rules apply at each phase. Instead of modifying the ALFA standard, we express such classification by adding an expression in the target clause that checks the value of the phase attribute (e.g., `target clause Attributes.session.phase == "retention"`). This allows the policy engine to filter and select policies and rules based on the current value of the phase attribute.

Listing 4.1 shows an example of an access control policy expressed in ALFA, and used to govern the use of patient data at the *Red Cross Hospital*. The target clause at line 2 specifies that the policy is classified as a `collection` policy and will be enforced upon collecting patient Electronic Medical Records (EMRs). The first rule of the policy (lines 4 to 7) specifies that policy engine must discharge an obligation to obtain patient consent if it has not been obtained yet. The second rule (lines 8 to 13) specifies that data collection is permitted given the following conditions: (1) the data subject (i.e., patient) consents the collection of their EMRs; and (2) the subject collecting data is a staff of the Red Cross Hospital; and (3) the purpose of collecting data matches the purposes consented by the patient. Finally, a default deny rule is added to deny any other cases.

Listing 4.1: Non-comprehensive example of data collection policy expressed in ALFA

```
1  policy collection {
2      target clause Attributes.session.phase == "collection"
3      apply firstApplicable
4      rule obtain-consent {
5          target clause Attributes.dataSubject.consent == "missing"
6          deny on deny { obligation obtainConsent { } }
7      }
8      rule allow-usage {
9          target clause Attributes.dataSubject.consent == "obtained"
10         condition Attributes.subject.institution == "Red Cross Hospital" and
11         Attributes.collectionPurpose in Attributes.dataSubject.consent.purposes
12         permit
13     }
14     rule default { deny }
15 }
```

Similarly, Listing 4.2 presents another example of an access control policy classified under the `processing` phase as specified in the target clause of the policy (line 2). The first rule (lines 4 to 7) allows staff of the Red Cross Hospital to process data, but the data subject (i.e., the patient) must be notified accordingly as specified in obligation `notifyDataSubject` at line 6. The second rule (lines 8 to 15) specifies that staff of other institutions are allowed to process the data given the following conditions: (1) the subject requesting data processing is a staff of a trusted institution; and (2) the purpose of processing data matches the purposes consented by the patient. In addition, the rule includes two obligations specifying that the data must be anonymised before processing and the patient must be informed about the use of their data. The last rule is default

rule that denies any other cases.

Listing 4.2: Non-comprehensive example of data processing policy expressed in ALFA

```
policy processing {
    target clause Attributes.session.phase == "processing"
    apply firstApplicable
    rule hospital-staff {
        target clause Attributes.subject.institution == "Red Cross Hospital"
        permit on permit { obligation notifyDataSubject { } }
    }
    rule research {
        target clause Attributes.subject.institution in Attributes.trustedParties
        condition Attributes.processingPurpose in
                        Attributes.dataSubject.consent.purposes
        permit on permit {
            obligation anonymize { }
            obligation notifyDataSubject { }
    }}
    rule default { deny }
}
```

4.2.2 Context Transformation Policies

Context transformation policies are semantically different than access control policy as they do not need to evaluate into an authorisation decision (e.g., `permit`, `deny`). Instead, these policies only need to modify attribute values based on predefined rules. Nonetheless, we use ALFA to express context transformation policies by exploiting ALFA obligations to specify attribute values. For demonstration purposes, we provide in Listing 4.3 a non-comprehensive example showing how ALFA can be used to express context transformation policies.

We use the `permitUnlessDeny` combining algorithm because it combines all obligations from all applicable rules that evaluate into `permit` as long as there are no applicable `deny` rules. The decision used in all rules is always set to `permit`, so all applicable rules will be enforced and attribute values will be modified accordingly. The first rule (lines 4 to 9) applies if the `data.category` attribute is required in the evaluation session. The rule sets the `data.category` to "`public benefit`" if the `data.type` is "`clinical trial`" or "`genomic`" data. This is achieved by enforcing the `updateDataCategory` obligation specified at lines 7 and 8.

Likewise, the second rule (lines 10 to 16) sets the `transferPurpose` attribute to "`untrusted`" if data transfer is requested for "`research`" purposes by a *non-research* institution. Such rules allow the policy engine to govern attributes coming from external third parties in a decentralised environment.

Listing 4.3: Non-comprehensive example of context transformation policy expressed in ALFA

```
1  policy context-transformation {
2      target clause Attributes.session.model == "health-data"
3      apply permitUnlessDeny
4      rule data {
5          target clause "Attributes.data.category" in Attributes.session.required
6          condition Attributes.data.type in ["clinical trial", "genomic"]
7          permit on permit{ obligation updateDataCategory {
8                  Attributes.data.category = "public benefit"
9      }}}
10     rule untrustedPurpose {
11         target clause "Attributes.transferPurpose" in Attributes.session.required
12         condition Attributes.subject.institution.type != "university"
13         and Attributes.transferPurpose == "research"
14         permit on permit { obligation updateTransferPurpose {
15                 Attributes.transferPurpose = "untrusted"
16     }}}
17 }
```

4.2.3 Lifecycle Transition Policies

Lifecycle transition policies are also semantically different than access control policies and do not need to evaluate into an authorisation decision. However, we also exploit obligations to express these policies using ALFA. The UCON+ model specifies that the lifecycle is a Deterministic Finite Automaton (DFA) and the lifecycle transition policy fires DFA events (i.e., labels) causing the phase to change according to the DFA's transition function. However, in the policy engine we use the lifecycle transition policy to directly modify the phase. This means that the DFA and its transition function are implicitly defined in the policy. Listing 4.4 shows a non-comprehensive example of a lifecycle transition policy that enacts the Usage Control (UCON) lifecycle DFA illustrated in Figure 4.1. Further examples of lifecycle transition policies are provided in Chapter 5.

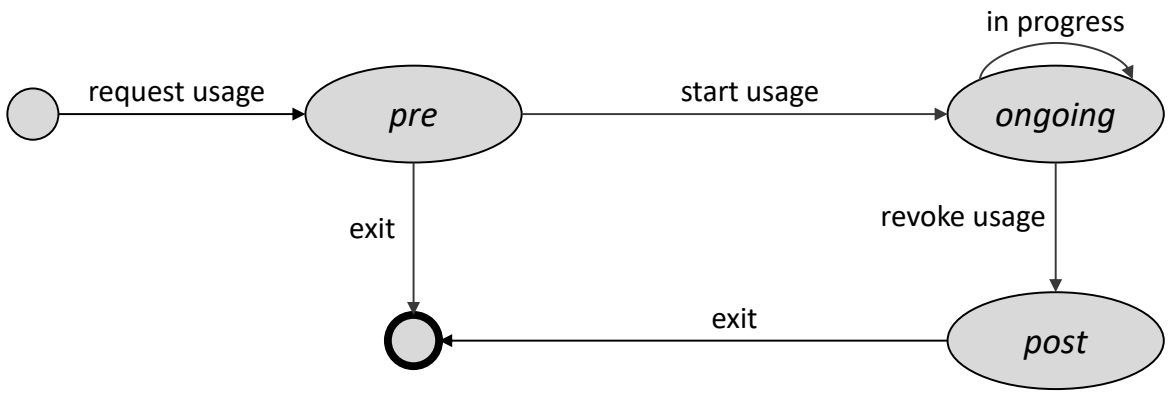

Figure 4.1: DFA illustrating the lifecycle of a UCON evaluation session

All sessions in the policy engine start with the `init` phase. Accordingly, rule `init` (lines 4 to 8) sets the phase to `pre`, implicitly firing the `usage requested` event in the DFA in Figure 4.1. Rule `usageAllowed` at lines 9 to 15 changes the phase from `pre` to `ongoing` if the decision is `permit` and all obligations are fulfilled. This corresponds to the `start usage` transition in the DFA. In contrast, rule `usageDenied` specifies that the session must transition to the `exit` phase and terminate if the `pre` decision is `deny` or obligations are violated. The last rule sets the phase to `post` when the session is in the `ongoing` phase and the obligations are violated, or usage is denied or ended. Similarly, the rest of the rules can be specified to completely enact the UCON lifecycle. By directly updating the value of the phase, the policy implicitly defines the DFA and its transitions.

4.3 Architecture

The architecture of the UCON+ policy engine is based on the XACML architecture [73]. The policy engine leverages the Publish/Subscribe (Pub/Sub) pattern, which enables asynchronous communications and loose coupling between components offering flexibility and scalability. This makes the architecture modular and allows adding, upgrading or substituting component instances where necessary. In this section, we outline the components of the logical architecture and describe how they interact to execute a UCON+ evaluation session.

Listing 4.4: Non-comprehensive example of UCON's lifecycle transition policy expressed in ALFA

```
policy ucon {
    target clause Attributes.session.model == "ucon"
    apply firstApplicable
    rule init {
        target clause Attributes.session.phase == "init"
        permit on permit {
            obligation requestUsage { Attributes.session.phase = "pre" }
    }}
    rule usageAllowed {
        target clause Attributes.session.phase == "pre"
        condition Attributes.session.decision == "permit" and
                  Attributes.session.obligations.status == "fulfilled"
        permit on permit {
            obligation startUsage { Attributes.session.phase = "ongoing" }
    }}
    rule usageDenied {
        target clause Attributes.session.phase == "pre"
        condition Attributes.session.decision == "deny" or
                  Attributes.session.obligations.status == "violated"
        permit on permit {
            obligation exit { Attributes.session.phase = "exit" }
    }}
    rule usageRevoked {
        target clause Attributes.session.phase == "ongoing"
        condition Attributes.session.decision == "deny" or
                  Attributes.session.obligations.status == "violated" or
                  Attributes.session.action == "end-usage"
        permit on permit {
            obligation endUsage { Attributes.session.phase = "post" }
    }}
};
```

4.3.1 Components

Figure 4.2 illustrates the architecture of the policy engine consisting of eleven major components. We categories the components into the following three categories: (1) *fixed* components have predefined behaviours and functionalities that cannot be modified or customised, and they are typically designed to perform specific tasks or provide specific features regardless of the environment of application domain; (2) *configurable* components provide a range predefined functionalities that can be adjusted or configured, allowing users or administrators to customise their behaviour according to specific requirements; (3) *programmable* components provide basic functionalities along with interfaces that allows them to be extended, modified or integrated with external systems to realise domain-specific requirements. The components are described in the following subsections.

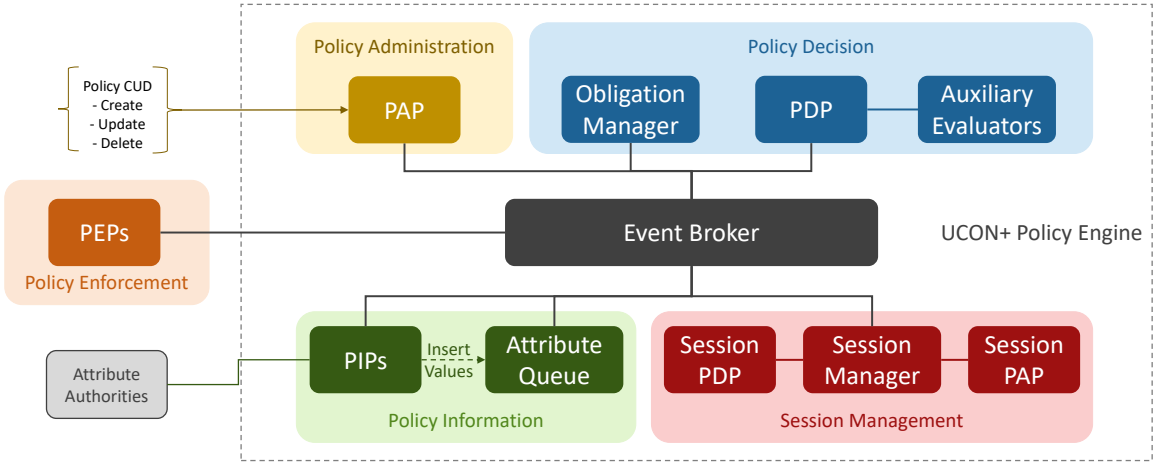

Figure 4.2: Architecture of the UCON+ policy engine

Fixed Components

- `Event Broker` enables event-driven communications between the components using the Pub/Sub pattern. It can be replaced with other brokers such as Kafka or Nats for cloud deployments.

- `Policy Decision Point (PDP)` is responsible for evaluating access control policies and making usage decisions and obligations. It leverages an ABAC evaluator for evaluating ABAC policies as well as additional auxiliary evaluators for evaluating additional expressions such as trust level evaluations.

- `Session PDP (SPDP)` evaluates context transformation and lifecycle transition policies.

- `Attribute Queue (AQ)` stores attribute update *events* in a queue to be processed by evaluation sessions. This allows each session to have its own snapshot of attribute values, which enables trusted context transformation based on the rules of the context transformation policy.

Configurable Components

- `Policy Administration Point (PAP)` stores and manages access control policies. It provides an administrative Application Programming Interface (API) to retrieve, create, update and delete policies. The PAP can be configured to use a variety of persistent or in-memory stores such as SQL, NoSql, Redis, etc.

- `Session PAP (SPAP)` stores and manages context transformation and lifecycle transition policies. It was split from the PAP of access control policies because UCON+ decouples the access control logic from context transformation and lifecycle transition, so different policy languages may be used. More importantly, the parties expected to create and manage context transformation and lifecycle transition policies are not the same parties that issue access control policies. This is because context transformation and lifecycle transition policies construct specific extensions of UCON+, so the parties that create such policies are expected to be experts in designing policy models rather than policy authors.

- `Session Manager (SM)` is a central component that handles sessions and enacts different extensions of the UCON+ model in each session according to the corresponding context transformation and lifecycle transition policies. It leverages the SPAP and SPDP to fetch and evaluate context transformation and lifecycle transition policies, and execute the session accordingly. It keeps track of all evaluation sessions running within the policy engine, which enables continuous monitoring and policy re-evaluation. It maintains the state of each session by storing all relevant information including its internal session attributes (e.g., phase) as well as a snapshot of external attribute values. It can be configured to use different databases for storing session information.

Programmable Components

- `Policy Information Points (PIPs)` are responsible for managing and monitoring attributes used by the policy engine. They interface with the systems that provide the attribute values (e.g., Active Directory), so their implementations depend on the specific attributes they manage and the corresponding application domain. For this reason, they are programmable interfaces that need to be implemented in each use-case.

- `Policy Enforcement Points (PEPs)` are the interfaces of the policy engine that integrate with the target application and enforce policy decisions. They intercept resource usage and send usage requests to the policy engine accordingly. They may also incorporate functions that execute obligations specified in policies. For this reason, PEPs are programmable interfaces whose implementation depends on each application domain. The policy engine allows multiple PEPs to enforce decisions and obligations related to the same session. This flexibility is particularly useful for data usage control in cloud environments or distributed architectures where multiple policy decisions must be enforced at different physical locations (e.g., databases, APIs, applications, caches).

- `Obligation Manager (OM)` handles obligations and determines how and where to execute them. It may also execute obligations that are internal to the policy engine (e.g., attribute updates). The OM monitors obligations and reports whether they were fulfilled or violated. Its implementation depends on each use-case and the specified obligations, hence it is a programmable component.

- `Auxiliary Evaluators` are interfaces to external evaluation engines leveraged by the PDP to evaluate non-ABAC expressions such as trust level expressions. They are programmable because their implementations depend on the external engines to be integrated.

4.3.2 Components Interactions

To provide a clearer understanding of how the architecture realises the UCON+ model, we outline the interactions between components within the system. We specifically identify two independent flows that may occur in parallel: the attribute update flow and the evaluation session flow.

Attribute Update Flow

The attribute update flow is a simple process that involves changing attribute values within the policy engine. As illustrated in Figure 4.3, Attribute Authorities (AAs) send attribute update events to the PIPs responsible for the corresponding attributes. The PIPs then add the attribute update events to the AQ and report such events to the SM, which may trigger policy re-evaluations consequently. Attribute updates are independent of evaluation sessions and are handled in parallel to policy evaluation.

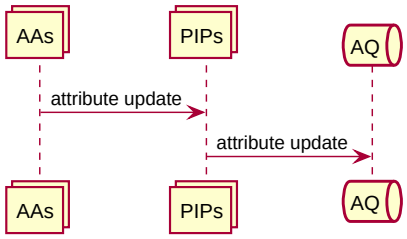

Figure 4.3: Attribute update flow

Evaluation Session Flow

The policy engine enacts an evaluation session through a loop of interactions that ends when the lifecycle reaches the exit phase. The SM is the central component that coordinates these interactions throughout the session. Figure 4.4 illustrates the sequence of interactions within a single step of the loop. The evaluation flow is described as follows.

(1) The evaluation session starts when a PEP sends an evaluation request to the policy engine.

(2) The SM creates a new session and fetches the relevant context transformation and lifecycle transition policies from the SPAP. These policies may be selected based a variety of conditions. For example, if the resource type is a data object, then the policies that construct a data-oriented usage control model would be selected.

(3) Once the context transformation and lifecycle transition policies are selected, the session enters a loop of evaluations and enforcements.

(4) First, the SM invokes the SPDP to evaluate the lifecycle transition policy.

(5) The SM updates the lifecycle phase according to the evaluation result.

(6) If the lifecycle reaches the exit phase, then the SM ends the session.

(7) Otherwise, if the phase did not change, the SM waits for attribute updates to occur. Note that in the very first step of the loop, the phase changes from the `init` phase to the first phase in the lifecycle (e.g., `collection`, `pre`), so the SM does not wait for updates in the first step.

(8) If the phase has changed or if attribute updates have occurred, the SM retrieves attribute values or attribute update events from the AQ.

(9) Thereafter, SM invokes the SPDP to evaluate the context transformation policy.

(10) The SM transforms the context according to the evaluation result.

(11) Once the context is prepared, the SM fetches the access control policies classified under the current phase from the PAP.

(12) Subsequently, it invokes the PDP to evaluate the access control policies.

(13) At this stage, policy evaluation is completed, so the SM sends the decision and obligation to the PEPs and OM respectively.

(14) The OM discharges the obligations to the relevant PEPs or parties responsible for enforcing them.

(15) Finally, the PEPs enforce the decision and obligations, and the OM reports the enforcement status back the SM.

The SM repeats steps (4) to (15) all over again until the lifecycle reaches the `exit` phase.

4.4 Experimental Performance Analysis

This section presents the results of an experimental performance analysis of the UCON+ policy engine. The experiments were conducted on a computer running Ubuntu 20.04LTS Linux OS, equipped with a Core i7-9850H CPU and 16GB RAM. We specifically measured the time required to evaluate ABAC policies expressed in ALFA. We conducted our tests with an increasing number of attributes to measure the impact of policy size on evaluation and parsing times. We also measured the overhead introduced by context transformation and lifecycle transition policies. It is necessary to note that the time needed by PIPs to collect attribute values adds to the overhead of policy evaluation.

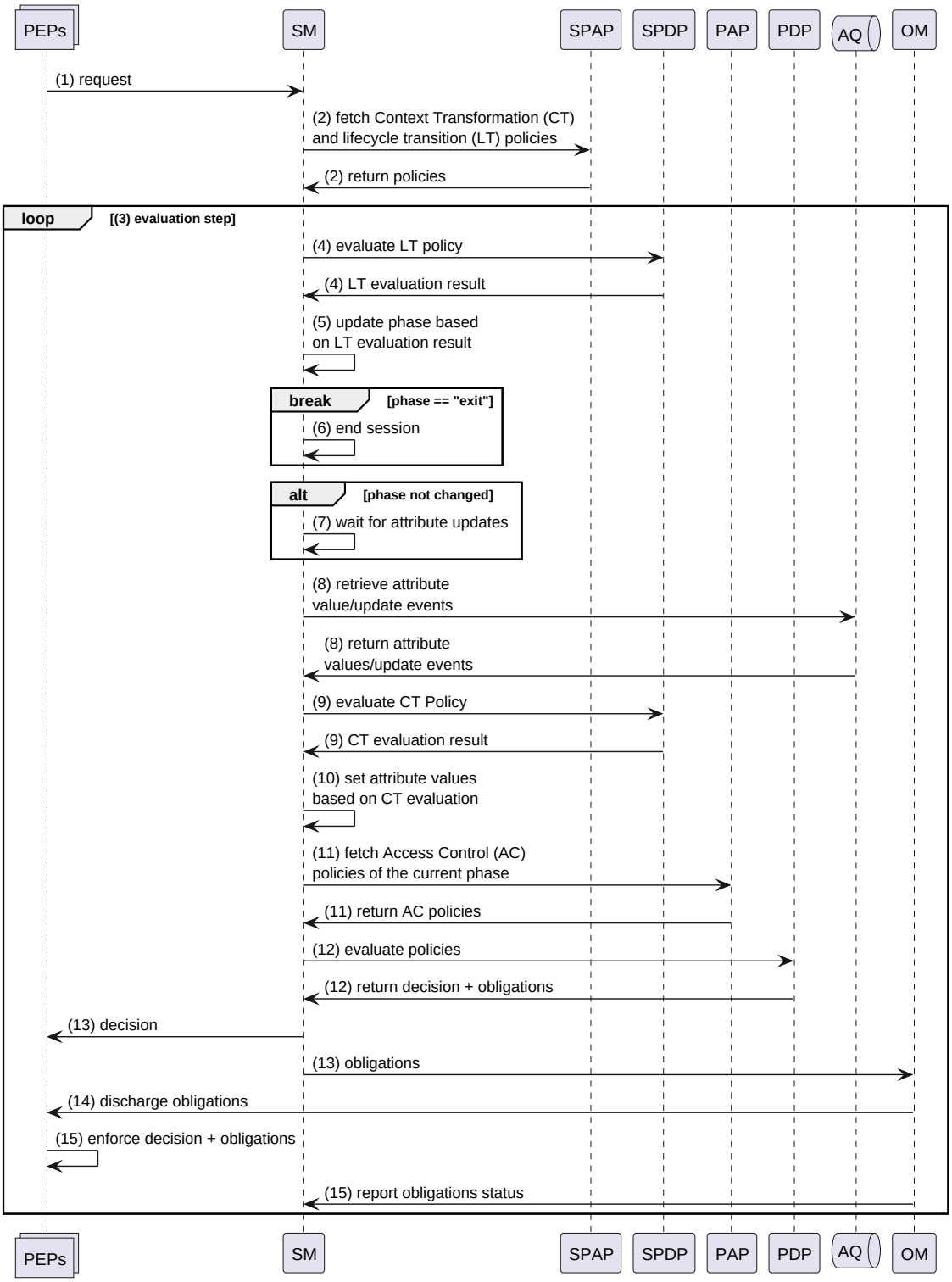

Figure 4.4: Flow of an evaluation session

However, attribute collection overhead is influenced by external factors such network delay or the PIP implementation, so we only measured policy evaluation overhead when all attribute values were already cached in the policy engine. The experimental results are described below.

4.4.1 Overhead of Access Control Policy Evaluation

According to Open Policy Agent (OPA) [2], high-performance environments require policy evaluation times that do not exceed one millisecond (1ms). However, other applications with lower performance requirements may tolerate longer evaluation times within the range of tens of milliseconds. The following experimental results show that our policy engine meets these requirements.

We measured and compared the total time required to parse and evaluate ALFA policies using our UCON+ policy engine to the total time required to parse and evaluate equivalent XACML policies using Balana, an open-source XACML evaluator[1]. We used four policies consisting of five, ten, fifteen and twenty attributes, expressed both in ALFA and XACML. The number of attributes in real-world ABAC policies varies widely depending on the specific use cases, and could be as little as two attributes or as much as fourteen attributes[2] [67]. We ran each test 1000 rounds and calculated the mean time required for policy parsing and evaluation. We observed a standard deviation of 27.1µs. The results are presented in Table 4.1 and Figure 4.5, demonstrating that our policy engine is lightweight, very efficient and highly optimised.

Table 4.1: Average time to parse and evaluate ALFA policies in UCON+ engine compared to XACML in Balana

Number of attributes	**5**	**10**	**15**	**20**
UCON+ policy engine (in ms)	0.707	0.837	0.989	1.119
Balana policy engine (in ms)	33.249	36.327	40.003	42.966

To distinguish between the overhead of policy parsing and policy evaluation, we measured the time to evaluate ALFA policies after they have been already parsed. The results are presented in Table 4.2. They show that policy evaluation in the UCON+ policy engine is highly optimised and efficient when policies are parsed beforehand and cached. They

[1] `https://github.com/wso2/balana`

[2] `https://github.com/wso2-attic/carbon-identity/blob/master/features/xacml/org.wso2.carbon.identity.xacml.server.feature/resources/policies/`

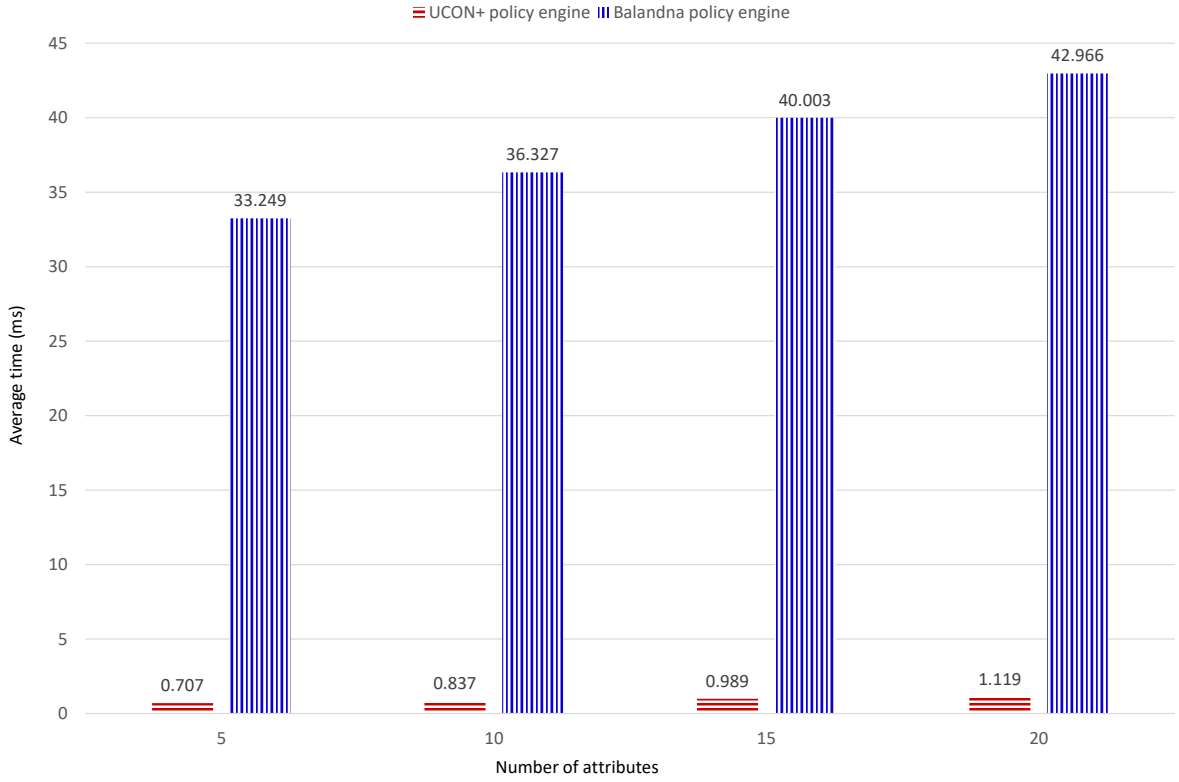

Figure 4.5: Average time to parse and evaluate ALFA policies in UCON+ engine compared to XACML in Balana

also show that the overhead increases linearly as the number of attributes used in the policy grows as illustrated in Figure 4.6.

Table 4.2: Average time to evaluate already-parsed ALFA policies

Number of attributes	**5**	**10**	**15**	**20**
Average time to evaluate a policy (in µs)	109.9	197.1	288.2	375.8

4.4.2 Overhead of Context Transformation and Lifecycle Transition Policies

UCON+ adds extra overhead to evaluation sessions because each access control policy evaluation must be preceded by context transformation and lifecycle transition policy evaluations. Context transformation and lifecycle transition policies are also expressed in ALFA, so the overhead of evaluating each of these policies depends on the number of

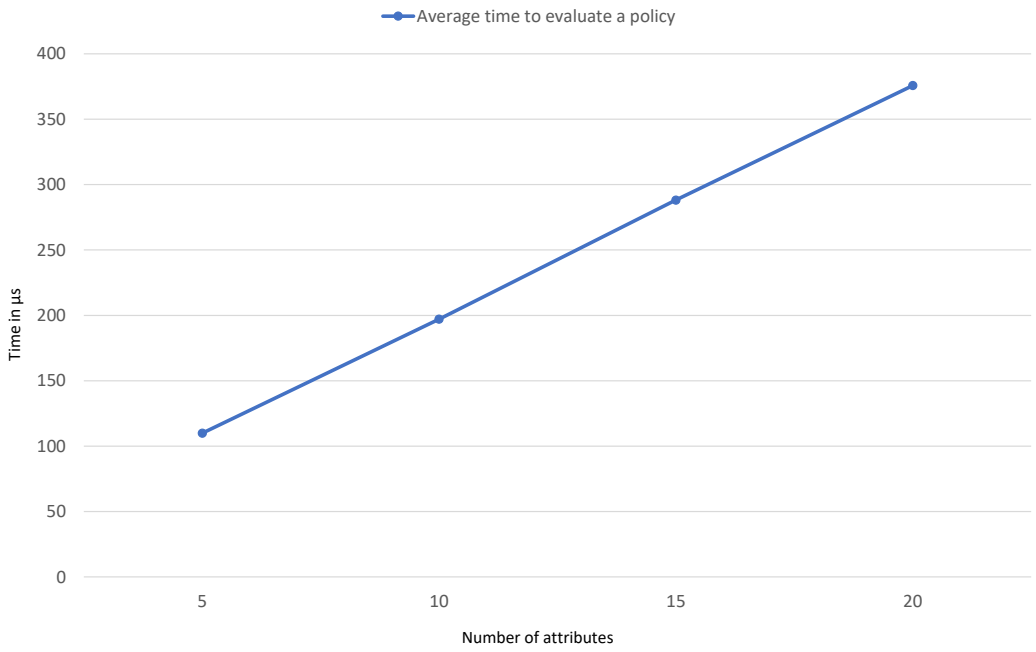

Figure 4.6: Average time to evaluate ALFA policies in the UCON+ policy engine

attributes as presented above. We note that these policies typically include a small number of attributes (e.g., no more than 5 attributes) as shown in the examples in Section 4.2. Nonetheless, context transformation and lifecycle transitions can be hardcoded in the implementation of the policy engine instead of being specified in policies. This improves the performance and efficiency of evaluation sessions at the expense of dynamic policy-based behaviour.

To provide a correct measure of the overhead incurred by context transformation and lifecycle transition policies, we implemented a hardcoded version of traditional UCON in our policy engine and used it as a frame of reference. We then created context transformation and lifecycle transition policies to construct traditional UCON and compared the performance to the hardcoded version. We specifically measured the time required for evaluating a session, from request to termination, using both the hardcoded and policy-based UCON. In both scenarios, the session followed exactly the same conditions and used the same access control policies. We conducted this test with two access control policies: a simple one involving two attributes only and a more complex one involving eight attributes. We repeated each test a 1000 times to accommodate for external factors, and we calculated the mean value.

The experimental results are outlined in Table 4.3 and illustrated in Figure 4.7. The results show that the overhead of context transformation and lifecycle transition policies is around 36% in the test involving a simple policy and drops to 19% in the test involving a complex policy. This means that the overhead incurred by these policies is near constant and becomes insignificant in complex environments involving complex policies.

Table 4.3: Time to evaluate a hard-coded and a policy-based UCON session

Number of attributes	**2**	**8**
Time to evaluate a session in hard-coded UCON (in µs)	966	2287
Time to evaluate a session in policy-based UCON (in µs)	1319	2739

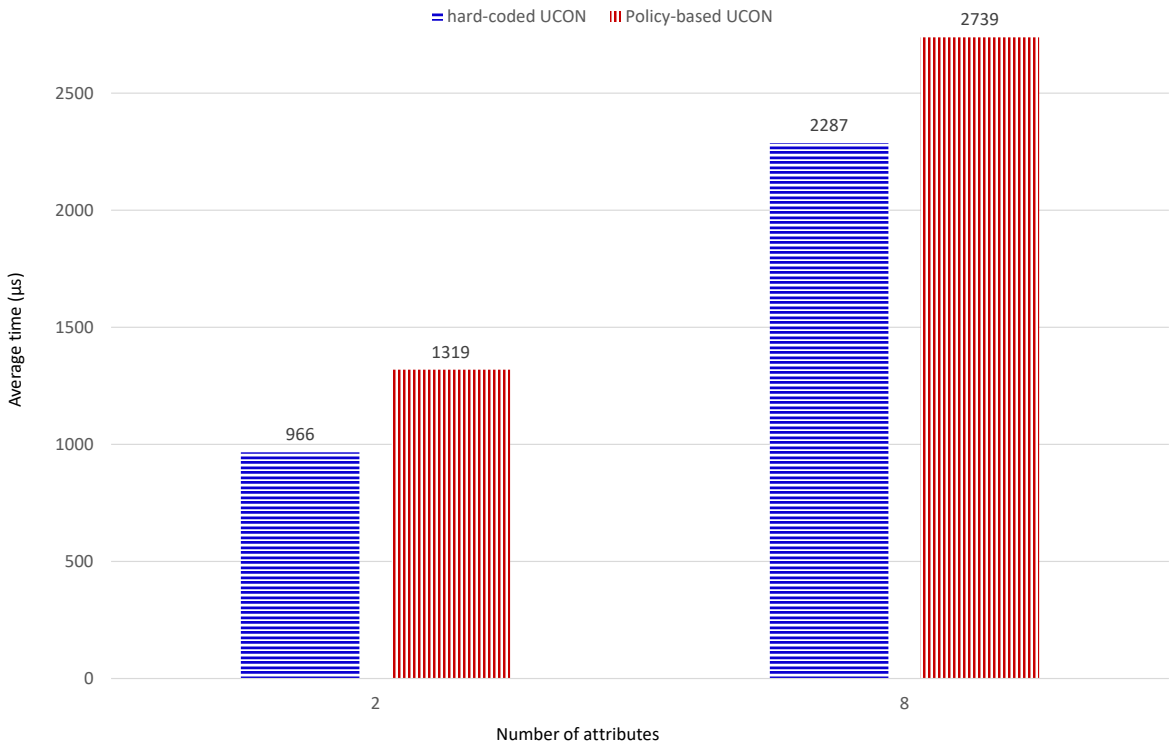

Figure 4.7: Time to evaluate a hard-coded and a policy-based UCON session

Chapter 5

Industrial Applications of the Policy Engine

This chapter outlines the industrial applications of the UCON+ model and policy engine.

5.1 Introduction

In this chapter we present the industrial applications of UCON+ that address our applied research goals. Four application domains stand out: Industrial/International Data Spaces (IDS), data hubs, smart vehicles as well as Self-Sovereign Identity (SSI) and credential transformation. We specifically describe the prototypes that were designed and developed for each one of the four application domains. We showcase how the policy engine was adapted to the requirements of each prototype, demonstrating the extensibility and flexibility of UCON+. We also outline the use-cases that demonstrate the use of the developed prototypes, and provide examples of policies expressed in Abbreviated Language For Authorisation (ALFA).

5.2 Industrial/International Data Spaces (IDS)

IDS have emerged as a new paradigm that aims to facilitate secure, sovereign and standardised data exchange between data providers and data consumer. It enables the data provider and consumer to negotiate a data sharing agreement that specifies their conditions and requirements for sharing and using the data. The IDS ecosystem incorporates several technical components that realise the concepts as illustrated in Figure 5.1.

The connector is the core component that serves as the gateway to the IDS ecosystem. The connectors of the data provider and consumer exchange control message to negotiate a data sharing agreement. They also exchange metadata with third party entities to enable trustworthy and verifiable execution of the data sharing agreement. Moreover, the connector interfaces with data sources and sinks (i.e., databases) of the provider and consumer to invoke and monitor data exchange.

Clearing houses act as intermediaries that oversee and validate the data exchange process. They collect logs and proofs from the data producer and consumer and attest the compliance with the data sharing agreement. The also resolve disputes to address any issues that may arise during the data sharing process.

Metadata brokers enable publishing and querying information about data providers and consumers as well as the data and services they provide. For example, data providers use the metadata broker to publish metadata about the data they provide such as the data type, size, quality, format, etc. On the other hand, data consumers publish information about the guarantees they can provide such as compliance with regulations as well as attestations by third parties.

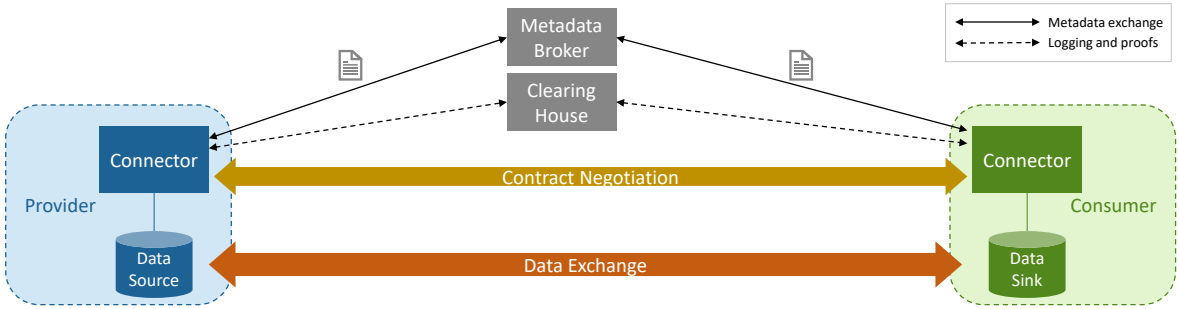

Figure 5.1: Technical components of an IDS

IDS [75] identifies Usage Control (UCON) as a key technology for governing the data exchange and usage according to the negotiated data sharing agreement. For this reason, we demonstrate UCON+ in an IDS to govern the data exchange and usage, as well as the contract negotiation between the provider and consumer. We particularly leverage UCON+ in Boot-X, which is a cloud-based implementation of IDS. This is shown in the open-source architecture of Boot-X[1]. We only focus on using UCON+ for policy-based governance on the producer and consumer sides, so we do not discuss interactions with the clearing houses and metadata brokers. To this end, we describe how the UCON+ policy

[1] `https://boot-x.eu/open-source-architecture/`

engine is used to drive the contract negotiation and govern data exchange and usage. We also discuss the evaluation lifecycle as well as the corresponding lifecycle transition and access control policies. Moreover, we present the Boot-X architecture highlighting how UCON+ is integrated.

5.2.1 Contract Negotiation

Contract negotiation is a key aspect of IDS that enables the data provider and consumer to negotiate a legally binding data sharing agreement. Both parties agree on the conditions and requirements for the data exchange and enforce them accordingly. The IDS protocol [48] specifies a negotiation process that involves exchanging offers and counteroffers until a mutual agreement is reached or one of the two parties rejects the negotiation. We leverage UCON+ to automate the negotiation process using policies that specify the conditions to be included in offers and counteroffers. The policies can also prompt for human input when necessary. To this end, we present the evaluation lifecycle used to enable the negotiation process and example policies that specify offers and counteroffers.

Negotiation Lifecycle

We devised a contract negotiation lifecycle consisting of two phases as illustrated in Figure 5.2. The same lifecycle is used both on the data provider and consumer sides to negotiate a contract. The lifecycle starts with the `negotiation` phase where policies specify offers and counteroffers. The lifecycle stays in this phase while the two parties exchange offers and counteroffers. If negotiation is rejected by either one of the two parties, the lifecycle terminates. Otherwise if a mutual agreement is reached, the lifecycle transitions to the `signing` phase where policies specify actions that must be taken upon signing the contract.

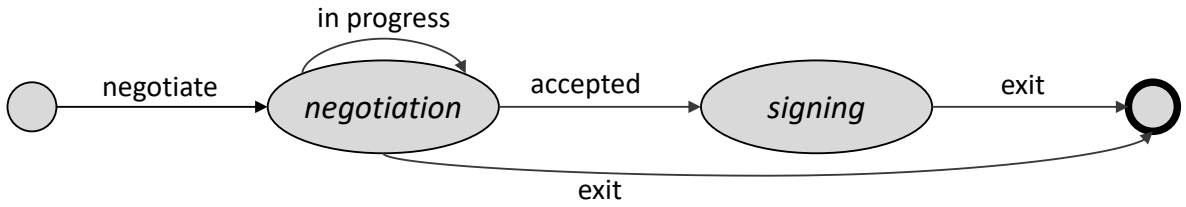

Figure 5.2: DFA illustrating the evaluation lifecycle of contract negotiation

Listing 5.1 presents the corresponding lifecycle transition policy. The first rule ini-

tialises the session by setting the phase to `negotiation`. The second rule keeps the session in the `negotiation` phase if the decision is `deny` and there are obligations to be enforced. A deny decision here means that the party rejects the offer, but is going to propose a counteroffer specified by obligations. The phase remains unchanged because this rule does not specify an obligation that modifies the phase, and because the `firstApplicable` combining algorithm is used, so none of the rest of the rules is considered. The `negotiationDenied` rule terminates the session if one of the two parties rejects the offer. This rule checks the response provided by the other party as well as the decision of the local party. For example, if we consider this lifecycle on the data provider side, the deny decision check in this rule corresponds to the data provider rejection, while the response check corresponds to the data consumer rejection. Rule `negotiationCompleted` sets the phase to `signing` if the two parties have reached a mutual agreement. The rule checks if the decision of the local party is `permit` *and* the response of the other party is `accepted`. Finally, the last rule terminates the session after signing the agreed upon contract.

Negotiation Policies

IDS specifies several templates for data usage policies that consider several criteria for usage conditions. We consider two criteria in this context: the data usage purpose as well as the number of times of using the dataset. Accordingly, we provide examples of ALFA rules used by the data provider to negotiate these criteria during the contract negotiation.

Listing 5.2 shows a non-comprehensive example of ALFA rules used to negotiate a contract by the data provider. The first rule specifies that the contract shall be accepted if the data consumer is requesting data for non-commercial purposes and the intends to use the data no more than five times. The `permit` decision of this rule indicates that the contract is accepted, so the lifecycle transitions to the signing rule as specified above. The second rule specifies a counteroffer when the consumer requests data sharing for research purposes with a number of usages greater than ten. The rule rejects the consumer request but sends a counteroffer with a number of usages acceptable for the data provider as expressed in the obligation. This demonstrates how policies can drive the contract negotiation process and decide whether to accept and reject an offer, and when to propose a counteroffer. The `commercialUse` rule prompts for human intervention in the negotiation process when the consumer requests data sharing for commercial purposes. This shows how policies can involve humans in the loop in specific cases of negotiation. The last rule applies when a contract has been negotiated and accepted and the lifecycle

Listing 5.1: Non-comprehensive example of the lifecycle transition policy of contract negotiation

```
policy negotiation-lifecycle {
    target clause Attributes.session.model == "contract-negotiation"
    apply firstApplicable
    rule init {
        target clause Attributes.session.phase == "init"
        permit on permit {
            obligation start { Attributes.session.phase = "negotiation" }
    }}
    rule negotiationInProgress {
        target clause Attributes.session.phase == "negotiation"
        condition Attributes.session.decision == "deny"
              and Attributes.session.obligations
        permit
    }
    rule negotiationDenied {
        target clause Attributes.session.phase == "negotiation"
        condition Attributes.session.decision == "deny"
               or Attributes.offer.response == "rejected"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}
    }
    rule negotiationCompleted {
        target clause Attributes.session.phase == "negotiation"
        condition Attributes.session.decision == "permit"
              and Attributes.offer.response == "accepted"
        permit on permit {
            obligation startRetention { Attributes.session.phase = "signing" }
    }}
    rule signed {
        target clause Attributes.session.phase == "signing"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}
    }
};
```

has transitioned to the `signing` phase. The rule specifies two obligations to sign the contract and to notify the legal team accordingly. These examples demonstrate how UCON+ is used to govern the contract negotiation process. Policies can be specified to check the contents of offers and accept, reject or propose counteroffers accordingly.

Listing 5.2: Non-comprehensive example of IDS negotiation rules of the data provider

```
rule acceptContract {
    target clause Attributes.session.phase == "negotiation"
    condition Attributes.consumer.offer.purpose != "commercial"
          and Attributes.consumer.offer.numberOfUsages < 5
    permit
}
rule researchUse {
    target clause Attributes.session.phase == "negotiation"
    condition Attributes.consumer.offer.purpose == "research"
           and Attributes.data.usageCounter > 10
    deny on deny { obligation counterOffer { numberOfUsages = 10 } }
}
rule commercialUse {
    target clause Attributes.session.phase == "negotiation"
    condition Attributes.consumer.offer.purpose == "commercial"
    deny on deny { obligation prompt {
        email = Attributes.provider.legal.email
    }}
}
rule sign{
    target clause Attributes.session.phase == "signing"
    permit on permit {
        obligation sign {}
        obligation notify { email = Attributes.provider.legal.email }
    }
}
```

5.2.2 Contract Enforcement

Once a mutual data sharing agreement has been reached, the data provider transfers the data to the data consumer, and the enforcement of the agreement begins. In this context, we consider policy enforcement on the data consumer side only. We describe the evaluation lifecycle used for enforcing the contract as well as the corresponding lifecycle transition policy. We also provide examples of policies that govern data usage on the consumer side.

Data Usage Lifecycle

The data usage lifecycle of the data consumer consists of two phases as illustrated in Figure 5.3. It starts with the `retention` phase where policies specify the conditions for storing data on the consumer databases. The lifecycle terminates when data retention is denied as per the data sharing agreement. The lifecycle transitions to the `processing` phase when data access is requested, and remains in this phase while data usage is in progress. Once data usage is completed or denied, the lifecycle moves back to the `retention` phase.

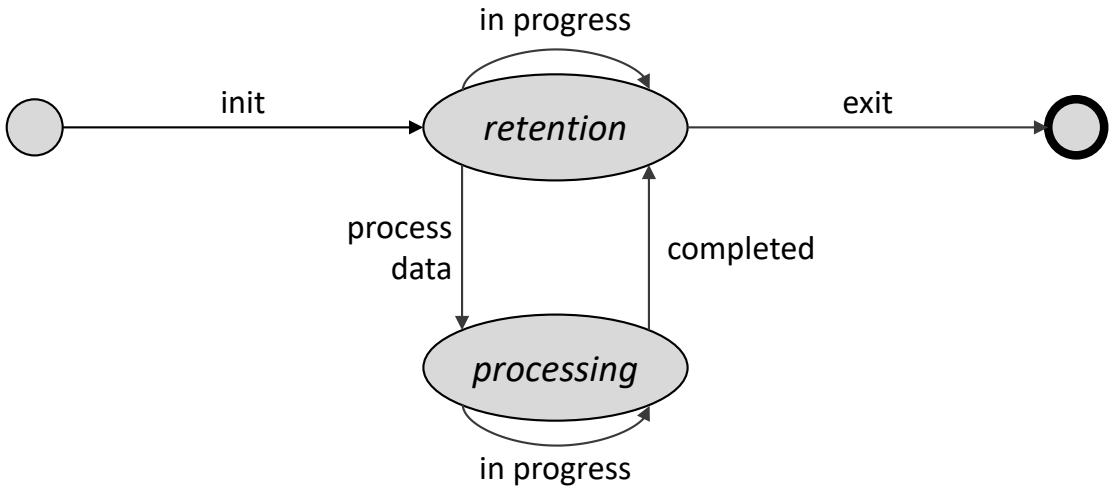

Figure 5.3: DFA illustrating the evaluation lifecycle of governing data usage on the consumer side

A non-comprehensive example of the corresponding lifecycle transition policy is provided in Listing 5.3. The `init` rule initialises the session by setting the phase to `retention`. The second rule keeps the lifecycle in the retention phase as long as data retention is allowed. Rule `retentionDenied` terminates the session when data retention is denied. The `processingRequested` rule changes the phase from `retention` to `processing` when data processing is requested. Finally, the last rule sets the phase back to `retention` if data processing is completed or denied.

Data Usage Policies

As mentioned above, IDS specifies policy templates the consider several criteria for usage conditions. We presented how these criteria can be negotiated by providing examples of contract negotiation rules used by the data provider. To show how UCON+ is used

Listing 5.3: Non-comprehensive example of the lifecycle transition policy of data usage on the data consumer side

```
policy ids-lifecycle {
    target clause Attributes.session.model == "ids"
    apply firstApplicable
    rule init {
        target clause Attributes.session.phase == "init"
        permit on permit {
            obligation init { Attributes.session.phase = "retention" }
    }}
    rule retentionInProgress {
        target clause Attributes.session.phase == "retention"
        condition Attributes.session.decision == "permit"
        permit
    }
    rule retentionDenied {
        target clause Attributes.session.phase == "retention"
        condition Attributes.session.decision == "deny"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}
    }
    rule processingRequested{
        target clause Attributes.session.phase == "retention"
        condition Attributes.session.action == "process-data"
        permit on permit {
            obligation startProcessing { Attributes.session.phase = "processing" }
    }}
    rule collectionCompleted {
        target clause Attributes.session.phase == "processing"
        condition Attributes.dataProcessing.completed
               or Attributes.session.decision == "deny"
        permit on permit {
            obligation toRetention { Attributes.session.phase = "retention" }
    }}
};
```

to govern data usage according to the negotiated contract, we provide in Listing 5.4 a non-comprehensive example of ALFA rules used by the consumer. The rules specify data usage conditions based on the criteria negotiated in the contract.

The first rules denies data retention when the number of usages of the data exceeds the number of usages agreed upon in the counter. The rule specifies obligations to delete the

data and to notify the data provider accordingly. When this rule applies, the corresponding Policy Enforcement Points (PEPs) automatically delete the data form the consumer databases. This demonstrates how the cloud service provides guarantees of contract enforcement. The `allowRetention` rule allows data retention as long as the number of usages has not exceeded the agreed upon number. It also specifies an obligation to encrypt data before storage. The last rule specifies the condition for data processing. It allows data processing if the purpose of the application processing data matches the usage purpose agreed upon in the contract. The rule also specifies an obligation to increment the data usage counter every time the data is used. This enables enforcing the number of usages criteria negotiated within the contract. Note that the rules use the criteria negotiated in the contract in their conditions. This is enabled by parsing these criteria from the contract and setting them as attributes in the UCON+ policy engine.

Listing 5.4: Non-comprehensive example of IDS data consumer usage rules

```
1  rule denyRetention {
2      target clause Attributes.session.phase == "retention"
3      condition Attributes.data.usageCounter >=
   Attributes.contract.numberOfUsages
4      deny on deny {
5          obligation delete { id = Attributes.data.id }
6          obligation notify { email = Attributes.contract.producer.email }
7  }}
8  rule allowRetention {
9      target clause Attributes.session.phase == "retention"
10     condition Attributes.data.usageCounter <
   Attributes.contract.numberOfUsages
11     permit on permit { obligation encrypt { algorithm = "AES" } }
12 }
13 rule allow {
14     target clause Attributes.session.phase == "processing"
15     condition Attributes.application.purpose == Attributes.contract.purpose
16     permit on permit {
17         obligation increment { id = "Attributes.data.usageCounter" }
18 }}
```

5.2.3 Architecture

We describe the logical architecture of the integration of UCON+ in Boot-X outlining its components illustrated in Figure 5.4. We also discuss how these components interact

during contract negotiation and data usage. Applications are central components to this architecture. On the data producer side, applications are used to provide Application Programming Interfaces (APIs) to share data. In contrast, applications on the data consumer side are used to collect data from the provider and process it. Applications interface with the data space connector, to initiate a data exchange and contract negotiation process. The connector also invokes applications to start sending data once a data sharing agreement has been negotiated. Applications also interface with databases and data storage systems to access, store and process data. UCON+ governs contract negotiation and data usage through PEPs incorporated in the different components of the architecture. For instance, the connector PEP enforces UCON+ policies that drive the contract negotiation. Similarly, PEPs in applications and databases enforce data exchange and usage policies. They restrict data processing in applications to the conditions of data sharing agreement. They also restrict access to data in databases and delete the data as specified by the agreement. The architecture shows the identity system and credential bridge components, which are used to identify subjects within the data space and issue credentials. We explain how these components are used in Section 5.5 below.

Throughout a data exchange process, these components interact as follows. (1) The data exchange process starts when an application on the data consumer side requests a dataset from a producer. The application specifically invokes the data consumer connector to start the negotiation process. (2) The connector then starts the negotiation process with the connector of the data provider. (3) Once negotiation is completed, the connectors of both the data consumer and data provider invoke UCON+ to start enforcing the agreement. (4) Applications are then invoked to start the physical data process. The data consumer applications access the APIs provided by the data provider to collect the data. Then they store the data in the database of the data consumer. (5) The data consumer application then starts processing the data until UCON+ policies request data deletion as per the agreement. (6) Finally, UCON+ notifies the data storage PEPs to delete the data.

5.3 Cloud Data Hub

Cloud data hubs emerged as efficient solutions for storing and processing large volumes of complex and various data. For example, the Health-x project [14] aims to build a federated cloud hub for sharing and accessing health data and utilise its potential for personal, research and medical purposes. Although this opens many possibilities, opportunities and insights, it also introduces security and privacy concerns. We address these concerns

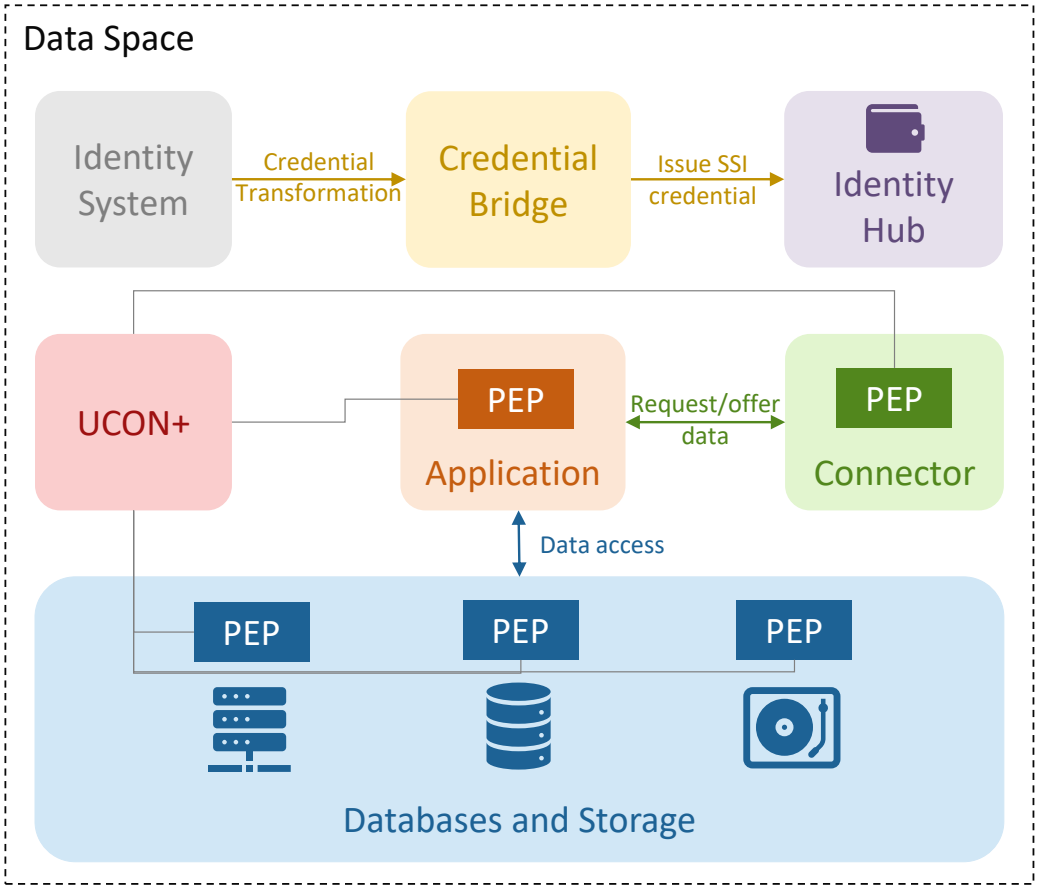

Figure 5.4: Logical architecture of integrating UCON+ in IDS

by leveraging the UCON+ policy engine in a cloud data hub to provide data governance and usage control as specified by regulations. The engine is used to devise a data lifecycle and govern the usage of data objects throughout the stages of the lifecycle.

We consider a single use-case in this prototype, but the concepts are applicable to other similar use-cases. The data hub in this use-case is used to provide digital services related to automotive applications and smart vehicles. Accordingly, the data hub collects and stores personal and vehicular data from users (e.g., car owners), and employs analytics to generate insights and provide services. To this end, we identify two main actors in this use-case as illustrated in Figure 5.5: (1) *data subjects* provide their data and use the provided vehicular services; and (2) *data scientists* (or engineers) process the data and apply data analytics to provide vehicular services. Both actors use software applications to interface with the cloud hub, and to store and process data as shown in the figure.

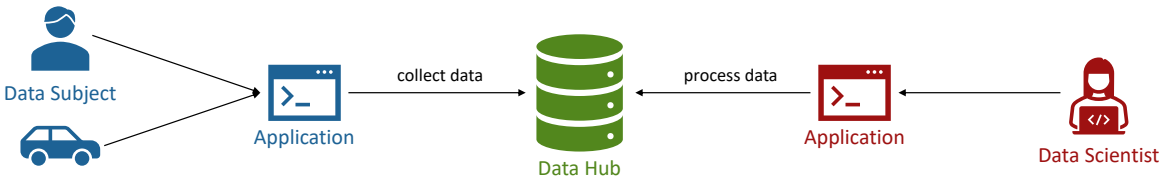

Figure 5.5: Actors of the cloud data hub use-case

We discuss the details of this use-case and prototype in the rest of this section. We specifically describe the devised data lifecycle and the corresponding lifecycle transition policy. We then outline the data governance policies and explain how they protect data, enable data subject rights, and provide transparency over data usage at each phase of the data lifecycle. Finally, we present the architecture of the data hub demonstrating how the policy engine is integrated and how policies are enforced.

5.3.1 Data Lifecycle

To provide data governance in the data hub, we construct an extension of UCON+ that captures the different stages of the data lifecycle. The data lifecycle is inspired by the General Data Protection Regulation (GDPR), which specifies how data must be collected, stored and processed, and that data must be deleted when it is not needed anymore. Accordingly, we define a data lifecycle illustrated in the DFA in Figure 5.6, and consisting of the following four phases: `collection`, `retention`, `processing` and `destruction`.

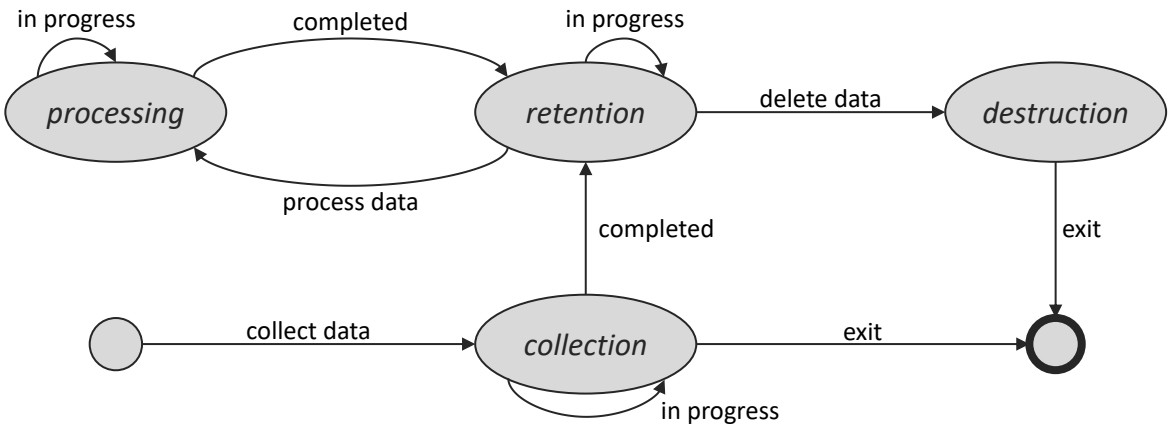

Figure 5.6: DFA illustrating the data lifecycle used in the cloud data hub application

The lifecycle starts with the `collection` phase where policies specify whether collecting a data object is allowed and what actions must be taken (e.g., obtain data subject

consent). The lifecycle stays in the `collection` phase as long as data collection is still in progress as shown in the `in progress` label and transition. If data collection is denied, the lifecycle terminates after firing the `exit` event. Otherwise, the `completed` event is fired once the data collection is completed, and the lifecycle transitions to the `retention` phase accordingly. Policies in the `retention` phase specify the conditions and requirements for storing data at rest. If data usage is requested, the lifecycle moves from `retention` to `processing` where policies specify the conditions for accessing and processing data. As illustrated in the figure, the lifecycle stays in the `processing` phase (i.e., in progress) until data processing is either completed or denied. The lifecycle then moves back to the `retention` phase after firing the `completed` event. Finally, the lifecycle transitions to the `destruction` phase when data retention is denied (e.g., because retention period has expired). The lifecycle terminates after enforcing `destruction` policies, which specify whether data can be archived or must be deleted.

Listing 5.5 presents a non-comprehensive example of the corresponding lifecycle transition policy used to enact the data lifecycle. The first rule specifies that if the data collection decision is `deny` *and* there are obligations to be executed (i.e., there are additional actions to be taken before collecting data), then the lifecycle must stay in the `collection` phase. The phase does not change because the rule does not include an obligation to update the phase. In addition, the `firstApplicable` combining algorithm is used, so if this rule is applicable then none of the other rules will be considered and therefore the phase remains unmodified. The second rule (lines 10 to 13) terminates the session if the data collection is denied and there are no more obligations to be enforced. This is captured by the `firstApplicable` combining algorithm, because the previous rule includes a check for the presence of obligations, so if the previous rule is not applicable, then there are no obligations to be enforced. Rule `collectionCompleted` updates the phase to `retention` when data collection is completed. The `retentionDenied` rule moves the lifecycle from `retention` to `destruction` if the evaluation decision of `retention` policies is `deny` or if obligations are violated. In contrast, if data processing is requested during the `retention` phase, the phase is updated to `processing` as specified by the `processingRequested` rule. The `destructionCompleted` rules terminates the session after evaluating and enforcing `destruction` policies. The default rule is added to keep the phase unmodified when none of the previous rules are applicable. This is equivalent to firing the `in progress` event in the DFA leading to local transitions. Other rules related to the `init` and `processing` phases are omitted due to size constraints, but the same concepts apply nonetheless.

The data lifecycle model depends on several factors, including data types, use-cases

Listing 5.5: Non-comprehensive example of the lifecycle transition policy used for the data lifecycle

```
policy data-lifecycle {
    target clause Attributes.session.model == "data-lifecycle"
    apply firstApplicable
    rule collectionInProgress {
        target clause Attributes.session.phase == "collection"
        condition Attributes.session.decision == "deny"
              and Attributes.session.obligations
        permit
    }
    rule collectionDenied {
        target clause Attributes.session.phase == "collection"
        condition Attributes.session.decision == "deny"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}}
    rule collectionCompleted {
        target clause Attributes.session.phase == "collection"
        condition Attributes.dataCollection.completed
        permit on permit {
            obligation startRetention { Attributes.session.phase = "retention" }
    }}
    rule retentionDenied {
        target clause Attributes.session.phase == "retention"
        condition Attributes.session.decision == "deny"
               or Attributes.session.obligations.status == "violated"
        permit on permit {
            obligation endRetention { Attributes.session.phase = "destruction" }
    }}
    rule processingRequested{
        target clause Attributes.session.phase == "retention"
        condition Attributes.session.action == "process-data"
        permit on permit {
            obligation startProcessing { Attributes.session.phase = "processing" }
    }}
    rule destructionCompleted {
        target clause Attributes.session.phase == "destruction"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}}
    rule default { permit }
};
```

or regulations, so there is no global and comprehensive lifecycle [99]. As mentioned above, the data lifecycle presented here is inspired by and based on the GDPR due to the European context of this work. Nonetheless, the lifecycle can be further adapted to meet other requirements or regulation by modifying the DFA as well as the lifecycle transition policy. For instance, a different lifecycle can be composed based on the Health Insurance Portability and Accountability Act (HIPAA), which specifies rules and conditions for data governance in four stages designated as `data creation`, `data at rest`, `data in transit` and `data disposal`[2]. This demonstrates the extensibility of UCON+ and its ability to cover and support a wide variety of requirements and applications.

5.3.2 Data Governance Policies

The UCON+ policy engine is used in the data hub to provide data governance throughout the data lifecycle and enable data subject rights. We particularly focus on three data subject rights. (1) *The Right to be Informed* grants data subjects the ability to be aware of why and how their data is processed and used. This entails providing data subjects with clear and transparent information about the purpose of processing their data as well as any events or updates that are related to their data. (2) *The Right to Restrict Processing* enables data subjects to limit the processing of their personal data for specific purposes or circumstances. (3) *The Right to be Forgotten* allows data subjects to withdraw their consent or object to the processing of their data. This entails the deletion of their data when there is no compelling reason for its continued processing. However, the data hub may be allowed to keep the data in certain circumstances (e.g., data is categorised as public interest data). Access control policies are used to express these rights and to specify the conditions for data usage. Policies or rules are classified by the phases of the data lifecycle allowing different conditions to apply in different phases. Accordingly, we show non-comprehensive examples of ALFA rules that specify data usage conditions at each phase of the data lifecycle. We assume that the `firstApplicable` combining algorithm is used to combine these rules.

Data Collection

Listing 5.6 shows an example of ALFA rules that are enforced in the `collection` phase upon collecting data from the subject. The first rule (i.e., `missingConsent`) specifies that if the data subject consent is not obtained yet, then data collection must be denied, but

[2]We do not provide a specification of HIPAA lifecycle as this is out of the context of this thesis, which is focused on European applications.

an obligation to obtain the consent must be executed. The session *does not* terminate after this rule, because the lifecycle transition policy leaves the phase unmodified if there are pending obligations in the `collection` phase as described above. Therefore, the session stays in the `collection` phase and re-evaluates the access control policies when an attribute is updated. Once the consent is obtained, the consent status attribute is updated and one of the other two rules applies. If the data subject agrees to the collection and use of their data, the second rule applies and data collection is consequently allowed. Otherwise, if the subject declines, the last rule denies data collection and the session terminates.

Listing 5.6: Non-comprehensive example of data collection rules

```
rule missingConsent {
    target clause Attributes.session.phase == "collection"
    condition !Attributes.subject.consent.status == "missing"
    deny on deny { obligation obtainConsent{} }
}
rule collectionAllowed {
    target clause Attributes.session.phase == "collection"
    condition Attributes.data.subject.consent.status == "agreed"
    permit
}
rule collectionDenied {
    target clause Attributes.session.phase == "collection"
    condition Attributes.data.subject.consent.status == "declined"
    deny
}
```

Data Retention

A non-comprehensive example of data retention rules is provided in Listing 5.7. The first rule denies data retention if the data subject revokes their consent or if the retention period expires. This moves the session to the `destruction` phase where data is deleted as specified by the lifecycle transition policy described above. Hence, the right to be forgotten is enabled by allowing the data subject to withdraw their consent, which leads to data deletion. In contrast, the `allowRetention` rule allows data retention as long as the data has not been stored for longer than the allowed retention period. This ensure that data is not kept indefinitely or longer than is needed. The rule also specifies an obligation to encrypt data before storage.

Listing 5.7: Non-comprehensive example of data retention rules

```
rule consentRevoked {
    target clause Attributes.session.phase == "retention"
    condition !Attributes.data.subject.consent.status == "revoked"
           or Attributes.data.retentionPeriod.expired
    deny
}
rule allowRetention {
    target clause Attributes.session.phase == "retention"
    condition !Attributes.data.retentionPeriod.expired
    permit on permit { obligation encrypt { algorithm = "AES" } }
}
```

Data Processing

Listing 5.8 presents a non-comprehensive example of data processing rules. The first rule checks the role attribute and denies data processing if the subject processing data is not a data scientist. The second rule allows data processing if the purpose of the software application processing data matches the purposes consented by the data subject upon collecting their consent. The rule also specifies an obligation to notify the data subject about the use of their data. This demonstrates how the right to restrict processing and the right to be informed are enabled. Specifically, the right to restrict processing is achieved by allowing the data subject to specify and update the data processing purposes that they accept. The right to be informed is achieved using the obligation that ensure that the data subject is notified when their data is being processed.

Data Destruction

Listing 5.9 shows a non-comprehensive example of ALFA rules applied in the `destruction` phase. The first rule specifies that data must be archived if it is categorised as public benefit data as per the GDPR. Otherwise, the second rule applies and specifies that data must be deleted. The decision of these rules is irrelevant because there is no action to be permitted or denied in the destruction phase. Instead, these rules are used only to specify what actions must be taken when the lifecycle reaches the `destruction` phase. We reiterate that the UCON+ model extends the two-value classical decisions (i.e., `permit`, `deny`), and allows other decisions to be defined and used. This becomes relevant in such situations where there is no action to be denied or permitted, so other decisions can be defined such as *execute obligations* to indicate that the decision is to execute obligations

Listing 5.8: Non-comprehensive example of data processing rules

```
rule denyProcessing {
    target clause Attributes.session.phase == "processing"
    condition Attributes.application.subject.role != "data-scientist"
    deny
}
rule allowProcessing {
    target clause Attributes.session.phase == "processing"
    condition Attributes.application.purpose in
Attributes.data.subject.consent.purposes
    permit on permit {
        obligation notify { recipient = Attributes.data.subject.email }
    }
}
```

only. ALFA, however, does not support such extension, so we use the `permit` decision even though there is no action to be permitted or denied.

Listing 5.9: Non-comprehensive example of data destruction rules

```
rule publicBenefit {
    target clause Attributes.session.phase == "destruction"
    condition Attributes.data.category == "public-benefit"
    permit on permit { obligation archive {} }
}
rule deleteData {
    target clause Attributes.session.phase == "destruction"
    permit on permit { obligation delete {} }
}
```

5.3.3 Architecture

In this section, we describe the logical architecture of the integration of the UCON+ policy engine in the data hub, showcasing how the policies outlined above are enforced. We identify three layers in the logical architecture as illustrated in Figure 5.7. The governance layer incorporates the services and systems that orchestrate the data hub and ensure compliance with regulations and organisational guidelines. This includes administrative and auditing tools as well as policy systems. We only focus on policy-based governance in this work, so we only show the UCON+ policy engine in the figure. The data layer

refers to all data storage systems such as databases, storage buckets and file systems, where collected user data is stored. The application layer includes software applications that collect and process data, and interface with the actors of the targeted use-case. Applications used to process data are deployed within the cloud data hub as shown in the Figure. In contrast, applications used to collect data and interface with the data subject are deployed remotely with respect to the data hub (e.g., mobile applications, smart car applications).

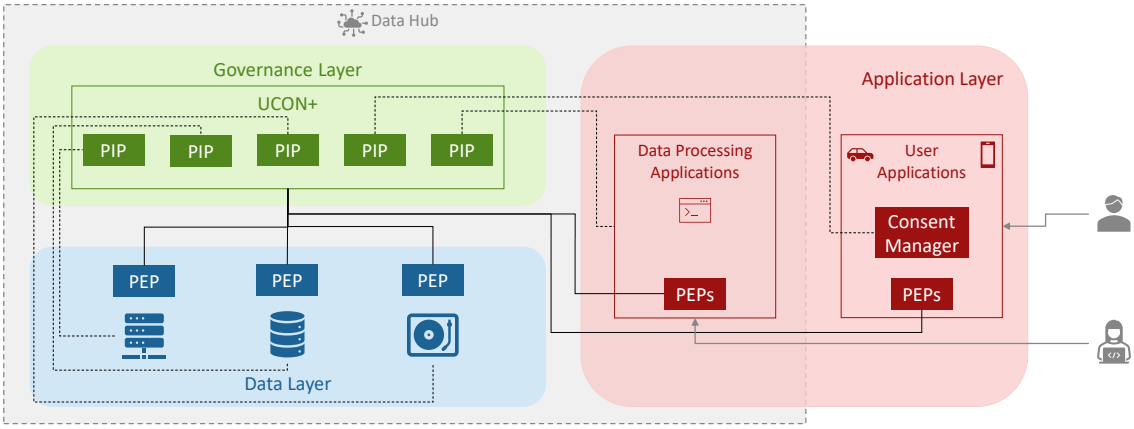

Figure 5.7: Logical architecture of the integration of UCON+ policy engine in the data hub

UCON+ interfaces with the data and application layers through PEPs and Policy Information Points (PIPs). Multiple PEPs are used to enforce data governance decisions and obligations in different components of the data and application layers. Similarly, PIPs retrieve attributes about applications, subjects and data objects from different components in the data and application layers. To demonstrate how this architecture enforces data governance in the data hub, we describe how its components interact at each phase of the data lifecycle based on the access control rules presented in Section 5.3.2.

Data Collection

Data collection starts when the user installs and uses one of vehicular applications provided by the data hub and its corresponding services. The application includes a PEP that invokes UCON+ to initiate a data governance evaluation session and determine whether data collection is allowed. The PEP passes the data object ID as well as the data subject ID upon invoking UCON+. UCON+ evaluates data collection policies and responds to

the PEP with an obligation to obtain data subject consent as specified by the policy rules described in Section 5.3.2. The PEP then invokes the consent manager, which is a submodule introduced to be integrated in applications and interface with users to manage their consent. Accordingly, the consent manager prompts the user to decline or consent the use of their data, and to specify the acceptable purposes of data processing. Once the user provides their consent, the consent manager pushes consent-related attributes to the PIP responsible for retrieving consent attributes in UCON+. The consent manager specifically passes the data subject ID along with the consent attribute values, enabling the PIP to resolve consent attributes to their corresponding data subjects. If the user has declined the consent, UCON+ notifies the PEP to deny to data collection as specified by the policy. Otherwise, UCON+ policy re-evaluation allows the data collection, and the PEP allows the application to start uploading data to the data layer. The PEP also notifies UCON+ that data collection policy enforcement is completed allowing the lifecycle to transition to the `retention` phase.

Data Retention

After transitioning to the `retention` phase, UCON+ starts monitoring attribute values and notifying PEPs in the data layer to enforce retention decisions. For instance, it may notify the PEPs to encrypt the corresponding data object as specified by the policy. The components of the data layer also interface with PIPs in UCON+ to set attribute values related to data objects. This includes metadata such as data creation date, allowing the corresponding PIP to calculate how long data has been store and whether the retention period has expired. Similarly, the consent manager in the user application allows the user to revoke or modify their consent and updates the corresponding PIP accordingly. This enables UCON+ to monitor relevant attribute values and update governance decisions accordingly. Therefore, if the user revokes their consent or if the data retention period expires, UCON+ denies data retention and the lifecycle transitions to the `destruction` phase to delete the corresponding data object.

Data Processing

Software applications that process data also involve PEPs that interface with UCON+ and enforce policies. The applications also interface with PIPs to set attribute values related to the application and to data processing. For instance, the application sends the intended purpose of data processing to the corresponding PIP in UCON+. Likewise, it passes the ID of the subject using the application to the corresponding PIP, which enables

the PIP to retrieve the role of the subject from relevant sources (e.g., Active Directory). This enables UCON+ to make data processing decisions based on the attributes provided through PIPs. For example, if the processing purpose set by the application is included in the purposes consented by the data subject, UCON+ allows data processing. The PEP included in the application enforces such decisions and allows or denies data processing accordingly.

Data Destruction

When the lifecycle reaches the `destruction` phase, UCON+ evaluates data destruction policies to determine the actions that must be taken. The examples of destruction rules provided above specify that data must be archived if it falls in the public benefit category or deleted otherwise. The data category attribute is provided by the data layer to UCON+ through PIPs. Alternatively, UCON+ may use a context transformation policy to set the category attribute based on other metadata such as the data type. To delete or archive data, UCON+ notifies relevant PEPs on the data layer to enforce these actions as specified by the policy.

5.4 Smart Vehicles

Automotive systems are transforming into digital systems and increasingly adopting Service-Oriented Architectures (SOAs) due to their flexibility [15, 30]. This introduces new security challenges as typical SOA security measures are not completely sufficient due to the safety-critical nature of vehicles [88]. Dynamic identity management and usage control are among the identified challenges, because vehicles are real-time mobile systems whose environmental conditions change continuously as they move. In addition, automotive digital systems may run software applications provided by third parties, and the behaviour of such applications is dynamic and cannot be known in advance, so they cannot be trusted unconditionally [88, 12, 16]. Therefore, access to vehicular resources must be continuously monitored and controlled to ensure correct, safe and secure usage as circumstances change. More importantly, credentials and capabilities of subjects (i.e. drivers, passengers, applications) need to be continuously managed, monitored and updated according to contextual changes.

We address these challenges by utilising the UCON+ policy engine. We specifically introduce SIUV: a Smart-car Identity management and Usage control system based on Verifiable credentials. SIUV is a vehicular Identity and Access Management (IAM) system

prototype that provides dynamic and stateful credential management as well as continuous monitoring and usage control. It incorporates a policy-based Security Token Service (STS) that manages, authenticates and verifies credentials, and issues capabilities in exchange using Verifiable Credentials (VCs). The STS exchanges external VCs that hold identity claims about subjects with internal VCs that specify subject capabilities. The STS uses policy-based decision making to specify how to exchange credentials taking environmental conditions into account. It also uses continuous monitoring and policy re-evaluation to manage the lifecycle of capabilities and adapt to changing situations. In addition to the STS, we also use the policy engine as a Usage Control System (UCS) to protect resources within the vehicle and enforce usage control. Following the trend of adopting SOAs in vehicles, we leverage an SOA-based architecture that integrates the STS in addition to several UCS instances to protect different resources within the vehicle.

5.4.1 SIUV Security Token Service (STS)

In a traditional STS, a particular set of capabilities may be bound to specific attributes. Thus, all subjects that have these attributes will always have the same capabilities regardless of the context. In addition, issued capabilities do not change even if attributes change. This usually results in subjects being either over-privileged or underprivileged. In contrast, SIUV STS introduced in this work issues contextualised capabilities based on the current circumstances.

The STS specifically collects and verifies external identity VCs issued by Identity Provider (IdP) and exchanges them with internal VCs that determine the capabilities of the corresponding subjects within the vehicle. It employs policy-based decision making to specify the capabilities to be issued based on the identity VCs of the subject as well as environmental attributes. The STS also uses session-based monitoring and manage the lifecycle of credentials starting from their issuance and lasting until their revocation or expiry. The STS reacts to changes in identity VCs or environmental conditions by re-evaluating its policies and revoking the issued capabilities if necessary. STS policies can also ensure graceful revocation of capabilities by specifying safety measures that must be taken upon or before revoking a credential.

SIUV STS leverages the UCON+ policy-engine with a specialised PEP that verifies and issues credentials as shown in Figure 5.8. The PEP interfaces with subjects or digital wallets, verifies identity VCs using the Verifier subcomponent, then sends a request to the policy engine. The policy engine collects the required attributes through PIPs, and evaluates relevant policies to determine what capabilities to issue and how to manage

them. The evaluation decision is then returned the PEP to issue internal capabilities using the Issuer subcomponent.

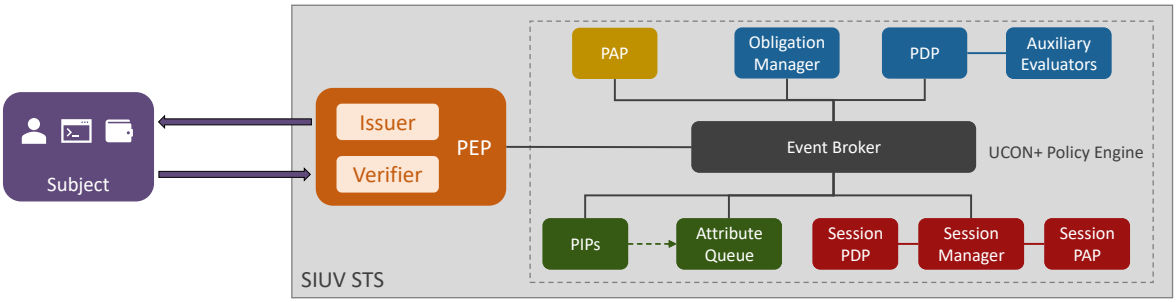

Figure 5.8: SIUV STS

Credential Lifecycle

The STS manages the lifecycle of a credential in a UCON+ evaluation session. It defines a DFA that reflects the lifecycle of a credential and specifies lifecycle transition policies to drive the lifecycle and fire events. The credential lifecycle is illustrated in the DFA in Figure 5.9. The credential lifecycle starts with the `issuance` phase when the credential is requested. Access control policies classified under the `issuance` specify whether the credential issuance is allowed or not. They also specify obligations that determine the content (i.e., claims) of the credential. If the credential issuance is denied or the obligations are violated, the lifecycle moves to the `exit` phase and the session ends. Otherwise, the STS issues the credential and the lifecycle transitions to the `usage` phase where policies specify whether the credential is still valid. The lifecycle remains in the `usage` phase as long as the credential is still valid, and moves to the `revocation` phase otherwise. `revocation` policies may specify safety measures that must be taken before revoking the credential, so the lifecycle stays in this phase until all measures are enforced. Finally, the lifecycle exits when safety measures are taken and the credential is revoked. The lifecycle may seem similar to UCON's tradition `pre-ongoing-post` lifecycle, but note that the last phase (i.e., the `revocation` phase) is continuous. This means that continuous monitoring is applied in this phase too, and the policy may be re-evaluated several times. In contrast, the `post` phase in UCON does not include continuous monitoring, and `post` policies are evaluated and enforced only once.

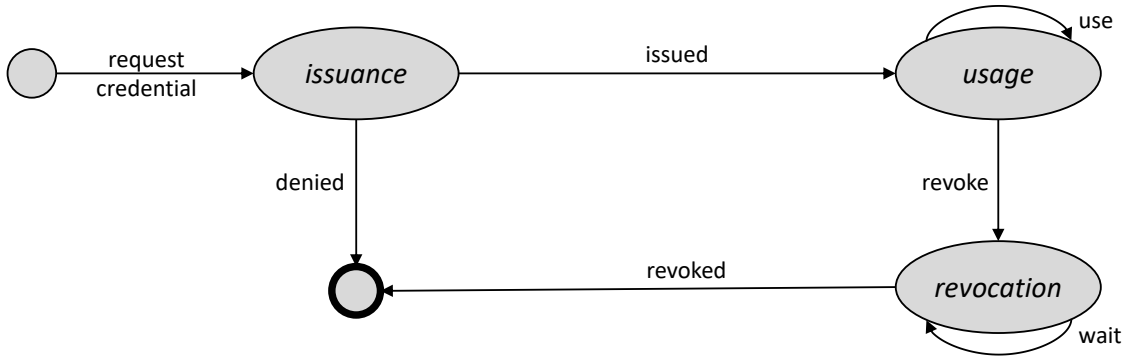

Figure 5.9: DFA illustrating the lifecycle of an STS evaluation session

STS Lifecycle Transition Policy

The STS uses a lifecycle transition policy to enact and implicitly define the credential lifecycle describe above. The lifecycle transition policy is presented in Listing 5.10. The first rule (lines 4 to 8) sets the phase to `issuance` when the session starts. The `credentialAllowed` rule sets the phase to `usage` if the decision is `permit` and all obligations are enforced (i.e., the credential has been issued). Rule `credentialDenied` at lines 16 to 21 sets the phase to `exit` and terminates the session if credential issuance is denied. Rule `usageRevoked` specifies that if the usage of the issued credential is denied, then the lifecycle must move from the `usage` phase to the `revocation` phase. Finally, the `credentialRevoked` rule sets the phase to `exit` and ends the session if the decision is `permit` and all obligations are fulfilled. A `permit` decision in the `revocation` phase denotes that the revocation of the credential is allowed, and obligation fulfilment indicate that the credential has been revoked. If the lifecycle is in the `usage` phase and the decision is `permit`, the `default` rule (last rule) will apply and the phase will remain unchanged. This means that the lifecycle stays in the `usage` phase as long as the credential usage is allowed. Similarly, if the lifecycle is in the `revocation` phase and the decision is `deny` (i.e., credential revocation is denied), the `default` rule applies and the phase remains unchanged.

5.4.2 SIUV Architecture

In this section, we describe the SIUV architecture and how its components together enable dynamic identity management and usage control. The architecture is illustrated in Figure 5.10.

Listing 5.10: SIUV lifecycle transition policy for managing credentials lifecycle

```
policy siuv-lifecycle {
    target clause Attributes.session.model == "siuv"
    apply firstApplicable
    rule init {
        target clause Attributes.session.phase == "init"
        permit on permit {
            obligation requestUsage { Attributes.session.phase = "issuance" }
    }}
    rule credentialAllowed {
        target clause Attributes.session.phase == "issuance"
        condition Attributes.session.decision == "permit"
              and Attributes.session.obligations.status == "fulfilled"
        permit on permit {
            obligation startUsage { Attributes.session.phase = "usage" }
    }}
    rule credentialDenied {
        target clause Attributes.session.phase == "issuance"
        condition Attributes.session.decision == "deny"
               or Attributes.session.obligations.status == "violated"
        permit on permit {
            obligation exit { Attributes.session.phase = "exit" }
    }}
    rule usageRevoked {
        target clause Attributes.session.phase == "usage"
        condition Attributes.session.decision == "deny"
        permit on permit {
            obligation endUsage { Attributes.session.phase = "revocation" }
    }}
    rule credentialRevoked {
        target clause Attributes.session.phase == "revocation"
        condition Attributes.session.decision == "permit"
              and Attributes.session.obligations.status == "fulfilled"
        permit on permit {
            obligation endUsage { Attributes.session.phase = "exit" }
    }}
    rule default { permit }
};
```

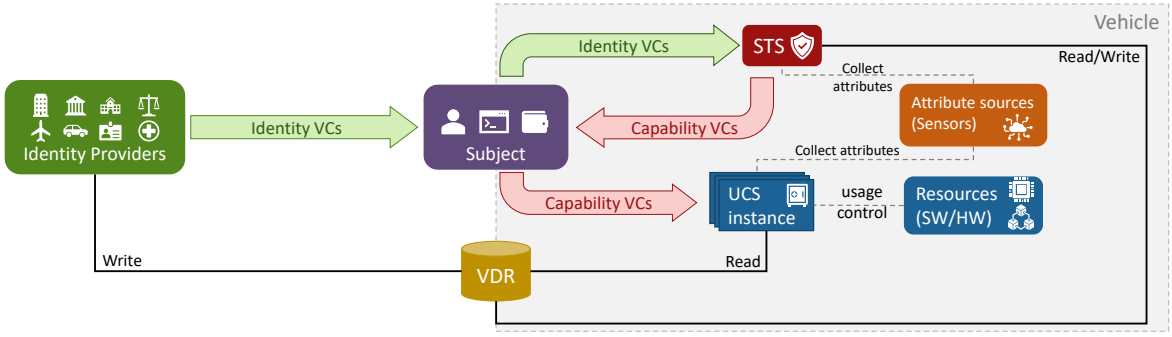

Figure 5.10: SIUV architecture

Components

IdPs are authorities that issue identity VCs about subjects that access resources within the vehicle drivers, passengers or software applications. An example of an issuer is the department of motor vehicles that issues driving licenses, or a service provider that asserts claims about a software application.

A subject may be a driver, a passenger or a software application that needs to access resources, and is represented by a digital wallet that holds the subject's VCs. The digital wallet interacts with the issuers and the STS to present/obtain VCs, and with UCS instances to request access to resources.

The Verifiable Data Registry (VDR) is a Distributed Ledger Technology (DLT) that holds revocation lists and issuers' public keys. The VDR may also be used to store other relevant information like metadata about proofs or schemas and structures of VCs.

The STS is the core component that provides dynamic identity and capability management of subjects. It exchanges external identity VCs with internal capability VCs. The combination of VCs and the STS protects against identity theft as capabilities issued by the STS are cryptographically verifiable and cannot be manipulated. This also allows *unlinkability* and maximizes privacy of subjects because they do not need to share identity information with the provider of the vehicle's applications and components.

UCS instances are used to protect resources within the vehicle. However, they do *not* control the low-level behaviour of the vehicle as this imposes a safety risk. Rather, they only control the usage of high-level APIs, services and functions exposed by the vehicle. UCS instances make usage decisions according to resource and environmental attributes as well as capabilities issued by the STS. The PEPs of UCS instances verify the capability VCs issued by the STS, and parse their claims, which are used as attributes in the policies. Therefore, the STS handles the global context of the whole vehicle, while UCS

instances manage localised authorisation contexts of individual resources. UCS instances use traditional Attribute-Based Access Control (ABAC) or UCON models depending on the type of protected resources and the corresponding requirements, so they use the lifecycle transition policies that enact ABAC and UCON lifecycles.

Revocation of VCs

When an IdP revokes an external VC, such as a driving license, the IdP updates the revocation list in the VDR. The update then gets communicated to all participating entities through the VDR. If the STS is using the updated VC in an active session, then a policy re-evaluation will be triggered in that session. Based on the policy re-evaluation, the STS may revoke all internal capabilities that depend on the revoked external VC. When the STS revokes internal VCs (i.e., capabilities), it also updates the VDR, thus the update gets communicated to all UCSs instances. This triggers a re-evaluation in the UCS sessions that are using the revoked internal VC, and the corresponding usage decisions are updated accordingly. As explained above, STS policies specify safety measures that must be taken before revoking capabilities, so internal VCs are not revoked unless such measures are enforced. Nonetheless, UCS policies must also specify safety procedures so that revocation of usage does not cause safety threats. For instance, a policy may specify obligations that safely stop the vehicle before completely revoking access or delay revocation of access until the vehicle stops.

5.4.3 Use-cases

To demonstrate SIUV, we describe three use-cases and present the corresponding policies.

Telemetry Data Collection Application

One relevant use-case of SIUV is the use of software applications that collect telemetry data. Suppose that the car owner installs an application that provides services based on telemetry data. The application requests a capability VC from the STS to allow it to collect telemetry data from the vehicle's sensors and resources. The STS evaluates relevant policies and decides whether to issue or deny the capability accordingly. The STS also monitors the capability and revokes it if the policies are not valid anymore. UCS instances on the other hand use the capability credential to decide whether to allow data collection from the resources they protect. They may also specify that data must be anonymised before collection.

A non-comprehensive example of the STS policy is provided in Listing 5.11. The first rule allows the issuance of the telemetry data collection capability, but specifies an obligation to obtain the consent of the car owner. If the owner consent is not obtained (i.e., obligation violated), the evaluation session terminates and the capability would not be issued as specified by the lifecycle described above. The second rule specifies that if the consent is revoked, then the usage of the capability must be denied. Finally, the last rule requests data deletion of already collected telemetry data and revokes the capability. Similarly, a non-comprehensive example of a UCS policy is provided in Listing 5.12. The policy allows the collection of location data if the subject has the telemetry data collection capability. It also specifies that the data must be anonymised before collection. Note that this is an ABAC policy, so it only consists of a `pre` rule.

Listing 5.11: Non-comprehensive example of a SIUV STS policy for the telemetry data collection use-case

```
policy telemetry {
    target clause Attributes.request.capability == "telemetry-collection"
    apply firstApplicable
    rule issueVC {
        target clause Attributes.session.phase == "issuance"
        permit on permit {
            obligation obtainConsent { type = "telemetry-collection" }
            obligation issueVC {
                capability = "telemetry-collection"
                id = Attributes.request.id
        }}
    }
    rule useVC {
        target clause Attributes.session.phase == "usage"
        condition !Attributes.owner.consent.telemetry-collection
        deny
    }
    rule revokeVC {
        target clause Attributes.session.phase == "revocation"
        permit on permit {
            obligation deleteData {}
            obligation revokeVC { id = Attributes.request.id }
    }}
}
```

Listing 5.12: Non-comprehensive example of a SIUV UCS policy for the telemetry data collection use-case

```
1  policy ucs {
2      target clause Attributes.request.action == "collect-data"
3                and Attributes.data.type == "location"
4      apply firstApplicable
5      rule pre {
6          target clause Attributes.session.phase == "pre"
7          condition Attributes.subject.capability == "telemetry-collection"
8          permit on permit { obligation anonymise {}}
9      }
10     rule default { deny }
11 }
```

Border Crossing

Another relevant use-case of SIUV is travelling between the borders of two countries where driving rules are different. For instance, the minimum age to drive in Denmark is seventeen, whereas it is eighteen in Sweden. Accordingly, consider a seventeen-year-old driver with a valid driving license in Denmark. The STS collects the driving license VC from the driver as well as any other required VCs (e.g., ID), verifies them then issues a capability VC allowing the subject to drive car. The capability VC is then used by UCS instances to grant access to the necessary components that allow the driver to drive the car (e.g., start the engine). If the car crosses the borders to Sweden, the location attribute changes, and the STS re-evaluates its policies consequently. In this case, the STS finds that the driving capability VC is not valid anymore, and it revokes it when the car safely stops. Policies may also specify obligations to notify the driver of the revocation.

This example is demonstrated in the following non-comprehensive policies. Listing 5.13 presents the STS policy. The first rule (lines 4 to 12) specifies that issuing a driving capability VC is allowed if the license is *not* expired and the driver's age is equal or greater than the minimum driving age. The `minimumDrivingAge` attribute is managed by the context transformation policy described below. The obligation of the first rule specifies the claims of the VC. The second rule (`useVC`) specifies that the use of the driving capability must be denied if the driver's age is less than the minimum driving age. Thus, when the car crosses the border, the `minimumDrivingAge` attribute changes and the driving capability will either be maintained or revoked accordingly. Rule `wait` denies the revocation of the driving capability if the car is not safely parked. It also specifies

an obligation to notify the driver that they do not meet the regulations. Finally, the last rule revokes the driving capability when the car is safely parked. The obligation specifies the ID of the credential to be revoked.

Listing 5.13: Non-comprehensive example of a SIUV STS policy for the border crossing use-case

```
policy driver {
    target clause Attributes.request.capability == "drive"
    apply firstApplicable
    rule issueVC {
        target clause Attributes.session.phase == "issuance"
        condition !Attributes.subject.license.expired
              and Attributes.subject.age >= Attributes.minimumDrivingAge
        permit on permit { obligation issueVC {
            capability = "drive"
            id = Attributes.request.id
        }}
    }
    rule useVC {
        target clause Attributes.session.phase == "usage"
        condition Attributes.subject.age < Attributes.minimumDrivingAge
        deny
    }
    rule wait {
        target clause Attributes.session.phase == "revocation"
        condition !Attributes.isCarParked
        deny on deny { obligation notifyDriver {
            message = "You do not meet the minimum age for driving here. Either
    park the car or return back."
        }}
    }
    rule safeRevoke {
        target clause Attributes.session.phase == "revocation"
        condition Attributes.isCarParked
        permit on permit { obligation revokeVC { id = Attributes.request.id }}
    }
}
```

Listing 5.14 shows an example of a context transformation policy used in SIUV. The first rule sets the `minimumDrivingAge` attribute to seventeen if the location is in Denmark. In contrast, the second rule sets the attribute to eighteen if the location is in Sweden.

Listing 5.14: Non-comprehensive example of a SIUV context transformation policy for the border crossing use-case

```
policy siuv {
    target clause Attributes.session.model == "siuv
    apply permitUnlessDeny
    rule denmark {
        target clause Attributes.location == "Denmark"
        permit on permit {
            obligation updateAttribute { Attributes.minimumDrivingAge = 17 }
    }}
    rule sweden {
        target clause Attributes.location == "Sweden"
        permit on permit {
            obligation updateAttribute { Attributes.minimumDrivingAge = 18 }
    }}
}
```

Car Rental Service

A car rental service is also another relevant use-case of SIUV as car owners can restrict the use of their cars according to rental agreements. For instance, a car owner can restrict the mobility of their rented car to a specific city or area. Thus, if the driver goes beyond the restricted area, SIUV can warn the driver and revoke their driving capability credential, and notify the car owner of the breach. Moreover, the car owner can define a specific time period during which the car can be used by the driver according to the rental agreement. Owners can also define policies that deny access to resources that are not needed by drivers who rent the car (e.g. diagnostic interfaces).

Listing 5.15 provides a non-comprehensive example of the STS policy of car rental service. The first rule issues a driving capability VC if the driver is the renter in the agreement. The `agreementBreach` rule is enforced when the driver goes beyond the agreed upon location. The rule still allows the use of the driving capability, but it notifies the driver and the owner that the rental agreement has been breached. The third rule (lines 19 to 23) denies the use of the driving capability when the agreement expires. Rule `wait` denies the revocation of the driving capability if the car is not safely parked and notifies the driver that the agreement has ended. The last rule revokes the driving capability when the car is safely parked.

Listing 5.15: Non-comprehensive example of a SIUV STS policy for the car rental service use-case

```
policy rental {
    target clause Attributes.request.capability == "drive"
    apply firstApplicable
    rule issueVC {
        target clause Attributes.session.phase == "issuance"
        condition Attributes.subject.id == Attributes.agreement.renter
        permit on permit { obligation issueVC {
            capability = "drive"
            id = Attributes.request.id
        }}
    }
    rule agreementBreach {
        target clause Attributes.session.phase == "usage"
        condition Attributes.location != Attributes.agreement.location
        permit on permit {
            obligation notifyDriver { message = "You have breached the rental agreement. Additional fees will be incurred" }
            obligation notifyOwner { agreementId = Attributes.agreement.id" }
    }}
    rule useVC {
        target clause Attributes.session.phase == "usage"
        condition Attributes.agreement.endDate < Attributes.currentDate
        deny
    }
    rule wait {
        target clause Attributes.session.phase == "revocation"
        condition !Attributes.isCarParked
        deny on deny {
        obligation notifyDriver { message = "You rental agreement has ended." }
    }}
    rule safeRevoke {
        target clause Attributes.session.phase == "revocation"
        condition Attributes.isCarParked
        permit on permit { obligation revokeVC { id = Attributes.request.id }}
    }
}
```

5.5 Credential Bridge

SSI and VCs emerged as new paradigms that empower individuals to own and control their credentials, and enable sharing credentials across different organisations and jurisdictions. SSI and VCs are not limited to digital identities (e.g., user accounts), but rather they are set to facilitate the digitalisation of other types of credentials such as university degrees, driving licenses and professional certifications. However, SSI needs to coexist with legacy identity and credential systems as organisations have heavily invested in them. Moreover, different jurisdictions have different requirements and regulations, and SSI alone cannot enable their interoperability.

To address these challenges, we extend the policy-based STS described in Section 5.4 into a general-purpose credential bridge that transforms credentials across various domains. Like the SIUV STS, the bridge leverages the UCON+ policy engine and uses policy-based decision making to transform credentials. However, the bridge is not limited to transforming identity credentials to capabilities, but rather enables several types of transformations. To this end, we outline in this section the different types of credential transformations enabled by the bridge. We also discuss the evaluation lifecycle of the transformation process and present the corresponding lifecycle transition policy. Finally, we describe the architecture as well as the considered use-cases with the corresponding transformation policies.

5.5.1 Types of Credential Transformation

As the credential bridge transforms credentials from one domain to another, we outline the types of transformations below. However, it is important to note that the credential transformation is not restricted to solely one of these types but can instead include a combination of them.

Credential Format Conversion

Format conversion is the simplest type of credential transformation. It entails reissuing the same claims of the original credential but in a different format. This transformation ensures that the content and meaning of the claims remain unchanged, but the structure and representation of the credential are adjusted to align with the target domain or framework. For example, if a credential is originally in the OpenID Connect (OIDC) format and needs to be used in a different domain that requires a VC format, the bridge verifies the OIDC credential and parses its claims, then issues a VC containing the same

claims. The bridge relies on transformation policies that specify the format based on the target system as well as other contextual information. Format conversion can be achieved using model transformation languages (e.g., ATLAS). However, the credential bridge adds a dynamic aspect as the transformation is based on conditions and policies. Besides, the rest of the transformation types cannot be achieved by model transformation languages as they require policy-based decision making.

Credential Claims Mapping

Another type of credential transformation is credential mapping, which involves exchanging the original claims with other values that are relevant to the target domain. For example, it may replace the courses and modules in a university transcript VC with their equivalent courses and modules in another university. Similarly, it may replace a birth date claim with an age claim based on the birth date in order to enhance privacy. Transformation policies specify how to map the claims and determine their values, ensuring accurate and meaningful transformations between different frameworks. This type of transformation does not necessarily require changing the format of the credential.

Credential Aggregation

The third type of credential transformation involves the aggregation of multiple credentials or claims into a single credential. In certain scenarios, the target domain may require a credential with a specific set of claims that are not originally provided by a single credential or identity provider. To address this requirement, the credential bridge collects and verifies all relevant credentials, aggregating them or their claims into a consolidated credential that meets the specified requirements. This process is similar to the concept of Verifiable Presentations (VPs), but the bridge goes beyond simple presentation and utilises transformation policies to determine the necessary claims, which credentials should be collected, and how to effectively aggregate them.

Credential Complementation

This is the most relevant type that demonstrates the value of the credential bridge. It involves fetching and verifying additional credentials to complement and supplement the credential to be transformed. For instance, to equate a university degree VC in a different country, the credential bridge may request additional credentials to complement the degree VC such as transcripts or accreditation credentials of the source university. This allows the bridge to enforce extra verifications and collect missing information ensuring a trustworthy

transformation. The policy-based decision making enables the bridge to determine the additional credentials to be fetched according to the requirements of each transformation case. For instance, policies can specify different information to be collected based on the university or country of origin of the degree VC.

5.5.2 Transformation Lifecycle

The credential bridge is not concerned with managing the lifecycle of credentials, but rather with transforming them from one domain or framework to another. Accordingly, the evaluation lifecycle outlines the stages of transforming a credential. The lifecycle specifically consists of two phases designated as `collection` and `issuance` as shown in the DFA in Figure 5.11. The lifecycle starts with the `collection` phase where the bridge verifies the input credentials to be verified and policies specify whether a transformation is possible and if additional information or credentials are required. The lifecycle remains in the `collection` phase while required credentials are being collected if applicable (i.e., `in progress` label is fired). If the transformation is denied, the `exit` label is fired and the lifecycle terminates. Otherwise, if the transformation is allowed and all required information and credentials are collected, then the lifecycle transitions to the `issuance` phase (i.e., `verified` label is fired). Transformation policies of the `issuance` phase specify the format of the issued credential as well as the claims to be included. Finally, the `exit` label is fired and the lifecycle terminates after enforcing the policies and issuing the credential.

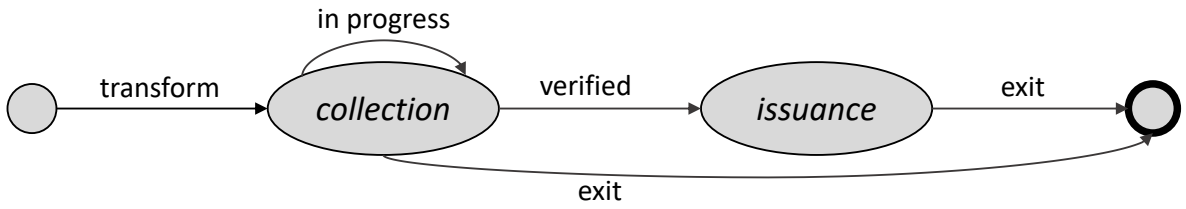

Figure 5.11: DFA illustrating the evaluation lifecycle of credential transformation

The corresponding lifecycle transition policy is presented in Listing 5.16. The `init` rule sets the phase to `collection` upon the initialisation of the evaluation session. The second rule (i.e., `collectionInProgress`) keeps the lifecycle in the `collection` phase when the decision is `deny` and there are obligations to be enforced (i.e., there are additional actions to be taken such as collecting additional credentials before transforming the credential). This is achieved by using the `firstApplicable` combining algorithm and the

absence of an obligation that changes the phase in this rule. Thus, when this rule applies, the `firstApplicable` combining algorithm discards the rest of the rules, so the phase remains unchanged. The `collectionDenied` rule terminates the session if the credential transformation is denied and there are no more obligations to be enforced. This is captured by the `firstApplicable` combining algorithm, because the previous rule includes a check for the presence of obligations, so if the previous rule is not applicable, then there are no obligations to be enforced. Rule `collectionCompleted` fires a transition from `collection` to `issuance` once credential transformation is permitted. Finally, the last rule terminates the session upon evaluating and enforcing credential issuance policies.

5.5.3 Architecture

The architecture of the credential bridge is similar to that of the SIUV STS with a minor modification as illustrated in Figure 5.12. It leverages the UCON+ policy-engine with a specialised PEP for verifying and issuing credentials. Thus, the PEP interfaces with the credential subject or their digital wallet, collects credentials from them and provides a transformed credential. The PEP also interfaces with PIPs and acts an attribute source providing claims parsed from credentials in the form of attributes. Moreover, the PEP interface with external IdPs or authorities to collect credentials from sources other than the subject and verify them using the verifier subcomponent. For example, upon equating a university degree credential, PIPs can collect credentials about the university that issued the degree from the university itself or other relevant sources.

Accordingly, the flow of a transformation process in this architecture is as follows: (1) the subject submits a credential to the PEP and requests a transformation; (2) the PEP verifies the submitted credential, parses it and invokes the policy engine by passing the credential claims as attributes; (3) the policy engine evaluates relevant policies (i.e., policies of the `collection` phase) and passes the corresponding obligations to the PEP to collect additional credentials if required; (4) the PEP prompts the subject for additional credentials or requests them from relevant authorities, then verifies and parses them, and passes the corresponding attributes to PIPs; (5) the policy engine then transitions to the `issuance` phase and evaluates policies that specify how to issue a transformed credential; (6) finally, the engine invokes the PEP to issue the transformed credential using the issuer subcomponent as specified by policy obligations; the PEP then sends the transformed credential to the subject.

Listing 5.16: Non-comprehensive example of the lifecycle transition policy of credentional transformation

```
policy bridge-lifecycle {
    target clause Attributes.session.model == "bridge"
    apply firstApplicable
    rule init {
        target clause Attributes.session.phase == "init"
        permit on permit {
            obligation start { Attributes.session.phase = "collection" }
    }}
    rule collectionInProgress {
        target clause Attributes.session.phase == "collection"
        condition Attributes.session.decision == "deny"
              and Attributes.session.obligations
        permit
    }
    rule collectionDenied {
        target clause Attributes.session.phase == "collection"
        condition Attributes.session.decision == "deny"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}
    }
    rule collectionCompleted {
        target clause Attributes.session.phase == "collection"
        condition Attributes.session.decision == "permit"
        permit on permit {
            obligation startIssuance { Attributes.session.phase = "issuance" }
    }}
    rule issued {
        target clause Attributes.session.phase == "issuance"
        permit on permit { obligation exit { Attributes.session.phase = "exit" }}
    }
};
```

5.5.4 Use-cases

We demonstrate the credential bridge in two use-cases and describe the corresponding policies.

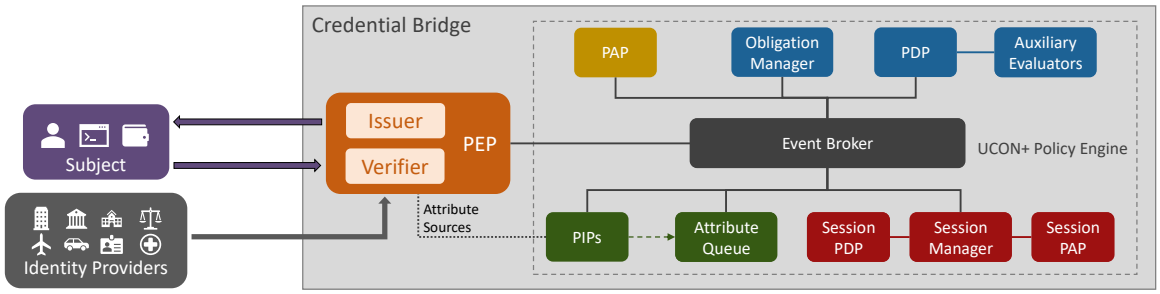

Figure 5.12: Architecture of the Credential Bridge

Equating University Degree

The first use-case considers equating a university degree VC in another university and country. The credential bridge is used by the target university to automate the recognition and equivalence process. Transformation policies are used to specify the conditions and requirements for equating the degree, as well as the claims to be included in the degree equivalence credentials. We consider equating a degree issued by a university that uses the Grade Point Average (GPA) grading system in a university that uses percentage grading. To this end, we provide two non-comprehensive examples of transformation policies, one for each phase of the transformation lifecycle described in Section 5.5.2.

The transformation policy of the `collection` phase is provided in Listing 5.17. The target clause of the policy (i.e., line 2) restricts the applicability of this policy to the `collection` phase only. The first rule in the policy specifies an obligation to collect the transcript VC from the subject if it was not provided already. When this rule applies, the PEP enforces this obligation by requesting the transcript from the subject wallet, which in turn prompts the subject to disclose the transcript. Similarly, the second rule specifies an obligation to collect the accreditation credential of the university that issued the degree. The PEP enforces this obligation by requesting the accreditation credential from the relevant authority such as the ministry of education in the country of origin. We assume that authorities that accredit universities are digitalised and provide APIs to obtain accreditation credentials in the form of VCs. The last rule of the policy specifies the requirements and conditions for equating the degree. It specifically checks if the accreditation of the source university is valid and if the courses listed in the transcript include the "Research Methodology" course. Moreover, the rule calculates the percentage grade of the student and checks if its equal to or greater than sixty (i.e., the passing grade). If these conditions are met, the policy allows the transformation process and

the session transitions to the `issuance` phase of the lifecycle. The `denyUnlessPermit` combining algorithm is used to enable the enforcement of all obligations that correspond to the `deny` decision at the same time. This combining algorithm combines and enforces all applicable `deny` rules together when there are no applicable `permit` rules. Otherwise, it enforces the first `permit` rule that applies.

Listing 5.17: Non-comprehensive example of a transformation policy used to for equating a university degree in the `collection` phase

```
policy collection {
    target clause Attributes.session.phase == "collection"
    apply denyUnlessPermit
    rule transcriptMissing {
        target clause !Attributes.subject.transcript
        deny on deny { obligation obtainTranscript {} }
    }
    rule accreditationMissing {
        target clause Attributes.subject.university.accreditation == "missing"
        deny on deny { obligation obtainAccreditation {} }
    }
    rule allowTransfromation {
        target clause Attributes.subject.university.accreditation == "valid"
        condition Attributes.subject.degree.grading == "GPA"
              and Attributes.subject.degree.grade * 25 >= 60
              and "Research Methodology" in Attributes.subject.transcript.courses
        permit
    }
}
```

Once all requirements are verified and complementary credentials are collected, the session transitions to the `issuance` phase where policies specify the claims of the credential to be issued. Listing 5.18 shows a non-comprehensive example of a policy that sets two claims in the degree equivalence credential to be issued. The first rule converts the grade to percentage form if the grading system of the source university is GPA. The rule sets the grade claim accordingly. The second rule checks if the percentage grade is above eighty, and sets the classification of the degree as `distinction` as per the rules and regulations of the target university. The `permitUnlessDeny` combining algorithm is used to combine all applicable `permit` rules and enforce their obligations at the same time. After setting all claims as specified by the policy, the PEP issues and signs the degree equivalence credential, and passes it to the subject wallet.

Listing 5.18: Non-comprehensive example of a transformation policy used to for equating a university degree in the `issuance` phase

```
policy issuance {
    target clause Attributes.session.phase == "issuance"
    apply permitUnlessDeny
    rule grade {
        target clause Attributes.subject.degree.grading == "GPA"
        permit on permit {
            obligation setClaim { grade = Attributes.subject.degree.grade * 25 }
        }
    }
    rule classification {
        target clause Attributes.subject.degree.grading == "GPA"
        condition Attributes.subject.degree.grade * 25 >= 80
        permit on permit { obligation setClaim { classification = "Distinction" }}
    }
}
```

IDS

IDS uses SSI credentials to identify and authenticate parties exchanging data in the data space. Accordingly, data provider and consumer must be able to present SSI VCs to each other upon starting a contract negotiation process. The VCs must include details about the subjects sharing and consuming data within each side. We leverage the credential bridge in Boot-X[3] to transform credentials from the internal identity systems of the data provider and consumer to SSI VCs.

The logical architecture illustrated in Figure 5.4 shows how the credential bridge is used in this context. When a user logs in, the identity system sends a request to the credential bridge to issue an SSI VC for the user and store in the identity hub of the data space. The identity hub is a component specified by IDS that holds SSI VCs of the corresponding participant of the data space. It is used by the connector during the contract negotiation process and passed to the other party in the data space for identification and authentication.

The credential transformation in IDS is simpler than the university degree use-case. It only involves exchanging an internal credential issued by the legacy identity system with an SSI VC. Thus, transformation policies specify the format of the credential as well as the claims to be included in the SSI VC. To this end, we show in Listing 5.19 a simple

[3] `https://boot-x.eu/open-source-architecture/`

non-comprehensive example of an ALFA policy used to issue an SSI VC in Boot-X. The `setFormat` rule sets the format of the credential to be issued to SSI VC. Similarly, the `setClaim` obligation specifies which claims to copy from the original credential to the SSI VC. The last obligation instructs the PEP to push the VC to the identity hub after issuing it.

Listing 5.19: Non-comprehensive example of an ALFA rule that issues an SSI VC in a data space

```
1    rule issueSsiVc {
2        target clause Attributes.session.phase == "issuance"
3        permit on permit {
4            obligation setFormat { format = "SSI VC" }
5            obligation setClaim {
6                role = Attributes.subject.credential.role
7                organisation = Attributes.credential.organisation
8            }
9            obligation pushToIdHub {}
10       }
11   }
```

Chapter 6

Conclusion

This chapter concludes the thesis by summarising the challenges and contributions, and outlining future directions.

6.1 Summary

As digital technologies continue to pervade various aspects of human life, addressing the ensuing privacy and security concerns becomes imperative. Access and usage control as well as policy-based governance are among the security mechanisms that are typically used to protect digital resources. In this thesis, we focused on several industrial applications of Usage Control (UCON) and policy-based governance. To address the diverse requirements of the targeted application domains, we also tackled a core research problem in existing UCON frameworks and proposed a new model accordingly. In this section we summarise the challenges, objectives and contributions of this thesis.

6.1.1 Challenges and Objectives

UCON is an authorisation-oriented mechanism concerned with managing the access rights that subjects can exercise on resources. In contrast, emerging security paradigms require a broader policy-based governance that transcends traditional access and usage control, and specifies how an entity (e.g., subject, object, process, etc.) should be generally handled in a specific context. For instance, data ecosystems (e.g., data hubs, data spaces) entail policy-based governance throughout the different stages of the data lifecycle from data collection to destruction. Other applications require different lifecycles based on their specific requirements. In contrast, UCON enacts a fixed lifecycle of three temporal phases

(i.e., `pre`, `ongoing` and `post`). Therefore, policy systems must enable the customisation of the lifecycle through which policy-based governance is applied. In addition, UCON allows attribute mutability, but does not incorporate a mechanism to govern attribute values and attribute updates. For this reason, policy systems must enable governing attributes using policies.

Besides solving the above-mentioned challenges, our main objective in this thesis was to leverage policy-based governance in the following four application domains: (1) in Industrial/International Data Spaces (IDS) to drive contract negotiation and govern data usage according to data sharing agreements; (2) in data hubs to govern data usage throughout the data lifecycle; (3) in smart vehicles to dynamically issue contextualised capabilities within the vehicle and manage their lifecycles; and (4) in a credential bridge that transforms credential between different domains or regulatory frameworks.

6.1.2 Contributions

We presented six contributions in this thesis that address the challenges and objectives mentioned above. (1) We proposed an extensible model for policy-based governance that concerns evaluation sessions in policy systems. The model defines general structures and common functions, and outlines extensible behaviour that can be refined and implemented by concrete extensions. Specifically, the model specifies the properties of the evaluation lifecycle, allowing extensions to define different and custom lifecycles. The model also specifies a rule-based mechanism to govern attribute values and drive the lifecycle according to conditional rules. Therefore, the model can be extended into various concrete extensions by specifying different rules resulting in a distinct behaviour tailored for specific requirements. (2) We introduced a policy engine that implements the model using a standard Attribute-Based Access Control (ABAC) policy language. The policy engine enables the use of policies to specify the extensible behaviour of the model. It specifically uses policies to drive the evaluation lifecycle and govern attribute values. Therefore, the policy engine can realise different extensions of the model without reimplementation as extensible behaviour is defined in policies. The policy engine leverages a modular architecture that allows adding, upgrading or substituting component easily when necessary. (3) We leveraged the policy engine in an IDS that is provided as a cloud service[1]. We specifically used the policy engine to drive the contract negotiation in the IDS based on

[1]The UCON+ policy engine and the credential bridge have been integrated in Boot-X as shown in the open-source architecture. Boot-X is a cloud-based data space service. `https://boot-x.eu/open-source-architecture/`

rules that specify whether to accept or reject offers, or to propose counteroffers. The engine was also used to enforce the negotiated data sharing agreement and govern data usage accordingly. (4) We also demonstrated the use of the policy engine in a data hub application. We used the engine to govern data objects throughout the different stages of their lifecycles. We provided examples of policies used to govern data usage and to define the data lifecycle. (5) We presented a dynamic identity management and usage control framework for smart vehicles using the policy engine. The framework includes a policy-based Security Token Service (STS) that issues contextualised capabilities for subjects within the vehicle, and manages the capabilities throughout their lifecycles. The STS can also revoke capabilities when changes occur and the corresponding policies are no longer valid. The framework takes safety measures in consideration. (6) Finally, we introduced a policy-based credential bridge using the policy engine. The bridge transforms credentials between different domains or regulatory frameworks according to policies that specify the requirements of each domain. The bridge enables seamless interoperability between different frameworks, and facilitates the transition towards Self-Sovereign Identity (SSI). We described two use-cases of the bridge including its use in an IDS[1].

6.2 Future Directions

The contributions presented in this thesis pave the way for future research that improves the expressiveness and adaptability of the proposed model, and complements the solutions we introduced for the targeted industrial applications.

In the context of IDS, future directions must aim at addressing end-to-end data sharing instead of one-to-one exchange. Current solutions only consider data sharing agreements between two parties, a data provider and consumer. However, data may be shared among many parties in a chain where each consumer becomes a provider for another party. Therefore, contract negotiation and data sharing agreements must consider the whole end-to-end path that data may traverse. Furthermore, verifiable proofs and evidence collection solutions must be developed to prove that the different parties of a data space have complied with the negotiated contract and enforced their obligations. For example, if the data sharing agreement specifies that the consumer must delete the data after a specific period, then the consumer must be able to provide a verifiable proof of the data deletion to ensure the fulfilment of the agreement.

The data hub solution provided in this thesis applies policy-based governance per individual data objects. However, different data objects may be related and their usages influence each other. For instance, if the policy specifies that a data object must be

deleted, then derivative data may also need to be deleted. For this reason, future work must address this challenge and provide a solution to collectively govern related data objects that may influence each other.

A full formal specification of UCON+ must also be provided as part of future research. As mentioned before, there is a work-in-progress that is already pursuing this goal [94]. Moreover, future work must aim at facilitating policy authoring and engineering using static and dynamic analysis methods based on the formal specification.

Bibliography

[1] Open Policy Agent. Policy Language, 2023. URL: https://www.openpolicyagent.org/docs/latest/policy-language/.

[2] Open Policy Agent. Policy Performance, 2023. URL: https://www.openpolicyagent.org/docs/latest/policy-performance/.

[3] Ali Al-Lawati and Luis Barbosa. A Framework for Intelligent Policy Decision Making Based on a Government Data Hub. In *Digital Transformation and Global Society: 4th International Conference, DTGS 2019, St. Petersburg, Russia, June 19–21, 2019, Revised Selected Papers 4*, pages 92–106. Springer, 2019.

[4] Mahmoud Ammar, Hassaan Janjua, Ashok Thangarajan, Bruno Crispo, and Danny Hughes. Securing the On-board Diagnostics Port (OBD-II) in Vehicles. *8th Embedded Security in Cars (ESCAR USA)*, 2020.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the Mirai Botnet. In *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110, 2017.

[6] Arno Appenzeller, Ewald Rode, Erik Krempel, and Jürgen Beyerer. Enabling data sovereignty for patients through digital consent enforcement. In *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 1–4, 2020.

[7] AUTOSAR. Explanation of Adaptive Platform Design, March 2019. URL: https://www.autosar.org/fileadmin/user_upload/standards/adaptive/19-11/AUTOSAR_EXP_PlatformDesign.pdf.

[8] Sebastian R Bader and Maria Maleshkova. Towards enforceable usage policies for industry 4.0. In *LASCAR@ ESWC*, pages 75–84, 2019.

[9] Gabriele Baldi, Yair Diaz-Tellez, Theo Dimitrakos, Fabio Martinelli, Christina Michailidou, Paolo Mori, Oleksii Osliak, and Andrea Saracino. Session-Dependent Usage Control for Big Data. *J. Internet Serv. Inf. Secur.*, 10(3):76–92, 2020.

[10] Subhajit Bandopadhyay, Theo Dimitrakos, Yair Diaz, Ali Hariri, Tezcan Dilshener, Antonio La Marra, and Alessandro Rosetti. DataPAL: Data Protection and Authorization Lifecycle framework. In *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pages 1–8. IEEE, 2021. Published. URL: `https://ieeexplore.ieee.org/abstract/document/9566212`.

[11] David E Bell, Leonard J La Padula, et al. Secure Computer System: Unified Exposition and Multics Interpretation. 1976.

[12] Cesar Bernardini, Muhammad Rizwan Asghar, and Bruno Crispo. Security and privacy in vehicular communications: Challenges and opportunities. *Vehicular Communications*, 10:13–28, 2017.

[13] Andrew Blower and Gerald Kotonya. RECON: A Real-time Entity Based Access Control for IoT. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 142–146. IEEE, 2019. URL: `https://doi.org/10.1109/IOTSMS48152.2019.8939218`.

[14] Susanne Boll and Jochen Meyer. Health-x dataloft: A sovereign federated cloud for personalized health care services. *IEEE MultiMedia*, 29(1):136–140, 2022.

[15] Ondrej Burkacky, Johannes Deichmann, Georg Doll, and Christian Knochenhauer. Rethinking car software and electronics architecture, February 2018. URL: `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture`.

[16] Ondrej Burkacky, Johannes Deichmann, Benjamin Klein, Klaus Pototzky, and Gundbert Scherf. Cybersecurity in automotive: Mastering the challenge, June 2020. URL: `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/cybersecurity-in-automotive-mastering-the-challenge`.

[17] Óscar Cánovas, Gabriel López, and Antonio F Gómez-Skarmeta. A Credential Conversion Service for SAML-Based Scenarios. In *Public Key Infrastructure: First European PKI Workshop: Research and Applications, EuroPKI 2004, Samos Island, Greece, June 25-26, 2004. Proceedings 1*, pages 297–305. Springer, 2004.

[18] Quyet H Cao, Madhusudan Giyyarpuram, Reza Farahbakhsh, and Noel Crespi. Policy-Based Usage Control for a Trustworthy Data Sharing Platform in Smart Cities. *Future Generation Computer Systems*, 107:998–1010, 2020.

[19] Enrico Carniani, Davide D'Arenzo, Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. Usage Control on Cloud Systems. *Future Generation Computer Systems*, 63:37–55, 2016.

[20] Ajay Chawla. Pegasus spyware – 'a privacy killer', 2021. URL: `https://europepmc.org/article/PPR/PPR381918`.

[21] Chung, David Ferraiolo, David Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC) Definition and Considerations, 2019. URL: `https://www.nist.gov/publications/guide-attribute-based-access-control-abac-definition-and-considerations-1`.

[22] Flavio Cirillo, Bin Cheng, Raffaele Porcellana, Marco Russo, Gürkan Solmaz, Hisashi Sakamoto, and Simon Pietro Romano. Intentkeeper: Intent-oriented data usage control for federated data analytics. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pages 204–215. IEEE, 2020.

[23] Maurizio Colombo, Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. A Proposal on Enhancing XACML with Continuous Usage Control Features. In *Grids, P2P and Services Computing [Proceedings of the CoreGRID ERCIM Working Group Workshop on Grids, P2P and Service Computing, 24 August 2009, Delft, The Netherlands]*, pages 133–146. Springer, 2009. URL: `https://doi.org/10.1007/978-1-4419-6794-7_11`.

[24] European Commission. The European Digital Identity Wallet Architecture and Reference Framework, 2023. URL: `https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework`.

[25] Council of the EU and the European Council. A Digital Future for Europe. URL: `https://www.consilium.europa.eu/en/policies/a-digital-future-for-europe/`.

[26] Stephane Couture and Sophie Toupin. What does the notion of "sovereignty" mean when referring to the digital? *new media & society*, 21(10):2305–2322, 2019.

[27] Daniele Micciancio. Deterministic Finite Automata (DFA), 2015. URL: `https://cseweb.ucsd.edu/classes/sp15/cse191-e/lec1.html`.

[28] Ferraiolo David and Kuhn Richard. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, volume 563. Baltimore, Maryland: NIST-NCSC, 1992.

[29] Harry Davies. Ted Cruz using firm that harvested data on millions of unwitting Facebook users. *the Guardian*, 11:2015, 2015. URL: `https://www.theguardian.com/us-news/2015/dec/11/senator-ted-cruz-president-campaign-facebook-user-data`.

[30] Johannes Deichmann, Benjamin Klein, Gundbert Scherf, and Stuetzle Rupert. The race for cybersecurity: Protecting the connected car in the era of new regulation, October 2019. URL: `https://mckinsey.com/industries/automotive-and-assembly/our-insights/the-race-for-cybersecurity-protecting-the-connected-car-in-the-era-of-new-regulation`.

[31] Günter Eggers, Bernd Fondermann, Berthold Maier, Klaus Ottradovetz, Julius Pfrommer, Ronny Reinhardt, Hannes Rollin, Arne Schmieg, Sebastian Steinbuß, Philipp Trinius, Andreas Weiss, Christian Weiss, and Sabine Wilfling. GAIA-X: Technical Architecture, 2020. URL: `https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-technical-architecture.pdf`.

[32] ESSIF Lab. European Self-Sovereign Identity Framework Lab, 2021. URL: `https://essif-lab.eu/`.

[33] Beatriz Esteves, Harshvardhan J Pandit, and Víctor Rodríguez-Doncel. ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 298–306. IEEE, 2021.

[34] GAIA-X European Association for Data and Cloud. Gaia-X Architecture Document - Federation Services, 2022. URL: `https://docs.gaia-x.eu/technical-committee/architecture-document/latest/federation_service/`.

[35] Georgios Fragkos, Jay Johnson, and Eirini Eleni Tsiropoulou. Dynamic Role-Based Access Control Policy for Smart Grid Applications: an Offline Deep Reinforcement Learning Approach. *IEEE Transactions on Human-Machine Systems*, 52(4):761–773, 2022.

[36] Vijay Gadepally and Jeremy Kepner. Technical Report: Developing a Working Data Hub. *arXiv preprint arXiv:2004.00190*, 2020.

[37] Maanak Gupta and Ravi Sandhu. Authorization framework for secure cloud assisted connected cars and vehicular internet of things. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, pages 193–204, 2018.

[38] Mohammad Hamad and Vassilis Prevelakis. Secure APIs for Applications in Microkernel-based Systems. In *ICISSP*, pages 553–558, 2017.

[39] Ali Hariri, Subhajit Bandopadhyay, Athanasios Rizos, Theo Dimitrakos, Bruno Crispo, and Muttukrishnan Rajarajan. SIUV: A Smart Car Identity Management and Usage Control System Based on Verifiable Credentials. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 36–50. Springer, 2021. Published. URL: `https://doi.org/10.1007/978-3-030-78120-0_3`.

[40] Ali Hariri, Amjad Ibrahim, Bithin Alangot, Subhajit Bandopadhyay, Alessandro Rosetti, Antonio La Marra, Hussein Joumaa, and Theo Dimitrakos. UCON+: Comprehensive Model, Architecture and Implementation for Usage Control and Continuous Authorization. In *Collaborative Approaches for Cyber Security in Cyber-Physical Systems*, pages 209–226. Springer, 2023. Published. URL: `https://doi.org/10.1007/978-3-031-16088-2_10`.

[41] Ali Hariri, Amjad Ibrahim, Theo Dimitrakos, and Bruno Crispo. WiP: Metamodel for Continuous Authorisation and Usage Control. In *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*, pages 43–48, 2022. Published. URL: `https://doi.org/10.1145/3532105.3535039`.

[42] Manuel Hilty, Alexander Pretschner, David Basin, Christian Schaefer, and Thomas Walter. A Policy Language for Distributed Usage Control. In *Computer Security–ESORICS 2007: 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24—26, 2007. Proceedings 12*, pages 531–546. Springer, 2007.

[43] Seongho Hong and Heeyoul Kim. Vaultpoint: A Blockchain-Based SSI Model that Complies with OAuth 2.0. *Electronics*, 9(8):1231, 2020.

[44] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. Cyber-Physical Systems Security — A Survey. *IEEE Internet of Things Journal*, 4(6):1802–1831, 2017.

[45] Renato Iannella, Michael Steidl, Stuart Myles, and Víctor Rodríguez-Doncel. ODRL Vocabulary and Expression 2.2, 2018. URL: `https://www.w3.org/TR/odrl-vocab/`.

[46] Renato Iannella and Serena Villata. ODRL Information Model 2.2, 2018. URL: `https://www.w3.org/TR/odrl-model/`.

[47] Amjad Ibrahim and Theo Dimitrakos. Towards collaborative security approaches based on the european digital sovereignty ecosystem. In *Collaborative Approaches for Cyber Security in Cyber-Physical Systems*, pages 123–144. Springer, 2023.

[48] International Data Space Association (IDSA). Dataspace Protocol - Working Draft. URL: `https://docs.internationaldataspaces.org/ids-knowledgebase/v/dataspace-protocol/`.

[49] Jostein Jensen. Identity Management Lifecycle - Exemplifying the Need for Holistic Identity Assurance Frameworks. In *Information and Communication Technology-EurAsia Conference*, pages 343–352. Springer, 2013.

[50] Christian Jung, Jörg Dörr, B Otto, M Ten Hompel, and S Wrobel. Data usage control. *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, pages 129–146, 2022.

[51] Euihyun Jung, Younghwan Choi, Jun Seob Lee, and Hyoung Jun Kim. An OID-Based Identifier Framework Supporting the Interoperability of Heterogeneous Identifiers. In *2012 14th International Conference on Advanced Communication Technology (ICACT)*, pages 304–308. IEEE, 2012.

[52] Basel Katt, Xinwen Zhang, Ruth Breu, Michael Hafner, and Jean-Pierre Seifert. A general obligation model and continuity: enhanced policy enforcement engine for usage control. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 123–132, 2008.

[53] Milen G Kebede, Giovanni Sileno, and Tom Van Engers. A Critical Reflection on ODRL. In *International Workshop on AI Approaches to the Complexity of Legal Systems*, pages 48–61. Springer, 2018.

[54] Dae-Kyoo Kim, Eunjee Song, and Huafeng Yu. Introducing Attribute-Based Access Control to AUTOSAR. Technical report, SAE Technical Paper, 2016.

[55] Jahoon Koo, Giluk Kang, and Young-Gab Kim. Security and Privacy in Big Data Life Cycle: a Survey and Open Challenges. *Sustainability*, 12(24):10571, 2020.

[56] Thomas Kurian. Engaging in a European dialogue on customer controls and open cloud solutions, 2020. URL: `https://cloud.google.com/blog/products/identity-security/how-google-cloud-is-addressing-data-sovereignty-in-europe-2020`.

[57] Antonio La Marra, Alessio Lunardelli, Fabio Martinelli, Paolo Mori, Andrea Saracino, Piero Castoldi, Francesco Martino, and Barbara Martini. Enhancing Security in ETSI Open Source MANO with Usage Control Capability. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 25–29. IEEE, 2019.

[58] Antonio La Marra, Fabio Martinelli, Paolo Mori, and Andrea Saracino. Implementing Usage Control in Internet of Things: A Smart Home Use Case. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 1056–1063. IEEE, 2017.

[59] Dmitrij Lagutin, Yki Kortesniemi, Nikos Fotiou, and Vasilios A Siris. Enabling Decentralised Identifiers and Verifiable Credentials for Constrained IoT Devices using OAuth-based Delegation. In *Proceedings of the Workshop on Decentralized IoT Systems and Security (DISS 2019), in Conjunction with the NDSS Symposium, San Diego, CA, USA*, volume 24, 2019.

[60] Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. Usage control in computer security: A survey. *Computer Science Review*, 4(2):81–99, 2010.

[61] Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. A prototype for enforcing usage control policies based on xacml. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 79–92. Springer, 2012.

[62] Aliaksandr Lazouski, Fabio Martinelli, Paolo Mori, and Andrea Saracino. Stateful Data Usage Control for Android Mobile Devices. *Int. J. Inf. Sec.*, 16(4):345–369, 2017. URL: `https://doi.org/10.1007/s10207-016-0336-y`.

[63] Tambiama André MADIEGA. Digital Sovereignty for Europe, 2020.

[64] Fabio Martinelli, Christina Michailidou, Paolo Mori, and Andrea Saracino. Too Long, Did Not Enforce: A Qualitative Hierarchical Risk-Aware Data Usage Control

Model for Complex Policies in Distributed Environments. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, pages 27–37, 2018.

[65] Barbara Martini, Paolo Mori, Francesco Marino, Andrea Saracino, Alessio Lunardelli, Antonio La Marra, Fabio Martinelli, and Piero Castoldi. Pushing Forward Security in Network Slicing by Leveraging Continuous Usage Control. *IEEE Communications Magazine*, 58(7):65–71, 2020.

[66] MATTR. OIDC Bridge, 2023. URL: `https://learn.mattr.global/docs/vii-platform/oidc-bridge`.

[67] Charles Morisset, Tim AC Willemse, and Nicola Zannone. A Framework for the Extended Evaluation of ABAC Policies. *Cybersecurity*, 2(1):1–21, 2019.

[68] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. A Survey on Essential Components of a Self-Sovereign Identity. *Computer Science Review*, 30:80–86, 2018.

[69] Andres Munoz-Arcentales, Sonsoles López-Pernas, Alejandro Pozo, Álvaro Alonso, Joaquín Salvachúa, and Gabriel Huecas. An Architecture for Providing Data Usage and Access Control in Data Sharing Ecosystems. *Procedia Computer Science*, 160:590–597, 2019.

[70] Andres Munoz-Arcentales, Sonsoles López-Pernas, Alejandro Pozo, Álvaro Alonso, Joaquín Salvachúa, and Gabriel Huecas. Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE. *Sustainability*, 12(9):3885, 2020.

[71] Huu-Ha Nguyen, Phu H Phung, Phu H Nguyen, and Hong-Linh Truong. Context-driven policies enforcement for edge-based iot data sharing-as-a-service. In *2022 IEEE International Conference on Services Computing (SCC)*, pages 221–230. IEEE, 2022.

[72] OASIS. Abbreviated Language For Authorization, 2015. URL: `https://www.oasis-open.org/committees/download.php/55228/alfa-for-xacml-v1.0-wd01.doc`.

[73] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01, 2017. URL: `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html`.

[74] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, 2000.

[75] Boris Otto, Sebastian Steinbuß, Andreas Teuscher, and Sebastian Bader. IDS Reference Architecture Model. Version 4.0. URL: `https://docs.internationaldataspaces.org/ids-knowledgebase/v/ids-ram-4/`.

[76] Jaehong Park and Ravi Sandhu. The UCONABC Usage Control Model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[77] Jaehong Park, Xinwen Zhang, and Ravi S. Sandhu. Attribute Mutability in Usage Control. In *Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security, July 25-28, 2004, Sitges, Catalonia, Spain*, volume 144 of *IFIP*, pages 15–29. Kluwer/Springer, 2004. URL: `https://doi.org/10.1007/1-4020-8128-6_2`.

[78] EU Parliament. General Data Protection Regulation, 2016. URL: `https://eur-lex.europa.eu/eli/reg/2016/679/oj`.

[79] Heinrich Pettenpohl, Markus Spiekermann, and Jan Ruben Both. International Data Spaces in a Nutshell. *Designing Data Spaces; Springer: Cham, Switzerland*, pages 29–40, 2022.

[80] Julia Pohle, Thorsten Thiel, et al. Digital sovereignty. *Practicing Sovereignty: Digital Involvement in Times of Crises*, pages 47–67, 2021.

[81] Alexander Pretschner, Enrico Lovat, and Matthias Büchler. Representation-Independent Data Usage Control. In *International Workshop on Data Privacy Management*, pages 122–140. Springer, 2011.

[82] Davy Preuveneers and Wouter Joosen. Towards multi-party policy-based access control in federations of cloud and edge microservices. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 29–38. IEEE, 2019.

[83] E Rissanen, H Lockhart, and T Moses. XACML v3.0 Administration and Delegation Profile Version 1.0. *Committee Draft*, 4, 2014. URL: `https://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-en.html`.

[84] Athanasios Rizos, Daniel Bastos, Andrea Saracino, and Fabio Martinelli. Distributed UCON in CoAP and MQTT Protocols. In *Computer Security: ESORICS 2019 International Workshops, CyberICPS, SECPRE, SPOSE, and ADIoT, Luxembourg City, Luxembourg, September 26–27, 2019 Revised Selected Papers 5*, pages 35–52. Springer, 2020.

[85] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero Trust Architecture. *NIST special publication*, 800:207, 2020.

[86] Wenqiang Ruan, Mingxin Xu, Haoyang Jia, Zhenhuan Wu, LuShan Song, and Weili Han. Privacy Compliance: Can Technology Come to the Rescue? *IEEE Security & Privacy*, 19(4):37–43, 2021.

[87] Marcel Rumez, Alexander Duda, Patrick Gründer, Reiner Kriesten, and Eric Sax. Integration of Attribute-based Access Control into Automotive Architectures. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1916–1922. IEEE, 2019.

[88] Marcel Rumez, Daniel Grimm, Reiner Kriesten, and Eric Sax. An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures. *IEEE Access*, 8:221852–221870, 2020.

[89] Babak Sadighi. Is ALFA a part of the OASIS XACML Technical Committee series of standards?, 2023. URL: `https://axiomatics.com/blog/is-alfa-a-part-of-the-oasis-xacml-technical-committee-series-of-standards`.

[90] Farzad Salim, Jason Reid, and Ed Dawson. An administrative model for UCON ABC. *Information Security 2010*, pages 32–38, 2010.

[91] Ravi Sandhu and Qamar Munawer. How to do discretionary access control using roles. In *Proceedings of the third ACM workshop on Role-based access control*, pages 47–54, 1998.

[92] Ravi S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, 1993.

[93] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[94] Ulrich Schöpp, Chuangjie Xu, Amjad Ibrahim, Fathiyeh Faghih, and Theo Dimitrakos. Specifying a Usage Control System. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*, pages 193–200, 2023.

[95] Julian Schütte and Gerd Stefan Brost. 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)*, pages 289–299. IEEE, 2018.

[96] Shalev Ben-David. Finite Automata, 2023. URL: `https://cs.uwaterloo.ca/~s4bendav/files/CS360S21Lec03.pdf`.

[97] Hira Siddiqui, Mujtaba Idrees, Ivan Gudymenko, Christof Fetzer, et al. Credentials as a Service Providing Self Sovereign Identity as a Cloud Service Using Trusted Execution Environments. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*, pages 210–216. IEEE, 2021.

[98] Anthony Simonet, Gilles Fedak, Matei Ripeanu, and Samer Al-Kiswany. Active Data: A Data-Centric Approach to Data Life-Cycle Management. In *Proceedings of the 8th Parallel Data Storage Workshop*, pages 39–44, 2013.

[99] Amir Sinaeepourfard, Xavier Masip-Bruin, Jordi Garcia, and Eva Marín-Tordera. A Survey on Data Lifecycle Models: Discussions Toward the 6Vs Challenges. *Technical Report (UPC-DAC-RR-2015–18)*, 2015.

[100] Manu Sporny, Dave Longley, and David Chadwick. Verifiable Credentials Data Model v1.1, 2022. URL: `https://www.w3.org/TR/vc-data-model/`.

[101] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized Identifiers (DIDs) v1.0, 2022. URL: `https://www.w3.org/TR/did-core/`.

[102] Vinh Thong Ta and Max Hashem Eiza. Dataprove: Fully automated conformance verification between data protection policies and system architectures. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022(1):565–585, 2022.

[103] Oliver Terbu, Torsten Lodderstedt, Kristina Yasuda, and Tobias Looker. OpenID for Verifiable Presentations - draft 18, 2023. URL: `https://openid.net/specs/openid-4-verifiable-presentations-1_0.html`.

[104] Andrew Tobin and Drummond Reed. The Inevitable Rise of Self-Sovereign Identity. *The Sovrin Foundation*, 29(2016):18, 2016.

[105] Kristina Yasuda, Michael B. Jones, and Torsten Lodderstedt. Self-Issued OpenID Provider v2, 2023. URL: `https://openid.net/specs/openid-connect-self-issued-v2-1_0.html`.

[106] Xinwen Zhang, Francesco Parisi-Presicce, Ravi S. Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005. URL: `https://doi.org/10.1145/1108906.1108908`.

[107] Johannes Zrenner, Frederik Oliver Möller, Christian Jung, Andreas Eitel, and Boris Otto. Usage Control Architecture Options for Data Sovereignty in Business Ecosystems. *Journal of Enterprise Information Management*, 2019.