



**HAL**  
open science

# Data centered Usage based Protection in a SMACIT context

Yuan Jingya

► **To cite this version:**

Yuan Jingya. Data centered Usage based Protection in a SMACIT context. Cryptography and Security [cs.CR]. Université de Lyon, 2021. English. NNT : 2021LYSEI044 . tel-03406822

**HAL Id: tel-03406822**

**<https://theses.hal.science/tel-03406822v1>**

Submitted on 28 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



N°d'ordre NNT : 2021LYSEI044

## **THESE de DOCTORAT DE L'UNIVERSITE DE LYON**

opérée au sein de  
**INSA-Lyon**

**Ecole Doctorale N° 512  
INFOMATHS**

**Spécialité/ discipline de doctorat :**  
Informatique

Soutenue publiquement le 8/7/2021 par :  
**Jingya YUAN**

---

## **Data centered Usage based Protection in a SMACIT context**

---

Devant le jury composé de :

M. Marco WINCKLER, Professeur des Universités, Université Côte d'Azur,  
Examineur  
M. Michael MARISSA, Directeur de Recherche, INNORENEW, Rapporteur  
M. Khalid BENALI, Maître de Conférences HDR, Université de Lorraine,  
Rapporteur  
Mme Genoveva VARGAS-SOLAR, Chargée de Recherches, CNRS, Examinatrice  
Mme Frédérique BIENNIER, Professeur des Universités, INSA-Lyon, Directrice de  
thèse  
Mme Nabila BENHARKAT, Maître de Conférences, INSA-Lyon, Co-encadrante de  
thèse

**Département FEDORA – INSA Lyon - Ecoles Doctorales**

<b>SIGLE</b>	<b>ECOLE DOCTORALE</b>	<b>NOM ET COORDONNEES DU RESPONSABLE</b>
<b>CHIMIE</b>	<b>CHIMIE DE LYON</b> <a href="https://www.edchimie-lyon.fr">https://www.edchimie-lyon.fr</a> Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	<b>M. Stéphane DANIELE</b> C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne directeur@edchimie-lyon.fr
<b>E.E.A.</b>	<b>ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE</b> <a href="https://edeea.universite-lyon.fr">https://edeea.universite-lyon.fr</a> Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	<b>M. Philippe DELACHARTRE</b> INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 philippe.delachartre@insa-lyon.fr
<b>E2M2</b>	<b>ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODELISATION</b> <a href="http://e2m2.universite-lyon.fr">http://e2m2.universite-lyon.fr</a> Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	<b>M. Philippe NORMAND</b> Université Claude Bernard Lyon 1 UMR 5557 Lab. d'Ecologie Microbienne Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX philippe.normand@univ-lyon1.fr
<b>EDISS</b>	<b>INTERDISCIPLINAIRE SCIENCES-SANTÉ</b> <a href="http://ediss.universite-lyon.fr">http://ediss.universite-lyon.fr</a> Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	<b>Mme Sylvie RICARD-BLUM</b> Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
<b>INFOMATHS</b>	<b>INFORMATIQUE ET MATHÉMATIQUES</b> <a href="http://edinfomaths.universite-lyon.fr">http://edinfomaths.universite-lyon.fr</a> Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	<b>M. Hamamache KHEDDOUCI</b> Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 hamamache.kheddouci@univ-lyon1.fr
<b>Matériaux</b>	<b>MATÉRIAUX DE LYON</b> <a href="http://ed34.universite-lyon.fr">http://ed34.universite-lyon.fr</a> Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	<b>M. Stéphane BENAYOUN</b> Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 stephane.benayoun@ec-lyon.fr
<b>MEGA</b>	<b>MÉCANIQUE, ÉNERGÉTIQUE, GENIE CIVIL, ACOUSTIQUE</b> <a href="http://edmega.universite-lyon.fr">http://edmega.universite-lyon.fr</a> Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	<b>M. Jocelyn BONJOUR</b> INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
<b>ScSo</b>	<b>ScSo*</b> <a href="https://edsciencessociales.universite-lyon.fr">https://edsciencessociales.universite-lyon.fr</a> Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	<b>M. Christian MONTES</b> Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 christian.montes@univ-lyon2.fr

\*ScSo: Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

## Acknowledgment

At the beginning of this dissertation and the end of this journey, I would like to thank all those who helped me to complete this work. Without much support and encouragement from professors, friends, the SOC team, family, and even my country, I would not have achieved my Ph.D. degree.

Firstly, I would like to thank my doctoral advisors Professor Frédérique Biennier and Mrs. Aïcha-Nabila BENHARKAT for the numerous scientific discussions, the insightful inputs on my work, and their guidance all along with my thesis. A special thanks will be given to Professor Frédérique Biennier as she has always been patient, responsible, efficient, and kind.

I would also like to thank the members of the jury who evaluated my work: Mr. Marco WINCKLER, Mr. Michael MARISSA, and M. Khalid BENALI for reviewing my dissertation, as well as Mrs. Genoveva VARGAS SOLAR for being part of my jury and their interest in my work.

I would also like to thank my fellow Ph.D. students in the SOC team for all valuable academic suggestions, relevant research ideas, and administrative supports during my research.

Furthermore, I do appreciate my country, more specifically, the Chinese Scholarship Council (CSC), for their financial support so that I could study in INSA de Lyon, France.

Finally, I owe my deepest gratitude to my family. My parents and grandparents provide a stably warm and sweet environment throughout my life.

## Abstract

Protecting Information Systems (IS) relies traditionally on security risk analysis methods. Designed for well-perimetrised environments, these methods rely on a systematic identification of threats and vulnerabilities to identify efficient control-centered protection countermeasures. Unfortunately, this does not fit security challenges carried out by the opened and agile organizations provided by the Social, Mobile, big data Analytics, Cloud, and Internet of Things (SMACIT) environment. Due to their inherently collaborative and distributed organization, such multi-tenancy systems require the integration of contextual vulnerabilities, depending on the a priori unknown way of using, storing, and exchanging data in the opened cloud environment. Moreover, as data can be associated with multiple copies, different protection requirements can be set for each of these copies, which may lead the initial data owner to lose control of the data protection. To overcome these limits, we propose a Data centered Usage-based Protection model relying on an IS Description model to set consistent protection for data assets. Protection means are defined according to both organizational and technical risks. To this end, we propose a GDPR compliant security and extended usage ontology which is used to define usage-control assertions coupling usage rights to security countermeasures so that data assets can be efficiently protected according to both organizational and technical dimensions. Thanks to a Blockchain-based usage control, our Data centered and Usage-based Protection architecture also allows tracking the way assets are used so their life-long protection can be checked.

# Table of Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Research context	1
1.2	Key research questions	3
1.3	Dissertation organization	5
<b>2</b>	<b>State of the art</b>	<b>6</b>
2.1	Information System security risks evaluation	6
2.1.1	Traditional risk and security engineering methods	8
2.1.2	SMAC-IT security Challenges	10
2.1.3	Risks, vulnerabilities, and countermeasures synthesis	15
2.1.4	Conclusion.	18
2.2	Security policy	19
2.2.1	Security policy ontologies	20
2.2.2	GDRP-based asset protection	31
2.3	Conclusion	34
<b>3</b>	<b>Data centered protection model</b>	<b>36</b>
3.1	Multi-layer information system description model	38
3.1.1	Static view of the Information System description	40
3.1.2	Integration of security requirements and protection means in the IS description model	45
3.1.3	Integration of the IS usage: a transaction-based model	49
3.1.4	Conclusion	50
3.2	Usage-based Protection model	50
3.2.1	Multi-dimension protection ontology	51
3.2.2	Usage-based policy model	56
3.2.3	Data-centric protection management	61
3.3	Conclusion	68
<b>4</b>	<b>Data-driven protection architecture</b>	<b>70</b>
4.1	Introduction	70
4.2	Data-driven and Usage-Based Protection architecture	71
4.2.1	Description of the main components	73
4.2.2	DUP organization	77
4.3	Using DUP	78
4.3.1	Terms of Usage negotiation	78
4.3.2	Usage derivation	87
4.3.3	Managing trusted transactions	92
4.3.4	Usage Governance	100
4.4	Evaluation	101
4.4.1	Experiment	101
4.4.2	Evaluation of DUP	118
4.5	Conclusion	121

<b>5</b>	<b>Conclusion</b>	<b>122</b>
<b>6</b>	<b>References</b>	<b>125</b>
<b>7</b>	<b>Annex: Smart Contract Patterns</b>	<b>132</b>
7.1	Exchange Smart Contract pattern	132
7.2	Usage Smart Contract pattern	135
7.3	Physical Smart Contract pattern	138
7.4	Tracking Smart Contract pattern	140
<b>8</b>	<b>Annex: Prototype key functions</b>	<b>142</b>
<b>8.1</b>	<b>ToU generation process part</b>	<b>142</b>
8.1.1	ToS generation	142
8.1.2	QoP evaluation	150
8.1.3	RoP generation	155
8.1.4	ToU evaluation	158
<b>8.2</b>	<b>Transaction generation part</b>	<b>161</b>
8.2.1	Business Transaction	161
8.2.2	Business Transaction generation	165
8.2.3	Usage Transaction	171
8.2.4	Usage Transaction generation	173
8.2.5	Physical Transaction	175
8.2.6	Physical Transaction generation	177
<b>8.3</b>	<b>Smart Contract generation</b>	<b>178</b>

## LIST OF FIGURES

FIGURE 1 CLOUD COMPUTING SECURITY STACK	12
FIGURE 2 UML CLASS DIAGRAM FOR SIMPLE ID CREDENTIAL ONTOLOGY [72]	21
FIGURE 3 VOWL DIAGRAM OF THE ACCESS CONTROL ONTOLOGY [73]	22
FIGURE 4 VOWL DIAGRAM OF THE WS-POLICY ONTOLOGY [74]	22
FIGURE 5 VOWL DIAGRAM OF THE CLOUD ACCESS CONTROL ONTOLOGY [78]	23
FIGURE 6 UML CLASS DIAGRAM OF THE DATA USER ONTOLOGY [79]	24
FIGURE 7 UML CLASS DIAGRAM OF THE PERSONAL PROFILE AND DIGITAL TRACES FROM[79]	24
FIGURE 8 UML CLASS DIAGRAM OF THE PURPOSE ONTOLOGY PICKED FROM[79]	24
FIGURE 9 UML CLASS DIAGRAM OF A SAMPLE ONTOLOGY TREE COMPANIES [80]	25
FIGURE 10 VOWL DIAGRAM OF THE SOCIAL NETWORKING SYSTEM ONTOLOGY[81]	25
FIGURE 11 MIND MAP OF THE ODRL XML COMPLEXTYPES [82]	26
FIGURE 12 VOWL DIAGRAM OF THE OREL ONTOLOGY [83]	27
FIGURE 13 MIND MAP OF THE BASIC PRIVACY ONTOLOGY [84]	27
FIGURE 14 POLICY ASSERTION MODEL BASED ON [85]	28
FIGURE 15 VOWL DIAGRAM OF THE USAGE CONTROL POLICY MODEL [85]	28
FIGURE 16 UML CLASS DIAGRAM OF THE ORGANIZATION DESCRIPTION -MODEL	41
FIGURE 17 UML CLASS DIAGRAM OF THE DATA DESCRIPTION MODEL	42
FIGURE 18 UML CLASS DIAGRAM OF THE USAGE DESCRIPTION MODEL	43
FIGURE 19 EXAMPLE OF BUSINESS SERVICES PROVIDED BY OLS AND THE ASSETS THEY REQUIRE	43
FIGURE 20 CLASS DIAGRAM OF THE CONCRETE SERVICE AND DATA IMPLEMENTATION MODEL	44
FIGURE 21 DUMP OF THE DATA BASE OF THE LOGICAL ASSET RELATED TO ALICE'S CONTACT INFORMATION AND ITS TWO CONTENTS STORED IN OLS OWN INFORMATION SYSTEM DESCRIPTION	44
FIGURE 22 MULTI-LAYER CLOUD-BASED DEPLOYMENT	45
FIGURE 23 CLASS DIAGRAM OF THE SECURED DATA DESCRIPTION MODEL	47
FIGURE 24 CLASS DIAGRAM OF THE SECURED USAGE DESCRIPTION MODEL	48
FIGURE 25 EXAMPLE OF PROTECTION LEVEL ASSOCIATED TO CHARLY'S CONTACT INFORMATION	48
FIGURE 26 MIND MAP OF THE PROTECTION GOAL ONTOLOGY	52
FIGURE 27 MIND MAP OF THE USAGE ONTOLOGY	52
FIGURE 28 MIND MAP OF THE BUSINESS AREA ONTOLOGY	53
FIGURE 29 CLASS DIAGRAM OF THE POLICY MODEL	56
FIGURE 30 MIND MAP DIAGRAM OF THE PRIVACY PROTECTION GOAL AND ASSOCIATED COUNTERMEASURE EFFICIENCY	57
FIGURE 31 MIND MAP DIAGRAM OF THE SECURITY SERVICES PROTECTION GOALS AND COUNTERMEASURE EFFICIENCY	57
FIGURE 32 MIND MAP OF THE PHYSICAL INFRASTRUCTURE PROTECTION AND COUNTERMEASURE EFFICIENCY	58
FIGURE 33 MIND MAP OF THE USAGE OPERATION INCLUSION RELATIONSHIPS	62
FIGURE 34 LOGICAL ASSET COMPOSITION CODE FRAGMENT	65
FIGURE 35 ROP EVALUATION CODE FRAGMENT	65
FIGURE 36 POLICY MATCHING CODE FRAGMENT	66
FIGURE 37 DUP ORGANIZATION	71
FIGURE 38 CONSENT NEGOTIATION USE CASE DIAGRAM	72
FIGURE 39 USAGE MANAGEMENT USE-CASE DIAGRAM	73
FIGURE 40 DATA-DRIVEN PROTECTION ARCHITECTURE	73
FIGURE 41 SEQUENCE DIAGRAM OF THE NEGOTIATION INTERACTION PROCESS SEQUENCE DIAGRAM	79
FIGURE 42 EXTRACTION OF LOGICAL ASSET PATTERNS THANKS TO THE ASSET COMPOSITION RELATIONSHIP	80
FIGURE 43 GLOBAL QUALITY OF PROTECTION EVALUATION FLOW CHART	82



FIGURE 44 CODE FRAGMENT RELATED TO THE INTEGRATION OF TOS FROM SUB-SERVICES	83
FIGURE 45 CODE FRAGMENT RELATED TO THE FINAL AGGREGATION TO SET THE TOS	83
FIGURE 46 PARTIAL DUMP OF THE DATA BASE SHOWING THE FINAL TOS ASSERTIONS GENERATED FROM THE MOTIVATING EXAMPLE	84
FIGURE 47 REQUIREMENT OF PROTECTION EVALUATION FLOW CHART	85
FIGURE 48 ROP AGGREGATION CODE FRAGMENT	85
FIGURE 49 PARTIAL DUMP OF THE DATABASE SHOWING THE CONSOLIDATED ROP STORED IN ALICE'S INFORMATION SYSTEM DESCRIPTION	85
FIGURE 50 MATCHING PROCESS	86
FIGURE 51 DESCRIPTION OF THE BUSINESS TRANSACTION CLASS DIAGRAM	87
FIGURE 52 USAGE TRANSACTION CLASS DIAGRAM	90
FIGURE 53 PHYSICAL TRANSACTION CLASS DIAGRAM	91
FIGURE 54 CLASS DIAGRAM OF THE SMART FACTORY'S SMART CONTRACT REGISTER ORGANIZATION	93
FIGURE 55 TRACKING IMPLEMENTATION PROCESS SEQUENCE DIAGRAM	100
FIGURE 56 USAGE MONITORING PROCESS SEQUENCE DIAGRAM	101
FIGURE 57 PROTOTYPE ARCHITECTURE	102
FIGURE 58 GLOBAL CLASS DIAGRAM OF THE INFORMATION SYSTEM DESCRIPTION USED IN OUR PROTOTYPE	103
FIGURE 59 OBJECT DIAGRAM DESCRIBING THE DIFFERENT USAGES ASSOCIATED TO THE SERVICES	105
FIGURE 60 PARTIAL DUMP OF THE MOTIVATING EXAMPLE DATA BASE: INITIAL TOS EVALUATION	105
FIGURE 61 PARTIAL DUMP OF THE MOTIVATING EXAMPLE: BASIC USAGES	105
FIGURE 62 PART OF THE TOS GENERATION REPORT FROM THE MOTIVATING EXAMPLE	106
FIGURE 63 PARTIAL DUMP OF THE DATABASE SHOWING THE RESULTS OF THE TOS GENERATION PROCESS	106
FIGURE 64 RESULT OF THE QOP GENERATION PROCESS	107
FIGURE 65 PARTIAL DUMP OF THE DATA BASE SHOWING THE FINAL TOS ASSERTIONS GENERATED FROM THE MOTIVATING EXAMPLE	107
FIGURE 66 PARTIAL DUMP OF THE DATA BASE SHOWING THE INITIAL QUALITY OF PROTECTION OF LOGICAL SERVICES	107
FIGURE 67 PARTIAL DUMP OF THE DATA BASE SHOWING THE INITIAL QUALITY OF PROTECTION OF PHYSICAL CONCRETE SERVICE	107
FIGURE 68 PART OF THE QOP GENERATION REPORT	108
FIGURE 69 PARTIAL DUMP OF THE DATA BASE SHOWING THE FINAL QUALITY OF PROTECTION EVALUATION	108
FIGURE 70 TERMS OF USAGE GENERATION REPORT	109
FIGURE 71 PARTIAL DUMP OF THE DATA BASE SHOWING THE FINAL TERMS OF USAGE POLICY	109
FIGURE 72 PARTIAL DUMP OF THE DATA BASE SHOWING THE PART OF ALICE'S INFORMATION SYSTEM DESCRIPTION RELATED TO ROP POLICIES	109
FIGURE 73 ROP EVALUATION ON OUR MOTIVATING EXAMPLE: DELIVERY CONTACT INFORMATION ROP MANAGEMENT ON ALICE SIDE	110
FIGURE 74 PARTIAL DUMP OF THE DATA BASE SHOWING THE CONSOLIDATED ROP STORED IN ALICE'S INFORMATION SYSTEM DESCRIPTION	110
FIGURE 75 PART OF THE BUSINESS TRANSACTION REFINEMENT REPORT PROCESS	111
FIGURE 76 USAGE TRANSACTION GENERATION PROCESS	111
FIGURE 77 PHYSICAL TRANSACTION GENERATION PROCESS	112
FIGURE 78 SMART CONTRACT FACTORY EXECUTION REPORT	112
FIGURE 79 SMART CONTRACT DEPLOYMENT	113
FIGURE 80 SEQUENCE DIAGRAM SHOWING THE SMART-CONTRACT DEPENDENCIES	115
FIGURE 81 DIFFERENT SMART CONTRACT DEPLOYMENT COSTS	116
FIGURE 82 COMPARISON OF THE EXCHANGE SMART CONTRACT DEPLOYMENT AND INVOCATION FUNCTION COSTS	116

FIGURE 83 COMPARISON OF THE USAGE SMART CONTRACT DEPLOYMENT AND INVOCATION FUNCTION COSTS	117
FIGURE 84 COMPARISON OF THE PHYSICAL SMART CONTRACT DEPLOYMENT AND INVOCATION FUNCTION COSTS	117
FIGURE 85 COMPARISON OF THE TRACKING SMART CONTRACT DEPLOYMENT AND INVOCATION FUNCTION COSTS	118
FIGURE 86 ROP, QOP AND MATCHING FUNCTION EXECUTION TIME BASED ON THE PICS PROTOTYPE	120

## LIST OF TABLES

TABLE 1 PRIVACY-PRESERVING TECHNIQUES	14
TABLE 2: SYNTHESIS OF THE DIFFERENT VULNERABILITIES, THREATS, AND COUNTERMEASURES	17
TABLE 3 DATA CENTERED VULNERABILITY AND THREATS	19
TABLE 4 XSD2OWL TRANSLATIONS FOR THE XML PICKED FROM [82](P3-4, TABLE 2)	26
TABLE 5 COMPARISON OF THE DIFFERENT ONTOLOGIES	30
TABLE 6 COUNTERMEASURES FOR DATA PROTECTION	32
TABLE 7 PROCESS MOTIVATION	54
TABLE 8 PROCESS CONTROL PURPOSE	54
TABLE 9 ASSET CONSUMPTION OPERATION	55
TABLE 10 COMPARISON OF OUR SYSTEM WITH OTHER BLOCKCHAIN-BASED SYSTEMS	121

## LIST OF ACRONYMS

ABAC	Attribute Based Access Control
ACL	Access Control List
AHP	<i>Analytical Hierarchy Process</i>
API	Application Programming Interface
ARAMIS	Accidental Risk Assessment Methodology for Industries
B2B	Business to Business
B2C	Business to Customer
BP	Business Process
BPDM	Business Process Definition Meta-model
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
BS	Business Service
BT	Business Transaction
CIA	Confidentiality, Integrity, Availability
CNAT	CoNtext Attribute
CNO	Collaborative Networked Organization
CRM	Customer Relationship Management
CSAC	Cloud Service Access Control
CUCON	Collaborative Usage Control
DDOS	Distributed Denial Of Service
DMP	Data Management Platform
DRM	Digital Right Management
DUP	Data-driven and Usage-based Protection
Ebios	Expression des Besoins et Identification des Objectifs de Sécurité
ECA	Event Condition Action
ELI	European Legislation Identifier
ERP	Enterprise Resource Planning
ETH	Ether - Ethereum currency
FMEA	Failure Modes and Effects Analysis
GAHP	Graphical AHP
GDPR	General Data Protection Regulation
GERAM	Generalised Enterprise Reference Architecture and Methodology
GFS	Global File System
GRAI	Graphe à Résultats et Activités Inter-reliées
HDFS	Hadoop Distributed File System
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System

IoT	Internet of Things
IP	Internet Protocol
IS	Information System
IT	Information Technology
MEHARI	MEthod for Harmonized Analysis of Risk
MES	Manufacturing Execution System
NIST	National Institute of Standards and Technology
	Organization for the Advancement of Structured Information
OASIS	Standards
OAT	Object ATtribute
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation
ODRL	Open Digital Rights Language
Onto-ACM	Ontology based Access Control Model
OrBAC	Organization Based Access Control
OSL	Obligation Specification Language
OWL	Web Ontology Language
PaaS	Platform as a Service
PT	Physical Transaction
QoP	Quality of Protection
QoS	Quality of Service
RAG	Red Amber Green sssessment
RBAC	Role Based Access Control
REL	Rights Expression Language
RoP	Requirements of Protection
SaaS	Software as a Service
SAT	Subject Attribute
SC	Smart Contract
SCADA	Supervisory Control And Data Acquisition
SCM	Supply Chain Management
SMACIT	Social Mobile Analytics Cloud Information Technology
SMAC-IT	Social Mobile Analytics Cloud Information Technology
SNA	Survivable Network Analysis method
SNO	Social Networking systems Ontology
SNS	Social Networking systems Ontology
SWRL	Semantic Web Rule Language
TLA	Temporal Logic Action
TM	Trust Management
TOE	Traget Of Evaluation
TOGAF	The Open Group entreprise Architecture Framework
ToS	Terms of Service
ToU	Terms of Usage

UCA	Usage Control Assertion
UCON	Usage Based Access Control
UCON ABC	Usage Control with Authorizations (A), obligations (B), and Conditions ©
UMP	Usage Management Policy
URI	Uniform Ressource Identifier
UT	Usage Transaction
VPN	Virtual Private Network
XaaS	Everything s a Service

# 1 INTRODUCTION

## 1.1 *Research context*

Globalized market trends and fast-changing business conditions induce significant changes for enterprises such as focusing on their core business and looking for new collaboration strategies in order to be more flexible, to adapt to the business reality (reduced Time To Market, customized production, sustainable production organization...). Such logic induces the development of outsourcing policies, promoting inter-enterprises collaborative business. These more or less ephemeral collaborative strategies involve both sharing a common project and / or common culture and building an ad-hoc or more formalized common collaborative process which will be the operational support of this collaboration. This collaborative process orchestrates different tasks managed by the partners, interacting with their own Information Systems (IS for short).

Different enterprise engineering methods such as GRAI Integrated Methodology<sup>1</sup>, GERAM<sup>2</sup>... aims at coupling the enterprise decision system and the industrial process to define a consistent Information System organization. These methods provide reusable patterns to guide Business Process (BP for short) design and description. Then, focusing on the way these Business Process are implemented, Enterprise Architecture Frameworks such as Zachman's framework<sup>3</sup>, or the Open Group TOGAF<sup>4</sup> (The Open Group Enterprise Architecture Framework) provide a set of guidelines and patterns to develop the associated software. These pattern-based engineering methods (for both enterprise engineering and software development) lead to set more or less standardized software components. By now, corporate information systems are made of different software applications, such as ERP (Enterprise resource Planning supporting production planning, orders and supply management, accounting functions...), SCM (Supply Chain management system managing interactions with suppliers), CRM (Customer Relationship Management system), MES (Manufacturing Execution System which is used to control and manage the different workshop equipment) and SCADA (Supervisory Control And Data Acquisition) or Cyber-Physical Systems management systems allowing the interaction with the production system. Taking advantage of the Cloud technology and of the Everything as a Service (XaaS) model, most of these software components (excepted those dedicated to the SCADA physical operations) are deployed in a SaaS mode, leading companies to share their own

---

<sup>1</sup> GRAI Integrated Methodology integrate modelling tools devoted to the Information System organisation to the decision model issued from the GRAI (Graphe à Résultats et Activités Inter-relées) developed by Guy Doumeings.

<sup>2</sup> GERAM (**Generalised Enterprise Reference Architecture and Methodology**) was developed by the IFAC/IFIP Task Force on Architectures for Enterprise Integration to set a common framework to manage different methods to support Enterprise Integration.

<sup>3</sup> <https://www.zachman.com/>

<sup>4</sup> <https://www.opengroup.org/togaf>

information with SaaS and Cloud providers. Mobile technologies, including Content Delivery Networks, also favor an efficient access to these distributed Cloud resources.

Moreover, the globalized environment and the reinforcement of B2B (Business to Business) and B2C (Business to Consumer) strategies have changed the way enterprise communicate and develop their marketing strategies. Social media are more and more involved to communicate with this global environment, develop marketing and advertising strategies. Taking advantage of the huge amount of data collected and managed by these systems, Analytics and Big Data systems provide companies tools to improve their decision process, product quality...

These Social media-based communication, Mobile environment, Cloud and Everything as a Service, Cyber-physical systems or Big Data and Analytics, known as SMACIT, are active drivers of the digital transformation and provide new opportunities such as industry 4.0, sharing economy... This increases the call for on-demand collaborative processes and efficient data sharing. This leads to extend the traditional Information System to fit these Collaborative Networked Organizations, opening the traditional Information System to support “on-demand” collaborative processes with different partners. This digital transformation leads to new security and privacy challenges.

First, Information Systems are more and more complex to fit the new collaborative and sharing economy challenges. This means that protecting these complex systems is harder and harder because such IS involves defining shared Business Processes (BP for short) and lots of data exchange among partners. This may increase risks and make it more difficult to provide consistent protection for shared data. Moreover, while traditional enterprise engineering methods as GRAI or frameworks as TOGAF take advantage of pattern-based engineering to improve system design and implementation, security is often neglected.

Second, Personal information is more and more integrated into Information Systems as digital transformation increase B2C models. Personal information can be collected from social media, interactions with consumers... Even if these data lakes represent new benefits for businesses, they also provide new challenges to ensure privacy and legal challenges. In other words, setting a GDPR<sup>5</sup> compliant Information System is not obvious. GDPR empowers end-user but they must manage their information protection consistency. Focusing on data consumers, they must integrate user consent management in their processes.

Third, traditional security models are built to protect well known Information Systems (i.e. with a clear and fixed perimeter). They are mostly control-driven so data may have different protections depending on the processes in which they are involved. Moreover, these systems are designed to manage cyber-risks on the Information Systems. They lack considering processes as potential threats or vulnerabilities for the data asset protection. Moreover, they do not allow user consent management and govern security deployment according to the context. Consequently, they do not fit this Collaborative Information System and SMAC-IT context as the multiple open workflows may lead to

---

<sup>5</sup> GDPR : General Data Protection Regulation (EU GDPR)



un-consistent protection and as the way data are processed may also be seen as a privacy /vulnerability/threat.

This context leads to a key challenge shared by data providers and data consumers: providing consistent and adaptive protection on data assets in this opened SMAC-IT context.

## 1.2 *Key research questions*

To address this global challenge, we identify three main research questions.

First, *protection requirements* must be managed consistently, although the asset is replicated in different information systems and even integrated into various assets. It means that data providers require to manage the way they dispatch copies and the usages they granted. This leads to **question 1: What should be the security strategy to set consistent protection?**

Second, the way data are used can also be seen as a vulnerability/threat. Fair usage management requires integrating business knowledge to identify precisely the way assets are used. This involves that *usage requirements* must be integrated to define the protection policy properly. This leads to **question 2: How to define fair usages in a protection policy?**

Third, to manage life-long data protection, security protection means and usages must be checked continuously. Consequently, *usage governance and tracking means* are requested by both data provider and data consumer to “prove” that assets are used and protected according to what has been approved by the data provider and the data consumer. **This leads to question 3: How to manage the usage proofs to support usage and protection governance?**

By now, traditional information systems are mostly protected in a control-driven way, i.e., paying attention to the way processes are organized and orchestrated to identify data assets they use. Several methods and standards have been developed to identify risks and address security issues for corporate Information systems (IS). These risk analysis and security engineering methods evaluate human resources, physical and environmental conditions, communication and operations, conformance, and incident management from each process before defining a set of policies, standards, procedures, and guidelines to mitigate risks. These risk analysis and security engineering methods are designed to face well-perimetrized environments. Unfortunately, collaborative networked organization and the shared Information Systems involved by the digital transformation rely on opened and agile organizations and make harder identifying the way assets are protected in such un-perimetrized environment. **To overcome this limit and answer question 1, we propose to set a data-driven strategy of protection instead of the control-driven one.** To this end, we propose a *multi-layer data-centered Information System description model* to capture the business organization (i.e. the contractual relationships linking the different parties), the static Information System organization (i.e. the data model and processes using them), and the description of the way the different copies of a data are used. By splitting data assets into two parts, i.e. the logical description for which protection is set and the physical instances involved in different transactions, consistent protection requirements

can be defined once for the logical asset and these requirements can be propagated to the different copies to manage the consistent protection of the asset.

Defining consistent protection requirements for a data asset involves also identifying “unfair” processing practices leading to data confidentiality or integrity abuses. Defining who (process or a dedicated actor) can accede to data and process it has led to usage and access control techniques. Access control can be seen as a fine-grained trust model to define which access action can be granted or denied. Various access control techniques have been defined from the simplest Access Control List, where only some well-identified trusted subjects can get access to an asset, to more complex Attribute-Based Access Control where properties are evaluated to decide to grant (or not) access to an asset. “Usage control policies” expand the “Attribute-based access control” model to replace the simple right associated with an access action to complex usages. These usage-based models encompass and enhance traditional access control models, Trust Management (TM) and Digital Rights Management (DRM). However, traditional security policy attached to services and assets may only provide basic usage actions and does not refer to precise business usage. The coarse description of usage and its operation context leads to underestimating risks from potential usages. Moreover, SMAC-IT changes data usage: data analytics and data mining processes extract knowledge and generate new data to serve business goals, considering that the way data is used can be a potential threat that may corrupt data protection efficiency. To overcome these limits and **provide a finer-grained description of usages to answer question 2**, we propose to expand the traditional protection and usage ontologies with

- new usage operations related to social services and analytic services (such as access delegation, mining,...)
- business knowledge describing a process motivation and its execution context.

By coupling this expanded protection with our multi-layer, precise usages and process context can be captured while defining the protection policy.

Focusing on usage control, the European Union General Data Protection Regulation (GDPR) equilibrates relationships between end-users and data consumers, allowing analytic purpose “by default” and empowering users to manage usage rights on their data, constraining consumers to report any data breach and to enhance transparency. This involves reporting (and prove) actions on data to show that the real usage complies with the usages that the data provider has accepted. Security events (namely security service deployment and security breach identification) must also be reported. To manage this “proof requirement” several works take advantage of the Blockchain immutability property to allow proving a consent, tracking data accountability and provenance... To **expand these works and answer question 3**, we define different smart contracts (exchange smart contract, usage smart contract, physical smart contract, and tracking smart contract) to manage usage consents and derive usage rights provided to different are generated by the smart contract factory according to the ToU<sup>6</sup> assertions and introduced to

---

<sup>6</sup> ToU: Terms of Usage

manage data anonymization, access control rules, tracking data accountability and provenance. To this end, we propose Terms of Usage refinement algorithms to retrieve or deduce the associated consent compliant transactions from high-level business ones to the very low-level physical access operations by implementing usage authorization flow chain and usage operation flow chain.

### **1.3 Dissertation organization**

From this context analysis and the key research questions, we organize this dissertation into 5 chapters, the general introduction, a state-of-the-art review, the presentation of the data-centered protection model, and the Data-driven Protection architecture.

Chapter 2 introduces risk management methods and expands protection requirements to those involved by SMACIT. This leads to study the way security policy are defined and the different ontologies that can be used to support them. Lastly, we present the GDPR main requirements before reviewing few blockchain-based works allowing us to prove consents or track usages.

Chapter 3 presents our formal data-centered protection model, fitting protection requirements associated with questions 1 and 2. We first define the multi-layer *Information System description model* allowing us to describe precisely data assets and processes using them as well as business relationships. Then we expand the protection ontologies to capture usage description, leading to define formally usage-based protection assertions.

Based on this formal model, chapter 4 presents our Data-driven and Usage-based Protection architecture. Taking advantage of the expanded protection ontology and usage control model defined in chapter 3, this architecture includes different components to deduce usage rights from the consent and the business knowledge stored in the Information System description-model. We define different kinds of transactions supported thanks to dedicated smart contracts to manage these usage rights consents and allow governing data asset usage and protection, fitting the requirements associated to question 3.

Lastly, chapter 5 sums up our work before presenting its current limitations and further works.

## 2 State of the art

Traditional information systems are well-perimetrized systems, integrating several interconnected components. These well-identified systems allow designing and deploying efficient control-centered protection to secure both infrastructure, processes, and data they use. However, the fast development of Social networks, Mobile computing environment, Big Data Analytics, and Cloud technologies (known as SMAC-IT) turns these information systems into multi-tenancy collaborative systems, with unknown limits. Paying particular attention to data protection means that a SMAC-IT based information system can dispatch multiple copies of data by using mobile and social media, so that the user has no more control over the way these multiple copies of data are used whereas the Cloud-based deployment may interconnect different services implementing various protection strategy, leading to inconsistent protection. This involves renewing the traditional risks engineering approach to integrate new risks related to the SMAC-IT integration.

To fit this requirement, this research work first reviews risk engineering approaches and SMAC-IT security challenges to identify the protection requirements such opened information systems must fit. Then, we focus on security policy models and on security ontologies to identify how these requirements can be considered. Lastly, particular attention is paid to personal data protection and on the GDPR requirements to manage usage consents and provide a global vision on collaborative SMAC-IT Information Systems protection challenges.

### 2.1 Information System security risks evaluation

Traditional information systems are mostly protected in a control-driven way, i.e. paying attention to the way processes are organized and orchestrated to identify data assets they use. Several methods and standards have been developed to identify risks and address security issues for corporate Information systems (IS). ISO/IEC 17799[1] and ISO/IEC 27002[2] offer guidelines for risk assessments and are based on risk assessment. They integrate human resources, physical and environmental conditions, communication and operations, conformance, and incident management while defining a set of policies, standards, procedures, and guidelines to mitigate risks.

A risk is defined as:

(Equ. 1) Risk=Threat X Vulnerability

Traditionally, Information System vulnerabilities are split into organization-related vulnerabilities and IT-related vulnerabilities. These IT vulnerabilities are often categorized into different layers such as hardware layer, network layer, O/S layer, middleware layer, and application layer. These infrastructure-related vulnerabilities include unsecured interfaces and APIs, as well as network protocols with reused IP addresses and logical network isolation, virtualization/multi-tenancy compromising failures. Failures may also be due to compromised software, taking advantage of un-adapted software management operations (updating, patching...), leading to major security breaches, as was recently

reported for the SolarWinds' Orion network monitoring system. By using a Trojan horse system, cybercriminals were able to infect more than 18,000 companies<sup>7</sup>.

Threats can be classified according to

- their effect: passive threats lead to read, capture and analyze data assets without modifying them whereas active threats lead to altering data assets of the IT system
- the threatening agent: it can belong to the organization owning the Information System (internal threatening agent) or not (external threatening agent)
- its purpose: intentional threats are related to an attack whereas unintentional threats are due to human errors, lack of knowledge of the system...

Focusing on the impact, threats affect one or several security services which [3]defines as:

- Confidentiality: this term covers two related concepts:
  - o Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals
  - o Privacy: Assures that individuals control their own personal information
- Integrity: This term covers two related concepts:
  - o Data integrity: Assures that information and programs are changed only in a specified and authorized manner
  - o System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system
- Availability: Assures that systems work promptly and service is not denied to authorized users.

Although these three basic security services known as the CIA<sup>8</sup> triad are used to describe fundamental security objectives for both data and computing services, two other services can also be added:

- Authenticity: is the property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.
- Accountability: is the security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. Systems must keep records of their activities to permit later legality and analysis to trace security breaches or to aid in transaction disputes.

---

<sup>7</sup> <https://www.switchfast.com/blog/over-18000-infected-from-solarwinds-data-breach-what-happened>

<sup>8</sup> CIA : Confidentiality, Integrity, Availability

- Non-repudiation: is the property that any authenticated action accepted by a party cannot be successfully disputed by its author or the party which has accepted it. It uses authenticity property and integrity service.

To sum up, the confidentiality-related risk is unauthorized disclosure, i.e. due to data exposition, interception or system intrusion. Integrity-related risks can be presented as masquerade, falsification, and repudiation. Availability can be damaged due to usurpation and disruption. Usurpation refers to misappropriation and misuse whereas disruption means system incapacitation, corruption, and obstruction. Of course, an attack can combine different actions, on both the infrastructure (transmission, storage, processing means) and on the organization itself (i.e. integrating human actors). Mitigating risks involves identifying precisely the associated threats and system vulnerabilities to select the most efficient countermeasures to set convenient security policies.

### 2.1.1 Traditional risk and security engineering methods

Assessing risks involves paying attention to the risk occurrence and the cost associated with its consequences. To this end, different methods can be used to provide a quantitative or qualitative risks evaluation:

- Ebios[4]proposes systematic reviews to (1) identify assets, considering data and functions provided by the system, (2) vulnerabilities and threats, paying attention to vulnerability and threat patterns, and (3) a guideline to evaluate costs related to attacks and cost of the attack itself to assess its occurrence probability and the global costs associated to the risks.
- MEHARI introduced by the Club de la Sécurité de l'Information Français (CLUSIF) defines a knowledge base to evaluate quantitatively risks and their impacts. It also proposes large knowledge bases guiding this risk evaluation process.
- ARAMIS methodology [5] offers an alternative to purely deterministic and probabilistic approaches to risk assessment of process plants (i.e. industrial risks in factories). This method, which can be adapted to Cyber risks management, proposes the following steps: identification of the major accidents and the reference accident scenarios, identification of the safety countermeasures and assessment of their performances, evaluation of safety management efficiency, and countermeasure reliability.
- OCTAVE [6] is a self-directed risk-based assessment method that uses the way by which an asset can be acceded (integrating both technical and organizational knowledge) to identify vulnerabilities and threats, guiding the cost evaluations ad proposing countermeasure patterns.
- SNA [7] is an iterative risk-driven process that refines an enterprise system architecture to resist, recognize, and recover from risks
- The methodologies defined in [8] and [9] referred GAHP<sup>9</sup> are several methods for risk assessment which provided AHP decision tree model and the mathematical and theoretic model for risk calculation.

---

<sup>9</sup> GAHP: Graphical *Analytical Hierarchy Process*

These different methods use the evaluation of the risks to classify them and define the strategic countermeasure to deploy, according to the security budget as proposed in Failure Modes and Effects Analysis (FMEA)[10] which multiplies the impact of an occurred risk and the likelihood of that risk to evaluate the risk priority. Unfortunately, reliable data on likelihood and cost may not be available. To overcome this limit, qualitative approaches can be used to qualify risks such as Red, Amber, Green (RAG), and Risk Urgency Assessment[11]. This assessment method provides a simple 3 level classification to assess risk occurrence probability and risk impact. This evaluation improves the risk mitigation decision process by prioritizing risks to choose those which will need mitigation according to the available budget.

Paying particular attention to data protection, access control can be seen as a fine-grained trust model, from the simplest Access Control List where only some well-identified trusted subjects can get access to an asset to more complex Attribute-Based Access Control (ABAC) where properties are evaluated to decide to grant (or not) the access to an asset [12]. For example, the Role-based Access Control (RBAC) [13] or the Organizational Based Access control OrBAC [14] strategies implement an ABAC<sup>10</sup> as organizational information (i.e. roles or the organizational units) that can be seen as a subject's attribute are used to restrict data access. Last but not least, Usage CONTROL (UCON) [15] is a promising approach for access control in open, distributed, heterogeneous, and network-connected computer environments as it allows identifying data operations themselves as potential threats on data. 'Usage control policies' are based on the 'Attribute-based access control' model which is a comprehensive extension that expresses variable usage actions. These usage actions are strongly related to the functional process and [16] proposes to integrate them in the BPEL process specification to define which usage can be granted or denied. This enriches "Rights" part instead of only giving the "grant/deny" of an 'access' action. It encompasses and enhances traditional access control models, Trust Management (TM) and Digital Rights Management (DRM), and its main novelties are mutability of attributes and continuity of access decision evaluation [17]. Different works have been developed to support this Usage Control approach:

The usage control method proposed by Alexander Pretschner, Enrico Lovat et al. presents a framework and its implementation for combining usage control enforcement with data flow tracking technology to enforce simultaneously usage control requirements at all relevant system layers [18]

- Manuel Hilty et al. proposes a transformation-based approach that contains a two-level usage control policy language and expresses a generic server-side architecture for implementing usage control[19][20].
- Prachi Kumari et al. also specifies distributed data usage control in the social network conditions[21]
- Florian Kelbert et al. proposes a data usage control in distributed systems, which extends a generic model for intra-systems and provides data flow tracking among the existence of copies of data [22]

---

<sup>10</sup> ABAC : Attribute-Based Access Control

This usage-based access control fits well the distributed and context of the SMAC-IT systems whose perimeter is unknown, provided that usages can be defined precisely. Different languages have been defined to describe these usages [20] and rights [23]. Some of them include organizational knowledge [25] or roles [26]. Thanks to formal models, policies can be expressed more efficiently (see [18] or [27]) before being integrated into applications [19].

Whereas these access-control-based strategies have been designed for well-perimeterized information systems, they provide only reduced protection for dynamic and opened information systems as undue usages can be a threat. Moreover, specific security vulnerabilities and threats are related to the Social, Mobile, Analytics, Cloud, and Internet of Things technologies.

### 2.1.2 SMAC-IT security Challenges

The explosion of multimedia big data in mobile and cloud computing has created unprecedented opportunities and fundamental security and privacy challenges as they are not just big in volume, but also unstructured and multi-models[28]. In what follows, we review the main security challenges associated with the integration of these technologies.

Social media, mobile and analytic concentrate on the service interaction, which generates value, while cloud computing is the foundation for these services providing IT implementation with the applications. There are numerous and diverse stakeholders who have different purposes participating in the services[29]. Social media can be associated to share, get credit and reuse each other's data and interpretations. Mobile can be associated to support user-service interactions. Analytic provides data analysis and processing services such as reusable workflows, and it also can support mining, integrating, and analyzing new and existing data to advance discovery.

Social media [30] provides supports to establish the value network that contributes to the establishment of relationships among the users. Data sharing and data archival are the main activities for social media. Social media are popular vectors of attacks leading to identity theft that includes profile cloning, profile porting which can lead to Sybil attacks. Other specific threats can be associated with social media such as unauthorized content sharing, content tagging, content publishing, dispatching/shared ownership of contents...

Mobile [31]allows "free and fluent access" from everywhere using various devices. Improving the quality of service in such a highly distributed environment relies on an adapted infrastructure allowing to upload or download data as close as possible to the user, expanding the Information System to "Content Delivery Network" services. Mobile applications can also provide context information related to location or time that implicitly reveals the privacy of the users. This mobile context increases threats related to data transmission due to wireless connections.

Analytics can be considered according to both a technological vision [32]and to a business /organizational vision [32]The technological part classifies the tools as statistical methods, machine learning, and knowledge-based methods. Each of them requires different usages on data and has different protection requirements. Statistical methods focus on the data association to collect explicit information, machine learning is concentrated on data deduction to generate new information, and knowledge-based



methods integrate data induction to reveal implicit information. So, analytic can be defined as the support for the data generation whereas the analytic process is controlled according to business perspectives (political, economic, sociological, regional...) that may affect data usage.

Cloud computing is a model for enabling convenient, on-demand network access, to a shared pool of configurable computing resources (eg., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [34]. NIST's cloud computing studies [35][36]classifies security and privacy policies under the purview of the cloud provider. [37]gives the context of cloud computing which can be classified into SaaS, PaaS, and IaaS. The Jericho Forum's security model [38] uses four dimensions to capture cloud security:

- Internal/External defines the physical location of the data,
- Proprietary/Open defines the state of ownership of cloud technology, services, interfaces, etc.
- Perimeterised/De-perimeterised architectures represent whether it operates within traditional IT perimeter.
- Insourced/Outsourced defines whether the service is under the control.

Last but not least, [39] and [40] illustrate the vulnerabilities, threats, and the relationships between threats, vulnerabilities, and countermeasures. [41] illustrates not only the vulnerabilities, threats in the IT part but also consider the organizational part. [42] starts to combine data aspect and context aspect and gives various requirement types in a different layer of cloud computing. [43] gives the state of the data in cloud computing which can be aggregated with the lifecycle of the data. To summarize this review, Figure 1 presents the cloud security stack.

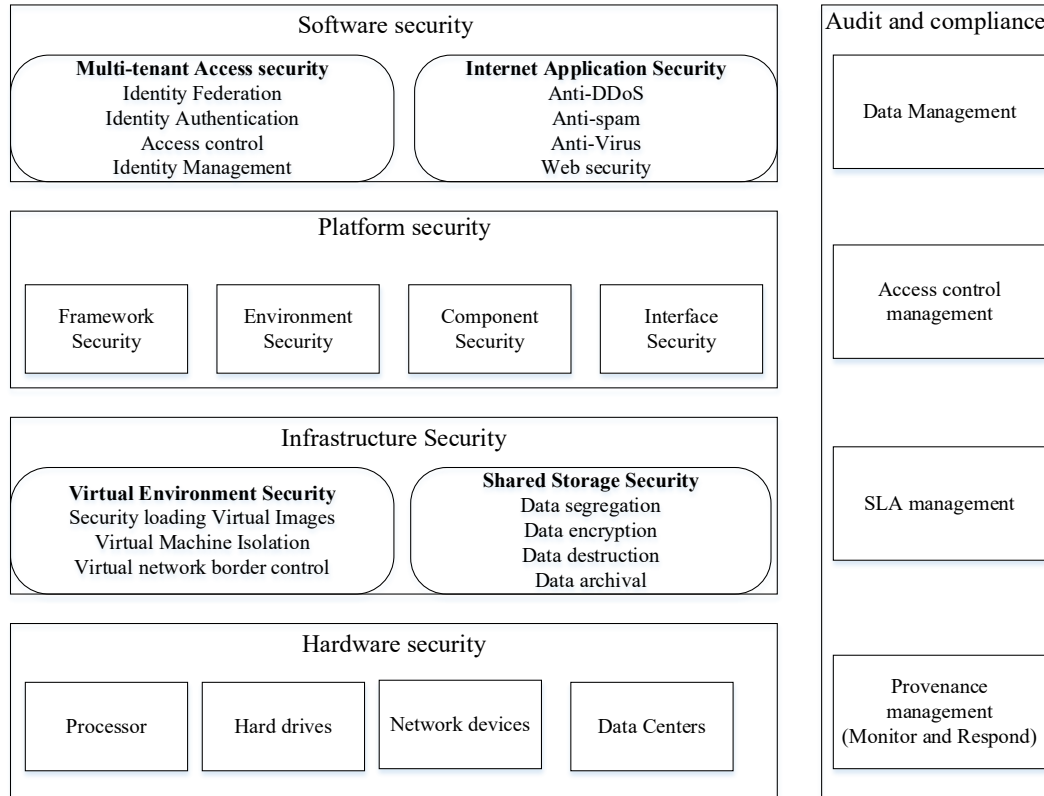


Figure 1 Cloud computing security stack

The privacy intrusion may lie in the domains of the social web, consumer and business analytics and governmental surveillance where the information gleaned can be used for nefarious purposes. [44] mentioned a sociological problem when private data is collected and mined by companies and proposed the requirement that users should stay informed about their personally relevant part which is publicly available on the social web. [45] also provides a major review of some privacy preservation mechanisms in big data and illustrates that privacy protection in the big data needs specification of privacy policies and integration of the enforcement monitors into the target analytics platforms. [46] is based on general data protection regulation to emphasize that the requirement of enhancing transparency and trust in co-controllership which means implementing appropriate controls to limit access to data. This work pays particular attention to statistics-related operation by integrating the stock of the effects of (potential) correlation. To manage such protection in a Big Data context, it also provides anonymization, encryption, and accountability controls features. As far as Cloud technology is concerned, [47] points out security and privacy challenges in multi-parties environment, especially tackling the XaaS outsourcing challenges, dynamic virtualization management, and multi-tenant shared operation, etc

[48] focuses on the security and privacy in mobile cloud computing which mentions the characteristics of mobile and associated issues. According to this work, the security requirements in mobility focuses on providing reliable information transmissions against malware, software vulnerabilities, and anomalous behavior of users.

Considering Analytics risks involves paying attention to Big Data processes and security challenges [49]. [50] defines the mining process into massive amounts of data to reveal hidden patterns and secret correlations named as big data analytics. It is characterized by the five V characteristics: variety, velocity, volume, veracity, and value. [51] mentions that it is not only difficult to store big data and analyze them with traditional applications, but also that it challenges privacy and security. [52] gives that a big data-driven security model should have the following characteristics:

- Focusing on data, it has to manage multiple data sources, multiple data types, large amounts, and fast-changing.
- For the usage, it will be analytic.
- For the IT support part, it provides cloud computing with N-tier infrastructure and a centralized warehouse for storage.
- For the management part, it considers advanced monitoring systems, active controls, standardized views, and a high degree of integration via security and risk management tools.

Although Big Data can be used to improve security and privacy by developing fraud detection systems and anomaly-based intrusion detection systems (IDSs)[53] it often leads to data related security and privacy troubles depending on data provenance. To identify the Big Data-related threats and vulnerabilities particular attention must be paid to the Big Data processes and their support tools. To this end, [54] defines a Big Data system according to different stages, including data acquisition and preparation relying on data cleaning and data integration processes, data analytic and mining, data visualization, and data interpretation. These different processes use various support technologies such as GFS [55] or HDFS [56] for the storage, MapReduce[57], etc. Following these stages, risks may affect privacy during the whole life cycle (collection, analytics, and publication), authenticity (related to data falsification, distortion), and access control as it may be difficult to manage roles and proper authorization while classifying data. To mitigate these risks, protection countermeasures are developed mostly regarding privacy violation risks such as location privacy protection, anonymous identifier protection, etc. [58] details risk in the knowledge discovery phase which includes data environment, analytic processing, and prediction for both IT-related risks and “organization” related risks. [59] defines five major countermeasure categories dealing with big data security challenges: anonymization, encryption, access control, and monitoring, policy, and governance frameworks. [60] also analyzes the security problems of big data and proposes protection strategies for big data security and privacy. It focuses on the organizational protection related to social networks.

Focusing on data sources [44] considers the social media and mobile systems as key parts of Big Data security. It stands at the user’s perspective and considers the awareness of the personally relevant part which is publicly available on the social web. It discusses how the growing proliferation and capabilities of mobile devices are creating a deluge of social media information and interactions which can affect privacy. It proposes an overview of protection means, including access control and metadata handling. It also points out two main privacy issues: homegrown problems are due to the user's own inconsistent behavior leading to a lack of protection damaging its own privacy whereas

external threats may be involved by other social media users who may share personal information in an unauthorized way.

[45] defines security and privacy challenges in a Big Data context. Information privacy is the privilege to have some control over how personal information is collected and used. Information privacy is the capacity of an individual or group to stop information about themselves from becoming known to other people than those they give explicitly the information to. Security is the practice of defending information and information assets through the use of technology, processes, and training from Unauthorized access, Disclosure, Disruption, Modification, Inspection, Recording, and Destruction. To mitigate these risks, privacy-preserving methods include de-identification, namely K-anonymity, L-diversity, and T-closeness with identifier attributes, quasi-identifier attributes, sensitive attributes, insensitive attributes, and equivalence classes. K-anonymity uses two regular techniques: suppression and generalization. The L-diversity model (Distinct, Entropy, Recursive) is an extension of the k-anonymity model which diminishes the granularity of data representation utilizing methods including generalization and suppression in a way that any given record maps onto at least k different records in the data. Furthermore, T-closeness is an improvement of l-diversity group-based anonymization that is used to preserve privacy in data sets by decreasing the granularity of a data representation. It also gives the comparative analysis and limitations of these privacy methods (see Table 1).

Methods	Definition	Limitations
<b>K-anonymity</b>	It is a framework for constructing and evaluating algorithms and systems that release information such that released information limits what can be revealed about the properties of entities that are to be protected	Homogeneity-attack, background knowledge
L-diversity	An equivalence class is said to have L-diversity if there are at least “well-represented” values for the sensitive attribute. A table is said to have L-diversity if every equivalence class of the table has L-diversity	L-diversity may be difficult and unnecessary to achieve and L-diversity is insufficient to prevent attribute disclosure
<b>T-closeness</b>	An equivalence class is said to have T-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness	T-closeness requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of a sensitive attribute in the overall table

*Table 1 Privacy-preserving techniques*

[62]also summarizes the protection of big data privacy which mentions the issues in the whole lifecycle of big data with cloud computing infrastructure. Meanwhile, it gives the associated protection approaches in data generation, processing, and storing phases considering security services. Lastly, [63] proposes a data-driven privacy control system based on CUCON (Collaborative Usage Control), which also considers big data techniques with business processes and tries to manage the consistency between already allowed rights and the potential given rights based on the effect of the business process's activities.

Focusing on Big Data support infrastructure, [61] considers the security and privacy involving cloud computing infrastructure in a Big Data context. This work defines different threats related to big data applications and the supporting architecture, including the network level, authentication level, data level, and generic types. It also provides different countermeasures associated with the different layers of cloud computing, such as file encryption, network encryption, logging, etc to mitigate these risks.

### 2.1.3 *Risks, vulnerabilities, and countermeasures synthesis*

To sum up, the introduction of SMAC-IT technologies leads to new technological and usage-related vulnerabilities and threats. The next table provides a synthesis linking data vulnerabilities, threatening agents, risks, and associated countermeasures. This shows that similarly to traditional IS protection, dedicated protection policies must be set to adapt the countermeasure deployment to the processing context.

Implementation-related Vulnerability		Data Vulnerability	Risk	Threatening agent		Countermeasure
Service interaction (Social, Mobile, Analytics)	Infrastructure configuration (Cloud computing)			Internal	External	
Inadequate security awareness or regulations for the social interaction	Insecure web-service and APIs	Data coarse-grained classification	Data leakage	Expose the data	Intrude the data	1. Access control 2. Security Awareness training 3. Data segregation 4. Detailed regulation
No boundary of the social interactions with multiple tenants	Multi-tenancy with logical segregation	Data coarse-grained classification	Data leakage Data un-availability	Compromise or overuse the application or Intrude the data with Malicious insider	NA	1. Strong isolation 2. Access control 3. Data distorting anonymization
Mobile communication at the public wireless communication	Insufficient Network protocol	Data coarse-grained classification Opaque data audit for derivation (lineage with composition)	Data leakage Data modification Data un-availability	NA	Man-in-middle attack IP spoofing ARP spoofing DNS poisoning RIP attack ARP poisoning Flooding DDoS (DoS) Cloud account Hijacking	Data encryption Data encapsulation Data distorting anonymization Wired communication Digital signature Timestamp Session management with Multi-factor certification
Random download and install the code under the mobile environment	The software has infrequent monitor, update, and patches	Opaque data audit for derivation (lineage with composition)	Data leakage Data modification Data un-availability	Misuse or misappropriate configure/install the application	Compromise the application software or install malicious code	attack detection application Regular update and patch the software Restrict the right to download or install the code

Implementation-related Vulnerability		Data Vulnerability	Risk	Threatening agent		Countermeasure
Service interaction (Social, Mobile, Analytics)	Infrastructure configuration (Cloud computing)			Internal	External	
No boundary of the Social interactions Multiple logins with the same account in the mobile Give rights of the analytics	loss of governance with insufficient audit and log	Opaque data audit for derivation (lineage with composition) Data romance	Data leakage	Expose data with no constraints	Recover data Analytic data	Restrict the usage of analytic Order others' usage at the specified location and time Set up specified social location and time for data exposure and withdraw instantly
Loss of organizational management	Insufficient physical protection	Data without backup or replication	Data unavailability	Damage data	Damage data	Replicate data Store the data in a location that has perfect protection with laws and regulations

*Table 2: Synthesis of the different vulnerabilities, threats, and countermeasures*

Note that N/A means not applicable

#### 2.1.4 Conclusion.

Focusing on IS protection, previous sections provide a rich background of risk assessment in information systems and usage protection development which can be used to support data-centered protection research. This review refers to traditional solutions to extend them to the SMAC-IT context, identifying security requirements and protection policy for data and process assets in open environments.

Focusing on data-centered protection, lack of security awareness leads to mis-identify and misclassify sensitive data as non-sensitive data. This trend is reinforced as different copies of the same data can be considered as different assets that are protected separately. Moreover, when audit operations or data origin checking cannot occur, this can be seen as an attack on data consistency, leading to damages that can affect the data authenticity and even its existence.

Paying attention to privacy, new requirements can be defined[29] such as

- Data un-linkability considers the data structure that is associated with data generation, data archival, and data sharing.
- Data usage transparency refers to data monitoring to control data transfer, usage, sharing, storage, and destruction.
- Data intervenability extends data governance i.e. it allows the data owner to intervene on the consumer side to manage its privacy requirements. “Intervenability” relies on SLA-based governance and need to consider the consequences of the whole lifecycle of the data

These privacy requirements, which can be combined with traditional security services, call for pre-protection, ongoing protection, and post-protection of data. Based on this analysis more complex data security/privacy services can be defined by composing basic security services:

- Data existence: on the data consumer side, it refers to data availability and the rights to keep it or delete it locally whereas on the data owner/ data producer side it refers to the way the data is governed to decide to withdraw it or not leading to intervene on the consumer side to control its deletion.
- Data authenticity: on the data consumer side, this service is associated with integrity and is associated with the right to modify (or write) the data whereas on the data owner/data provider side this will refer to the data transparency, i.e. providing audit and origin certification services.

Different vulnerabilities and threats can be considered depending on the data life-cycle status. To classify them, we use the data lifecycle steps defined in[37][30], i.e. generation, transfer, usage, sharing, storage, archival, destruction. (see Table 3).



<b>Lifecycle operation</b>	<b>Data vulnerability</b>	<b>Data threats</b>
Data generation	Data identification and segregation	Data breach
Data transfer	Data encapsulation	Data modification, breach
Data usage	Data audit and log evidence	Data breach, modification
Data sharing	Data authorization	Data protection inconsistency
Data storage	Data replication	Data loss
Data archival	Data locality	Data breach, unretrieved data
Data destruction	Data persistency	Data breach

*Table 3 Data centered vulnerability and threats*

Once evaluated, these security and privacy requirements must be managed by defining convenient security policies.

## **2.2 Security policy**

A security policy is a definition of secured behaviors for stand-alone or collaborative business organizations or IT architectures. A security policy consists of a set of rules specified using abstract vocabularies to give some constraint or condition on the usage or deployment of an entity owned by a participant [65]. The policy has three aspects: the policy assertion, the policy owner, and policy enforcement. Policy assertions refer to the way the security services are implemented. The service consumer independently of any agreement with the policy owner may assert them. Policy enforcement<sup>2</sup> ensures that the policy assertion is consistent with the real world. The OASIS [64] also extends the security policy by defining a security model relating the IT architecture with the human/social organization. Based on these social structures, legitimate permissions, obligations, and roles are generated for people concerning the IT system and the security mechanisms. This model characterizes security in 6 key concepts: confidentiality, integrity, authentication, authorization, non-repudiation, and availability and deploys the SOA systems in terms of three primary layers which are network layer, transport layer, and application layer. By assessing threats in these different layers, the security model can be used to specify the necessary security response mechanisms.

To this end, [66] uses EBIOS methods to generate a security policy which first needs to identify security goals and context and determine the security requirements with the risk assessment. [67] introduces a tool to simulate and analyze the security policy specified using the OrBAC model. OrBAC aims at modeling a security policy centered on the organization which provides specified policy at an organizational level and enforced policy to concrete it. There are also multiple types of research related to the model-driven security policy. Most of them confirm the common criteria approach [68] which depicts the relationship between security requirements and the TOE (Target of Evaluation). Their approach is based on refinement of the security requirements into a TOE summary

specification expressed in the security target and each lower level of refinement represents a design decomposition with additional design detail. In this way, the policy is defined with abstraction levels of functional specification, high-level design, low-level design, and implementation considering the security environment and security objects with threats and organizational policies.

Different policy languages can be used to support usage descriptions. [20] presents the Obligation Specification Language (OSL) for a wide range of usage control requirements. It also presents translation between OSL and two rights expression languages (RELS) [23] from the DRM area [24]. It classifies the policies into obligations and provisions. [25] proposes a formal technique that combines the use of access control policies expressed in the OrBAC language together with specifications based on the B-method. It models language for system specification and also gives a formal expression of security properties modeling the relationships between the security policy and system. [26] provides a common means of specifying security policies that map onto various access control implementation mechanisms. The language focuses on roles to group policies relating to a position in an organization, relationships to define interactions between roles, and management structures to define a configuration of roles and relationships of an organizational unit. [18] provides a two-level usage control policy. It first proposes specification-level usage control policies which is a temporal logic with explicit operators for cardinality and permissions. Then it gives implementation level policies which can be expressed as event-condition-action (ECA) rules. [27] developed a logical specification of the UCON<sub>ABC</sub> model with a temporal logic of actions (TLA) with a sequence of states expressed by system attributes. [19] especially gives the usage control policy in web applications. Here is a brief description of the basic usage control model by illustrating UCON policies: UCON model [69] has three parts: Authorization Models, Obligation Model, and Condition Model. In the authorization model, user credentials and resource attributes are checked before granting permission for the requested resource. Obligation models comprise the obligation monitoring of access requests. The condition model relates to the system infrastructure, environment, and business session. All these aspects should consider the attributes of subject and object to give the rights. It identifies three types of subjects: consumer, provider, and Identifier, Consumers are the subjects who request to perform a certain action on an object. Providers are the individuals who own services and issue the rights to the requesting party. The identifier is the entity whose confidential information is incorporated within a digital object. While, three types of rights (actions) are specified namely consumer, provider, and identifies rights which indicates the set of actions or privileges on digital objects. UCON model also classifies the objects as privacy sensitive and privacy non-sensitive objects that determine whether the object contains critical information of identifiers, subject or not. The basic usage control policy will describe the syntax, semantic, and vocabulary for describing variable attributes related to a complex system. The policy model defines the policy grammar and semantic, and the vocabulary used to describe domain knowledge which bridges grammar with the associated domain.

### 2.2.1 Security policy ontologies

There have been many works that apply ontology technology to security policy systems to capture domain knowledge. The ontology allows the construction of domain

knowledge by defining the relation between concepts. The knowledge represented with ontology can be used to reason about entities within that domain and find new knowledge, thus describes the domain. The Web Ontology Language (OWL)[70][71] is a widely adopted knowledge presentation language for ontology construction that presents concepts in a structured way, defining the Class-Subclass relation. Ontologies provide a formal specification of concepts and their interrelationships and play an essential role in complex web service environments, semantics-based policy model.

[72] presents POLICYTAB (see Figure 2), a policy inheritance and composition framework based on credential ontologies, formalizes these representations and the related constraints in Frame-Logic. The ontology shares information about credentials and their attributes, which is needed for establishing trust between negotiating parties.

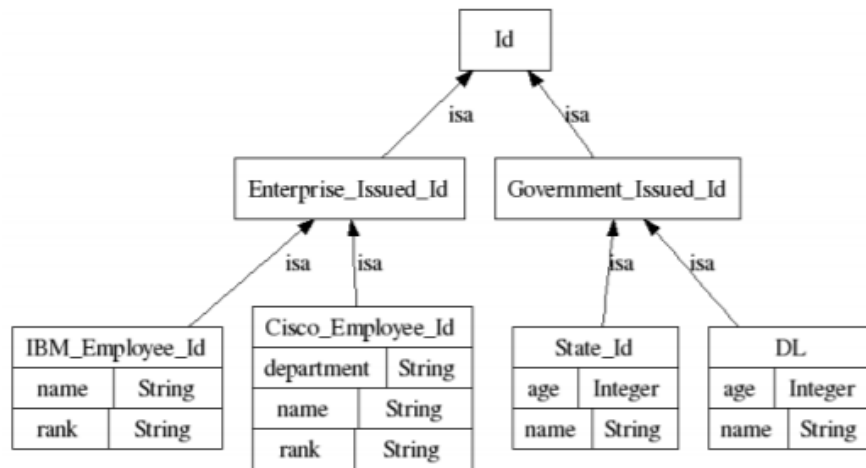


Figure 2 UML class diagram for Simple ID Credential Ontology [72]

[73] provides ontology-based access control model (Onto-ACM) approach which offers a mechanism for securing applications and systems considering the conditions based on context-aware technologies in the cloud computing environment and applies the access level of resource access based on ontology reasoning and semantic analysis method. It defines the context information ontology of the user and the administrator by using OWL based on the ontology class, including basic information, the resource time, and the terminal, among others (see Figure 3).

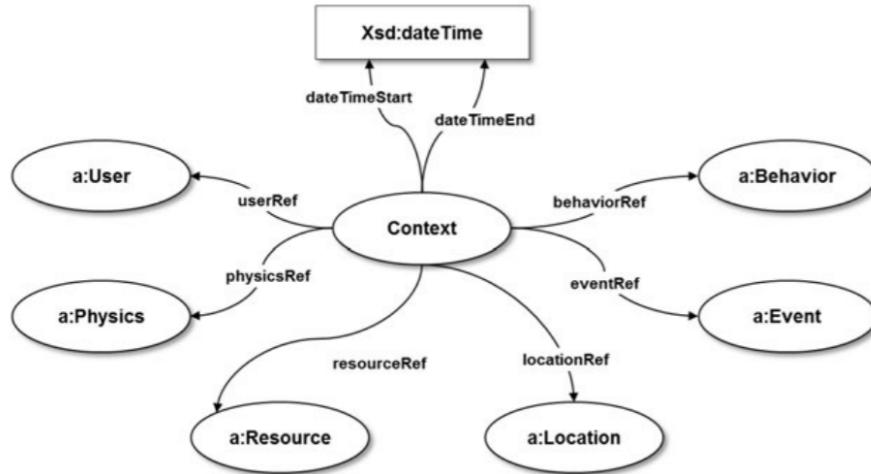


Figure 3 VOWL<sup>11</sup> diagram of the access control ontology [73]

[74] extends the WS-Policy to represent QoS policies for selecting adequate services to meet service consumer needs which apply ontological concepts to WS-policy in order to enable semantic matching.

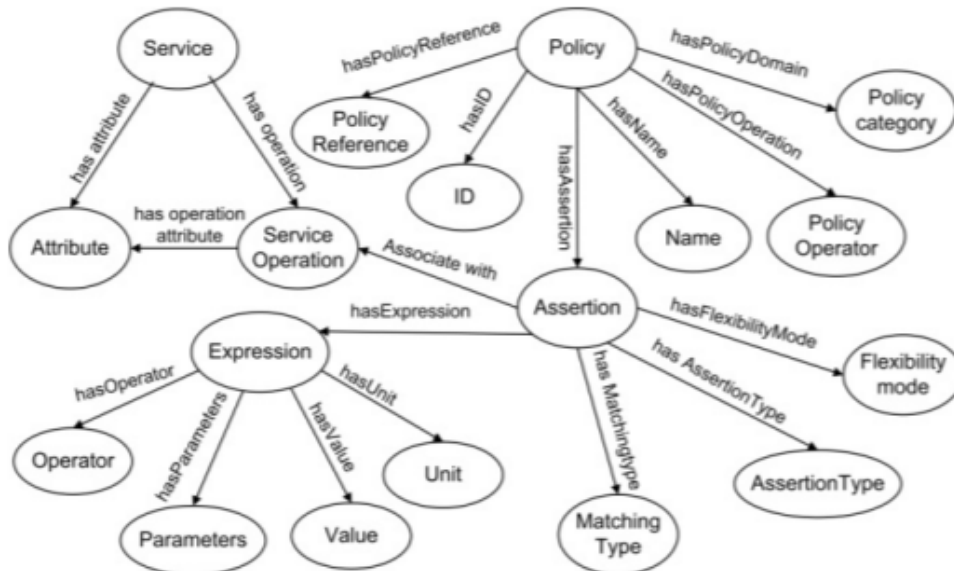


Figure 4 VOWL diagram of the WS-policy ontology [74]

However, the OWL reasoners applied to this WS policy ontology are mainly used for their class / subclass deduction capabilities [73]. This limit can be overcome with SWRL [75]. The Semantic Web Rule Language (SWRL) combines sub languages of OWL with those of the Rule Markup Language. It takes advantage of both the ontology and the rule-based knowledge to draw inference which can add new facts to the knowledge base [76]. [77] combines the OWL with the RBAC model for access control. It means that OWL

<sup>11</sup> See <http://vowl.visualdataweb.org/v2/>

constructions can be extended to model attribute-based RBAC or more generally attribute-based access control. This ontology provides the basic RBAC concepts including subjects, objects, roles, role assignments, and actions. It defines the class with actions, subjects, and objects. To control access, it introduces two important Action subclasses for permitted and prohibited actions: PermittedAction and ProhibitedAction. The RBAC ontology defines some key properties that a subject or object can have (depending on the details of the representation) and leaves the specification of additional properties and subclasses to the specific domain model. The Cloud Service Access Control (CSAC) mechanism [78] considers payment status and service level as the two essential characteristics of cloud service. Ontological terms are used to represent the user's payment status and access control policies and provide necessary semantic information to execute policy conflict analysis and access denying rules.

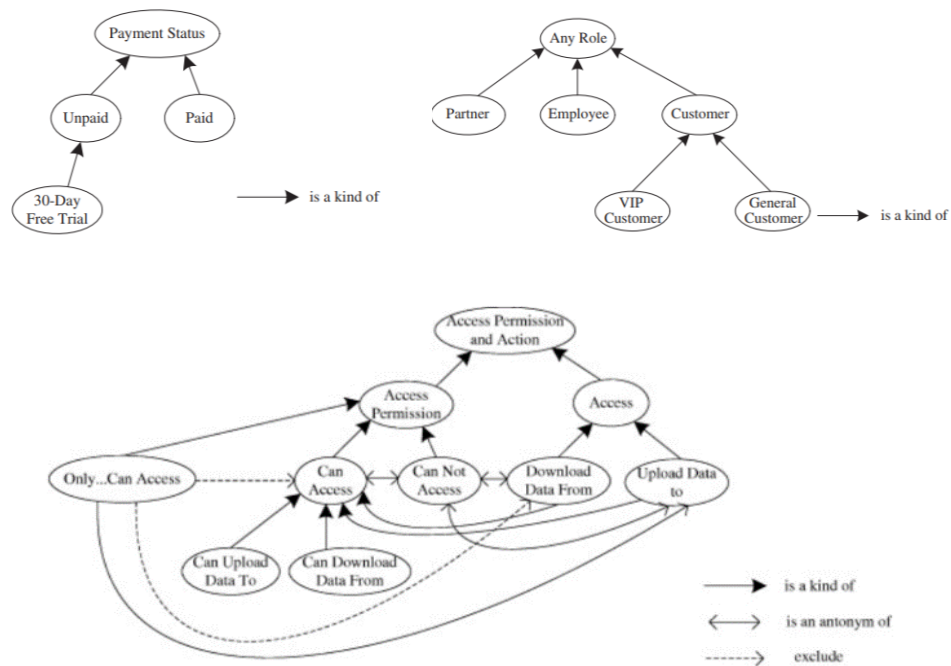


Figure 5 VOWL diagram of the Cloud access control ontology [78]

[79] considers privacy protection policies which give three types of ontology in the Description Logic (DL), log-based ontologies, and rules combination for the semantic enforcement. It includes data user ontology, data type ontology, and purpose ontology

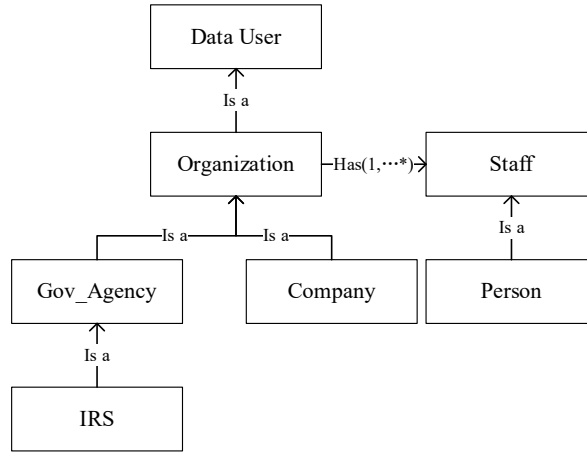


Figure 6 UML class diagram of the data user ontology [79]

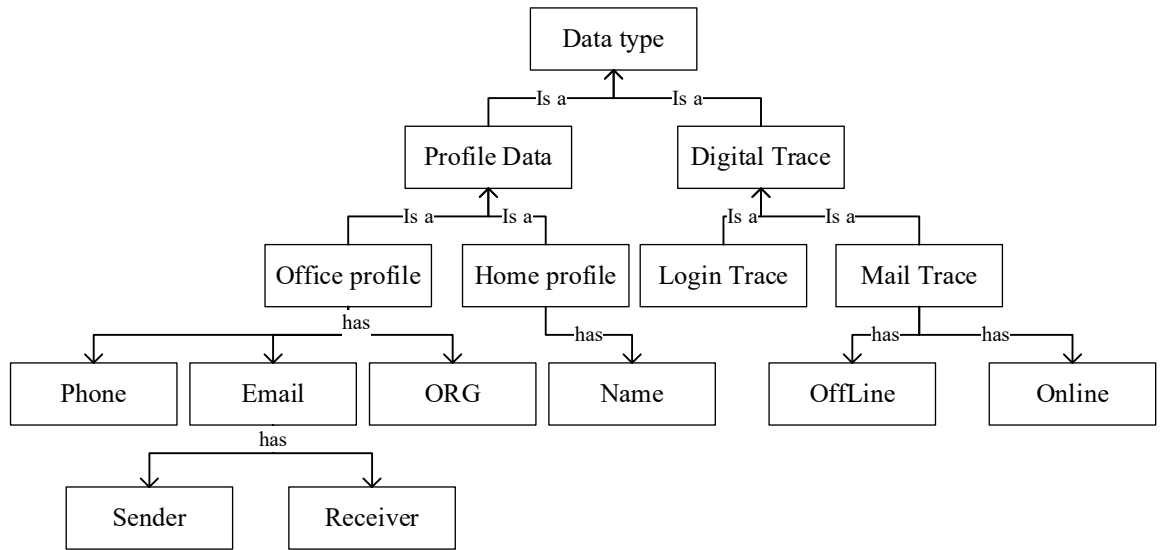


Figure 7 UML class diagram of the personal profile and digital traces from [79]

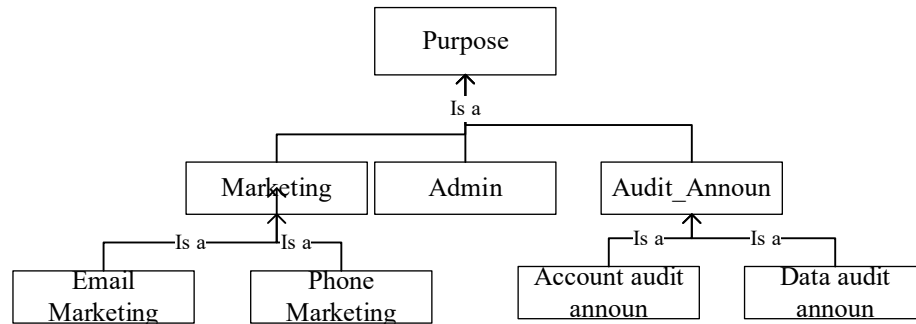


Figure 8 UML class diagram of the purpose ontology picked from [79]

[80] defines the role definition using ontology information, and the transformation operations in ontology trees which aims at providing an appropriate policy with an exact role for every tenant.

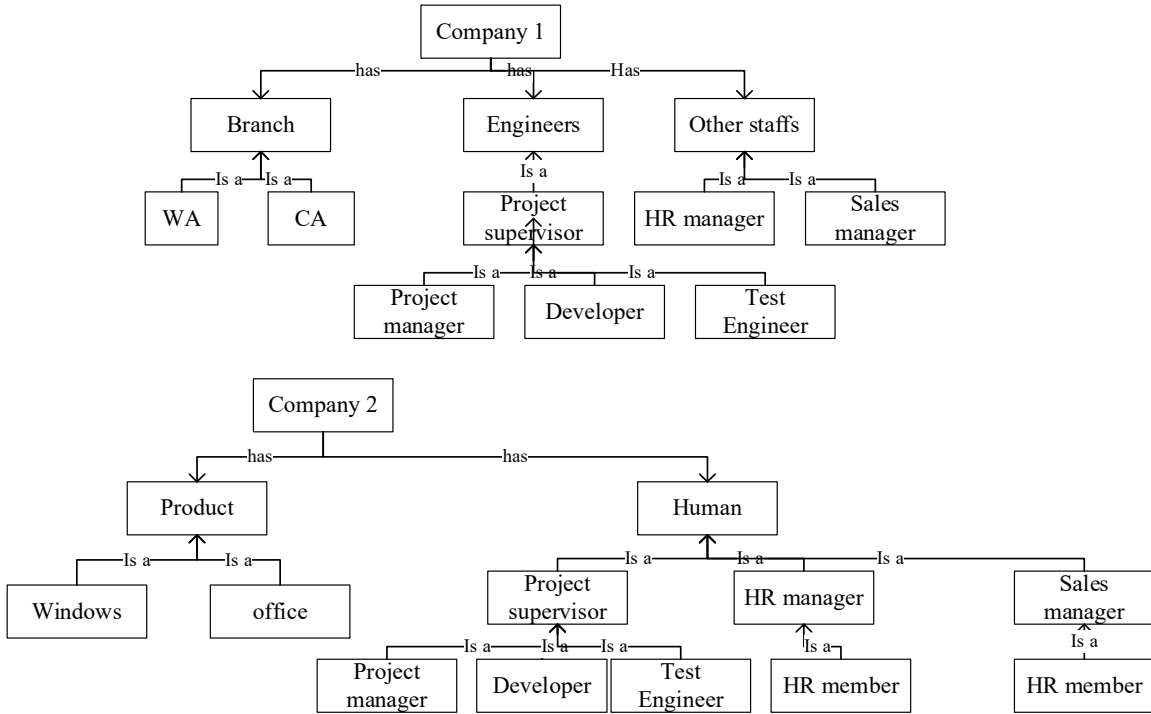


Figure 9 UML class diagram of a sample ontology tree companies [80]

The model proposed by [80] enables expressing much more fine-grained access control policies on social network knowledge. It relies on a Social Networking systems Ontology (SNO) that models key entities and their relationships typically found in Social Networking Systems (SNSs). The Entity concept is the root of all concepts in SNO, with three immediate descendants: DigitalObject, Person, and Event. The DigitalObject concept models any object with digital, typically visualizable content. The Person concept models human users in the context of SNSs.

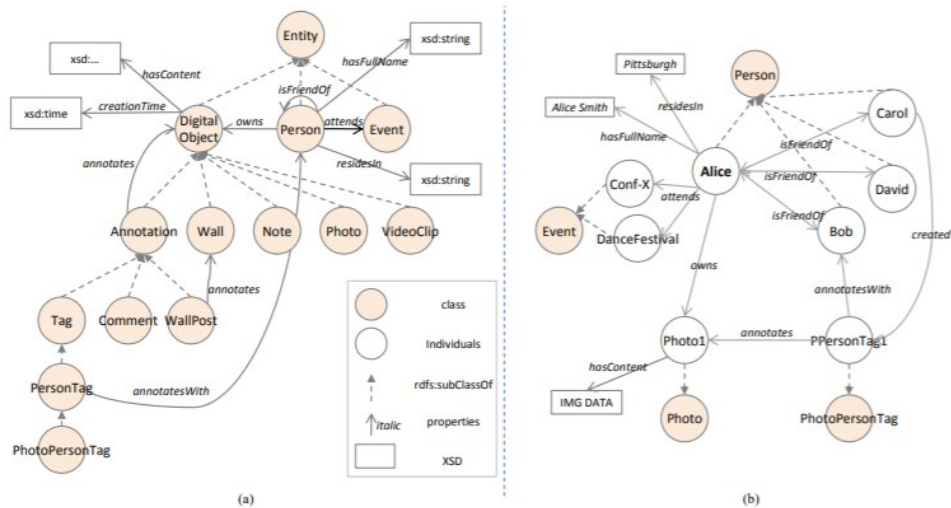


Figure 10 VOWL diagram of the social networking system ontology[81]

[82] gives an approach that formalizes ODRL (Open Digital Rights Language) semantics based on semantic web ontologies. The resulting ODRL ontologies have been connected to IPRonto and provide Digital Right Management to the Internet.

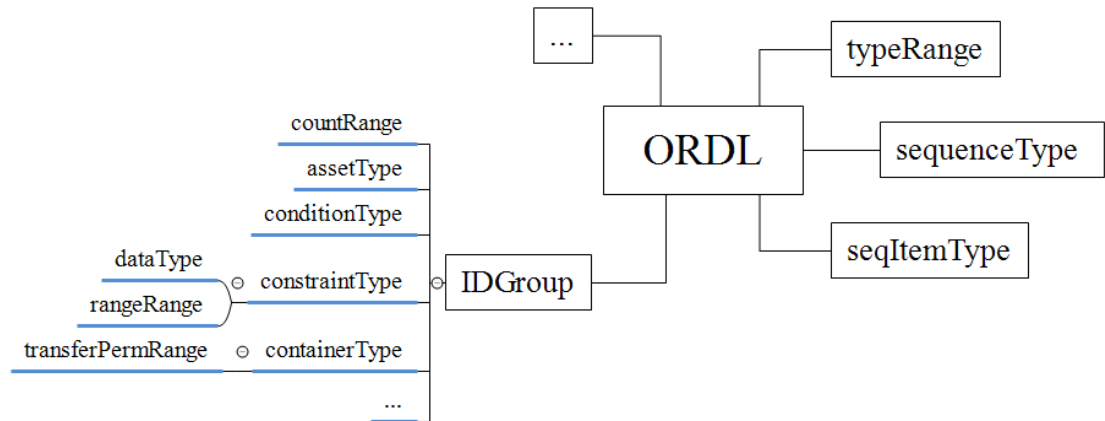


Figure 11 Mind map of the ODRL XML complexTypes [82]

XML Schema	OWL	Shared informal semantics
element attribute	owl:DatatypeProperty  ObjectProperty	Named relation between nodes or nodes and values
element @substitutionGroup	rdfs:subPropertyOf	Relation can appear in place of a more general one
element@type	rdfs:range	The relation range kind
complexType  group attributeGroup	owl:Class	Relations and contextual restrictions package
complexType// element	owl:Restriction	Contextualised restriction of a relation
extension restriction @base	rdfs:subClassOf	Package concretises the base package
@maxOccurs  @minOccurs	owl:maxCardinality  minCardinality	Restrict the number of occurrences of a relation
sequence choice	owl:intersectionOf  unionOf	Combination of relations in a context

Table 4 XSD2OWL translations for the XML picked from [82](P3-4, Table 2)

[83] proposes an Ontology-based Rights Expression Language, called OREL which allows not only users but also machines to handle digital rights at a semantics level. It uses the terms “Act”, “Agent”, “Resource”, “Time” and “Place” from MPEG RDD (Rights Data Dictionary) as well as their hierarchy and relationships to build this ontology.



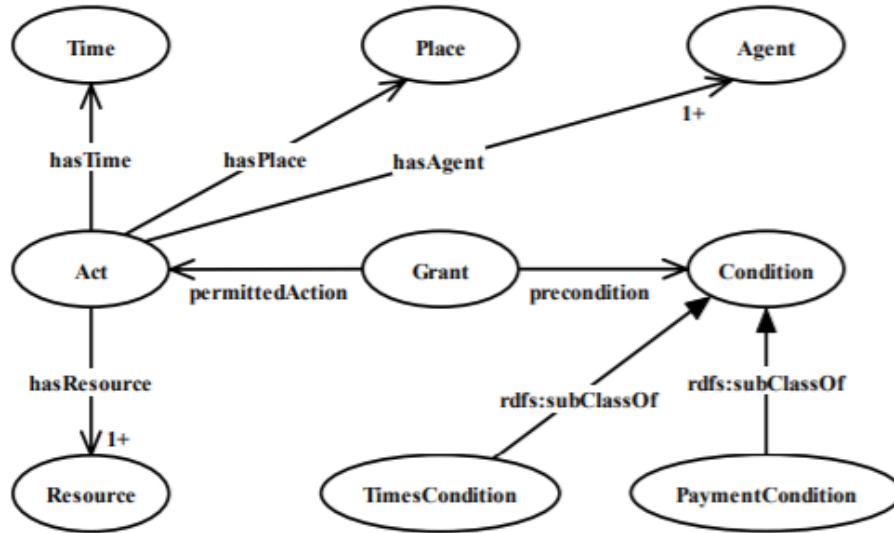


Figure 12 VOWL diagram of the OREL ontology [83]

[84] presents a base privacy ontology for e-services and a privacy framework for Service-Oriented Architecture (SOA). The ontology offers a base vocabulary that can be extended to create ontologies specific to a given service domain and operating environment.

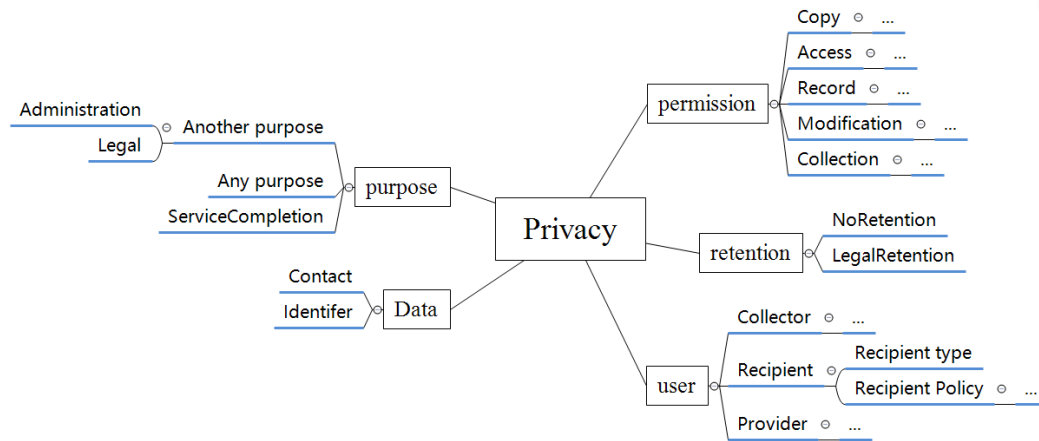


Figure 13 Mind map of the basic privacy ontology [84]

Focusing on data protection and usage control, [85] formalizes policies as a set of assertions defining rights, obligations and conditions. These conditions are focused on different attributes related to the subject (SAT) who will get the right, the object on which the right is granted (OAT) and context (CNAT) (see Figure 14 presenting the policy assertions and Figure 15 presenting the usage control policy model). This work defines both usage actions and organizational regulation so that both “technical” and “organizational” security countermeasures can be integrated.

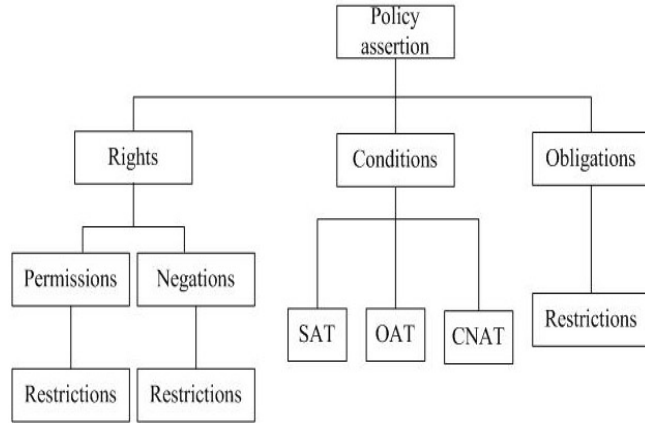


Figure 14 Policy assertion model based on [85]

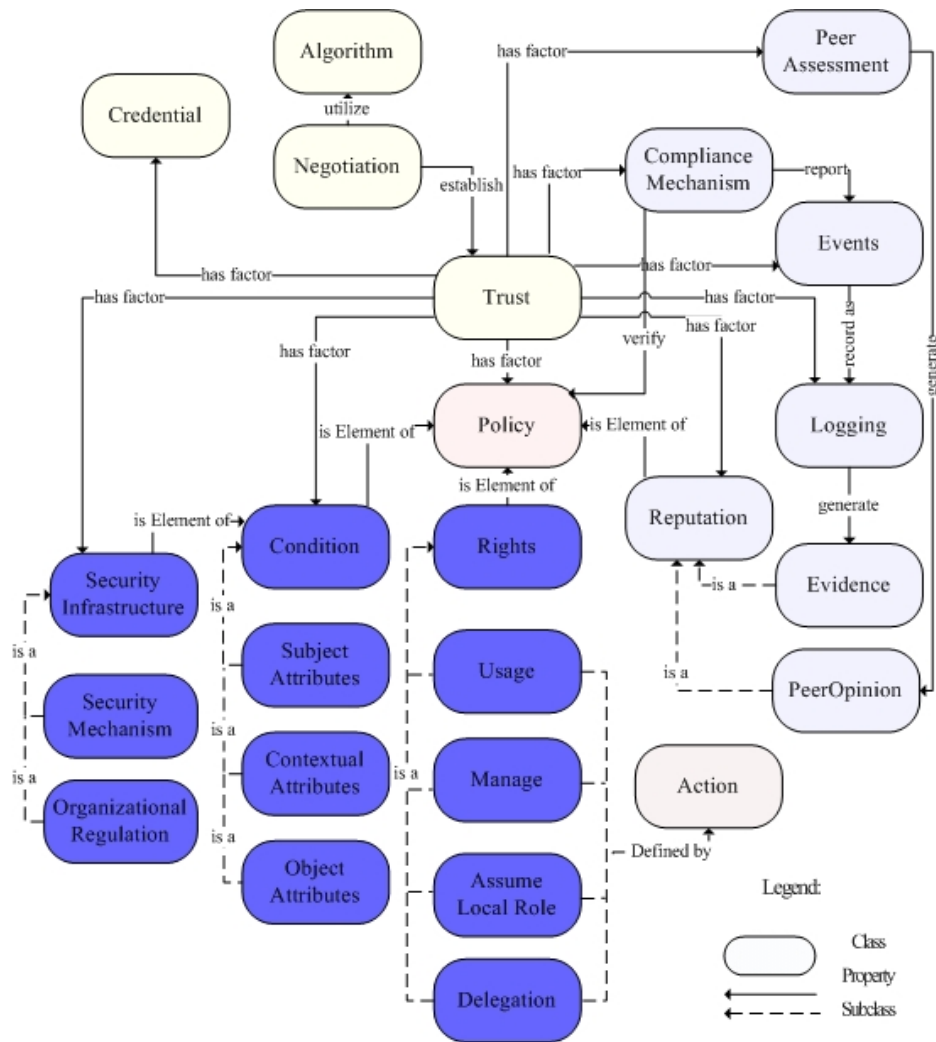


Figure 15 VOWL diagram of the Usage Control Policy model [85]

This brief state of the art shows that a large variety of ontologies are defined to describe policies, access control rules, protection means and even organizations involved in access control / usage rules. Most of them are devoted to access control, integrating different protected resources (data or services) and knowledge on the subject, i.e. the entity who will get the right. Different attributes are used to define rights, obligations and conditions. We have identified different criteria to compare these works (see Table 5)

1. **Control purpose:** defines for what the policy is designed: it may be service selection, access control...
2. **Control object:** defines the attributes which are used to describe 'Rights', 'Obligation' and 'Condition'. It usually has the range shown as the quantity or quality. These attributes may deal with operation conditions (including time, location...), the knowledge a subject has on a stakeholder, the trust and reputation level of a subject, the intended usage qualification, the quality of service (including the quality of protection) or attributes related to the object itself.
3. **Subject Attributes:** defines the attributes of the party requiring the access (i.e. the subject) used in the access control process. It may be a role, a group identification (such as a friend, stranger, or acquaintance usually used in social media)
4. **Object Attributes:** defines the attributes of the asset that is protected by the policy (i.e. the object) that are conditioned the access. It can be the type, content, or an operation service
5. **Policy extension:** defines if the policy is extended from some existing ones or regulation constraints
6. **IT implementation:** defines the specific characteristics of the IT implementation which may generate extra risks and usage, such as risk generated by the multi-tenant in the cloud computing architecture or the risk generated by the web service.
7. **Environment:** characterizes environment which has specific IT implementation and gives extra information for the control object.

Ref.	Control purpose	Control object	Subject Attribute	Object Attribute	Policy extension	IT implementation	Environment
[72]	Access control	Trust	Role	not available	Organizational and governmental regulation	Web Service	Social media
[73]	Access control	Operation Condition	Role	Resource	Permission level including cloud service context	not available	Cloud
[74]	Service Selection	Quality of Service	Service attributes	Policy	Policy definition and enforcement	Web Service	not available
[78]	Access control	Trust + Service	Role and Reputation	Service level	Access permission	not available	Cloud
[79]	Access control	Purpose	Role	Data type	P3P and EPAL-based privacy protection policies	Web Service	not available
[80]	Access Control	Purpose + Service	Role	not available	RBAC	Multi-Tenancy Architecture	not available
[81]	Access control	Knowledge	Roles and Social Relationship	Digital Object	Access permission	not available	Social media
[82]	Access control	Object	not available	Data type	ODRL	Web Service	not available
[83]	Access control	Operation Condition	Role	Resource	REL	not available	not available
[84]	Service selection	Quality of Service	Role	Data type	Contextual policy definition	Web Service	not available

*Table 5 Comparison of the different ontologies*

Based on this review, security and privacy management in SMAC-IT systems call for providing consistent protection to data assets as these assets can be shared by different processes and information systems. To manage the “organizational” protection, Business Process models must also be considered. [86] BPMN extended meta-model [87] and BPDM [88] These Business Process models allow specifying business processes as set of tasks, defining events and flows to connect them. Organizational knowledge is addressed by defining parties and roles related to the tasks. Nevertheless, none of these models integrate a precise specification of usages on data resources. This calls for integrating Usage control models and the Business Process models to support a consistent and contextualized usage definition to protect efficiently data resources.

Paying particular attention to the Big Data and the Data Science context, legal regulation is also needed to define data owner and data consumer rights and duties so that data privacy and security can be managed.

## 2.2.2 *GDPR-based asset protection*

As stated previously, SMAC-IT generates new risks leading to privacy violations. Paying particular attention to social media, end-users may disclose personal data unconsciously while surfing on the Internet, lose control of the data they submit / that are collected as different stakeholders with different privacy policies can share these data. To limit these risks, particular attention should be paid to the privacy policies, allowing users to understand them before consenting to provide their data and to control the protection level they provide. Focusing on the legal environment, several laws aiming at protecting personal data and privacy have been developed, mostly on the European Union side, leading to the GDPR, deployed in 2018. The General Data Protection Regulation (GDPR) strengthens and unifies data protection for individuals in the European Union [89]. GDPR transfers the burden of proof on the service provider side instead of the end-user who consumes the service and provides its personal information. Whereas GDPR allows statistics-related usages leading to data fusion or information mining from a large set of anonymized data, service providers have to state and prove usages they have for a particular data by managing users' consents accordingly, reporting any security breach to the data owner. Due to these requirements, service providers must describe all the actions to be done on the different data provided by users depending on their category. Service providers have also to manage data collection [90] and track data flows between stakeholders [91] or between their own activities.

### 2.2.2.1 *GDPR main requirements*

The European Legislation Identifier (ELI) ontology which supports the definition of legal resources with attributes and terms through URI templates, can be used to show that GDPR legal resources and concepts can be linked to each other as well as to other resources thanks to URI [93]

Focusing on the social media environment, different risks must be considered such as disclosure of private data unconsciously while surfing on the Internet, lack of control on information submitted and transfer to third parties, or data sharing with third parties, where each party has its privacy policy. These risks are obviously due to a lack of transparency and non understandable privacy policies. To overcome this limit and fit the GDPR transparency and user consent management requirements, [92] categorizes data according to the main types defined in GDPR and uses supervised machine learning techniques to classify privacy policies in a three-scale risk-based categorization so that risk indicators are set to draw users' attention.

Focusing on Big Data, [91] points out the protection provided by GDPR. It also provides a legal analysis showing why big data and protection policy affect each other and need to be balanced. While purpose limitation constraints the usage of data, it is difficult to provide complete information on (may be unknown a priori) Big Data processes and to provide associated monitoring information. Data minimization may compromise the Big Data process itself. For example, removing identifiers to achieve pseudonymity can potentially undermine the quality of the results as the data are altered. This may also prevent the aggregation of different datasets. Last but not least, while sensitivity levels are defined for data categories to prevent data leakage for sensitive data, Analytics processes can bypass these protections, mining both sensitive and insensitive data. To face these risks,

users often prohibit automated analysis, reflecting the deep distrust towards these automated processes.

Other works also pay particular attention to the data collection sources such as data collected by IoT devices. Such devices are often unprotected and may capture different kinds of often sensitive personal data. For example, medical devices capture health parameters, while household appliances collect precise living habits... As these devices are interconnected in larger smart systems, they can be seen as a major security and privacy breaches. [95] couples an IoT ontology to a security and privacy ontology to manage GDPR requirements. This ontology aims at making the smart devices more autonomous while managing their access control decision by inferring data access rights and enforcing the privacy policy compliance at the execution level.

Risks related to the big data process are due to the mining processes, looking for hidden patterns, and the proliferation of personal information. Focusing on data privacy protection, [96] defines both pseudonymous and anonymous data. Pseudonymous data are personal data that can no longer be linked to a person unless additional data or processes are used. Anonymous data are personal data that can no longer be linked to a person even if external data and processes are used. Data Management Platforms (DMP) proposed in, [96] offers tag management, users' segmentation, media integration, campaign analytics, and user analytics. Tag management supports data audit and data control; user segmentation supports the role's identification. Media integration, Campaign analytics, User analytics technological platforms support value propagation, value generation, and value exchange. This paper also provides some countermeasures for privacy protection showed in Table 6.

External part	Internal organizational part	Internal technological part	Data part
<b>Browser extension on the user's system</b>	Instant messaging Internet application Remote logins VOIP and games and other communication anonymity and pseudonymity systems	Type-0 remailers Anonymizer.com Onion routing The freedom networks Java Anon proxy Tor GNU... Protocols for unified communication	Data encryption

*Table 6 Countermeasures for data protection*

GDPR provides the “general analysis” principle which integrates pseudonymization to reduce the risks while using the information contained in large databases. [97] focuses on the way pseudonymization impacts big data processing. While anonymization prevents consumers from identifying, providing a guarantee of privacy and breaks the risk of big data, pseudonymization may allow indirect re-identification, requiring additional steps such as providing distributed additional information.

Pseudonymization and privacy-by-design can be combined with additional technical and organizational countermeasures (such as technical and functional internal policies) to improve privacy protections.

Focused on smart-cities applications, [98] provides a new Big Data-assisted public policy-making process that implements “privacy by design”. It provides a refined definition of the big data-assisted policy-making process, defines the privacy-aware public policies, and formalizes a privacy compliance assessment based on GDPR. Data are classified into two sets: one gathers data collected before the definition of the policy (retrospective data) and the other one gathers data collected after the deployment of the policy (perspective data), collected for the evaluation of the proposed policy. The situation analysis considers data exploration (acquisition), data option(preparation) of the data consumer for initializing the policy. The action plan will consider the data from the data owner aspect and predict the action from the data consumer aspect. It includes data partitioning, feature/value selection, and voting for the final policy. Lastly, the implementation provides an open-source Apache Foundation framework showing the main components of the big data platform which is also used to evaluate and determine the final policy. It gives the Model-Based Big Data Analytic as a Service approach for the compliance assessment.

Despite the protection means these works propose, particular attention must be paid to the way users may approve or refuse to share their personal data. This leads to consent management and questions for consent proofs. Due to its immutability characteristics, Blockchain can appear as a promising solution to achieve this goal.

Paying attention to the burden of proof devoted to the data consumer, GDPR involves that data owners may control the usage of their data during their life-cycle. This means managing user consents accordingly and reporting any security breach to the data owner. To fit these legal obligations, several works have been developed either(i) to identify both information and processing categories in traditional Enterprise Architecture models in order to simplify the data usage control [90] or (ii) to manage data collection and tracking data flows between stakeholders [91] which involves tracking proofs.

#### *2.2.2.2 Blockchain-based protection*

Focusing on the way “fair and accepted usage” can be proved, several works have focused on blockchain technology due to its immutability property. Blockchain is an integrated multi-field infrastructure [99], combining cryptography, peer-to-peer networks, and distributed consensus algorithm to manage transaction-based synchronization. Blockchain technology has six main key characteristics: decentralized, transparent, consensus-based, anonymity, immutable and open source. Blockchain technology has started in 2008 with the Bitcoin used to support financial transactions. Parties involved in these transactions are defined thanks to addresses associated with cryptographic keys. This mechanism provides anonymity to parties Id. Blockchain 2.0 provides automated transactions, i.e. First introduced in Ethereum, smart contracts allow storing automated transactions in a Blockchain, leading to the Blockchain 2.0 model. Ethereum is an open-source blockchain platform combining Smart Contract, offering a decentralized virtual machine to handle the contract, by using its digital currency called ETH. People can create many different services, applications, or contracts on this platform [100]. Smart contracts

are automatically executed to control user's digital assets according to the participants' rights and obligations.

Blockchain-based data protection solutions are based on blockchain technology for enabling users to preserve their IoT data privacy while eliminating the need to trust a centralized regulator. For instance, [101] proposed a distributed data storage system, which used blockchain to maintain data access control and data storage model. [102] proposed a decentralized personal data management system, which uses blockchain to keep track of both data and access transactions. Considering the blockchain immutability property, it can be worthy used (i) to manage access control function such as the smart contracts embedding access control rules [103], (ii) to manage data encryption key protecting data access [104], (iii) to manage user consents[105] or (iv) to track data accountability and provenance tracking [73] usage operation thanks to smart contracts generated according to the data usage policy [105].

### **2.3 Conclusion**

While risk and security engineering methods provide a strong basis to identify sensitive assets, threats, and vulnerability in traditional and well perimetrised information system, their intrinsic “process-driven” approach may lead to security breaches due to inconsistent protection when a data asset is replicated in different sub-systems using different protection strategies. Moreover, these methods used to identify systems vulnerabilities and threats are not suitable for opened and unperimetrized environments where potential usages, detailed processes, and infrastructure may be out of control.

The integration of Social, Mobile, Analytics, Cloud, and even Internet of Things technologies (SMAC-IT) in Collaborative Information Systems involves paying attention to new vulnerabilities and threats due to the intrinsic usage they have on data assets. In such opened and rather evolving environment, usages may also be considered as threats or vulnerabilities leading to asset dispatching, privacy violation... This involves that SMAC-IT system cannot be protected anymore thanks to these traditional risks and security engineering approaches: potential usages and associated vulnerabilities must be integrated into data protection policy, requiring extended usage and protection ontologies.

To fit the extended, collaborative, and sharing organization model carried out by SMAC-IT, legal regulation has also evolved. The European Union General Data Protection Regulation requires service providers to provide fair information to the service consumer and manage usage consents. This legal constraint has led to the development of pre-formalized consents regarding data that may be collected and processed. Focusing on the data provider (i.e. the service consumer), these consents do not provide precise information on the usage scope and there is no way to guarantee that consents are consistent with the data protection strategy. Paying attention to the service provider side (i.e. the data consumer), few works have been focused on Blockchain technology to manage and store consents. Nevertheless, these works are mostly devoted to consent management and do not allow controlling the way data assets are protected.

To overcome these limits, we identify two main challenges:



- Defining a data-centered and usage-based control model allowing defining consistent and life-long protection on data assets, paying particular attention to potential usages considered as potential threats
- Defining a consent-based usage governance model allowing to derive precise usage authorization from global consents and track the exact way data assets are processed and exchanged to ensure the consistent protection

### 3 Data centered protection model

As stated in the state of the art, SMAC-IT challenges new security organizations to provide consistent life-long protection to data. Such distributed environment challenges information privacy management as data assets can be exchanged and used by several parties (service providers, hosting platform providers...). Moreover, as this distributed environment can be supported by highly distributed infrastructure components, the global ecosystem can be constrained by different national legal rules as the different actors involved (service provider, hosting platform manager...) may use and transfer data assets from one country to another one. According to a societal point of view, this complex and distributed organization may lead to “unfair” practices as data owners may lose control of their assets as several copies are shared by multiple stakeholders. This trend is reinforced by the reduced privacy management abilities offered by data consumers/service providers: service consumers can only select service providers according to a (maybe subjective) trust level and then they have to accept or not the protection and privacy conditions proposed by the selected service provider. Regarding specific risks related to SMAC-IT, particular attention must be paid to social networks as they provide coarse-grained access control on data they store and as several uncontrolled copies of data can be made. This involves managing information security and privacy continuously among different collaborative service chains. It involves also being able to manage finer-grained privacy requirements related to an “information life cycle”, propagating these requirements for all the information copies and integrating the way the information asset evolves.

These challenges can be illustrated thanks to a simple use case. This motivating example integrates collaborative business and end users’ interactions. It relies on an online shopping platform (called later Online Shopping), shared by different companies proposing products and services to clients and exchanging information to propose “manufactured-on-demand” products. To reduce the carbon footprint, Company A uses the 3D printers hosted by company B to “manufacture” the product as close as possible to the client. Online Shopping also shares data with MyAnalytics company which uses the customer data to establish recommendations and provide marketing analysis to Online Shopping company.

Alice, Bob, and Charly, three friends sharing some personal information, often use an Online shopping platform. Each of them has particular privacy requirements. Bob trusts Alice, his sister, and share lots of personal information with her. Each of them can use the other contact information for a delivery purpose (“permanent delegation for delivery”). As Charly has very weak privacy requirements, Alice and Bob forbid him from sharing their personal information.

Alice browses the Online shopping application which provides products from different suppliers to buy a gift for her brother Bob. In her Personal Information System, Alice shares some of Bob’s personal information such as his contact information, his identity information, photo... Alice will have to share Bob’s contact information with Online Shopping and its sub-contractors to manage the gift delivery. The Online shopping platform collects user’s view history and connection information as well as other information related to the product Alice intends to buy. Consequently, Alice will have to “share” different personal information such as traces of her online activity, financial

information related to her payments, her address (for billing/delivery purpose), and what she has bought with Online Shopping platform, MyAnalytics (which can also mix this information with other sources) and Company A and B (the product suppliers).

Alice also interacts with other online platforms, Personal Information Management Systems, social networks... using her computer or smartphone. Protecting consistently her personal information is difficult for Alice as she interacts with different systems, providing their Terms of Service (ToS for short). Moreover, she cannot get any information to check if her information is protected and used according to the ToS she accepted.

The online Shopping platform is responsible for Alice's personal information protection and, according to the GDPR, must be able to prove that it uses and protects this information according to the exact ToS. While exchanging data with its partners (MyAnalytics, Company A or B), Online shopping must check the business purpose of the external service requesting information to verify if this is allowed by the ToS Alice has approved. It also has to transfer this ToS to the service provider.

Charly, a friend of Alice and Bob also uses the Online Shopping application. Although he knows Bob's contact information, Bob has never allowed him to share this data with other parties. Charly has weak privacy requirements and he often publishes his personal information without restriction. He, therefore, agreed, as part of his interaction with the online shopping application, that his own contact information, his full profile... can be used for marketing purposes and even can be shared with other parties.

Based on this simple motivating example, we can identify different challenges:

- Focusing on data providers, a unified vision on the way a data asset is used and shared is necessary to provide consistent protection. Moreover, "data usage delegation" must be managed to control whether the asset security level can be controlled or not.
- Focusing on the data consumers, different protection policies must be deployed on assets depending on their origin and their associated Terms of Service. Similarly, usages must be well defined and control to ensure that only "fair usages" are achieved. . Of course, data assets must be protected consistently, regardless the IT component processing them.

As said in the state of the art, traditional risk engineering methods use the different processes as a guideline to identify threats, vulnerabilities and the assets to protect. This leads to define specific protection for each Information System component so that the different copies of a same data asset will be associated to different protection policies, depending on the component using them. Moreover, these methods are designed for well-known information systems where data assets and processes are well-identified. Unfortunately, the SMAC-IT and collaborative context leads to opened and evolving Information System organization, adding collaborative processes or new partners so that risks must be re-evaluated continuously. Focusing on Usage Control, usage actions are

To overcome these limits and provide a consistent protection to data assets, even in opened context, we propose to capture the Information System organization in a single model, from business context and business relationships among parties to IT support components. By this way the current Information System perimeter may be identified. This Information System model will integrate an abstract description of both data assets and

processes using them as well as a description of the IT components supporting these processes and of the replicated instances of the data assets stored and processed by these IT components.

Such an Information System description model will provide:

- a unified vision of data assets and their replicated instances
- a precise description of the way these data assets are used.

Thanks to this Information System model, requirements of protection can be defined once for each data asset depending on this data asset sensitivity and value. Then, these requirements of protection can be used to control the protection policy associated to each replicated instance.

Moreover, the business knowledge captured by the description of the business relationships and context as well as the abstract process description enriches the description of usage actions on data assets. This will allow a finer-grained definition of “fair usages”. To manage this point, we propose to expand the Collaborative Usage Control model proposed in [85] to integrate business context and process information to define contextualized “fair” usages. To his end, a usage ontology including a business taxonomy and generic usage actions will be set.

By this way, a data-centered protection model is set. Data protection is defined consistently although data assets are replicated and used in various processes and context. Then, policy aggregation and matching process can be defined to control continuously the way an asset is protected.

In this chapter, we introduce first the Information Description system model, describing the collaborative organization, data assets and processes using them. This system will provide a centralized description of data assets so that security requirements can be defined once and are “propagated” to the different copies of the data asset. As security threats and vulnerabilities can be due to the way data assets are used and process, we expand the traditional security and protection ontologies to manage usages. Thanks to the integration of business knowledge, fair usages can be defined precisely. By this way, Requirements of protection and Quality of Protection policies can be set, compared and combined with usage control function.

### **3.1 Multi-layer information system description model**

To manage the current asset protection and allowed usages, we design an Information System (IS for short) description model to store the IS collaborative ecosystem description, identifying the different service providers and their relationships, the different data assets description, and the description of their replicated content. More precisely, this information system description model is organized in three main layers:

- *the organization layer* gathers information on the different actors (human beings/enterprise / organizational unit) that own, store, or process logical assets (also called logical data assets) with meta-data description. Contracts, including security agreements, are associated with the relationships between actors. Actors are also

associated with different business areas / organizational units according to their competencies or to the Business Processes and business activities they have to manage.

- *the “logical” layer* gathers descriptions of both data assets and the way they may be processed. Logical data assets are described thanks to meta-data and are associated with security requirements depending on their sensitivity level. These data assets can be combined to set more complex assets, aggregating atomic ones. Processes are defined thanks to abstract services, described thanks to activity patterns related to their business domain. Each of these abstract services is also characterized by its interface, described thanks to data objects (which can be seen as service interface objects) associated with logical assets.

- *the implementation layer* stores the identification of the real physical instances associated with the data object. These physical copies of logical assets are called “containers”. These “containers” are consumed by “concrete services” associated with IT or manual application services. Each of these “concrete” services is provided by an actor and has its security policy which may be inherited from the data consumer’s generic security policy, enriched with the target usage of the data object. Paying attention to the cloud-based multi-tenant implementation, each concrete service deployment may use other support services (cloud or network services), using a “support” relationship.

This multi-layer Information system description model is split according to two main “vertical” dimensions:

- The first one is related to the static Information System organization to provide a unified reference description model so that data assets and the way they can be used and protected can be identified. This part of the Information system description model will later be called “static dimension”. It stores logical data description and the way they are used and processed, describing the services using them. Data assets and services are described semantically, using metadata and dedicated models integrating knowledge of the specific business domain. We extend this data semantic description by integrating information on the data source (internal, external, or mixed), the data origin (collected, created by the target business process or generated thanks to an analytical algorithm), and data representation form (multiple physical copies) to describe asset. Moreover, we define the service referring to the collaborative business process (including the business purpose and collaboration purpose), abstract usage with functional means (i.e., the target usage within its operated generic communication and processing application), and concrete execution call and physical infrastructure.

- The second one is related to the dynamic information usage life-cycle, i.e. the implementation of information evolve. Besides multiple copies of an asset, we also need to protect the derivation of the asset resulted from potential usages. While the usage is associated with different services which can compose or decompose with purpose and implementation. At last, the data-centric information model will refer to internal or collaborative services and extend the consistent protection to various physical instances and logical asset groups.

Thanks to this data-centric information model data-centered protection policy can be managed and shared by multiple copies of a data asset so that consistent protection can be set to provide life-long protection on data assets.

### 3.1.1 *Static view of the Information System description*

#### 3.1.1.1 *Organizational layer*

This layer is used to describe the different actors involved in the information system and their relationships (see Figure 16). An actor can be a human being, a legal entity, or a part of a legal entity (i.e. an organization entity). Organization entities can be defined recursively and consequently may include sub-organization entities. Actors can be gathered into trust groups.

Actors take part in different processes. These processes are defined thanks to Business Services that are interconnected to reflect the way the process is organized. Each Business Service refers to a business area. It is described functionally thanks to a set of meta-data associated with this business area. Business services can be defined recursively, i.e. a Business Service may include other Business Services. We call Elementary Task a Business Service that does not include other business services.

The data required/produced by these business services are defined thanks to Logical Asset Patterns. A Logical Asset Pattern may be defined recursively (i.e. it can include sub-logical asset patterns). Each Logical asset pattern is described using different characteristics associated to meta-data. Each of these characteristics is related to different types. Depending on the business area a data belongs to, several models defined thanks to logical asset patterns can be reused. For example, as far as personal data are concerned, some typical categories can refer to:

- The personal aspect considers social viewpoints. In our use case, it includes traces of online activity, home address, photos, and identity information.
- The financial aspect is related to transaction payments. In our use case, it will consider the bank account, payment password, and payment certification code, etc.
- The management aspect refers to business process management considering organizational viewpoints. The use case includes product delivery address, product contact information, product order information. Actors are linked thanks to contractual relationships. Each contract defines the common goal, what each actor provides and consumes. Contracts can refer to generic models such as partnership agreements, data processing consent, “Service agreements”, financial agreements... More precisely a contract consists of a set of clauses, each of them dedicated to relating actors to particular elements of the information system.

From our motivating example, we can identify different actors: “human actors” such as Alice, Bob, and Charly, legal entities such as Online Shopping, Company A and Company B or organizational entities such as Online Shopping Marketing department, accounting department, delivery department, after-sale department, ordering department... Contracts link Online Shopping with Company A and Company B, specifying a business purpose, financial, and quality of service clauses. Different trust groups are set: Online shopping trusts Company A, B, and My Analytics, Alice trusts Bob but not Charly...

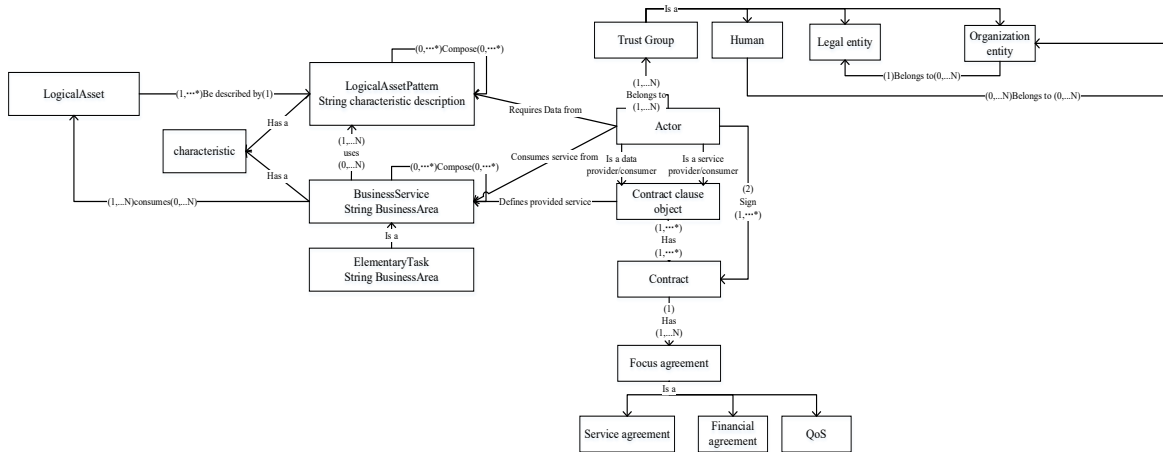


Figure 16 UML class diagram of the organization description -model

Focusing on Online Shopping IS Description model, it includes different Logical asset Patterns, such as Delivery contact information, Billing contact information, Bill information, Product description... and business services such as online ordering associated with the Online Ordering Workflow which includes the ordering control business service, the billing business service, the production, and delivery business service... We can note that this production and delivery business service is split into different elementary tasks (one to schedule the production, another to customize the product) and sub-workflows shared with partners (“print and deliver” business service). This last Business Service refers to agreements between Online Shopping and Company A and B.

### 3.1.1.2 Logical layer: describing the Information System

In this layer, we describe the Information System “IT Twin” -model, paying attention to data and process descriptions. These descriptions are used to identify assets to define requirements of protection depending on each asset value/sensitivity. This IS Description model includes a data description -model, a data usage -model, and a security -model.

The data description -model (see Figure 17) is used to describe logical data assets, called Logical Assets. A logical asset is a pivotal element used to describe precisely an information asset. The logical asset is defined as a set of other logical assets or atomic assets. Each logical asset is defined semantically thanks to a set of meta-data defined as characteristics in our model. Each of these characteristics, is associated with a characteristic type. The Logical Asset also refers to the Logical Asset Pattern to capture the knowledge related to the IS organization. Logical Asset patterns can also be described recursively (i.e. a logical asset pattern may include other logical asset patterns)

From our use case, focusing on Alice's Personal Information System description, Alice manages a Logical Asset Pattern called Contact Information split into sub-patterns as name, address, phone number, and email. She uses this pattern to organize her own contact information, the contact information related to her brother Bob and her other friends such as Charly. Consequently, different logical assets are defined as My Contact Information, Bob’s contact information, and Charly’s contact information sharing the same

description. Similarly, Online shopping manages different Logical asset patterns such as delivery contact information and billing contact information sharing this contact information pattern.

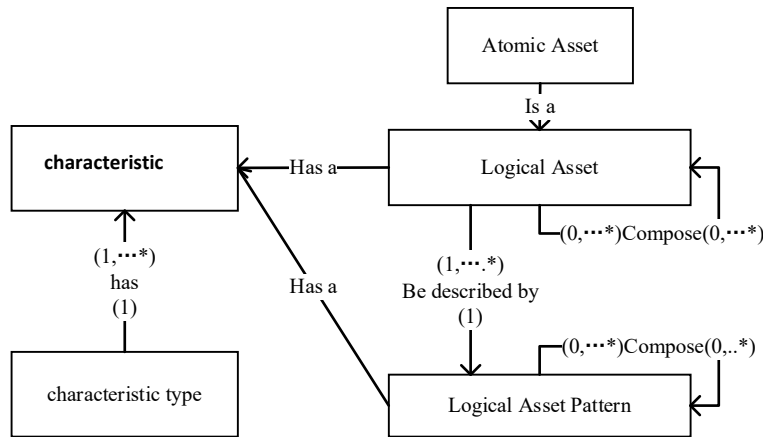


Figure 17 UML class diagram of the data description model

The data usage description model (Figure 18) describes IT processes (such as Email Service) using the data. This process description consists of a set of logical services defined thanks to their functionality. These logical services represent the IT twins of Elementary tasks, providing the implementation description of the associated support services. These logical services can be defined recursively as a service may include a composition of logical sub-services. We call atomic service a logical service that does not embed any other services.

When Alice interacts with Online Shopping to buy a gift for Bob, she will have to integrate the “online ordering” service in her information system. Focusing on the Online shopping side, different services are defined to implement this “online ordering” workflow: The Logical order management service supports the interactive task with the customer to get its requirements, the billing logical service is in charge of the payment...

Logical services may transform input data into output data by using internal information. To support this description, we introduce a data object concept that represents a logical service interface description. A data object refers to a data description, using a logical asset pattern. Data objects are used to set the relationship defining the different operations a logical service may have on a particular type of data. In this way, usages on a given type of data asset can be defined precisely. For example, an Email Service requires a password as the Logical Asset Pattern and String type when it wants to authenticate the Login user and requires an email as Logical Asset Pattern and word doctype when it wants to send.



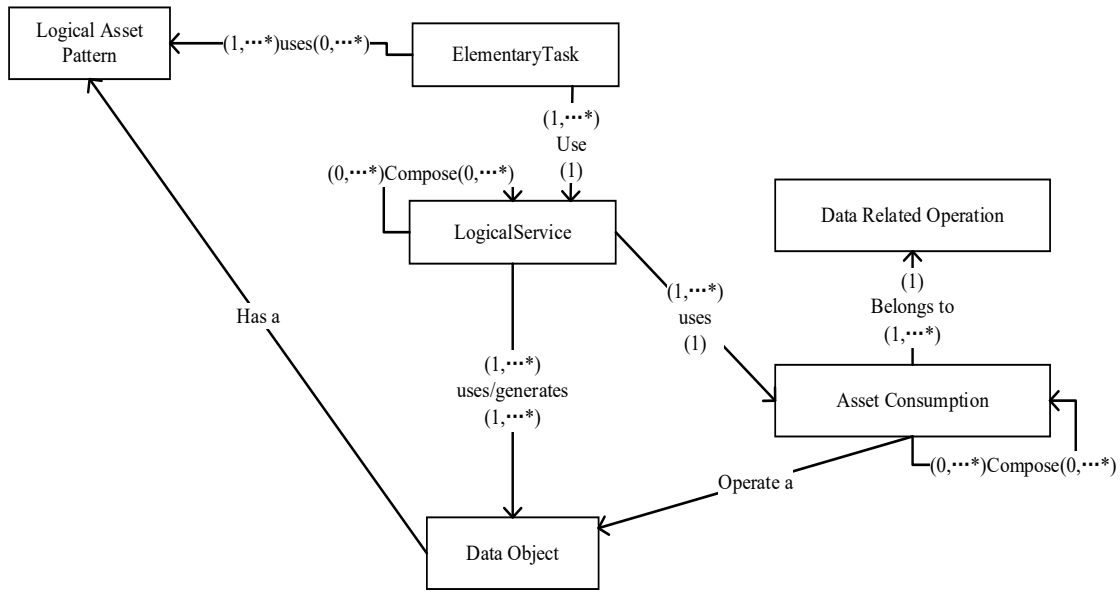


Figure 18 UML class diagram of the usage description model

From our motivating example, Figure 19 presents the dump of the Data Base records of the Logical asset pattern used by elementary tasks.

<input type="checkbox"/>	BusinessServiceId	metaData
<input type="checkbox"/>	8	ContactInformation
<input type="checkbox"/>	8	Account
<input type="checkbox"/>	8	OrderTransaction
<input type="checkbox"/>	8	FinancialTransaction
<input type="checkbox"/>	8	ActivityOfTrace
<input type="checkbox"/>	8	ContactInformation
<input type="checkbox"/>	8	Account
<input type="checkbox"/>	8	OrderTransaction
<input type="checkbox"/>	8	FinancialTransaction
<input type="checkbox"/>	8	ActivityOfTrace
<input type="checkbox"/>	7	ContactInformation
<input type="checkbox"/>	7	OrderEstablishment
<input type="checkbox"/>	9	OrderEstablishment
<input type="checkbox"/>	10	ContactInformation
<input type="checkbox"/>	11	OrderEstablishment
<input type="checkbox"/>	12	Address
<input type="checkbox"/>	14	OrderEstablishment
<input type="checkbox"/>	14	AccountId
<input type="checkbox"/>	14	PayMentInformation
<input type="checkbox"/>	13	OrderEstablishment
<input type="checkbox"/>	30	OrderEstablishment
<input type="checkbox"/>	30	FinancialTransaction
<input type="checkbox"/>	30	Account

Figure 19 Example of Business Services provided by OLS and the assets they require

### 3.1.1.3 Implementation layer

The implementation layer considers the physical representations of the data and the real “concrete services” describing application means (see Figure 20) Data may be associated with different physical copies, stored separately. We call “container” the physical representation of a logical asset. A container can be associated with an IT format such as files, pixmaps, memory regions or network packets, etc., or to a non-IT format (a paper copy...). Each container may be transmitted, used by operations, stored... by different concrete services. Concrete services providing IT applications such as Data Base, OperatingSystem, etc. can compose, process, and transform containers to generate new ones.

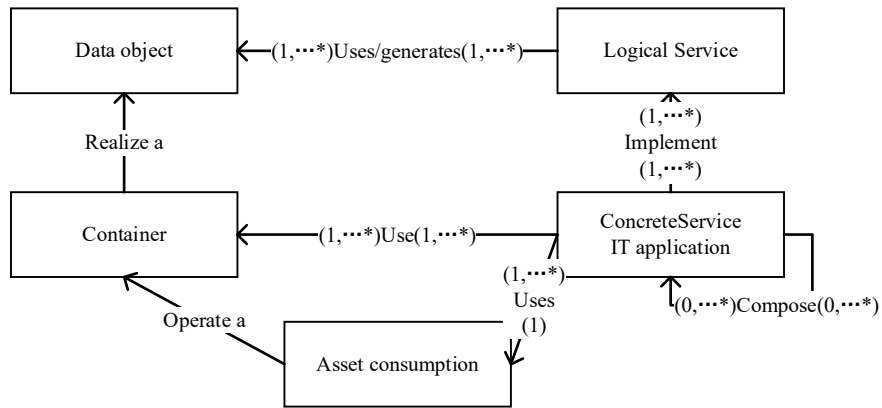


Figure 20 Class diagram of the concrete service and data implementation model

The connection between the implementation layer and the logical layer is achieved by adding relationships between concrete services to logical services and by linking containers to data objects and logical assets.

From our example, Online shopping has to manage physical copies that are associated with the information provided by Alice. Her contact information is processed locally by Online Shopping whereas the delivery contact information is associated with two containers: the first one is processed locally by Online shopping to integrate the delivery information on the bill. The second one is sent to the sub-contractor who is in charge of the final delivery of the product (Company B) (see Figure 21).

id	metaData	extensionDescription	UserName	Role
1	ContactInformation	OnlyForDelivery	Alice	Customer

id	content	formatType	containerDescription	metaData	logicalAssetId
2	INSALYON	8B pdf	1 page	ContactInformation	1
3	INSALYON	8B pdf	1 page	ContactInformation	1

Figure 21 Dump of the data base of the Logical asset related to Alice’s contact information and its two contents stored in OLS own Information System description

To define the way the system is deployed and implemented, concrete services are described as a composition of different implementation services, following the cloud multi-layer model (see Figure 22).

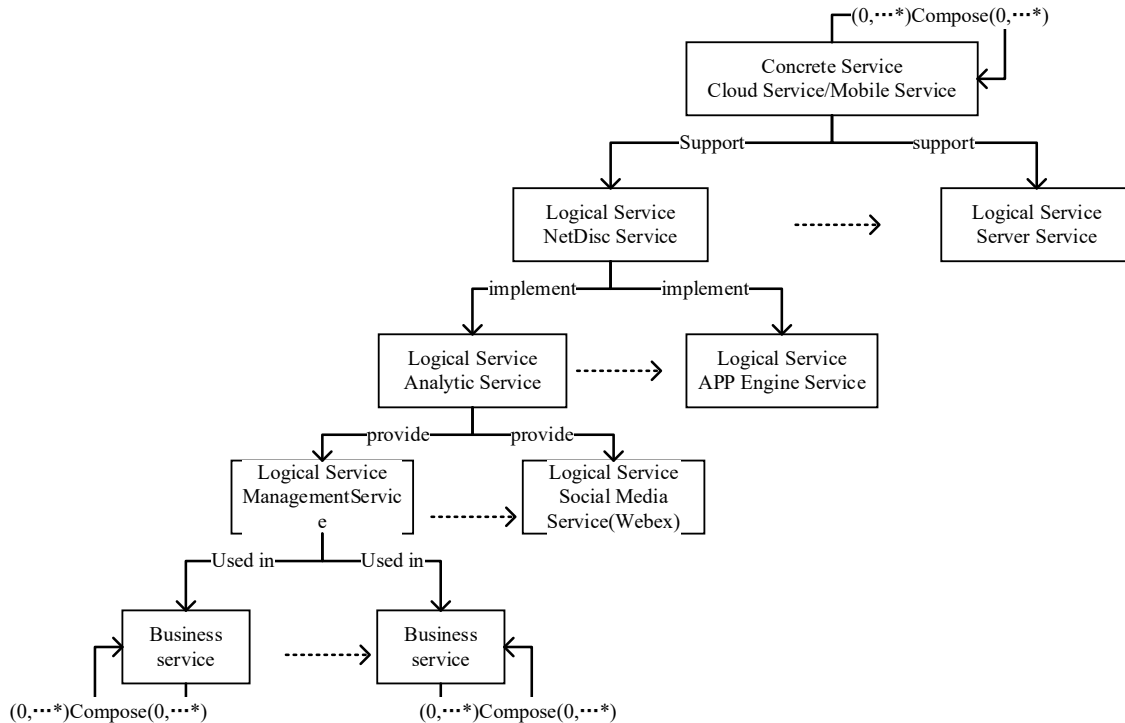


Figure 22 Multi-layer cloud-based deployment

### 3.1.1.4 Conclusion

This Information System description enriches the Business Process model with organizational and data resource knowledge. Processes are defined thanks to Business Processes (the Business Service) and Business Activities (Elementary tasks). Focusing on the Processes implementation, IT tasks are described as logical services and concrete services. This allow integrating both applications and other support services in the process description. Compared with traditional Business Process models, our Information System model also integrates both global information class description (the logical asset pattern) and information instances abstract data resources (the logical assets). This will allow defining a tuned and fine-grained security policy for each instance. Focusing on the process description, the recursive description of business services, allows defining different descriptions, more or less precise, for the processes. This will protect the process confidentiality while creating a shared collaborative business process as sub-processes and tasks can remain private. Lastly, asset consumption linking containers and concrete services allow defining precisely usage actions on assets.

### 3.1.2 Integration of security requirements and protection means in the IS description model

After describing the IS organization and implementation, particular attention must be paid to the way security is managed. As seen in the state-of-the-art chapter, protecting an information system involves identifying and mitigating organization-based and technological-based vulnerabilities and threats. Providing consistent protection to data and services that process them involves defining security requirements and deploying security countermeasures among the different layers of the IS description model.

### 3.1.2.1 Organizational layer security -model

As said previously, this layer is used to describe the different actors involved in the information system and their relationships as well as the business services they offer. This layer addresses organization-based security, paying attention to established relationships between parties, Business Process organization, and trust management. This organization-based security model allows defining undue processes as potential threats and capturing organizational vulnerabilities. More precisely, integrating security requirements and counter-measures in this layer relies on:

- Particular clauses in the contract are related to defining the data asset sensitivity, providing the key characteristics to set requirements of protection, and evaluate the quality of protection policies accordingly to define how data ought to be / will be protected
- Trust information is associated with each actor: this information is used to define if the associated actor usually respect (or not) the security clauses
- Using a precise connection between actors and services allow defining precisely who will be in charge of the different part of the process / who can accede to a particular data/service
- Relationships between actors define who will be involved in the information system deployment regarding a particular business service.
- The logical asset pattern sensitivity level is used to set the security requirements regarding the three security services (confidentiality, integrity, and availability).

Deploying this organizational security model requires capturing the way partners collaborate, focusing on business services execution, and paying attention to the real assets exchanged and processed along with the workflows.

From our use-case, the contract linking Online shopping with company B integrates dedicated clauses to define that delivery contact information can be used and shared only for delivery purposes. It also integrates security clauses to identify secured communication channels between Online Shopping and Company B.

### 3.1.2.2 Logical layer security -model

patterns. Thanks to the sensitivity level associated with the logical asset patterns, security requirements, and security protection levels can be defined for logical assets: they will be “at least” protected as requested in the Logical Asset pattern they belong to.

These sensitivity levels are used to derive security Requirements of Protection (RoP) associated with data assets (namely Logical Assets and Logical Asset Patterns). Focusing on the way assets are processed by logical services, the security policy attached to the logical service description allows identifying the Quality of Protection (QoP) associated with a logical service. Both Requirements of Protection Policy and Quality of Protection policy are defined using a policy model. A policy consists of a set of assertions, each of them addressing particular vulnerabilities or threats to mitigate/reduce the associated risk. As in a traditional information system, we can use the asset semantic description to evaluate its sensitivity level. To guide this analysis, existing security engineering methods identify some vulnerabilities, value assessment patterns depending on the data type or on the service

functional description, leading to “standard” pattern-based requirements of protection. As security countermeasures are defined to face threats against the basic security services (confidentiality, integrity, availability), we extend usage description with the associated threats it can represent and we add extra security-related information to the logical data description with identifier/sensitivity attributes. Attributes can be non-sensitive or sensitive, i.e., allowing identifying a particular asset of the system:

- Explicit\_Identifier is an attribute that explicitly identifies a data and/or its owner. For example, a bank account number is an explicit identifier.
- Quasi\_Identifier is a set of attributes that allow a less precise identification. It includes
  - (1) Implicit\_Identifier which are a set of attributes that are necessary to characterize the data / its owner (for example the name of the bank account owner, its birth date, and the bank owning the bank account can be used to retrieve a bank account identification)
  - (2) Deduction\_Identifier which identifies or represents the attributes that can be used in an analytics process to identify a data / a data owner

Figure 23 and Figure 24 present the secured data description model and the secured usage description model.

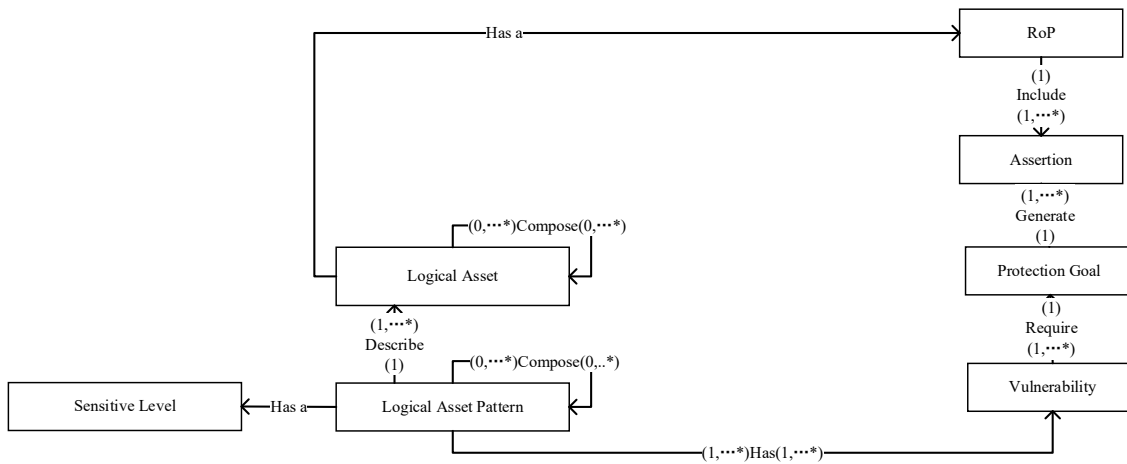


Figure 23 Class diagram of the secured data description model

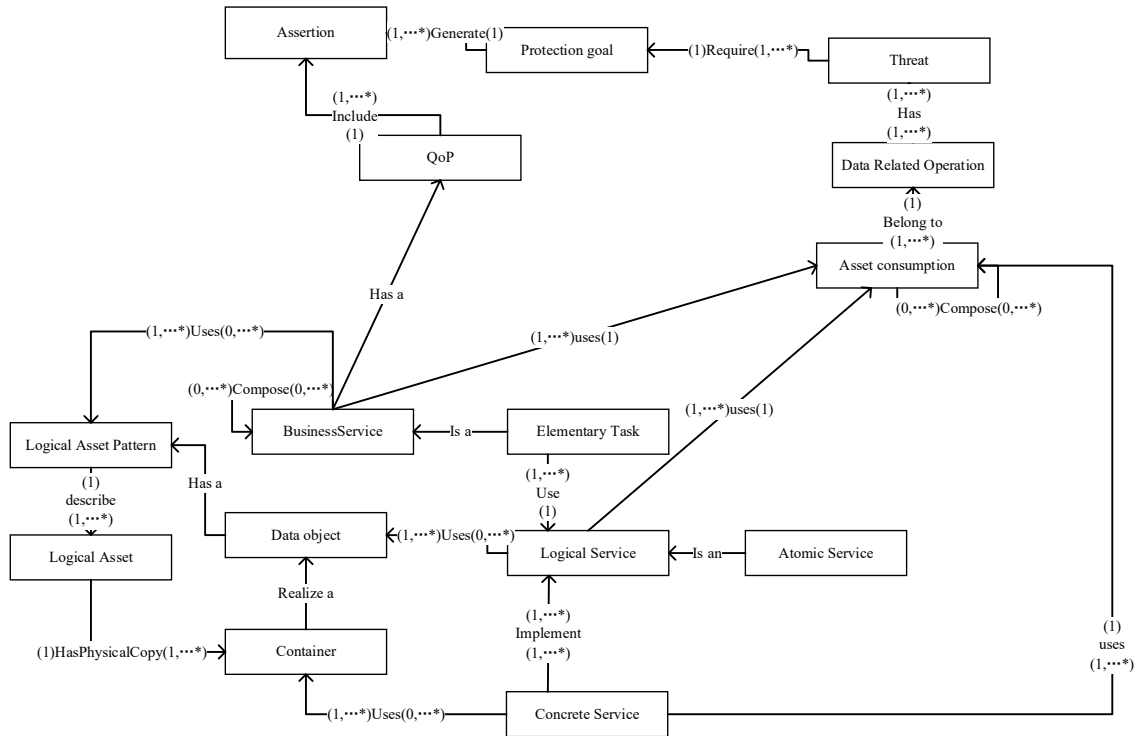


Figure 24 Class diagram of the secured usage description model

From our use-case, Online Shopping can define precisely the Quality of Protection means provided by its different logical services whereas specific requirements of protection are associated with the different “contact information” instances depending on the ToS concluded with the different parties. For example, Alice’s Billing contact logical asset and Alice’s delivery contact logical asset is highly sensitive whereas Charly’s billing and delivery contact logical assets exhibit a low sensitivity level (as Charly accepts everything and has already published them on social media). Figure 25 presents a dump of the data base of the requirement of protection associated to Charly’s contact information.

<input type="checkbox"/>	id	metaData	CountermeasureLevel	CountermeasureContext	lid
<input type="checkbox"/>	14	Address	medium	UsageDelegation	2
<input type="checkbox"/>	15	Address	high	DataIntegrity	2
<input type="checkbox"/>	16	Address	low	DataConfidentiality	2
<input type="checkbox"/>	17	Telephone	medium	UsageDelegation	3
<input type="checkbox"/>	18	Name	low	UsageDelegation	4
<input type="checkbox"/>	19	ContactInformation	medium	UsageDelegation	1
<input type="checkbox"/>	20	ContactInformation	high	DataIntegrity	1
<input type="checkbox"/>	21	ContactInformation	low	DataConfidentiality	1

Figure 25 Example of Protection level associated to Charly’s contact information

### 3.1.2.3 Capturing security in the implementation layer

Similar to the logical layer, security policies are attached to the different concrete services and containers. Security Policies associated with containers are used to identify the security process that has to be executed before generating/acceding to the content associated with the container. For example, such assertions may be related to encryption

algorithms, to obfuscation algorithms to manage the container confidentiality pre-protection whereas hash algorithms may be used to generate “signed contents” to manage integrity pre-protection. Focusing on the security policy associated with Concrete Services, assertions are associated with concrete security countermeasures implemented by the concrete service to mitigate threats and vulnerabilities. Paying particular attention to the usage-based security assertions defined at the logical layer to manage a “fair and due usage” on logical assets, usage-based access control assertions must be used to support access authorization to the data stored in containers. In this way, consistent protection can be derived while deploying the concrete services and their related infrastructure services.

From our usage case, Online Shopping will manage different access authorizations for the containers associated with Alice’s billing contact information which can be processed by Online Shopping, and its anonymized copy that will be processed by MyAnalytics.

### 3.1.3 *Integration of the IS usage: a transaction-based model*

After providing the static organization of the Information System and its security policies, we integrate different types of transactions to describe the way it is used. The dynamic view of the IS description addresses the way logical assets are used and processed. To this end, we integrate transactions in the IS Description model.

A transaction is defined by two parties, exchanging data to achieve a given service. A secured transaction is defined as a transaction constrained by a security policy associated with the exchanged data. This security policy is used to control the protection means and the allowed/denied usages for these exchanged data.

This leads us to define:

- *Business transaction*: this transaction is set between a service provider and a service consumer. It describes an execution instance of a Business Service to fulfill a service consumer need. It relates the business service provider and the business service consumer to a Business Service. This Business Transaction is associated with a logical asset and a logical asset pattern. And the transaction is used to define precisely the exchanged asset, provided by the service consumer and consumed by the Business Service to achieve a given goal. This model allows capturing the data asset provenance and the potential usage associated with this asset. As a Business Service may embed sub-business services organized in different service chains, an asset may be shared and transferred among different partners. To manage this flow, we define an asset’s delegation relationship to manage the way assets’ ownership is shared or transferred from one actor to another for a more precise business purpose to achieve the common goal. In this way, usage authorization and security requirements can be propagated to intermediate Business Services and their related actors.
- *Usage transaction*: this transaction is set between a service provider and a service consumer. It is generated from a Business transaction and is focused on a P2P transaction to precise the IT operations that will be managed by the logical services supporting the Business Services involved in the business transaction. This transaction links to the data object, defining the logical service interface to select the convenient logical assets involved in this transaction. Precise usage operations and the security

policy associated with this logical service are used to define the specific security policy associated with this transaction.

- *Physical transaction*: this transaction is associated with the real concrete service and the container associated with the logical asset. Precise operations and protection assertions defined for the concrete service are used to define and deploy the adapted security countermeasures.

To this end, we integrate transactions linking parties, services, and real assets to manage the information flows between data providers and data consumers and between services. This dynamic view allows defining events to manage the way a data asset is processed and evolves. Moreover, by adding specific security policies, the global asset protection can be tuned depending on the trust level associated with the parties involved in the transaction.

From our use case, Alice initiates a Business transaction with Online Shopping when she starts the online ordering workflow. To this end, Online shopping proposes a parametric Terms of Usage to Alice so that she can tune exactly the rights associated with the different assets that will be shared with Online Shopping and its sub-contractors. As Alice accepts to share her billing contact information for online ordering and marketing purposes, different usages will be generated for both workflows. Regarding the delivery contact information, usages will be restricted to the necessary services related to the delivery purpose, whether these services are hosted by Online Shopping or Company B.

#### 3.1.4 Conclusion

Our Information System Description model has been designed to provide a unified view of data assets and the way they are used. Thanks to our multi-layer organization, organizational relationships can be captured as well as trusted relationships among different partners. The logical layer provides a unified view of the Logical Assets so that consistent protection can be defined on the different copies of these assets. Security policies are used to define both requirements of protection and the quality of protection provided by the different services so that the current asset protection level can be evaluated. By integrating transactions, usages can be captured as well as events allowing to manage the asset life-cycle. Nevertheless, providing consistent protection, including the way an asset is used, requires defining a consistent usage-based protection model.

### 3.2 Usage-based Protection model

As shown in the State of the Art, SMAC-IT requires integrating potential usages on assets as potential threats or vulnerabilities. While our multi-layer Information System Description model provides a unified vision on data assets and on their protection, Usage-based protection must be defined more precisely. We, therefore, identify, in our multi-layered organization, different types of uses. Focusing first on the “organizational layer”, contracts are used to describe the relationships between actors, defining precisely the parties (human being/enterprise / organizational unit) that own, store, or process data assets. Actors are also associated with different business areas / organizational units according to their competencies or to the Business Processes and business activities they have to manage. According to this vision, usages are defined according to particular business areas/business processes. Actors involved in these processes may delegate their usage rights to “sub-



contractors”]; refine the authorization to fit sub-business process requirements... Consequently, usages defined for this layer are mostly related to the business motivation and scope.

Second, the “logical layer” provides a finer-grained description of the usages and the processes using them. In this layer, processes are defined thanks to logical services, described with activity patterns related to their functional domain. Each of these logical services is also characterized by its interface, described thanks to data objects associated with logical assets. This leads to set a “logical usage” protection policy for logical assets, defining which “macroscopic” operations may be authorized for given logical assets to achieve the business goal specified by the organizational usages.

Third, the “implementation layer” defines precisely the containers (i.e., the physical copy of a logical asset) which are processed by concrete services. Each of these “concrete” services is provided by an actor and has its security policy which may be inherited from the data consumer’s generic security policy, enriched with the target usage of the data object. Considering the cloud-based multi-tenant implementation, each concrete service deployment may use other support services (cloud service), using a “support” relationship. Focusing on this “concrete usage” protection, particular attention must be paid to data leakage or data loss risks. This requires defining precisely access operations on the containers, fitting the logical usages. This usage-based protection policy must integrate also threats and vulnerabilities from the system infrastructure, paying particular attention to protection countermeasures.

While traditional security ontologies can be used to define generic security requirements/countermeasures, they do not encompass usage-related requirements. To overcome this limit, we propose a multi-dimension ontology to capture both organizational, usage, and security knowledge to define usage-based security policies.

### 3.2.1 *Multi-dimension protection ontology*

The protection model is proposed to support a consistent protection and usage control on the multiple copies of a data asset according to its sensitivity and value. Each asset protection is defined in a Requirement of Protection (RoP) policy which is used by the data provider to specify who can access the asset. Protection policies are specifications that constrain the behaviors on the assets and describe the capabilities on security, Quality of service, etc. to ensure the protection of assets while releasing them to data consumers. To this end, we design a protection goal ontology (see Figure 26), gathering system protection goals, specific data protection goals, and protection means integrated into the policy. These data protection goals expand the traditional security services with privacy management. They also extends the traditional security countermeasures such as those defined in th NRL-SO ontology [109] in order to include dedicated countermeasures adapted to the SMACIT context [110].

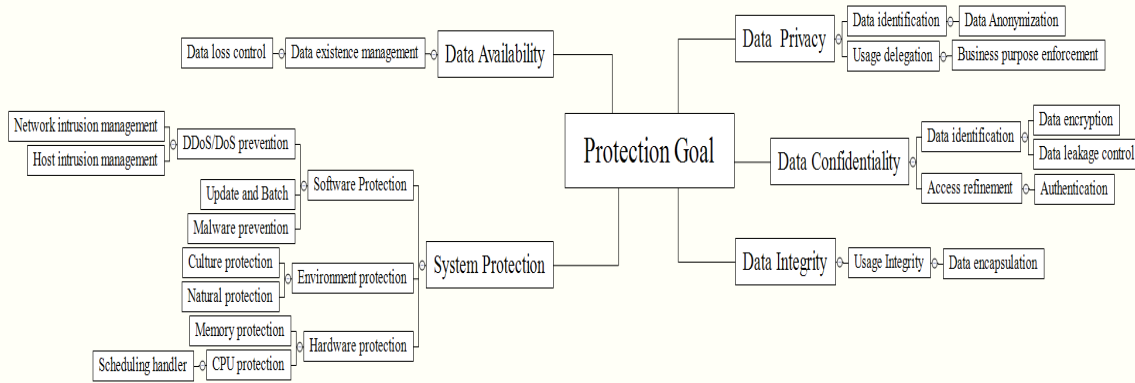


Figure 26 Mind map of the Protection goal ontology

As security breaches may also be due to the way assets are processed and shared, we assume that usages must be defined, setting specific assertions to specify the way assets are used and protected. The  $UCON_{ABC}$  model [18] relies on a usage ontology to define rights that are associated with basic usage operations (i.e., Create, Read, Update, Delete), subjects who will get a usage right, objects which define the asset on which the usage right is granted, obligations associated to protection means and restrictions associated to time or environment constraints. We expand this ontology (see Figure 27) to

- Integrate more complex data-related operations, to define more precisely the way data are processed, exchanged, replicated, and stored
- Integrate organizational knowledge to define the subject (a user, a group, or an organizational entity)
- Integrate business knowledge to define more precisely the usage context, specifying a process motivation, process purpose, and business scope related to business areas.

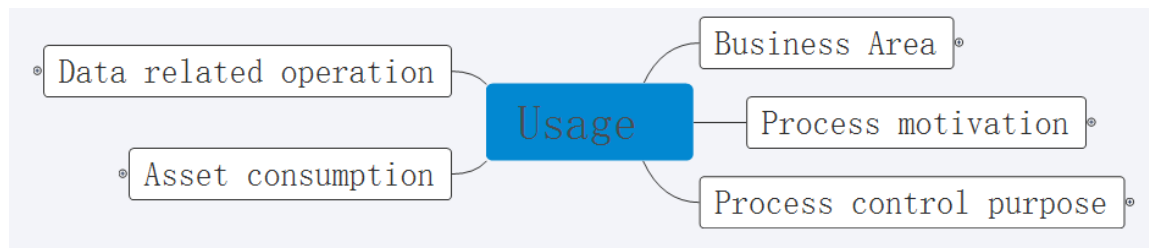


Figure 27 Mind map of the Usage ontology

Following our multi-layer system, usages may refer to one or several layers. Analyzing the threats and vulnerabilities in the business service collaboration is mostly related to the way data are shared by the different actors related to the different business areas (see Figure 28). These “shared usages” are rights associated with data-related operations. These operations lead to generic usage consequences including data exchange, disclosure or replication, etc. Sensitive and value of logical assets having “share” usages result in the threats and vulnerabilities associated with the privacy violation due to these collaboration processes or the actors involved. Integrating these threats and vulnerabilities context implies taking an interest in collaboration purpose which encompass the business

scope (characterized as a business area) and process motivation, including particular motivation associated with analytics or social media related processes. Lastly, particular attention must be paid to usage delegation as such an operation may transfer the data ownership to other parties.

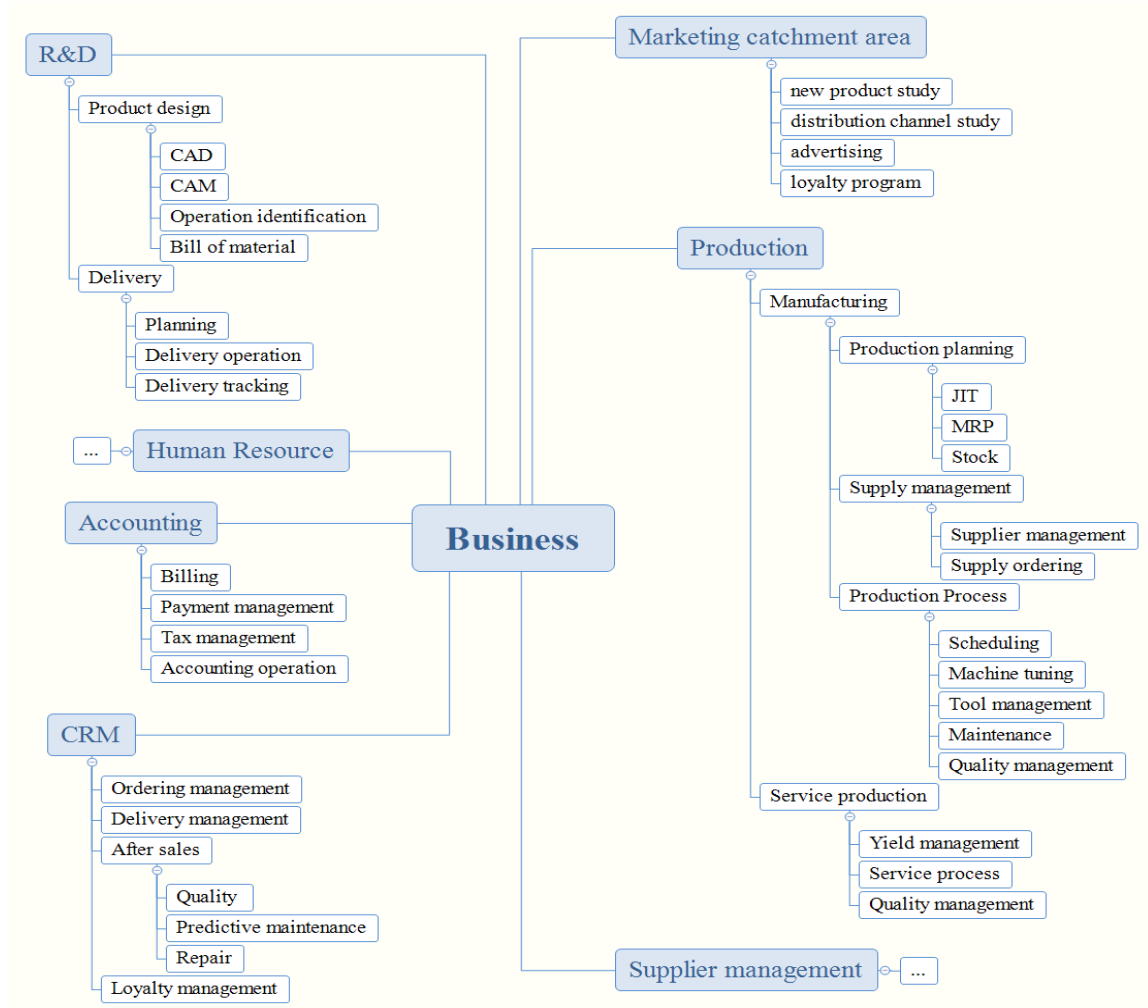


Figure 28 Mind map of the Business area ontology

Then, the business service-IT service-specific usage is operated in the logical layer. These logical usages define more precisely the way data are processed to achieve a given data related operation. For example, data disclosure can be specified into show, track, read and spread. Usage delegation constraints from the inherited business service and readable form of containers having specific processing usages result prevent data leakage or loss from threats and vulnerabilities because of application context and support state of IT service. Integrating the deployment context involves paying attention to infrastructure services such as those provided by cloud computing and mobile support. Protection countermeasures can be defined to manage the different security services that may be affected by these usage operations.

At last, the concrete service-IT-application implementation describes the internal infrastructure executions, leading to specify “physical usage”, i.e., atomic usage operation

supporting a logical usage. Physical usage consists of a copy, write, move operations on the data physical instances (containers). Dedicated countermeasures can be set to mitigate software, hardware, and data delivery channel vulnerabilities and threats.

More precise definitions of the process motivation can be found in Table 7, process purposes are set in Table 8, and asset consumption operations are provided in Table 9.

Process motivation	Definition
<b>Data discovery</b>	Find and collect raw data (internal data of an organization or external data received/bought from partners or other actors) to support the analytics process
<b>Data visualization</b>	Model and visualize data to present valuable insights to various participants in an understandable form.
<b>Data processing</b>	Transform and execute data. It helps to use data operation and data cleaning to generate data sets/stacks or a needed format for a specific process.
<b>Data exploitation</b>	Analyze and mine data. It helps to get valuable or useful insights from the data by performing specific algorithms.

*Table 7 Process motivation*

Process control purpose	Definition
<b>Preparation</b>	Define a logical service to indicate the development scope of its inherited elementary task. It will inquire about containers before the elementary task which can link different elementary tasks for a business service.
<b>Execution</b>	Define a logical service to indicate the development scope of its inherited elementary task. It will operate containers during the elementary task.
<b>Governance</b>	Define a logical service to indicate the development scope of its inherited elementary task. It will manage the output containers after the elementary task.

*Table 8 Process control purpose*

Asset Consumption	Definition
<b>Sell</b>	Delegate the ownership of a logical asset to the external actors (for money)
<b>Share</b>	Delegate a given right on a logical asset to the external actors (being paid or not)
<b>Refine</b>	Delegate a given right on a logical asset to local actors
<b>Revoke</b>	Revoke the delegated right on a logical asset from external actors or local roles
<b>Transfer</b>	Exchange the content associated to a logical asset (send a container)
<b>Withdraw</b>	Remove the exchanged logical asset and all its associated containers
<b>Modify</b>	Change the value of a logical asset
<b>Update</b>	Change/Derive the new value of the content of a logical asset
<b>Store</b>	Register the logical asset and its content
<b>Spread</b>	Publish (without access control) the value of the logical asset
<b>Show&amp;Track</b>	make available the value of a logical asset to the authorized actors and send an alert when the value of the asset is changed
<b>Read</b>	Provide the value of the logical asset to the authorized actors
<b>Generate</b>	Use a logical asset to create a new one thanks to a dedicated process including analytics processes
<b>Analyze</b>	Use existing assets to extract “hidden value” thanks to analytics and mining processes which may be irrelevant to the logical asset
<b>Extract</b>	Split the logical asset into multiple sub logical assets
<b>Aggregate</b>	Compose sub logical assets into a global logical asset
<b>Copy</b>	Generate a new physical copy of the content associated with a logical asset
<b>Write</b>	Define/update the content of a physical copy (container) of a logical asset
<b>Move</b>	A container is created and attached to a new logical asset by copying and deleting a previous one.
<b>Delete</b>	Delete a container
<b>Execute</b>	Transform a container

*Table 9 Asset consumption operation*

As security breaches may also be due to the way assets are processed and shared, we assume that Terms of Usage must be defined, providing a consistent consensus mixing Requirements of Protections of the data provider and the Quality of Protection / Terms of Services promised by the data consumer. QoP policies are used to express a data consumer’s predefined promises about the protection it offers to the assets it requires. ToS

assertions are also provided by the data consumer, declaring the way data assets may be / will be used under different conditions and legal environments. This requires defining a Usage-based assertions model to manage consistently usage and their related protection means.

### 3.2.2 Usage-based policy model

Our usage ontology and protection goal ontology offer a vocabulary and knowledge set to describe the objects and properties in the real world that should be regulated in policies. To provide assets consistent protection, our RoP/QoP/ToS model is built using our multi-layer IS Description model, allowing us to mix business and technical knowledge to set convenient security protection. Coupling these business and technical dimensions to define policy assertions allows investigating threats and vulnerabilities conditioned by business service collaboration, business service-IT service-specific usage, and IT service application implementation. Protection consistency is continuously checked by comparing the data owner's Requirement of Protection (RoP) and the data consumer Quality of Protection (QoP) / Terms of Service (ToS).

Our policy model (see Figure 29) consists of a set of assertions. Each assertion is designed to protect an asset, either a logical data object or a container, fitting its sensitivity level. The usage refers to the way the asset will be processed, including the allowed actions. The data object and container will enrich the usage with specified purposes and the environmental conditions. Last but not least, the subject identifies the actors that will be allowed to achieve the processing actions depending on the context (i.e. the actor under the responsibility of whom the actions will be executed).

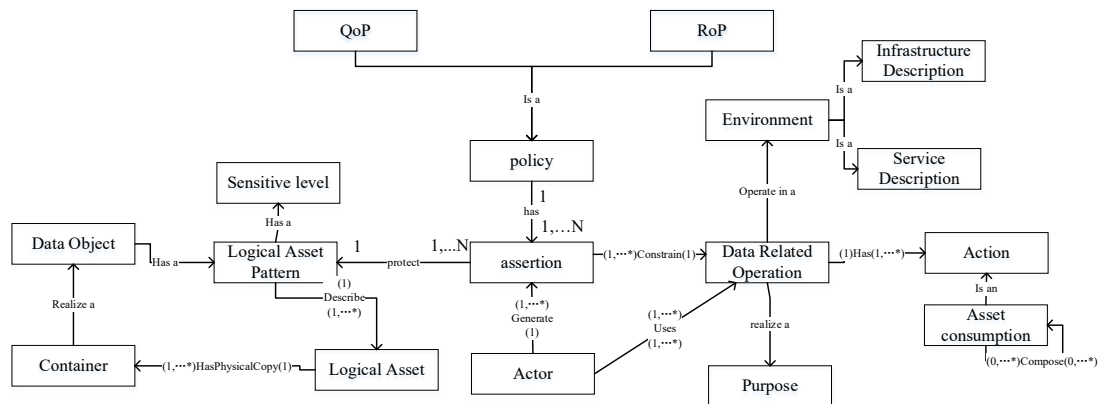


Figure 29 Class diagram of the policy model

Based on this model, RoP or QoP/ToS are defined as a set of multiple assertions. Each protection assertion is defined as a proposition—an expression of some property of the IS Description model whose usage protection can be measured by examining the asset state and checking that the enforcement of the countermeasures and the usage operations are consistent with each other. Taking advantage of our IS Description model and our protection ontology, RoP assertions include

- Data sensitivity level for each security service. These sensitivity levels are defined using a 4-level scale (No need/low/medium/high)
- Usages defined according to Business Scope and data-related operations

Focusing on the QoP and Usage policies, assertions are related to

- Protection countermeasures related to the different security services. Similarly to the RoP, we integrate a 3-level scale to assess the protection efficiency of the countermeasure (low/medium/high). We enrich our protection goal ontology with countermeasures efficiency. Figure 30 focuses on the protection in the organizational layer. Figure 31 focuses on the protection in the Logical layer. At last, Figure 32 concentrates on protecting the implementation layer.
- Usages are defined according to Business Scope, usage operations, and the contextual protection they propose.

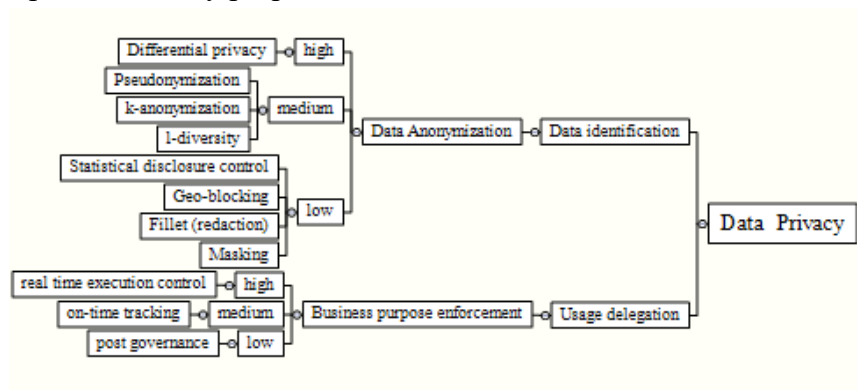


Figure 30 Mind map diagram of the privacy protection goal and associated countermeasure efficiency

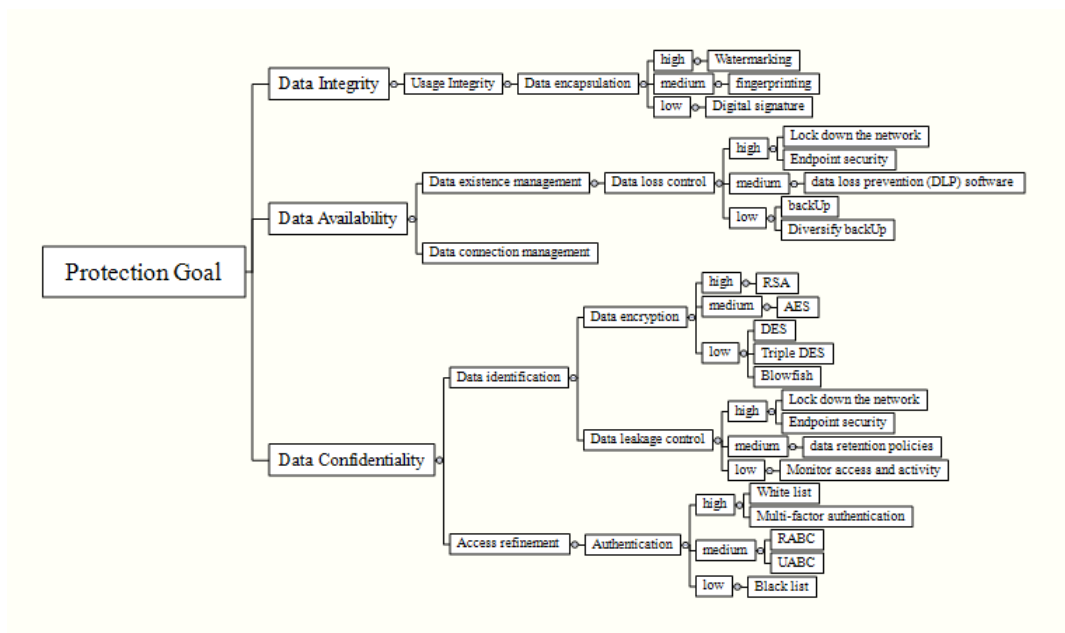


Figure 31 Mind map diagram of the security services protection goals and countermeasure efficiency

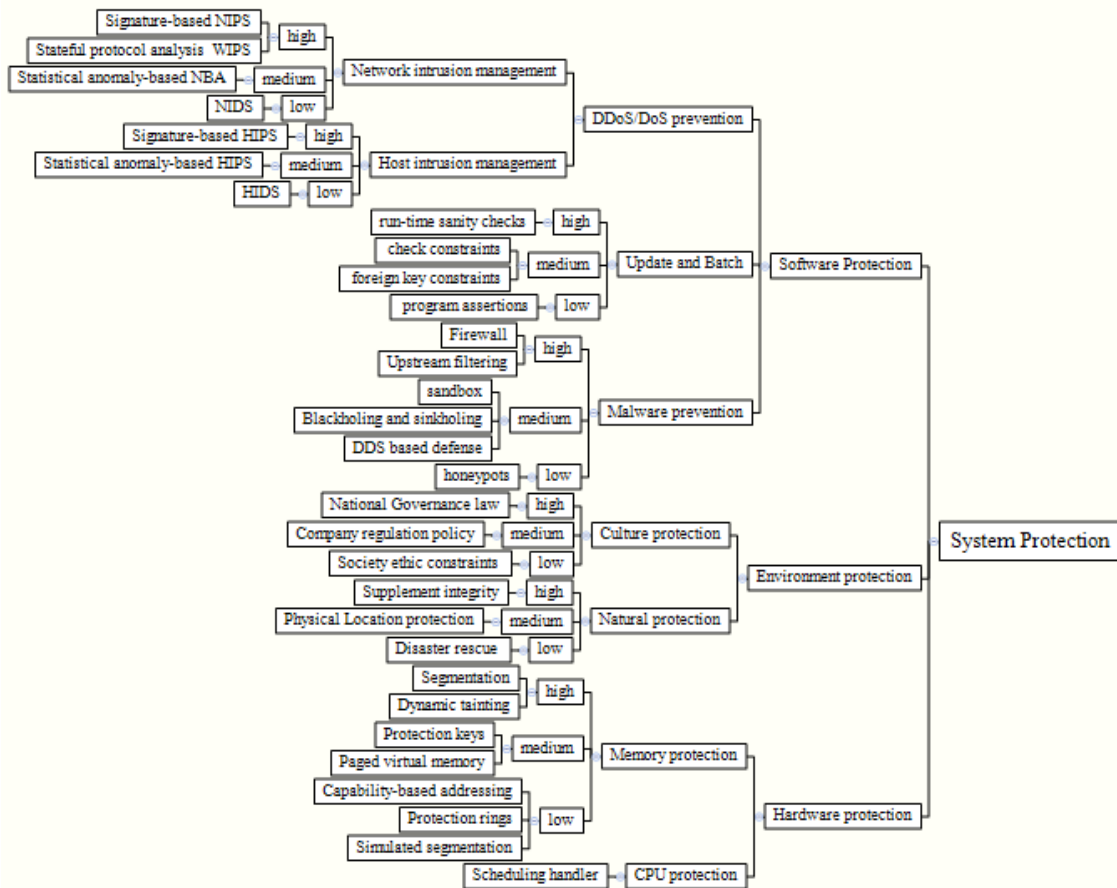


Figure 32 Mind map of the Physical infrastructure protection and countermeasure efficiency

Terms of Usage policies are used to gather usage-based assertions. These ToU can be seen as a contract linking a data provider and a data consumer defining the approved usages by the data provider, defined as consent.

A usage model is defined as a t-uple (S, O, U) describing the subject(S) that will consume an object (O) in a usage (U) way. We take advantage of our multi-layer IS model to identify and manage different usage operations. Focusing on the organizational layer, the usage model defines business services collaboration. An actor (S) consumes a logical asset (O) operating a business service operation(U). The business service operation includes its purpose (business area and process motivation), an operation method (asset consumption operation (business usage type)), and the operation context (data-related operations). Focusing on the way the Business Service implementation, Logical Services operating for a Business Service Interface Object (the Subject) consume Data Objects (the object O) according to the logical service operation (U). In this logical layer, the logical services operation includes their scope according to the business service (process control purpose), operation method (asset consumption operation (logical usage type)), and operation context (IT/manual functional service description). This allows deriving asset consumption operations from the data-related operations according to dependencies



identified from our protection ontology. Similarly, logical services operation context requires infrastructure support from concrete services. Concrete Services operating for Logical Service Interface objects (S) consume containers (O) according to the concrete service operation (U). The concrete services operation includes operation method (asset consumption operation (physical usage type)) and operation context (infrastructure environment). In this way, logical usages are defined as a composed usage that is implemented by physical usages. This generic usage-based model, providing usage-based authorization, is also enriched with security countermeasures/protection means.

This Terms of usage consent, signed by two parties, gathers a description of both data assets, their potential usages with consensual protection level requirements, and protection countermeasures provision. This model leads to a Usage Control Assertions (UCA) doubled approved by the data provider (who will provide the asset) and by the data consumer (who will use the asset). As usage can be defined globally, an assertion development process is set to allow the data consumer to generate more precise usage assertions from an original consent. Each of these assertions is defined as a t-uple (Equ. 2):

$$\text{(Equ. 2) } UCA = (AS, AO, S, O, U, PO, OSP, CSP)$$

- AS defines the assertion status, i.e., if the assertion is originated from the original consent or has been inferred from another assertion
- AO defines the asset owner related to the rule. This owner is specified as a set of two attributes:
  - o Assignee attribute defines the organizational entity or simple user owning the asset
  - o Assignee status defines if it is the original owner or a delegate representing the owner
- S is the subject, i.e., the party that will get the right on the asset. This subject can be an organizational entity, a simple user, or an IT business service in charge of a part of a business process. Similar to the asset's owner description, it includes 2 attributes:
  - o Assignee attribute defines the organizational entity or simple user that may use the asset
  - o Assignee status defines if it is the original owner or a delegate representing the initial asset consumer
- O is the object, i.e., the exchanged asset having Logical asset pattern attribute relating to logical asset, data object and container whose usage is regulated by the assertion. This exchanged asset is associated with a unique identifier shared by both parties and related to the corresponding logical assets stored in each party information system
- U is the usage purpose regulated by the assertion. It is specified as a set of attributes:
  - o BuP denotes the business purpose. It can refer either to a business area or to a more precise business activity
  - o PrP denotes the process purpose (including the process motivation and the process control purpose)
  - o ACP denotes the asset consumption purpose, it can refer to a physical usage type (such as copy, write...) or to logical usage type defined as

transfer, modify, store, show... or to business usage type (such as share, revoke...and relating to the data related operation (data fusion, data exchange, data replication...))

- PC is the Protection Context. It is defined by a set of attributes:
  - PG denotes the protection goal, i.e. the security service (confidentiality, availability, integrity, non-repudiation) or the quality of service that must be provided by the subject
  - CCtX denotes the countermeasures context, i.e. the set of countermeasures that must be deployed to provide consistent protection.
- OSP defines the Asset Owner Signing Party. It is specified thanks to 2 attributes
  - OSC is the signing owner authentication certification. When an assertion is inferred from a global one, this attribute refers to the “parent assertion”
  - OSK is the owner's signature parameters.
- CSP defines the Asset Consumer Signing Party. It is specified thanks to 2 attributes
  - CSC is the signing consumer authentication certification. When an assertion is inferred from a global one, this attribute refers to the “parent assertion”
  - CSK is the consumer signature parameter.

From our motivating example, when Alice consents to share contact information with online shopping (OLS for short) while ordering a product as a gift to be delivered to her brother Bob, two usage assertions are originally defined:

- One is defined for Alice’s contact information that will be used by the ordering and billing processes (Equ. 3)
- The other is defined for Bob’s contact information that will be used by the delivery process. This assertion accepts that this last asset can be shared with other parties involved in the delivery process (Equ. 4).

(Equ. 3)  $UCA1=(AS=Original, AO=Alice, S=OLS, O=Alice's\ contact\ information\ Id, U-BuP=Order, U-ACP=Read, PC-PG=Confidentiality+integrity, OSP-OSC=Direct, OSP-OSK=Alice\ key, CSP-CSC=direct, CSP-CSK=OLS\ key)$

(Equ. 4)  $UCA2=(AS=Original, AO=Alice, S=OLS, O=Bob's\ contact\ information\ Id, U-BuP=Deliver, U-ACP=share, U-ACP=show\&\ track, PC-PG=Confidentiality+integrity, OSP-OSC=Inferred, OSP-OSK=Bob's\ consent\ assertion\ key, CSP-CSC=direct, CSP-CSK=OLS\ key)$

Online Shopping infers these assertions to generate sub-consent assertions associated with more precise activities and sub-contractors. For example, online shopping will extract the Deliver Business Usage Purpose from UCA2 to select the corresponding Business Process and its related sub-processes from its information system Description model. This extracted Delivery Business Process consists of two sub-processes: the order delivery preparation process managed by OnLine Shopping and the Client Final Delivery process managed by Company B. As UCA 2 includes a “delegate” right, a new usage assertion formalizing the inferred consent between OnLine shopping and Company B can

be created. This assertion will allow Company B to process Bob's contact information, paying attention to Confidentiality and Integrity requirements. This last consent will be signed according to UCA2 and by company B (Equ. 5).

(Equ. 5) UCA3=(AS=Inferred, AO=OLS, S=CompanyB, O=Bob's contact information Id, U-BuP=Deliver, U-ACP=show&track, PC-PG=Confidentiality+integrity, OSP-OSC=Inferred, OSP-OSK=UCA2-key, CSP-CSC=direct, CSP-CSK=CompagnyB key)

### 3.2.3 *Data-centric protection management*

Managing consistent and life-long protection for data assets involves controlling the way copies of these data assets are protected and consumed. To this end, we define a usage protection governance including pre-protection and post-evaluation. Pre-protection refers to the Terms of Usage management and post-evaluation relies on usage governance. Each time a Business Transaction is initiated, the service provider must define the global protection level and potential usages for these different assets. Similarly, the data provider has to define the global RoP associated with the required set of assets. It must be defined by aggregating the sensitivity level and particular protection requirements of each asset. To this end, we define different policy aggregation mechanisms to identify consistent Requirements of Protection and the current Quality of Protection of a given asset. We take advantage of our 4-level protection sensitivity and 3-level protection efficiency to manage this aggregation process. As far as usages are concerned, our business ontology is used to manage the inclusion of different business areas/business scopes whereas relationships among data-related operations / logical operations and physical operations (see Figure 33) is used to evaluate "usage inclusion".

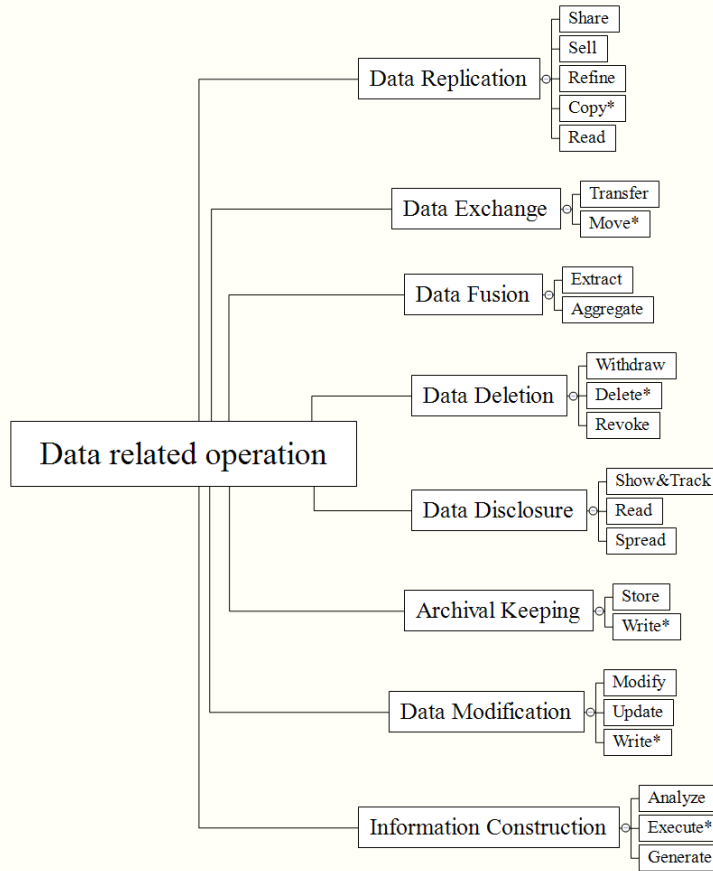


Figure 33 Mind map of the usage operation inclusion relationships

Formally, aggregating two policies  $i$  and  $j$  involves aggregating each assertion of the policies. To this end, assertions are classified according to the subject and the purpose. To build the resulting policy  $P$ , we use a 2-steps algorithm, adding the first policy's assertions after aggregating them with the similar assertion of the second policy, and then adding the remaining assertions from the second policy (see Algorithm 1). Regarding the assertion aggregation strategy, we define two strategy:

- Negative aggregation keeps the most restrictive value picked from both assertions (this is a deny by default strategy). It consists in keeping the most precise or the highest sensitivity / protection efficiency level
- Positive aggregation keeps the laxest value picked from both assertions (authorize by default). It consists of keeping the less precise usage or the lowest sensitivity/protection efficiency level.

*Algorithm 1 Policy aggregation algorithm*

---

**Input:** Policy  $P_i$  to be aggregated in policy  $P_j$ , aggregation strategy (lax or strict)

**Output:** Aggregated policy  $P$

---

- 1 Function Aggregate ( $P_i, P_j$ , aggregation strategy) /\*  $P_i$  and  $P_j$  are the two policies to aggregate \*/
- 2 Policy  $P = \text{null}$

```

3  For each assertion n belonging to  $P_i$  (noted  $A_{i,n}$ )
4      Subject  $\leftarrow A_{i,n}.subject$ 
5      Purpose  $\leftarrow A_{i,n}.purpose$ 
6      Select Assertion from  $Policy_j$  where Assertion.subject=subject and Assertion.purpose
=      Purpose
7      If no assertion is found,
8          Create a new assertion A
9           $A \leftarrow A_{i,n}$ 
10         Add assertion A to the policy P
11     Else
12         Create a new assertion A
13          $A \leftarrow$  Assertion-aggregation ( $A_{i,n}, A_{j,k}$ , aggregation strategy)
14          $A_{j,k}$  status  $\leftarrow$  Processed
15         Add assertion A to the policy P
16     End if
17 End for
18 For each assertion k belonging to  $P_j$ (noted as  $A_{j,k}$ )
19     If  $A_{j,k}$  status  $\neq$  processed
20         Create a new assertion A
21          $A \leftarrow A_{j,k}$ 
22         Add assertion A to the policy P
23 End For
24 Return(P)
25 End Function

```

---

As said previously, Requirements of Protection are associated with logical assets. Taking advantage of the inclusion relationship linking logical assets, RoP can be propagated according to a 2-steps process: a forward process selects all the logical assets to propagate the global RoP. Once adjusted, RoP is identified for each atomic data asset, a backward process is launched to propagate this RoP to the “embedding” logical assets. At each step, security is enforced by selecting the negative aggregation strategy. Algorithm 2 presents the way a RoP policy is generated for an asset  $i$ .

*Algorithm 2 Requirement of protection generation algorithm*

---

**Input:** Logical asset for which the requirement of protection policy must be defined

**Output:** RoP policy P

---

```

1  Function CreateRoPPolicy (LogicalAsset)
2  Select LogicalAssetPattern LAP from LogicalAsset.type
3  Select RoPPolicy from LAP
4  Create RoP policy
5  RoP  $\leftarrow$  RoPPolicy
6  If LogicalAsset is not an Atomic asset
7      Select all logical assets j included LogicalAsset
8      For each LogicalAssetj involved in LogicalAsset
9          Select LogicalAssetj RoP Policy  $ROP_j$ 
10         If no  $ROP_j$  is found
11              $ROP_j \leftarrow$  CreateRoPpolicy (LogicalAssetj)
12         End If
13         RoP  $\leftarrow$  Aggregate(RoP,  $ROP_j$ , negative aggregation)
14     End For

```

15 **End If**  
16 **End Function**

---

Focusing on the current protection level associated with a logical asset, the protection level is proposed by the business service which consumes the logical asset according to its requested logical asset pattern. Furthermore, logical services will support the business service. To the end, the protection level of the logical asset in a business service depends on authentication to protect the ownership of shared data. Then the data object which characterizes a logical service interface connecting the logical asset pattern with the required container. We use the “logical asset to container” relationship to select all data objects related to the copies of the logical asset for which the current Quality of Protection has to be evaluated and a resulting global QoP is computed by aggregating assertions of this different QoP, selecting the positive aggregation strategy (see Algorithm 3), using the policy assertion aggregation algorithm defined in Algorithm 1.

*Algorithm 3 Evaluation of the current Quality of Protection of a data*

---

**Input:** Logical Asset LA

**Output:** QoP Policy

---

```
1 Function Evaluate-QoP(Logical Asset LA)
2   Select QoP Policy QoP related to Logical Asset LA
3   If no QoP policy found
4     QoP<-CreateQoP Policy
5   End if
6   Select all Data Object k related to logical asset LA
7   For each Data Object k DOk
8     Select QoP Policy QoPDO related to DOk
9     If no QoP policy found
10      Select Container C related to DOk
11      Select QoP policy QoPC related to C
12      QoPDO<- CreateQoP Policy
13      QoPDO<-QoPC
14      Associate QoPDO to DOk
15    End If
16    QoP <- Aggregate (QoP, QoPDOk, positive aggregation)
17  End for
18  Select all Logical Assets LAi involved in LA
19  P<-Create Policy
20  For each Logical asset LAi involved in LA
21    P<- Evaluate-QoP (LAi)
22    QoP<- Aggregate(QoP, P, positive aggregation)
23  End for
24  Return(QoP)
25 End Function
```

---

A Requirement/protection policy matching function (see Algorithm 4) is also added to determine if a service associating the QoP of a logical asset fits the RoP of this logical asset or to check the global consistency of the current protection for a data compared to the requirements of protection).

*Algorithm 4 RoP and QoP matching process*

---

**Input:** RoP and QoP policy to compare

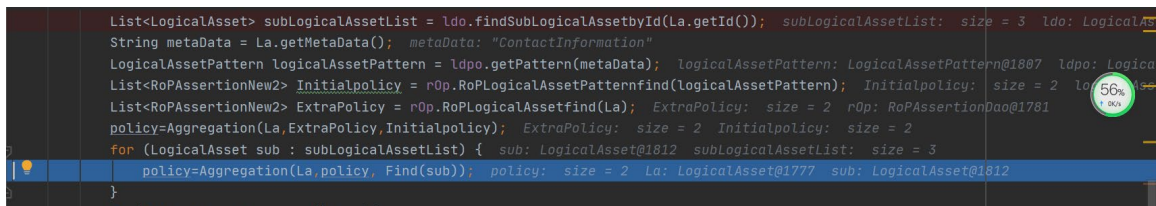
**Output:** Success or Failed

---

```
1 Function RoPandQoPMatching (RoP, QoP)
2 Select all Policy Assertion  $A_i$  from QoP
3 For each assertion  $A_i$  from QoP
4     Subject  $\leftarrow A_i$ .Subject
5     Purpose  $\leftarrow A_i$ .Purpose
6     Select Assertion  $A_k$  from RoP where  $A_k$ .Subject = Subject and  $A_k$ .Purpose
    = Purpose
7     If aggregate ( $A_k, A_i$ , restrictive)  $\ll A_i$  then
8         return(failed)
9     End If
10 End for
11 Return(success)
12 End Function
```

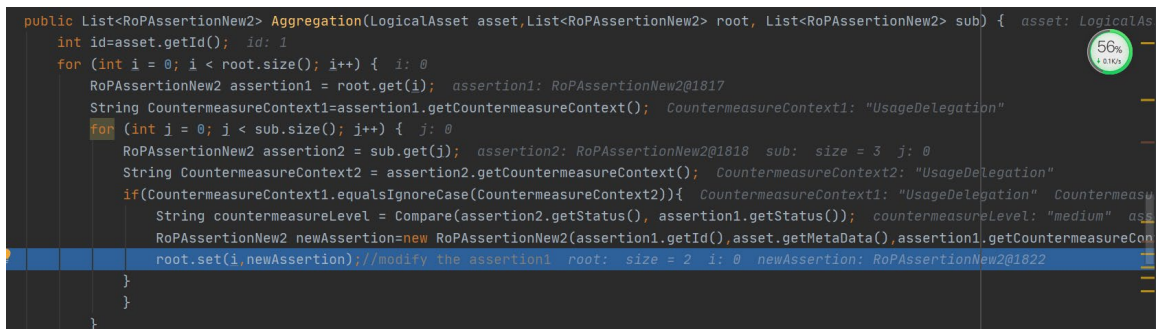
---

From our motivating example proof of concept, we present the logical asset aggregation Figure 34, ROP evaluation Figure 35 and policy matching Figure 36 code fragments.



```
List<LogicalAsset> subLogicalAssetList = ldo.findSubLogicalAssetById(La.getId()); subLogicalAssetList: size = 3 ldo: LogicalAs
String metaData = La.getMetaData(); metaData: "ContactInformation"
LogicalAssetPattern logicalAssetPattern = ldpo.getPattern(metaData); logicalAssetPattern: LogicalAssetPattern@1807 ldpo: Logica
List<RoPAssertionNew2> InitialPolicy = rOp.RoPLogicalAssetPatternfind(LogicalAssetPattern); InitialPolicy: size = 2 lo
List<RoPAssertionNew2> ExtraPolicy = rOp.RoPLogicalAssetfind(La); ExtraPolicy: size = 2 rOp: RoPAssertionDao@1781
PolicyAggregation(La,ExtraPolicy,InitialPolicy); ExtraPolicy: size = 2 InitialPolicy: size = 2
for (LogicalAsset sub : subLogicalAssetList) { sub: LogicalAsset@1812 subLogicalAssetList: size = 3
    PolicyAggregation(La,Policy,Find(sub)); Policy: size = 2 La: LogicalAsset@1777 sub: LogicalAsset@1812
}
```

Figure 34 Logical asset composition code fragment



```
public List<RoPAssertionNew2> Aggregation(LogicalAsset asset,List<RoPAssertionNew2> root, List<RoPAssertionNew2> sub) { asset: LogicalAs
int id=asset.getId(); id: 1
for (int i = 0; i < root.size(); i++) { i: 0
    RoPAssertionNew2 assertion1 = root.get(i); assertion1: RoPAssertionNew2@1817
    String CountermeasureContext1=assertion1.getCountermeasureContext(); CountermeasureContext1: "UsageDelegation"
    for (int j = 0; j < sub.size(); j++) { j: 0
        RoPAssertionNew2 assertion2 = sub.get(j); assertion2: RoPAssertionNew2@1818 sub: size = 3 j: 0
        String CountermeasureContext2 = assertion2.getCountermeasureContext(); CountermeasureContext2: "UsageDelegation"
        if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){ CountermeasureContext1: "UsageDelegation" Countermeasu
            String countermeasureLevel = Compare(assertion2.getStatus(), assertion1.getStatus()); countermeasureLevel: "medium" ass
            RoPAssertionNew2 newAssertion=new RoPAssertionNew2(assertion1.getId(),asset.getMetaData(),assertion1.getCountermeasureCon
        root.set(i,newAssertion);//modify the assertion1 root: size = 2 i: 0 newAssertion: RoPAssertionNew2@1822
    }
}
```

Figure 35 RoP evaluation code fragment

```

public boolean match(String metaData, List<RoPAssertionNew2> rop, List<UsageManagementPolicy> tou) { metaData:
    boolean result=true;
    for (int i = 0; i < tou.size(); i++) {
        UsageManagementPolicy assertion1 = tou.get(i);
        if (assertion1.getAssetDescription().equalsIgnoreCase(metaData)) {
            String countermeasureContext1 = assertion1.getCountermeasureContext();
            for (int j = 0; j < rop.size(); j++) {
                RoPAssertionNew2 assertion2 = rop.get(j);
                String countermeasureContext2 = assertion2.getCountermeasureContext();
                if ((countermeasureContext1.equalsIgnoreCase(countermeasureContext2))) {
                    String countermeasureLevel = Compare(assertion1.getStatus(), assertion2.getStatus());
                    if(countermeasureLevel.equalsIgnoreCase(assertion2.getStatus())!=true){
                        result=false;
                    }
                }
            }
        }
    }
}

```

Figure 36 Policy matching code fragment

As said previously, post-protection refers to usage governance, affecting both the asset life-cycle (i.e. the way an asset is generated, replicated, and consumed) and its security status (secured/compromised / out of control). Considering the data-centered protection concept, two types of events are required for providing security protection policy:

- Information life-cycle events: these events are used to represent the life-cycle of a logical asset by describing how a logical asset is generated, modified, and deleted. According to these events, security protection policy can acquire each initial and target state of a container during the life-cycle transformation while controlling the physical execution with IT protection by managing the concrete service with system protection
- Usage events: these events are associated with the way the asset is consumed. The usage event describes which actor uses which logical asset, for which purpose and with which logical service will be used to manage the business process with usage protection. Usage events may affect the asset ownership (when rights are delegated to other parties) or the security status of the asset (when “unfair usages” occur)
- Security events: these events are associated either with security failure notification or unfair usage detection.

Consequently, a life-cycle state, an ownership state, and a security state are attached to each asset.

The logical asset life-cycle state is associated with different values:

- Created: means that the asset is registered logically
- Instantiated means that a first container is attached to the logical asset
- Replicated means that multiple containers are attached to the logical asset
- Deleted means that all containers attached to the logical asset are deleted
- Unusable means that none of the containers can be read anymore or the asset is expired.

Containers life-cycle are associated with the following states:

- Copied means that content is copied to a new container
- Prepared means that content is written/executed to generate this container



- Secured means that protection countermeasures (such as obfuscation, encryption...) have been deployed on this content
- Transferred means that the container has been moved to the data consumer
- Deleted means that the data consumer has deleted the content associated with this container
- Updated means that the content of the container has been executed/written by the data consumer
- Unusable means that the container cannot be read anymore.

Ownership state refers to two sub-states:

- Ownership origin: defines the original owner of the asset. It can take the different values:
  - o Originated means that the party storing this logical asset has created or generated it
  - o Lent means that this logical asset has been provided for a given purpose (i.e., Business Service). It means that the party in charge of this Business Service can share it with other parties involved in the workflow associated with this business service for this given purpose.
  - o Partly delegated means that this logical asset has been provided by the owner for different business purposes.
  - o Fully delegated means that this logical asset has been provided by the owner without restrictions on usages.
- Usage-related ownership status:
  - o Private means that the actor is the data owner of the asset and do not share this ownership
  - o Shared means that the owner has shared the ownership of this asset with another party
  - o Public means that the owner has widely distributed copies of this asset without any control
  - o Conditioned means that this asset has been provided by another party and that the current asset owner has limited rights to it

Lastly, a security state is used to define whereas the asset is protected or not. It includes

- Secured means that the asset is totally under control and that all the necessary protection means are deployed
- Controlled means that the asset protection is under control and that protection means will be deployed when it may be necessary
- Partly under control means that at least one container associated with this asset is not fully secured
- Out of control means that the asset has been spread and that the security level of its containers cannot be managed

- Compromised means that a security event has affected one of the security services associated with this asset which includes damaged, tampered, leaked, vanished, interrupted conditions

These different states are updated according to the data transfer operation, to the granted authorization provided to data consumers, or any security event issued from processes using data.

When Alice interacts with OLS for the product delivery service, she will be requested to “share” her assets such as account information, delivery address information, and browsing history to OLS. While the OLS provides the protection contract to describe the detailed implementation (which is ToS) and associated quality of protection policy (QoP). Alice will protect the requested logical assets to evaluate the sensitive level from its logical asset pattern and identify an extra protection policy of logical assets referring to its data source. Then she can generate the Requirement of protection policy (RoP) of each logical asset. For example, Alice will generate RoP of delivery address information by aggregating the basic protection policy of delivery address information, basic protection policy of sub-logical asset pattern (such as email address). Meanwhile, if the delivery address information provided by Alice is received from other actors, the RoP of delivery address information will also add the extra protection policy of delivery address information requested by its data provider. Finally, Alice will match the protection contract with her RoP of each asset, and negotiate with OLS to determine the final Terms of usage consent (ToU) to protect each asset used in this product delivery service. The negotiated terms of usage consent will enforce ToS and QoP operating by OLS. When OLS interacts with companyA and other third parties by sharing Alice’s asset, OLS also defines its RoP of the asset while aggregating the RoP from Alice to enforce the operations in companyA, etc.

Comparing our data-driven and usage-based protection control strategy to the protection challenges and requirements from the motivating example, it will promise the consistent protection of each asset used in different business services because of the RoP of the logical asset. Then the RoP of logical asset and QoP of the logical asset in a business service will detail the protection from the business usage of logical asset and physical usage of multiple containers to make logical consistent protection and physical consistent protection. It will support protecting the asset considering its logical asset pattern’s information inheritance during a business transaction in the organizational layer and logical asset’s multiple containers during a physical transaction in the implementation. At last, Alice can control and retrieve each usage on her asset during usage transactions in the logical layer.

### **3.3 Conclusion**

As inconsistent protection is one of the major security breaches against data protection, we have proposed a data-centric and usage-based protection model. This model is designed to manage consistently protection requirements on data assets in opened environments. To this end, we have proposed a multi-layer Information System Description model gathering logical assets, their different copies, and the way they are processed. This model allows defining once data asset protection requirements and “propagating” these requirements of protection to the different replicated contents so that the asset can be

protected consistently. This turns the traditional “process-driven” security risk engineering approach into a data driven strategy, based on the data asset intrinsic value. To ease this data centered protection requirements definition, we have proposed to set these requirements for each basic security services (namely confidentiality, Integrity and Availability) according to a discrete scale. This answers **question 1: What should be the security strategy to set consistent protection?**

Besides this data-driven protection strategy, “internal” security threats and vulnerabilities must be considered in opened organization. In fact the way a data asset is used may also be a major security breach. This has lead us to integrate “fair usages” in the data asset security policy. To this end, we have expanded the Collaborative Usage Control model, integrating business knowledge and defining different kinds of operation in our protection ontology so that fine-grained usages can be defined including the way an asset can be used to face particular SMACIT risks. A fine-grained usage model is built to derive data usage authorization from generic consents. By setting a process that generates fine-grained usage assertions from a global consent, users can have a more comprehensive view of the rights they grant while protecting efficiently their assets. These Terms of Usage assertions are stored locally by each party (data owner and data consumer) in their own Information System description model. This fine-grained contextual usage model answers **question 2: How to define fair usages in a protection policy?**

Fitting the GDPR requirements and providing consistent life-long protection needs to expend this model to set a Distributed Usage governance and tracking system allowing both data provider and data consumer to “prove” that data are used and protected according to consents approved by the data provider and the data consumer.

## 4 Data-driven protection architecture

### 4.1 Introduction

Thanks to the definition of the Information System Description model, our data-centered protection model defined in the previous chapter provides a data-asset-centered vision on requirements of protection and on the way these data assets are used and replicated. By extending the traditional security and usage ontologies to capture business knowledge, specific threats and vulnerabilities related to SMACIT due to (potential) unfair usages can be integrated into the protection policy specification. Providing life-long protection on data assets requires defining a Data-driven Usage-based Protection architecture to provide usage-based protection governance for both data providers and data consumers.

Whereas GDPR empowers data providers to control the way their data are used and protected, data providers have to manage consistently their asset protection and the authorization they grant to different services. Focusing on the data provider, we identify two main requirements:

- Managing consistent Requirements of Protection: characterizing the value associated with each data asset is a key point to identify the main requirements associated with the different security services (namely confidentiality, integrity, and availability). These requirements of protection must be fulfilled whether the protected data is included in a group of the asset or managed individually. The consistency of the requirements of protection must be checked each time a Term of Service is approved to verify that the “new authorizations” will not lead to inconsistent protection
- Usage control and governance features aim at providing the data provider trusted information on the authorized operations for a given asset. Integrating these operations in the usage governance loop will allow the data provider to check whether the operations fit the global consent he has provided or not. This will also make the data provider properly know the status of its logical assets and containers, supporting the life-long protection governance.

Focusing on the data consumers, the main requirements are derived from the GDPR and are related to:

- *the negotiation and generation of adapted Terms of Services*, integrating the Quality of Protection and potential fair usages necessary to support a Business Process. This requires deriving the initial consent to manage “fair usages” and the involved parties associated with the different services supporting the Business Process execution.
- *the usage control and governance*: a trusted usage-based authorization process must be set and tracked to support the burden of proof requirement. In this way, data consumers can report on the exact usage they made of a given asset. By integrating security events monitoring, this usage governance feature will also report security/protection failures to the data owner, as requested by the GDPR.

To manage these requirements and achieve these goals to answer **question 3: “How to manage the usage proofs to support usage and protection governance?”** we propose

a Data-driven and Usage-based Protection architecture which unifies the Information System description and protection requirements and policies. In this chapter, we present this architecture and its main components before detailing the way these components interact to support the Terms of Usage negotiation, necessary to manage the protection consistency, Usage derivation to control usages, and Usage governance to manage the life-long protection of assets.

## 4.2 Data-driven and Usage-Based Protection architecture

To manage both data owners and data providers requirements, we design a Data-driven and Usage based protection service (DUP for short) service. This service is used by both data providers and data consumers to manage consent and usage proofs (see Figure 37). Our Data-driven and Usage-based Protection service is loosely coupled to each party's information system. This service is designed using a 3-tiers architecture:

- the information system interface provides the entry-point to the DUP to integrate data assets, processes and manage consents and usage operations requests
- The Data-driven and Usage based protection is built on our Information System description model. It provides two main features: on one hand it manages asset requirements of protection and consents and on the other hand it manages usage proofs
- The consent and usage operation persistency tier is implemented in a blockchain. A smart Contract factory is added in the DUP core to manage the interface with the Blockchain.

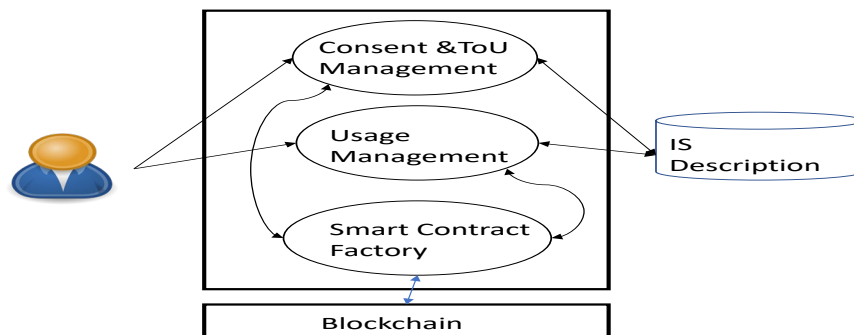


Figure 37 DUP organization

Focusing on the consent management, each time a data owner initiates a Business Process with a service provider, a Terms of Usage (ToU) associated to the data owner's consent is set. To this end, the service provider invokes the ToU generation from the DUP service. This Terms of Usage integrates the different operations and protection means that will be involved in the Business Process. Once established, the ToU is signed by the service provider and sent to the data owner. Our DUP service assists the data owner in the ToU negotiation by evaluating the requirements of protection of the required assets. If the data owner accepts the ToU, he also signs it and invokes its smart contract factory to deploy this initial ToU in the Blockchain. Once the Service provider gets the double approved ToU he launches the transaction derivation process before invoking the smart contract

factory to store all sub-consents derived from the initial ToU. A wallet manager component is added to manage the different keys of the signing parties. Figure 38 presents the global use-case diagram associated to this consent negotiation part.

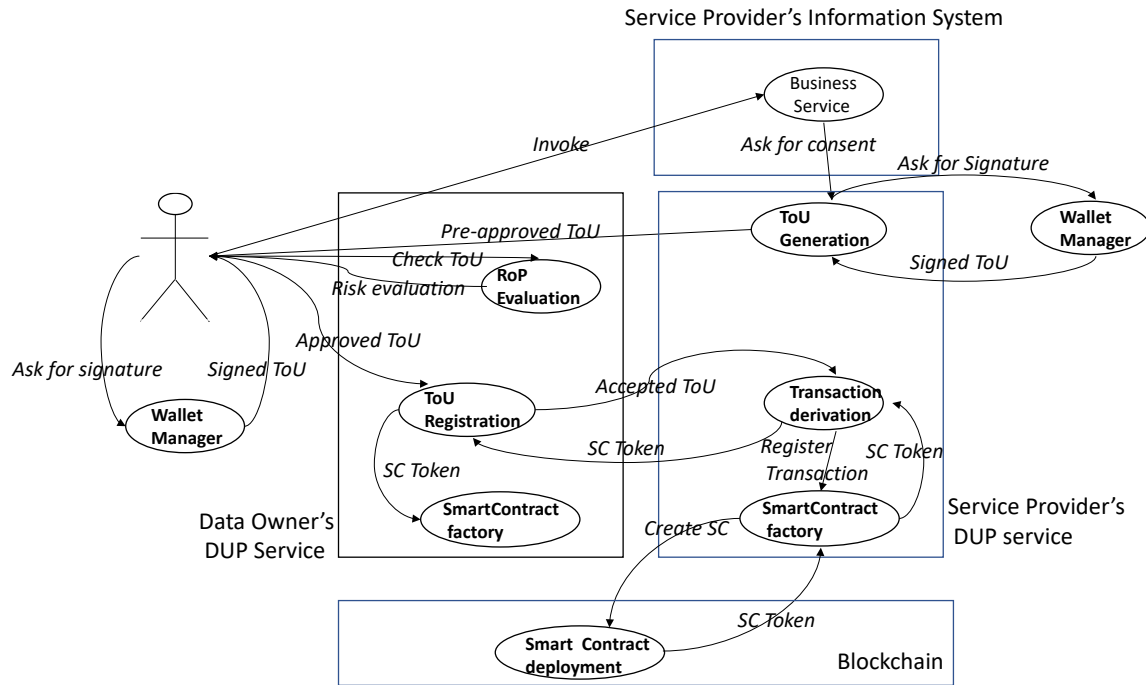


Figure 38 Consent negotiation use case diagram

When a Business Process is executed, the DUP service is used to manage usage approval and register usage operation proofs. When a service operation requests an asset, the usage tracker is launched to provide the associated ToU. To this end, the Usage Tracker interacts with the smart contract factory to get the ToU token and register the operation's proof. By this way, the ToU token can be provided to the data owner while requesting a data asset. Focusing on the data owner side, the data owner can invoke the Usage Monitoring to check a ToU token validity, retrieve operation proofs... so that he can follow the way its assets are used. Figure 43 presents Usage management use-case diagram.

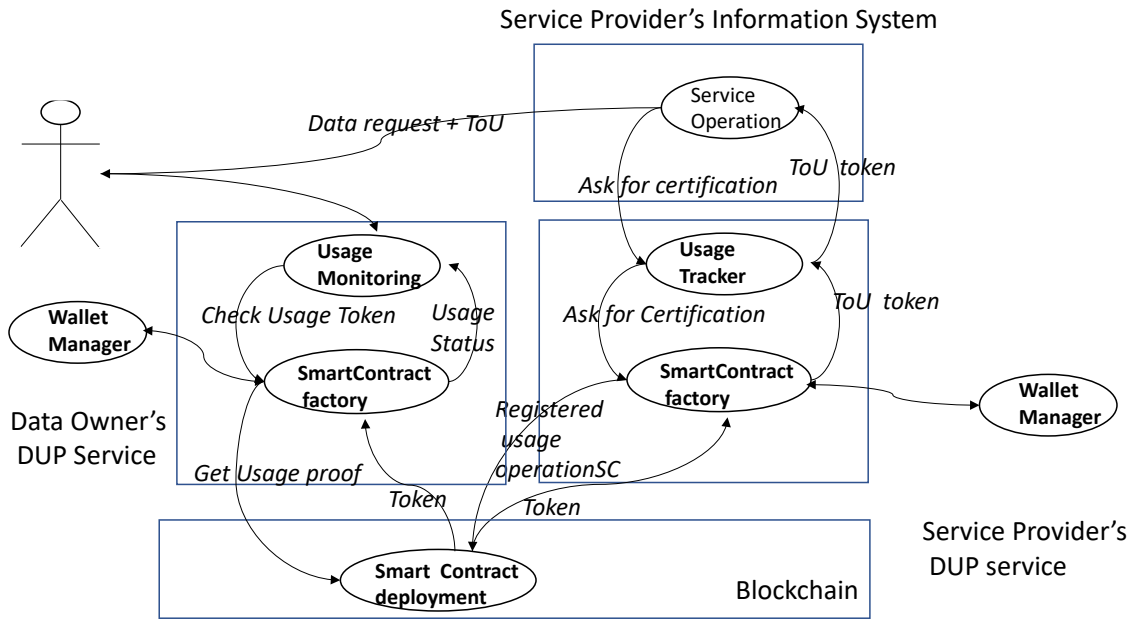


Figure 39 Usage management use-case diagram

This architecture gathers different components providing different functions to data consumers and data providers [111]. We present first its different components before detailing the way this DUP can be deployed and used.

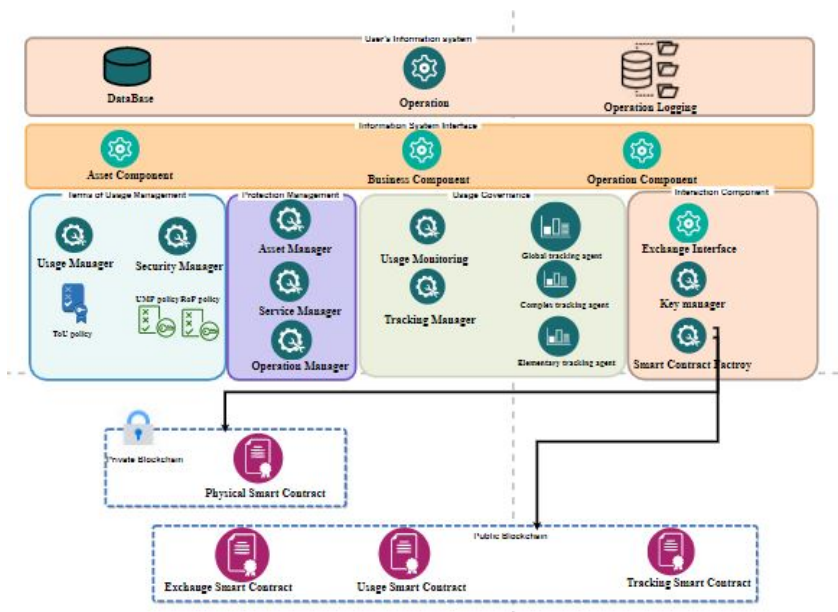


Figure 40 Data-Driven Protection Architecture

#### 4.2.1 Description of the main components

As shown in Figure 40, we use a multi-tier organization for our DUP architecture: the key functions associated with the data provider and data consumer requirements are split into different core components: The Protection Management, Terms of Usage

Management, and usage governance management components. We add an Information System Interface component to plug this DUP on the information system for which assets are protected and the Interaction Interface component to manage the Exchange Interface component which is used for “off-chain” communication and the Smart Contract factory component which is used to provide a single interface to the blockchain component for “on-chain” communication.

The **Information system Interface** is used to manage the interaction with the information system. It is used by both Data providers and Data Consumers to manage the relationship between their Information System description, stored in a dedicated instance of our IS Description model, and their own Information System. This component integrates three sub-components to interact with the real information system:

- *(Data) Asset interface*: it is in charge of capturing the requested data semantic description associated with a data object and defining the associated selection operation to get the proper data from the information system. By this way, it can connect the logical data asset to data elements of the information system. Once invoked, it returns the requested value to the container
- *Business interface*: This component is used to retrieve the services fitting the business requirements/business purpose. It can interact with different information systems depending on the collaboration agreements. To this end, allowed-business purposes are associated with the different partners' information systems so that services can be searched and retrieved from different systems.
- *Operation interface*: This component is in charge of the interaction with concrete services and operations tracking. To this end, it includes log-files management and information extraction features.

The **Protection Management component** is in charge of the consistent protection policy initialization. It is devoted to aggregating assets Requirement of Protection, and service Usage Management Protection based on consolidation feature. It manages the way assets and services are combined to identify precisely which asset is used by which processing element and requires which countermeasure protections. This component provides different functions:

- *Process confidentiality protection*: this function is used by data consumers from the service manager and the operation manager to define which trusted groups can get a precise description of a process, i.e. can be allowed to get the precise identification of the business services and their precise usages. This function can also be used to set dedicated usages and QoP policies for a given service. This function which includes a usage consolidation feature is designed to gather Terms of service (ToS for short) especially data related operations belonging to “hidden services” and the Quality of Protection (QoP for short) consolidation feature to define the laxest protection level provided by the process for the different Logical Asset Patterns it uses.
- *Requirement of Protection management*: this function is used by the data providers from the asset manager to tune their Requirements of Protection (RoP for short) for different assets. This function provides a Requirement of Protection consolidation feature which consolidates the RoP of sub-assets to manage the global RoP.



- An “RoP consistency checking” function is also provided to identify the current protection level of an asset according to the granted usages

The **Terms of Usage management component** is in charge of managing protection consistency. The component is devoted to generating the convenient Terms of Usage, adjusting assets Requirements of Protection and requested services Usage Management Protection. It provides different functions:

- *Evaluation of the Term of Usage* managed by security manager: this function is associated with a particular Business Service that will be used by a given Data Provider. this function interacts with the Protection Management to collect the assets Requirement of Protection and service Usage Management Protection of a given service. While, the service Usage Management Protection will be considered as the draft Terms of Usage, paying attention to the trusted group to which the Data provider belongs. While this function is used to generate ToU by matching assets Requirement of Protection and service Usage Management Protection.
- *Consent approval* managed by security manager: this function is used by data providers and data consumers to adjust, sign the ToU, and register the double approved ToU consent. Once a draft ToU is generated by a data consumer, this consent approval function is launched by the data consumer to sign this. Once the data provider also approves the ToU, its signature is added and this function launches the Smart contract factory component to register this consent in the Blockchain before storing this consent registration in each party IS Description model description.
- *Consent checking* managed by security manager: this function is used by the data provider and data consumer to get useful information related to a registered consent (identified by a Blockchain Smart Contract token which can be a usage-authorization token, usage-operation-authorization token, or access-control-authorization token). This function interacts with the Smart Contract factory to extract the approved ToU associated with this token from the Blockchain.
- *Usage delegation* managed by usage manager: this function is used by Data Consumers as delegators to interact with the Smart Contract factory to generate the different smart contracts related to the usages associated with a given ToU identified by its registered tokens.
- *Usage authorization* managed by usage manager: this function is used by Data Consumers as delegates trying to get a registered token for given usage operations. This function defines the required logical asset patterns, usage operations and associated usage management protection assertions with its authentication signature and interacts with the Smart Contract factory to get the token associated with a given Smart Contract for the usage operations.

The **Usage governance component** is in charge of tracking and evaluating the enforcement of business usage and concrete service according to the terms of usage policy. It provides different functions:

- *Usage checking* by the usage monitoring: this function is used by data providers to check if a usage associated with a registered token is compliant with the approved ToU. This function interacts with the Smart Factory component to check the delegation

certification chain linking a usage-authorization token granted to a logical service to the initial consent.

- *Data Transfer certification* by the usage monitoring: this function is used by the data provider to register that a container is sent according to a data consumer request with a usage-operation-authorization token.
- *Usage tracking* by the tracking manager: this function is used by data consumers to register the access-control-authorization token used by the concrete services while “consuming” a given container.
- *The security monitoring function* by the tracking manager: is used to register security events sent by the infrastructure and captured by the tracking agents.

The **key manager** is in charge of authenticating and managing the keys associated with the different users of our DUP system. This component is implemented in a distributed way:

- *The User Certification component* is deployed on the DUP user side. It manages the user private keys
- *The Portfolio Manager* is deployed on the DUP service. It manages the DUP user public keys.

These two components are connected via a secure VPN-based channel. The key manager provides three main functions:

- *Public Key identification*: this function provides the data provider or data consumer public keys used to control the certification or to encrypt the authorization provided to this user
- *Usage certification*: this function can be used by the data provider or the data consumer. It provides a signature encrypted by the user private key so that the user can be authenticated and the proposed usage can be certified
- *Authorization collector*: this function is used by both the data provider and the data consumer. It uses a private key to decrypt the token

All these functions rely on asymmetric encryption: the user and the DUP component manage pairs of keys:

- *Authentication keys*: the user publishes its public key and uses its private key to encrypt a message to be authenticated. Its public key is used to decrypt this message so that the user can be authenticated.
- *Authorization keys*: the user publishes its public key that is used by the sender to encrypt the authorization token and it uses the private key to decrypt it so that only the user owning the private key can decrypt it and use it.

The **Smart Contract Factory** is used to manage the interactions with the blockchain. Based on the Terms of Usage policy, different smart contracts are used to control the way data assets are exchanged, protected, and processed. The smart contract factory is in charge of generating these smart contracts. Different patterns have been identified and Terms of Usage assertions are used to identify the requested parameters of these patterns. The smart contract factory provides different functions

- *Smart Contract publication*: this function is used by the DUP service on behalf of a data consumer when it represents as a delegator. It interacts with the Usage delegation function from the Usage manager. It consists of generating the exchange, usage, or physical smart contract according to a set of parameters identifying the owner and subject public keys, logical asset patterns definition, usage assertions, and the authorization tokens from the Usage delegation belonging to the initial ToU consent.
- *Usage delegation*: this function is used by the DUP service on behalf of a data consumer when it represents as a delegator. It interacts with the Usage authorization function from the Usage manager to get the assertions of the given usage operations provided by a delegatee. The identity of delegatee has been authenticated thanks to the authentication signatures that Usage authorization function provides. Then, the delegation is evaluated according to the usages defined in the target smart contract (identified by its token). While, it provides an authorization token for the delegatee with the given usage right.
- *Usage certification*: this function is used by the DUP service on behalf of the data provider. It interacts with the Usage checking function from the Usage monitoring. It consists of getting the usage description associated with a token. This function uses the data provider's public key to encode the certified usage so that only the data provider identified in the ToU consent can check this usage.

#### 4.2.2 DUP organization

The DUP architecture is deployed as a Service that can be used by different data providers and data consumers. Each of them has to manage its own Information System and their interactions. The DUP service captures these interactions to manage the consent and usage certification, usage authorizations... and uses the functions provided by its different components to manage the data asset protection and the usage governance.

As the data-centered protection and the usage control rely on Peer-to-Peer transactions between data providers and data consumers, our DUP service has to be deployed by both data providers and data consumers to manage and certify these transactions. To allow deploying it in an asymmetric context, i.e., when a data provider or a data consumer does not use this DUP service, “shadow users” associated with data providers or data consumers who are not registered as DUP users are created. The Information System description of a shadow user is created “on the fly”, i.e. it gathers the descriptions of the assets and processes in the transactions it captures.

More precisely, when a data provider interacts with an untrusted data consumer, a “shadow data consumer” is created. A dedicated User Certification and a User portfolio manager are created on our DUP. A key generation process is launched to create a pair of keys to authenticate this new data consumer. Based on the Terms of Service proposed by the Data Consumer, a basic IS description is set. In this way, DUP will manage locally this “shadow” data consumer until the data consumer deploys DUP and the Terms of Service is used to launch the DUP delegation process. Of course, the reduced vision on the information system of the data consumer does not allow generating business and usage transactions from the original consent. It means that these transactions are created “on the fly” each time the Data Provider has to provide a Logical Asset to a concrete service. These

transactions are approved by the data provider and the “Terms of Service” which act as a delegate of the data consumer.

In a symmetric way, when a data consumer using DUP interacts with a new data provider, a shadow data provider is created. A dedicated User Certification and a User portfolio manager are also created on our DUP. A key generation process is launched to create a pair of keys to authenticate this new data provider. This shadow user will “sign” the ToU and stores the different tokens without checking them.

### **4.3 Using DUP**

#### *4.3.1 Terms of Usage negotiation*

Our Usage-based protection model relies on the constant evaluation of the Quality of Protection and potential usages provided by services to check if this protection fits the protection level required by a given asset according to its sensitivity level. From our IS Description model, logical assets are associated with sensitivity levels. Their requirements of protection are evaluated according to a discrete scale associated with the different security services. Focusing on the process part, services are associated with the Quality of Protection (QoP for short) they promise. This QoP policy is formalized as a set of assertions describing the usages and protection countermeasures. The QoP efficiency regarding the different security services can be evaluated by aggregating the different countermeasures protection efficiency. The Terms of Usage management component (ToU manager for short) is designed to negotiate the protection and control contracts between the data owner and the data consumer (see Figure 41).

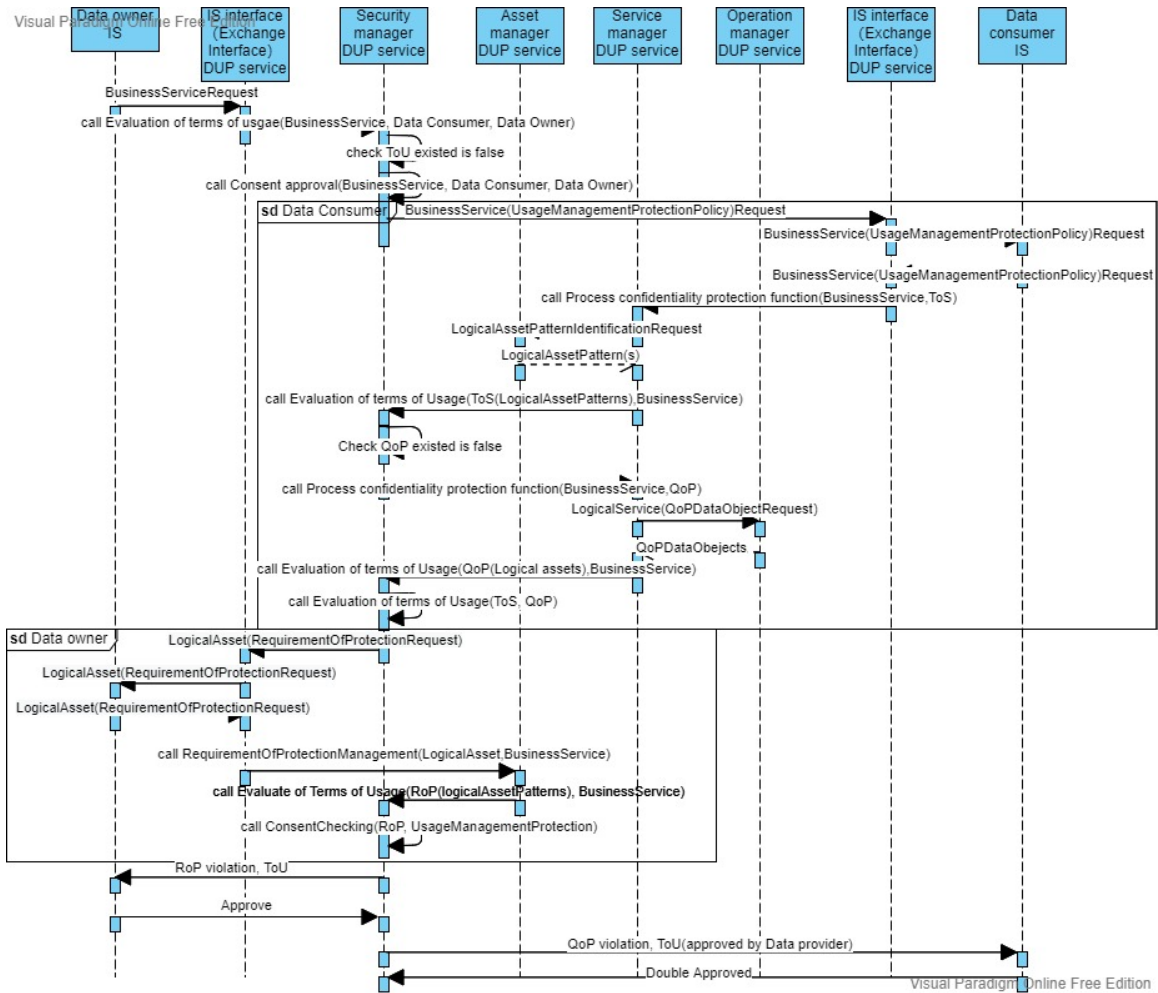


Figure 41 Sequence diagram of the negotiation interaction process sequence diagram

The ToU management component is launched each time when a Business Service requests data. It first checks if a ToU has already been generated for this data owner to support this transaction or not, searching the IS Description model to retrieve transactions linking the requested service and to the data owner. If no transactions are found, the Terms of the Usage negotiation process is launched. The ToU is evaluated according to both Data Owner RoP and Data Consumer Usage Management Policy (UMP for short), including the description of usages and protection. To this end, the Security Manager starts on the data consumer side, by identifying the required assets description (i.e. the associated meta-data) and the associated Terms of Service (ToS for short). This process is managed thanks to three steps:

- First, the Protection management component collects the necessary assets and identifies the way they will be used. To this end, the protection management component uses Service Manager to identify the Logical Asset Patterns associated with the required service as well as the data-related operations requested by the service on this logical asset pattern. It browses recursively the Logical Asset Pattern composition to extract the different meta-data describing

this logical asset pattern (see the code fragment Figure 42). The algorithm providing the draft Terms of Service describing the usages of the required assets (see Algorithm 5) aggregates the ToS assertions of the different assets associated to this pattern, using a lax aggregation strategy (see Algorithm 6).

- Second, the Protection management component checks if a Quality of Protection is associated with the required service or not (see Figure 43). If the service has not yet a defined QoP, the security manager interacts with the operation manager to get recursively all sub-services used to support the required service, requesting the QoP policy / implemented protection means. It takes advantage of our ontology and the evaluation of the protection efficiency associated with the different security means to identify the associated protection level (see Algorithm 7).
- Third, the final Usage Management Policy is generated by the Protection management component (see Algorithm 8). The service QoP and the draft ToS protection are evaluated thanks to the policy aggregation algorithm defined using a lax aggregation rule (i.e., keeping the less protecting level for each protection assertion).

Figure 46 presents the ToS assertions generated for our motivating example. The full process ToS generation process associated to this experiment is presented section 4.4.1.

```
List<LogicalAssetPattern>subLogicalAssetList=LogicalAssetPatternDao.findSubLogicalAsset(lap); LogicalAssetPatternDao: LogicalAssetPattern
for(LogicalAssetPattern lp:subLogicalAssetList) {
    List<BusinessAuthorizationQoP> assertions=ServiceRecursive(lp, businessNode);
    tosFinal = Aggregation(tosFinal,assertions,businessNode, lap); businessNode: BusinessNode@1937 lap: LogicalAssetPattern@1908
}
```

*Figure 42 Extraction of Logical Asset Patterns thanks to the asset composition relationship*

*Algorithm 5 Algorithm describing the evaluation of the draft ToS associated to a service*

---

**Input:** Business Service BS

**Output:** ToS policy

---

- 1 Function Evaluate-ServiceDraftToS(BS)
- 2 **Select** Terms of Service policy ToS associated to Business Service BS
- 3 **If** the ToS policy does not exist
- 4     TOS<-Create Terms of Service policy
- 5     P<-Create Terms of Service Policy
- 6     **Select** all Logical Asset Pattern LAP <sub>j</sub> consumed by BS
- 7     **For** all Logical Asset Pattern LAP <sub>j</sub> consumed by BS
- 8         P ← Evaluate-DataToS(LAP <sub>j</sub>, BS)
- 9         ToS ← Aggregate (ToS, P, negative aggregation)
- 10     **End for**
- 11 **End if**
- 12 **Return**(ToS)
- 13 **End Function**

---

*Algorithm 6 Evaluation of the current ToS assertions for a Logical Asset Pattern  
for a service*

---

**Input: Logical Asset Pattern Data, Business Service Service**

**Output :**

---

```
1  Function Evaluate-DataToS(Data, Service)
2  Select Terms of Service Policy ToS associated to Data and Service
3  If ToS does not exist
4      ToS<-Create Terms of Service policy
5      Associate ToS to Data
6      Associate ToS to Service
7      P<-Create Terms of Service policy
8      Select all Logical Asset Pattern LAP j included in Data
9      For all Logical Asset Pattern LAP j included in Data
10         P ← Evaluate-DataToS(LAP j, Service)
11         ToS ← Aggregate (ToS, P, negative aggregation)
12     End for
13     Select all Business Service Service i included in Service consuming Data
14     For all Business Service Service i included in Service
15         P ← Evaluate-DataToS(LAP j, Service i)
16         ToS ← Aggregate (ToS, P, negative aggregation)
17     End For
18     Select all Data related operation DRO n related to Data and Service j
19     For each Data related operation DRO n
20         Reinitiazlize P
21         A ← Create usage assertion
22         A.DataRelatedOperation<- DROn
23         A.LogicalAssetPattern<- Data
24         A.Service <- Service
25         ToS ← Aggregate (ToS, P, negative aggregation)
26     End for
27     Return(ToS)
28 Else Return (ToS)
29 End if
30 End Function
```

---

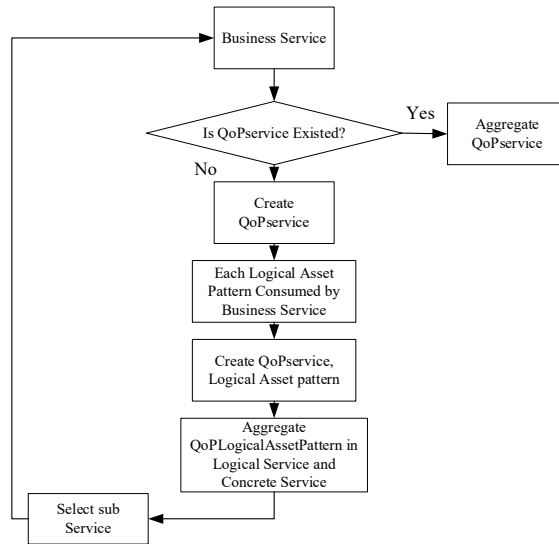


Figure 43 Global Quality of Protection evaluation flow chart

Algorithm 7 Evaluation of the current Quality of Protection of a service

---

**Input: Business Service BS**

**Output: QoP Policy QoP**

---

1. Function Evaluate-ServiceQoP(Service)
  2. **Select** QoP policy QoP associated to Service
  3. **If** QoP does not exist
  4.     QoP<- Create QoP Policy
  5.     Associate QoP to Business Service BS
  6.     Create QoP Policy P
  7.     **Select** all Business Services Service j included in BS
  8.     **For** all Business Service Service j included in BS
  9.         P<-Evaluate-ServiceQoP(service<sub>j</sub>)
  10.        QoP<-Aggregate (QoP, P, positive aggregation)
  11.     **End for**
  12. **End if**
  13. **Return**(QoP)
  14. **End Function**
- 

Algorithm 8 Evaluation of the current ToS for a service

---

**Input: Business Service BS**

**Output: ToS Policy**

---

1. Function Evaluate-ToS(BS)
2. ToS<-Create Terms of Service Policy
3. Associate ToS to BS
4. DraftToS<-Create ToS Policy
5. DraftQoP<-Create QoP Policy
6. DraftToS ← Evaluate-ServiceDraftToS(BS)



7. DraftQoP ← Evaluate-ServiceQoP(BS)
  8. ToS ← Aggregate (DraftToS, DraftQoP, negative aggregation)
  9. Return(ToS)
  10. End Function
- 

```

public List<BusinessAuthorizationQoP> ServiceRecursive(LogicalAssetPattern lap, BusinessNode businessNode) throws SQLException {
    BusinessService bs= businessServiceDao.findById(businessNode.getId()); bs: BusinessService@1898
    List<BusinessAuthorizationQoP> tosFinal=new ArrayList<>(); tosFinal: size = 0
    boolean flag=false; flag: false
    if(qoPDao.isToSExist(lap,bs)==true){
        flag=true; flag: false
        tosFinal=(qoPDao.ToSfind(lap,bs)); tosFinal: size = 0 bs: BusinessService@1898
    }else{
        //using the aggregation algorithm of ToS
        List<BusinessNode> child=businessNode.getBusinesschildren(); child: size = 5 businessNode: BusinessNode@1879
        for(BusinessNode sub:child) { sub: BusinessNode@1895 child: size = 5
            BusinessService subService = businessServiceDao.findById(sub.getId()); subService: BusinessService@1894 businessServ
            //select the subBS consuming logicalAssetPattern
            if (qoPDao.whetherConsumeLAP(lap, subService) == true) { qoPDao: QoPAssertionDaoNew@1887 lap: LogicalAssetPattern@18
                List<BusinessAuthorizationQoP> assertions=ServiceRecursive(lap, sub);
                tosFinal=Aggregation(tosFinal,assertions,businessNode,lap);
                /**it need to aggregate the assertion with the existed assertion and insert to the database.
                 * 1. it will aggregate: process motivation, usagename, data related operation, permission
                 * 2. after the aggregation, it will change the logicalassetpattern and service
                 * to generate a new businessauthorizationassertionQoP
                 * and the new assertion will be inserted in the database.
                 * **/
            }
        }
    }
}

```

Figure 44 Code fragment related to the integration of ToS from sub-services

```

public List<UsageManagementPolicy> AggregationFinal(List<BusinessAuthorizationQoP> root, List<QoPAssertionNew> sub) throws SQLException {
    List<UsageManagementPolicy> UMP=new ArrayList<>(); UMP: size = 0
    for (int i = 0; i < root.size(); i++) { i: 0
        BusinessAuthorizationQoP assertion1 = root.get(i); assertion1: BusinessAuthorizationQoP@1838 root: size = 6 i: 0
        // String UsageName1= assertion1.getUsageName();
        String CountermeasureContext1=assertion1.getCountermeasureContext(); CountermeasureContext1: "DataIdentification"
        for (int j = 0; j < sub.size(); j++) { j: 0
            QoPAssertionNew assertion2 = sub.get(j); assertion2: QoPAssertionNew@1840 sub: size = 5 j: 0
            String CountermeasureContext2 = assertion2.getCountermeasureContext(); CountermeasureContext2: "DataIdentification"
            // String UsageName2=assertion2.getUsageName();
            if((CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2))){ CountermeasureContext1: "DataIdentification" Counter
                String countermeasureLevel = Compare(assertion2.getStatus(), assertion1.getDecision()); countermeasureLevel: "low" assa
                UsageManagementPolicy newAssertion=new UsageManagementPolicy(id: -1,assertion1.getBid(),assertion1.getAssetDescription(),
                UMP.add(newAssertion);//modify the assertion1
            }
        }
    }
}
}

```

Figure 45 Code fragment related to the final aggregation to set the ToS

AssetDescription	qid	ServiceDescription	CountermeasureContext	CountermeasureLevel	UsageName	ProcessMotivation	DataRelatedOperation
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageDelegation	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageIntegrity	low	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataIdentification	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	HardwareProtection	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageDelegation	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageIntegrity	low	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataIdentification	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	HardwareProtection	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	UsageDelegation	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	UsageIntegrity	low	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	DataIdentification	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	HardwareProtection	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	transfer	DataDiscovery	DataExchange
AccountInformation	36	ProductDelivery	DataExistenceManagement	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	UsageDelegation	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	UsageIntegrity	low	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	DataIdentification	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	HardwareProtection	high	Refine	DataProcessing	DataReplication

Figure 46 Partial dump of the data base showing the final Tos assertions generated from the motivating example

Once this Usage Management Policy (which is the current ToS for a service) is generated, the protection management component sends it to evaluate this proposal as potential Terms of Usage via the Exchange interface. On the data owner side, the process starts by identifying the Requirements of protection associated with the required assets. Similar to the ToS evaluation on the data consumer side, this process consolidates the Requirements of Protection of the sub-assets (see Figure 47):

- First, the Protection management component extracts the Logical Asset Pattern description from the ToS and sends it to the Asset manager to identify the “candidate assets” fitting this description. The asset manager selects the Logical Asset Patterns using a similar description (thanks to the meta-data). Based on this selection, it provides a list of candidate logical assets. Interactions with the user or the IS system interface can be set to select the convenient logical assets that will be associated with this transaction.
- The second step is managed by the Protection management component. For each selected asset, it checks the ownership status and extracts the associated Requirements of protection.
- The third step consists of aggregating the different ToS and RoP associated with the required group of assets to set the consolidated RoP. To this end, the Protection management component aggregates the requested assets RoP, using a strict aggregation rule (i.e., the more protecting and more reduced usage authorization strategy) (see Figure 48) to provide a consolidated RoP.

Figure 49 presents the results of the RoP consolidation stored in Alice’s own Information System Description data base. The full process associated to this experiment is presented section 4.4.1

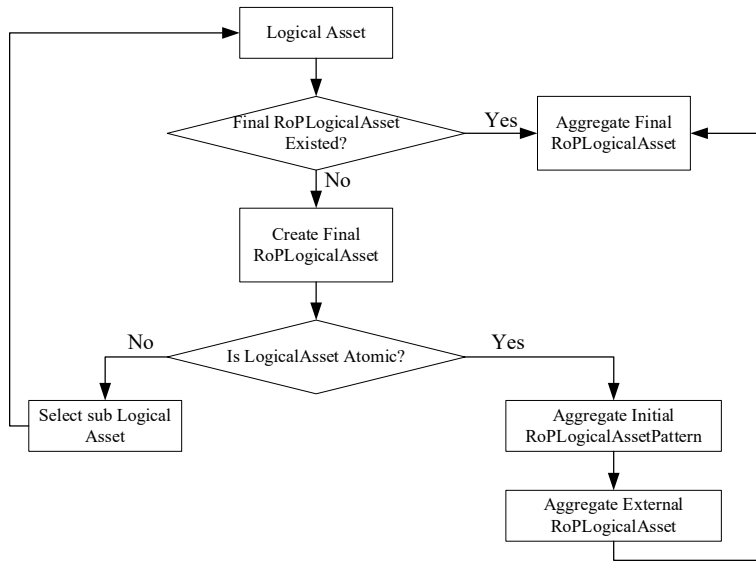


Figure 47 Requirement of protection evaluation flow chart

```

public List<RoPAssertionNew2> Aggregation(LogicalAsset asset, List<RoPAssertionNew2> root, List<RoPAssertionNew2> sub) {
    int id=asset.getId(); id: 1
    for (int i = 0; i < root.size(); i++) { i: 0
        RoPAssertionNew2 assertion1 = root.get(i); assertion1: RoPAssertionNew2@1817
        String CountermeasureContext1=assertion1.getCountermeasureContext(); CountermeasureContext1: "UsageDelegation"
        for (int j = 0; j < sub.size(); j++) { j: 0
            RoPAssertionNew2 assertion2 = sub.get(j); assertion2: RoPAssertionNew2@1818 sub: size = 3 j: 0
            String CountermeasureContext2 = assertion2.getCountermeasureContext(); CountermeasureContext2: "UsageDelegation"
            if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){ CountermeasureContext1: "UsageDelegation" Countermeasu
                String countermeasureLevel = Compare(assertion2.getStatus(), assertion1.getStatus()); countermeasureLevel: "medium" ass
                RoPAssertionNew2 newAssertion=new RoPAssertionNew2(assertion1.getId(),asset.getMetaData(),assertion1.getCountermeasureCon
                root.set(i,newAssertion);//modify the assertion1 root: size = 2 i: 0 newAssertion: RoPAssertionNew2@1822
            }
        }
    }
}
  
```

Figure 48 RoP aggregation code fragment

id	metaData	CountermeasureLevel	CountermeasureContext	lid
54	Address	medium	UsageDelegation	2
55	Address	high	DataIntegrity	2
56	Address	low	DataConfidentiality	2
57	Telephone	medium	UsageDelegation	3
58	Name	low	UsageDelegation	4
59	ContactInformation	medium	UsageDelegation	1
60	ContactInformation	high	DataIntegrity	1
61	ContactInformation	low	DataConfidentiality	1

Figure 49 Partial dump of the database showing the consolidated RoP stored in Alice's Information System description

This consolidated RoP is then compared to the proposed Usage management protection policy (UMP policy for short), using the policy matching algorithm (see Figure 50). This matching process generates a ToU restricting the initial protection to RoP conditions is set. Of course, if the proposed protection does not fit the aggregated RoP, the Data Owner can be notified and may decide to modify the RoP accordingly. Then the Data Owner signs this ToU and sends it for approval to the Data Consumer Usage manager who will sign it to set its approval.

```

public boolean match(String metaData, List<RoPAssertionNew2> rop, List<UsageManagementPolicy> tou) { metaData:
    boolean result=true;
    for (int i = 0; i < tou.size(); i++) {
        UsageManagementPolicy assertion1 = tou.get(i);
        if (assertion1.getAssetDescription().equalsIgnoreCase(metaData)) {
            String countermeasureContext1 = assertion1.getCountermeasureContext();
            for (int j = 0; j < rop.size(); j++) {
                RoPAssertionNew2 assertion2 = rop.get(j);
                String countermeasureContext2 = assertion2.getCountermeasureContext();
                if ((countermeasureContext1.equalsIgnoreCase(countermeasureContext2))) {
                    String countermeasureLevel = Compare(assertion1.getStatus(), assertion2.getStatus());
                    if(countermeasureLevel.equalsIgnoreCase(assertion2.getStatus())!=true){
                        result=false;
                    }
                }
            }
        }
    }
}

```

*Figure 50 Matching process*

Once the ToU is doubled approved, a Business Transaction is created in the data provider IS Description model to store this contractual agreement. Focusing on the data provider side, the exchanged asset is used to encapsulate the Logical asset linking to the Logical Asset Pattern that will be exchanged according to the approved ToU assertions specifying the set of authorized Data related operations and the identification of the Business Service (this service can be created as a local artifact if necessary). The “Usage Protection” relationship is used to link the exchanged asset to the related ToU assertion. On the data consumer side, Logical assets associated with the Logical asset pattern and the data provider are created if necessary. Then these logical assets are associated with the Logical Asset Pattern and the Business Service will process them thanks to the Exchanged Asset. The “Usage Protection relationship” is used to link the Logical Asset with the ToU assertion. By this way, each party updates its IS Description model to integrate this new contractual relationship (see Figure 51). This Business Transaction is associated with the signed initial Consent.

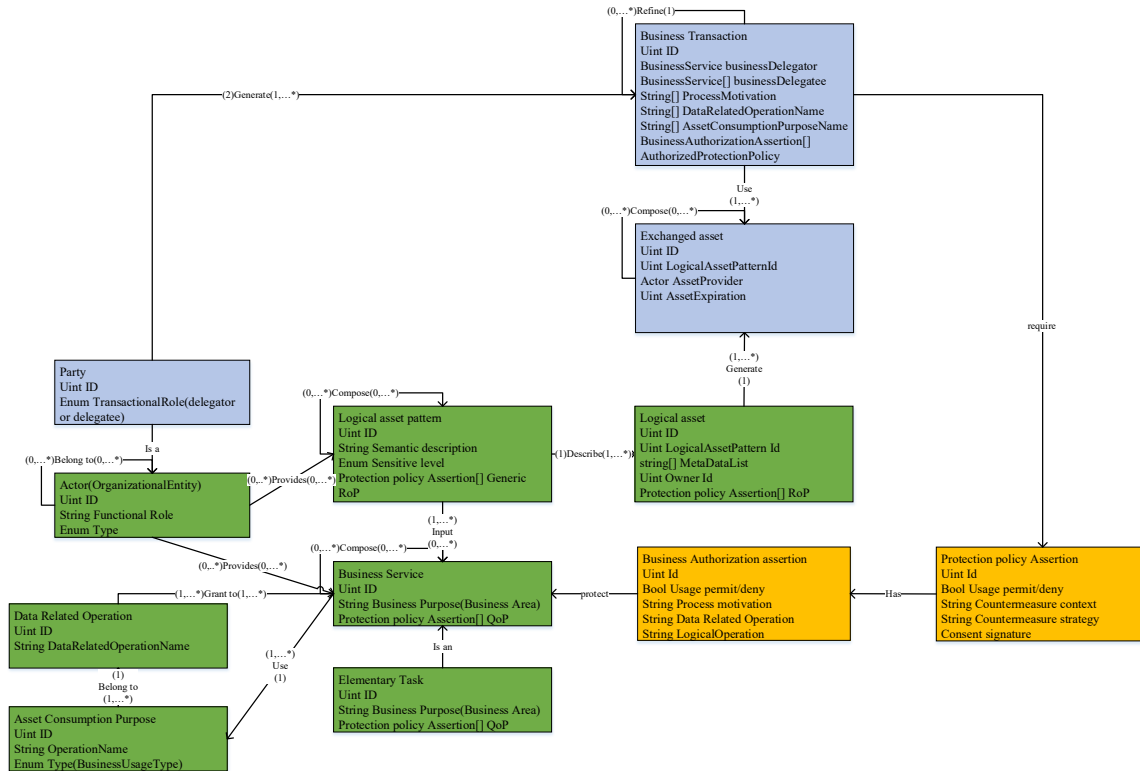


Figure 51 Description of the Business Transaction class diagram

### 4.3.2 Usage derivation

Once the Terms of Service has been approved, the usage derivation process is launched by the protection management component to create the different business, usage, and physical transactions. This process is managed by the data consumer, fitting its Business Process organization. It takes advantage of our usage model defined in section 3.2.2, sees (Equ. 2) which allows linking transactions thanks to the usage “approval delegation”. This usage derivation is managed using three steps:

- Refinement of the Business Service workflow to create all Business Transactions related to sub-business services
- Usage transaction derivation to create the precise usage transaction related to the logical service supporting an elementary task, i.e. a Business Service which does not include any sub-business service.
- Physical transaction derivation is used to generate the physical transactions authorizing the concrete service implementing the logical service to use the physical operation to achieve an authorized usage.

#### 4.3.2.1 Business Transaction refinement

From our IS Description model, generic usages are defined for each business service thanks to Logical Operation, linking the Business service, Logical Asset patterns, and Data related operations, process motivation, asset consumption purpose (which is defined as Business usage type for the organizational layer) that will be used by this

Business Service to “consume” this particular Logical Asset Pattern. The Business Transaction establishes a link between two parties, the data provider and the data consumer who agree that a Business Service (which may include sub-business services) will “consume” a set of logical assets according to precise usages on these assets (namely process motivations, data related operations, and asset consumption purposes). As a Business Service can be defined recursively, the initial transaction defined thanks to the ToU must be “refined” to identify precisely usages on a given asset achieved by a given sub-business service. To this end, we propose a refinement process that first invokes the Terms of Usage Management Component to get the ToU policy. Then this refinement process (see Algorithm 9) invokes the Service Manager to identify the Business Service involved in a given transaction and all its sub-business services. The recursive Business Transaction generation process (see Algorithm 10) is launched for each sub-business service. This generation process creates the Business Transaction associated with this Business Service, selects its embedded business services, and generates recursively the sub-business transactions. Following our model, the generated Business Transaction is associated with their “father transaction” on behalf of which they are created.

*Algorithm 9 Business Transaction refinement algorithm*

---

**Input: Business Transaction BT**

**Output: Success/Failed**

---

1. RefineBT(BT)
  2. Select Business Service BS involved in Business Transaction BT
  3. Get Business Authorization Policy P involved in this Business Transaction BT
  4. Select all Business Service Service  $i$  included in BS
  5. For each Business Service Service $i$
  6.     GenerateBusinessTransaction(P, Service  $i$ , BT)
  7. End For
  8. Return(Success)
  9. End Function
- 

*Algorithm 10 Business Transaction generation process*

---

**Input: ToS policy P, Business Service BS, “Delegating Business” Transaction DBT**

**Output: Created Business Transaction BT**

---

1. Function GenerateBusinessTransaction(Policy P, Business Service BS, Business Transaction DBT)
2. BT<- Create Business Transaction
3. Associate BT to Business Service BS
4. Associate BT to DBT thanks to the “On behalf “relationship
5. Associate BT to the policy P
6. Select all Logical Operation LO  $j$  used by BS
7. **For each** Logical Operation LO $j$
8.     Select the Logical Asset Pattern LAP involved in this Logical Operation LO $j$
9.     Select the Data Related Operation DRO involved in this Logical Operation LO $j$
10.    Select the Process Motivation PM involved in this Logical Operation LO $j$
11.    Select the Business Usage Type involved in this Logical Operation LO $j$

12. Select Exchanged Asset EA referring to Business Transaction BT and to Logical Asset Pattern LAP
  13. **If** EA does not exist
  14.     Create Exchanged Asset EA
  15.     Associate EA to Business Transaction BT
  16.     Associate EA to Logical Asset Pattern LAP
  17.     **End If**
  18. Select all Policy assertions Ak from policy P referring to Logical Asset Pattern LAP and Data Related Operation DRO and Process Motivation PM
  19. For each policy assertion Ak
  20.     Select the Logical Asset LA protected by Policy Assertion Ak
  21.     **If** EA is not associated to LA
  22.         Associate EA to LA
  23.     **End if**
  24.     **End For**
  25.     Associate EA to DRO and Policy P
  26. **End for**
  27. **Return(BT)**
  28. **End Function**
- 

#### 4.3.2.2 *Usage Transaction generation process*

Once the Business Transactions are generated, the Protection management component launches the usage transaction derivation process. This process takes advantage of the Usage Transaction model picked from the Information System Description model (see Figure 52). This process retrieves all Business Transactions associated with a ToU policy before launching the usage transaction generation process for each of these retrieved Business Transactions. The usage transaction generation process checks if the Business Service involved in the Business Transaction is defined as a workflow or as an elementary task. For each elementary task, the corresponding logical service is selected and the associated usage operations are retrieved. The usage operation links Data object, Logical service, asset consumption purpose (which is described as Logical usage type for the Logical layer), and process control purpose to the business transaction. The usage transaction generation process uses both the IS Description model knowledge to extract the data object to which the asset consumption purpose operates and the Business Transaction description to identify the exchanged asset associated with this data object via Logical Asset Pattern for this transaction. Once all information is fixed, the usage transaction is created. Once all information is fixed, the usage transaction is created and related to these different elements (see Algorithm 11 and Algorithm 12).

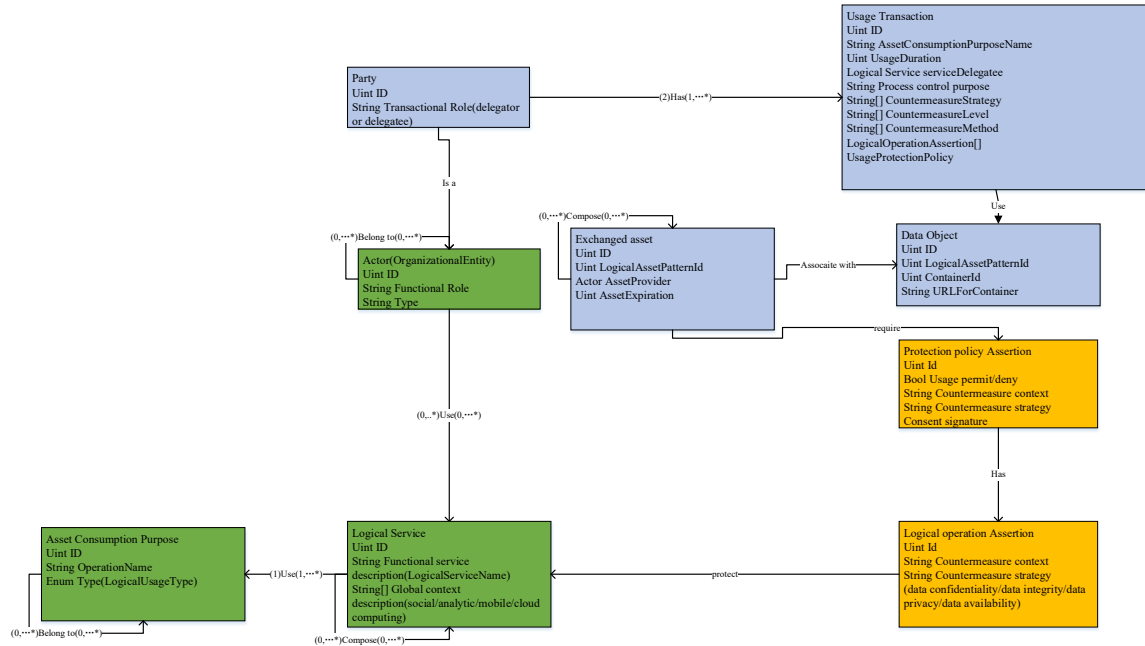


Figure 52 Usage Transaction class diagram

Algorithm 11 Usage Transaction derivation process

**Input:** ToS policy P

**Output:**

1. Function Usage Derivation(Policy P)
2. **Select all Business Transactions BT<sub>i</sub> associated to approved policy P**
3. **For each** Business Transaction BT<sub>i</sub>
4.     GenerateUsageTransaction(P, BT<sub>i</sub>)
5.     **End For**
6. **Return()**
7. **End Function**

Algorithm 12 Usage Transaction generation

**Input:** ToS Policy P, Business Transaction BT

**Output:**

1. Function GenerateUsageTransaction(Policy P, Business Transaction BT)
2. Select the Business Service BS associated to BT
3. **If** BS is an Elementary task
4.     Select Logical Service LS supporting BS
5.     Select all Usage Operations UO(s) used by Logical service LS
6.     **For each** Usage Operation UO<sub>j</sub>
7.         Select Data object DO associated to Logical Asset Pattern LAP and to Usage Operation UO<sub>j</sub>
8.         Select Logical Usage Type LUP associated to Usage Operation UO<sub>j</sub>
9.         Select Data Related Operation DRO associated to LUP implementation
10.         Select Policy assertion A from Policy P authorizing DRO for LAP



11. Select Exchange Asset EA associated to Logical Asset Pattern LAP and to Business Transaction BT
12. Select Logical Asset LA associated to EA
13. Select Logical Operation Policy LOP associated to LA and UO;
14. UT<-Create Usage Transaction
15. Associate UT to BT according to “on behalf” relationship
16. Associate UT to DO
17. Associate UT to LUP
18. Associate UT to LS
19. Associate UT to LOP
20. Associate UT to Policy assertion A
21. Associate DO to LA
22. **End for**
23. **End if**
24. **Return ()**
25. **End Function**

#### 4.3.2.3 *The physical Transaction generation process*

Once the Usage Transactions are generated, the Protection management component launches the physical transaction derivation process. This process retrieves all Usage Transaction associated with a ToU policy before launching the physical transaction generation process for each of these retrieved Usage Transactions. For each usage transaction, the corresponding logical service is selected and the associated concrete services are retrieved. Then physical operations used by these concrete services are also retrieved. The physical operation also links concrete service, container, asset consumption purpose (defined as physical usage type for implementation layer) and data object to the Usage transaction. This physical transaction generation process uses both the IS Description model knowledge (see the physical transaction class diagram Figure 53) to extract the data object to which the asset consumption purpose operates and the Usage Transaction description to identify the Exchanged asset with the associated Data object, and with the container for this transaction. Once all this information is fixed, the physical transaction is created and related to these different elements (see Algorithm 13).

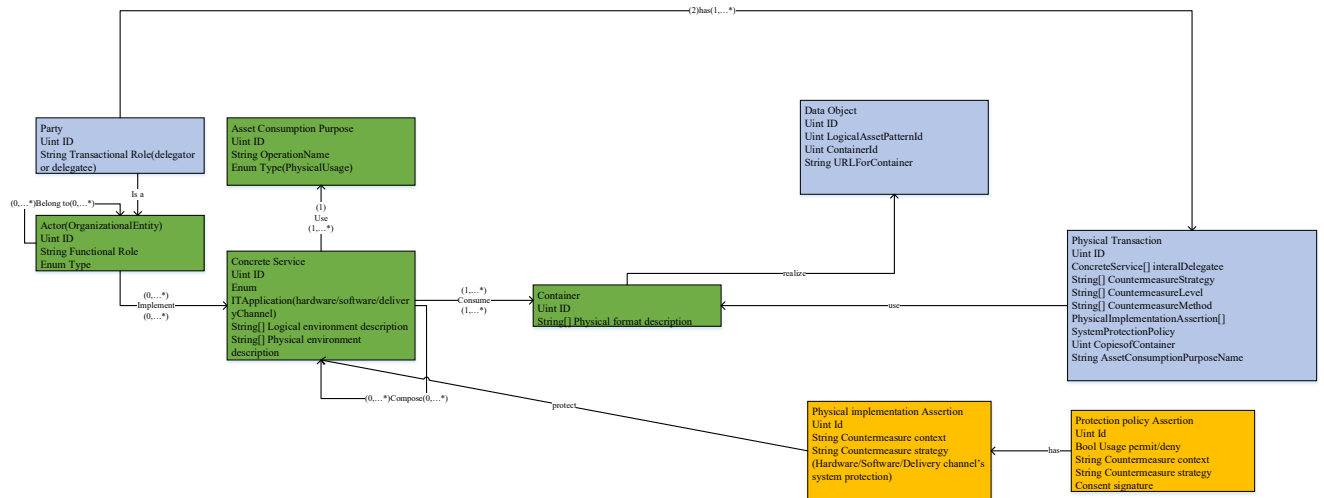


Figure 53 Physical transaction class diagram

### Algorithm 13 Physical Transaction generation

---

**Input:** Usage Transaction UT, Policy P

**Output**

---

1. GeneratePhysicalTransaction(Usage Transaction UT, policy P)
2. Select Logical Service LS involved in UT
3. Select all concrete services CS<sub>i</sub> implementing LS
4. **For each** Concrete Service CS<sub>i</sub>
5.     Select all Physical Operation PO<sub>j</sub> associated to Concrete service CS<sub>i</sub>
6.     **For each** Physical Operation PO<sub>j</sub>
7.         Select Data Object DO associated to Physical Operation PO<sub>j</sub>
8.         Select Physical Usage Type PUT associated to Physical Operation PO<sub>j</sub>
9.         Select Logical Asset LA associated Data Object DO
10.         PIP<-Create Physical implementation policy
11.         PIP<-Select Physical Implementation Assertions from Policy P associated to  
LA
12.         Select Container C associated to DO and to UT
13.         **if** Container C does not exist
14.             Create Container C
15.             Associate C to UT
16.             Associate C to LA
17.         **End if**
18.         PT<-Create Physical Transaction
19.         Associate PT to UT according to “on behalf” relationship
20.         Associate PT to DO
21.         Associate PT to PUT
22.         Associate PT to CS
23.         Associate PT to C
24.         Associate PT to PIP
25.         Associate PT to ToU policy P
26.     **End for**
27. **End for**
28. **Return()**
29. **End Function**

---

#### 4.3.3 Managing trusted transactions

Once defined, transactions are turned into Smart Contracts thanks to the Smart Contract Factory. This component is designed as the generic interface coupling our system to the blockchain which provides the immutable proof of consents and authorizations used by data consumers. This component is invoked by

- the Terms of Usage management component to generate smart contracts associated with the different transactions
- the Protection management component to get the pre-authorization token for a given service
- the Usage governance component to describe the target usage associated with a given authorization token.

To achieve these requirements, the Smart Contract factory interacts with the Key manager to get the Blockchain Account Id of the party involved in a given transaction and manage and store the description of the smart contract associated with the transactions. The Smart Contract Factory provides two main components to interact with the Blockchain:

- The SmartContractGenerator interacting with Smart Contract publication function is in charge of extracting the usage information associated with a given transaction to generate the convenient assertion that will be deployed in a smart contract.
- The UsageTracker is in charge of (1) interacting with Usage certification function providing the usage-operation-authorization token (which has been updated by the data transfer certification function to register the container) associated with a Physical Transaction and (2) interacting with Usage delegation function to certify that an authorization token has been given by the convenient smart contract.

This component uses an internal smart contract register (see Figure 54) to keep a local description of the smart contracts, who has created them, and which transaction they prove.

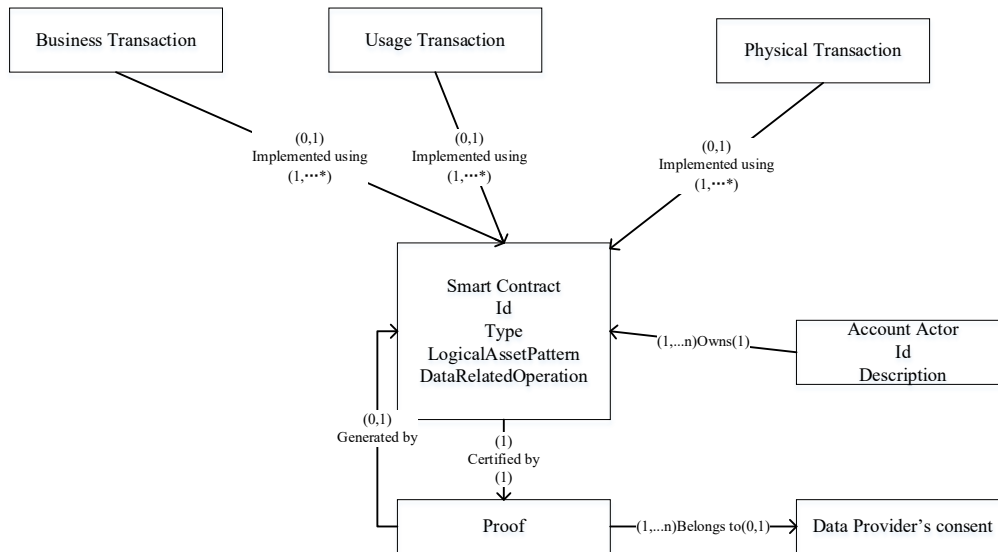


Figure 54 Class diagram of the Smart Factory's smart contract register organization

#### 4.3.3.1 Smart contract generator

This component is launched by the Terms of Usage Management, providing the “original” Business Transaction and the ToU consent signed by the data provider. The smart contract generator is associated with a recursive process that explores the Business Transaction hierarchy to generate the associated smart contracts (see Algorithm 14).

Our Smart Contract model is designed to “push” an approved usage assertion (defining the target asset and the usage parameter) from a data consumer (called by the delegator) to a given data consumer (called later the delegatee) following the initial ToU policy which generates tokens signed by a data provider. The usage assertion is approved “off-chain” by the delegator which delegates the usages to the delegatee. To this end, our smart contract pattern consists of different parts:

- *The approved policy*: define the asset and the allowed usage operation certified by this smart contract. A signature checking is used to verify the authenticity of this assertion. We define different types of assertions depending on the smart contract type:
  - *Business Authorization assertion*: it consists of a structure gathering the business purpose, requested data, and authorization status of the usage description based on our ontology to set the process motivation, the data-related operation, and asset consumption operations(which includes business usage type and logical usage type).
  - *Logical operation assertion*: it consists of a structure gathering operational conditions provided by a logical service, requested data, and protection countermeasure setting of asset consumption operations(which is logical usage type).
  - *Physical implementation assertion*: it consists of a structure gathering operational conditions provided by a concrete service, requested data, and protection countermeasure setting of asset consumption operations(which is physical usage type and is an atomic operation).
- *The ExchangedAsset* function is invoked by the delegator account to define the description of the asset and the previous delegator identity.
- *The UpdateBusinesspolicy* function is invoked by the delegator account to define the associated with the usage-authorization tokens of an exchanged asset from the initial ToU consent.
- *The UpDateToS* function is invoked by the delegator account to describe the description of the Terms of service operations belonging to the delegatee account
- *The Business Usage function* is invoked by the delegator account to define the delegatee account and to provide the exchanged asset to this account via the business usage type.
- *The DelegationMatch* function is invoked by the delegator to evaluate the delegation of exchanged assets to the Terms of service operations.

These functions provide the generic smart contract template of the exchange smart contract. We define precise implementations associated with the exchange smart contract and to the usage smart contract as these smart contracts do not use the same types of usages. The solidity code associated with this Exchange Smart Contract template is provided in the annex (see section 7.1). Similarly, the Usage Smart Contract template is defined in section 7.2.

The Smart Contract generation process consists of extracting the different Exchanged Assets associated with a Business Transaction, identifying their potential usages to provide this necessary information to the Blockchain Actor account in charge of deploying the Smart Contract. The Smart Contract is in charge of generating the certified pre-authorization assertion for the operation implementing the given usage. This certification is managed according to a signed global ToU assertion stored in the Smart Contract. This signed ToU assertion is either the consent assertion signed by the data provider for the “root transaction” or a “delegated authorization” provided by another smart contract associated with the “father transaction” identified thanks to the « on behalf » relationship. This involves that the smart contract creation process has first to identify the

delegator account to pre-manage the authorization delegation before deploying the smart contract. Once the exchange smart contract associated with the Business Transaction is created, Usage smart contracts and physical smart contracts are created. The last step of the process consists of identifying all the “sub-business transactions”, i.e. the transactions that are generated “on behalf” of the current transaction to create the smart contracts associated with these sub-transactions (see Algorithm 14).

*Algorithm 14 Smart Contract generation process*

---

**Input: Business Transaction BT**

**Output: Success or Failed**

---

1. SmartContractGeneration(BT)
2. Select Original Transaction OT from “On behalf relationship” from BT
3. **If** no Original Transaction is found
4. Original  $\leftarrow$  true
5. Delegator  $\leftarrow$  BT.DataProvider
6. Select Terms of Usage Consent associated to BT
7. **Else**
8. DelegatorSC  $\leftarrow$  Select Smart Contract Id SC associated to Business Transaction OT
9. Original  $\leftarrow$  False
10. Consent  $\leftarrow$  OT.Consent
11. **End if**
12. Select Business Service BS associated to BT
13. Select Terms of Service ToS associated to BS
14. Purpose  $\leftarrow$  BS.Process purpose
15. Delegatee  $\leftarrow$  Select Actor Account associated to BS from portfolio
16. Select all Exchanged Asset EA<sub>i</sub> involved in BT
17. **For each Exchanged Asset EA<sub>i</sub>**
18. Get UsageAuthorizationTokens as Token associated to EA<sub>i</sub>
19. **If** Original == False
20. Token  $\leftarrow$  AskforDelegation(DelegatorSC, EA<sub>i</sub>, ToS, Token, Delegatee)
21. **If** no Token generated
22. Return(failed)
23. **End If**
24. Certificator=DelegatorSC
25. **Else**
26. Token  $\leftarrow$  Sign(Delegator, EA<sub>i</sub>, ToS, Consent)
27. Certificator=Data-Provider
28. **End if**
29. SmartContract-ID  $\leftarrow$  CreateExchangeSC(ExchangedAsset EA, Delegatee, ToS, Token, ToU-Id)
30. Register (SmartContract-ID, SmartContract-Type=Exchange, Actor Account=Delegatee, Asset=ExchangedAsset, usage=ToS, proof=Token, Certified by Certificator)
31. EA<sub>i</sub>.SC  $\leftarrow$  SmartContract-ID
32. **If** BS.type == Elementary Task
33. Select Usage Transaction UT associated to BS using “On behalf” relationship
34. Select Logical Service LS associated to UT
35. Select Data Object DO associated to UT

```

36.      Usage-operation-authorization token<-DO.Token
37.      Select Logical Usage Type LUP associated to UT
38.      ServiceDelegate ← LS actor account
39.      Token ← AskforUsageDelegation(Delegator, EAi.SC, Consent, LUP,
          ServiceDelegate)
40.      If no Token generated
41.          Return(failed)
42.      End if
43.      SmartContractId ← CreateUsageSC(DataObject DO, Delegator, LUP,
          UsageOperationToken, UsageAuthorizationToken, ServiceDelegate)
44.      Register (SmartContractID, SmartContract-Type=Usage, Actor Account=
          ServiceDelegate, Asset=LA, usage=LUO proof=Token, Certified by
          SmartContract-ID)
45.      PhysicalSmarContractGeneration(UT, Consent)
46.      End if
47. End for
48. Return(Success)
49. End Function

```

---

- The *AskForDelegation* function provides the delegator's account the necessary information to invoke its smart contract to delegate authorization tokens and exchange assets to the delegatee account. It consists of
  - invoking the *BusinessUsage* function to declare the usage for which the pre-authorization must be generated and declare it will exchange the asset to the delegatee account.
  - invoking the *DelegationMatch* function to certify the delegation, providing the exact Terms of Service and the Delegated account which will get this token that is provided back to the Smart Contract generator process.
- The *CreateExchangeSC* function provides the Consumer account the necessary information to deploy the exchange smart contract on the Blockchain. To generate the smart contract, the actor uses the Smart Contract Pattern provided as a solidity code and merges it with the authorization token to the smart contract code description (see 7.1) so that the smart contract can be deployed. The actor invokes the *ExchangedAsset* function and the *UpDateBusinesspolicy* function to configure the asset description and the allowed operations on this asset.
- The *Sign* function extracts the corresponding assertion from the consent signed by the delegator.

Similarly, the *UsageSmartContractGeneration* function captures the knowledge stored in the Usage transaction. The *AskForUsageDelegation* function provides the necessary information to allow the actor in charge of the Business service to invoke the associated smart contract to extract the usage-authorization and provide the usage-operation-authorization for the Logical operation assertion. The *CreateUsageSC* function provides the Consumer account the necessary information to deploy the usage smart contract on the Blockchain. Similar to the exchange smart contract generation, the actor merges the solidity code associated with the Usage Smart Contract Pattern with the Usage authorization token to generate and deploy the usage smart contract.

The actor account in charge of the Usage smart contract is used as a gateway to connect the public blockchain with the private blockchain, which stores the consent and the pre-authorized operations to the private blockchain which is used to control the real access operations for the concrete services.

Focusing on the Physical Smart Contract generation, the process (see Algorithm 15) is managed for each Usage Transaction. First, the list of associated Physical Transactions is extracted, each physical transaction is defined to authorize a physical operation on a container provided that the concrete service provides the necessary countermeasures. The assertion on behalf of which the concrete operation will be granted is certified by the Usage Smart Contract associated with the Usage Transaction.

*Algorithm 15 Physical Smart Contract Generation Process*

---

**Input: Usage Transaction UT, Approved Consent CO**

**Output: Success or failed**

---

1. Function PhysicalSmartContractGeneration(UT, CO)
  2. Get Usage Actor Account UsageActor associated to UT
  3. Get UsageSC ID from UT
  4. Select all Physical Transaction PT<sub>i</sub> associated to UT
  5. **For each** Physical Transaction PT<sub>i</sub>
    6. Select Concrete Service CS associated to PT<sub>i</sub>
    7. Select Container C associated to PT<sub>i</sub>
    8. Select Data Object DO associate to Container C
    9. Usage Operation Authorization Token ← DO.Token
    10. Select Physical Usage Type PUT associated to PT<sub>i</sub>
    11. Delegatee ← Select CS Actor Account from the portfolio
    12. AccessControlAuthorizationToken ← AskforPhysicalDelegation (UsageSC, DO, C, PO, Delegatee, CO)
    13. If no AccessControlAuthorizationToken
    14.     Return(failed)
    15. Else
    16.     SmartContract-ID ← CreatePhysicalSC(Container C, Delegatee, PUT, AccessControlAuthorizationToken, UsageActor, UsageOperationAuthorizationToken)
  17.     **End If**
  18. **End for**
  19. **Return(Success)**
  20. **End Function**
- 

The *AskForPhysicalDelegation* function selects the Usage Smart Contract and its associated actor from the Smart Contract Factory register. It provides this actor's account the necessary information to invoke the smart contract to delegate AccessControlAuthorization token and container to the delegatee implementing the Concrete Service. It consists of invoking the *GrantPhysicalUsage* function to declare the usage for which the pre-authorization must be generated and declare that it will grant the container to the delegated account(delegatee).

The *CreatePhysicalSC* function provides the UsageActor account the necessary information to deploy the Physical smart contract on the Blockchain. The Physical Smart Contract Pattern (provided as a solidity code see section 7.3) is merged with the Usage Operation authorization token to get the physical smart contract code that will be deployed by the actor account associated with the Logical Service implementing the Usage Transaction.

At runtime, the actor associated with the Concrete Service will get the approved AccessControlAuthorization Token to certify that the asset will be processed according to what has been accepted from the *Comparison* function with the evaluation of the physical operation description.

#### 4.3.3.2 *Usage Tracker*

This component is used to provide and check the token associated with a given transaction. It provides two main functions interacting with the blockchain:

- *GrantPhysicalUsage* function (see Algorithm 16) managing (1) is used to send the usage-operation-authorization token to provide a container to the actor account associated with a concrete service when the requested operation matches the usage assertion stored in the physical smart contract. To this end, it first retrieves the Physical Transaction and then invokes the *Comparison* function (Algorithm 17)
- *CertifyConsent* function managing (2) is used to provide the list of the certification approval linking a granted usage to the original consent assertion (Algorithm 18).

#### *Algorithm 16 GrantPhysicalUsage function*

---

**Input: Concrete service CS, Business Transaction BT**

**Output: Usage Token UT**

---

1. GrantPhysicalUsage(ConcreteService CS, Business Transaction BT)
  2.  $PT \leftarrow$  RetrievePhysicalTransaction(CS, BT)
  3. Select CS-Actor Account associated to CS
  4. Select Physical Smart Contract SC-ID associated to PT
  5. Select Physical Usage Type PUT associated to CS
  6. Select Container C associated to CS
  7. Get UsageOperationAuthorizationToken associated to Container
  8. UsageToken UT  $\leftarrow$  PhysicaldelegationMatch(PhysicalSC, LAPC, PO, CS-Actor)
  9. If UT  $\neq$  Null
  10.     Register UsageToken associated to PT
  11. End if
  12. Return(UT)
  13. End Function
- 

#### *Algorithm 17 Process used to identify the physical transaction*

---

**Input : Concrete service CS, Business Transaction BT**

**Output: Physical Transaction PT**

---

1. Function RetrievePhysicalTansaction(ConcreteService CS, Business Transaction BT)
2.  $PT \leftarrow$  NULL
3. **Select Business Service BS associated to BT**



```

4  If BS.Type=Elementary task
5      Select all usage transaction UTi associated to BT
6      Found ← False
7      While (not(Found) or UTi list not empty)
8          Select all Physical transaction PT associated to each UT(UTi)
9          While (not(Found) or PTj list not empty)
10             Select Concrete Service S associated PTj
11             If S == CS then
12                 Found ← True
13                 Return(PTj)
14             Endif
15             PTj<- Next Physical Transaction from the list
16         End While
17         UTi<-Next Usage Transaction from the list
18     End While
19 Else
20     Select each business transaction BTi included in BT
21     Do
22         PT<-RetrievePhyscalTransaction(CS, BTi)
23     Until PT!=NULL or all BTi are explored
24 End If
25 Return(PT)
26 End function

```

---

The *CertifyConsent* Function uses the Smart Contract register to manage the certification chain. At each step, it interacts with the smart contracts to get the verification token. The process starts from the proof provided by the smart contract. Then the consent certification function identifies the smart contract which has delivered this proof and manages the origin of this proof recursively. The *GetCertification* function sends the token to be verified to the actor owning the smart contract. This actor will return the Verification token associated with the SC.

*Algorithm 18 Recursive retrieval of the Initial token*

---

Input: Proof Token P

Output: Consent

---

1. RetrieveOriginalConsent(ProofToken)
2. **Select** SmartContract SC from Smart Contract Factory register where Proofcertification==ProofToken
3. **If** SC.ProofCertifiedbyDataProvider==True
4.     **Select Business Transaction BT associated to SmartContract SC**
5.     Return(BT.signed consent)
6. **Else**
7.     GetCertification(SC, SC.Actor, ProofToken)
8.     **Select** SmartContract Father-SC associated to SC thanks to Generated by relationship
9.     New-Token ← Father-SC.Token
10.     Return(RetrieveOriginalConsent(New-Token))
11. **End If**

4.3.4 Usage Governance

The Usage Governance component is designed to manage the different life-cycle events, usage events, and security events associated with the different assets. It interacts with the Protection management component to capture assets and related operations to manage the life-long protection.

For the container operation governance in a Logical service and concrete services, the data provider can evaluate the current quality of protection of a logical asset by integrating the way the different containers are protected. This is based on the asset QoP evaluation algorithm developed in Algorithm 3. Each time, a concrete service is launched and wants to accede to a container to support the business service. the Information System interface component captures this service invocation and invokes the Usage Tracking function provided by the tracking manager to get the credentials associated with the requested operation on this container. This Usage Tracking function extracts the Usage and Physical transactions associated with this container and its data object and invokes the Smart Contract Factory to get the credential associated with the requested operations and countermeasure efficiency. The Smart Contract Factory selects the Smart Contract Id and the Actor account associated with the Physical Transaction. The Tracking Manager registers this token and sends it back to the protection management component as proof for the data provider. This authorization token is associated with the container so that all accesses on the container can be reported. Paying attention to the life-long asset protection, the tracking implementation process includes different sub-processes called tracking agents (global tracking agent, complex tracking agent, and elementary tracking agent) in charge of managing the state of Containers (Figure 55)

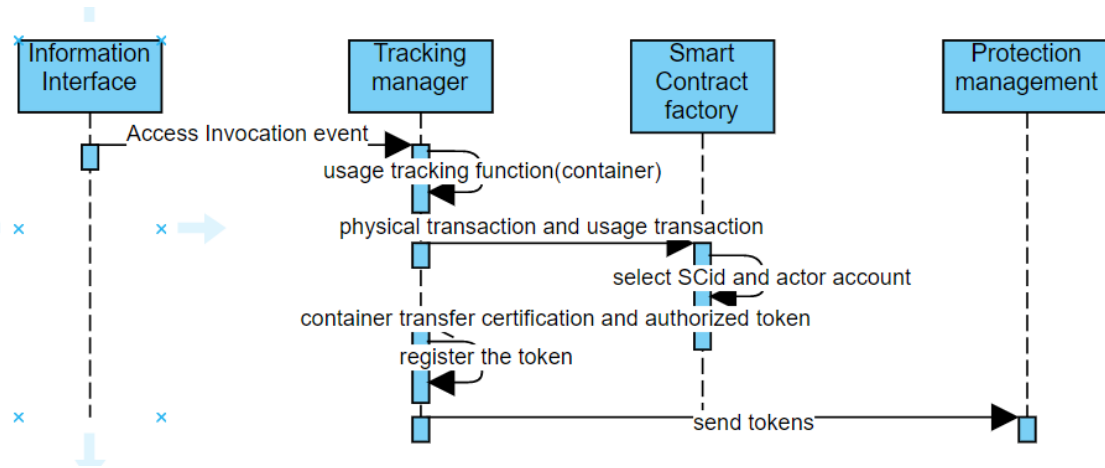


Figure 55 Tracking implementation process sequence diagram

For the exchanged asset authorization governance from business services, few functions can be used to monitor the way assets are used. In this way, the data consumer can prove that the operations it uses on a given exchanged asset are compliant with the initial consent. Each time, an exchanged asset is exchanged, the protection management component gets the token associated with the authorization provided by the data consumer

(Figure 56). The protection management component invokes the usage checking function provided by usage monitoring to check this authorization token and monitor this new authorization. To this end, the Usage Monitoring has first to interact with the Smart Contract factory to check the token and identify the business transaction associated with the declared usages. It launches the security monitoring certification function to start the (Business service monitoring) Global tracking agent, (Logical service monitoring) Complex tracking agent, and (concrete service monitoring) Elementary tracking agent to manage the creation of a new container associated with the exchanged asset and authorized usage. It provides the container Id and the related usages to the Protection management component so that pre-protection means can be deployed. Once the container is defined and instantiated, the Protection management component invokes the Data transfer certification function to register the transfer of this container. The Data transfer certification function interacts with the Smart Contract factory to generate a tracking smart contract (see section 7.4 providing the associated solidity code) which certifies the data transfer and notifies the data provider to have the container operation governance.

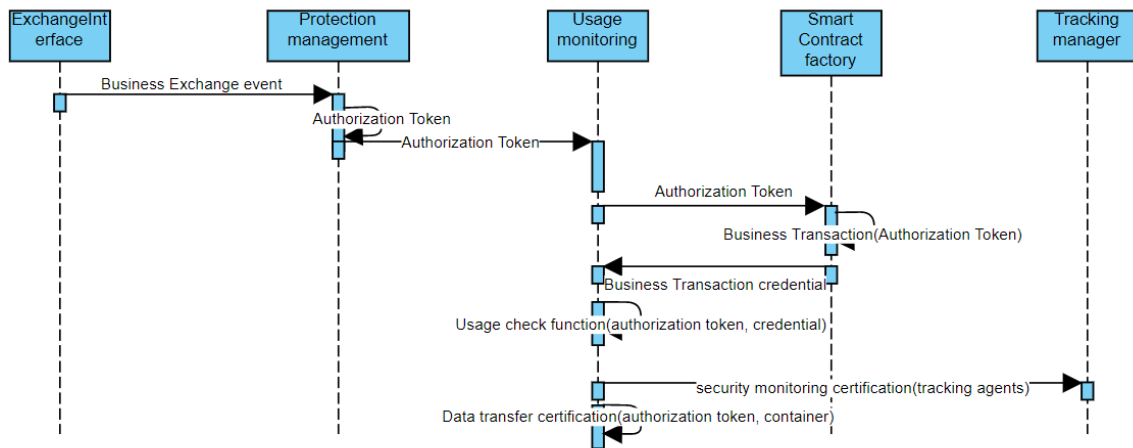


Figure 56 Usage monitoring process sequence diagram

## 4.4 Evaluation

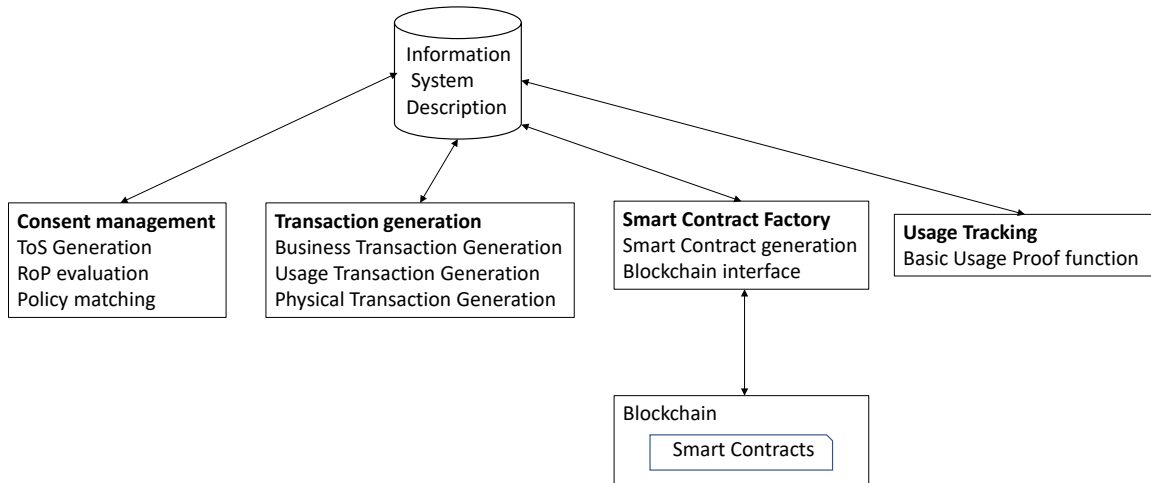
To evaluate our distributed data-driven protection architecture, we use a simple use case to compare our DUP architecture with other systems. To this end, we have developed key components to evaluate data asset protection terms of service... or to generate smart contracts. These components are loosely coupled as they have interconnected thanks to our Information System description model.

In this section, we first introduce our experiment before comparing our results with those provided by other ontologies, consent management and protection systems presented in the state of the art section.

### 4.4.1 Experiment

The prototype we build to validate our Data-driven and Usage-based protection model integrates key functions of DUP. We use mysql Ver 14.14 Distrib 5.5.62, for Win64

(AMD64) to store the Information System Description and have developed the Consent Management, the Transaction Generation, the Smart Contract Factory and Usage Tracking function using Java: IntelliJ IDEA jdk 12.0.1, Maven, junit4.11 and mysql-connector-java:8.0.22. The Blockchain is deployed using Ganache.



*Figure 57 Prototype Architecture*

The different parts of the prototype are loosely coupled thanks to the Information System Description Data Base (see the associated class diagram Figure 58). The code is provided in annex (see section 8).

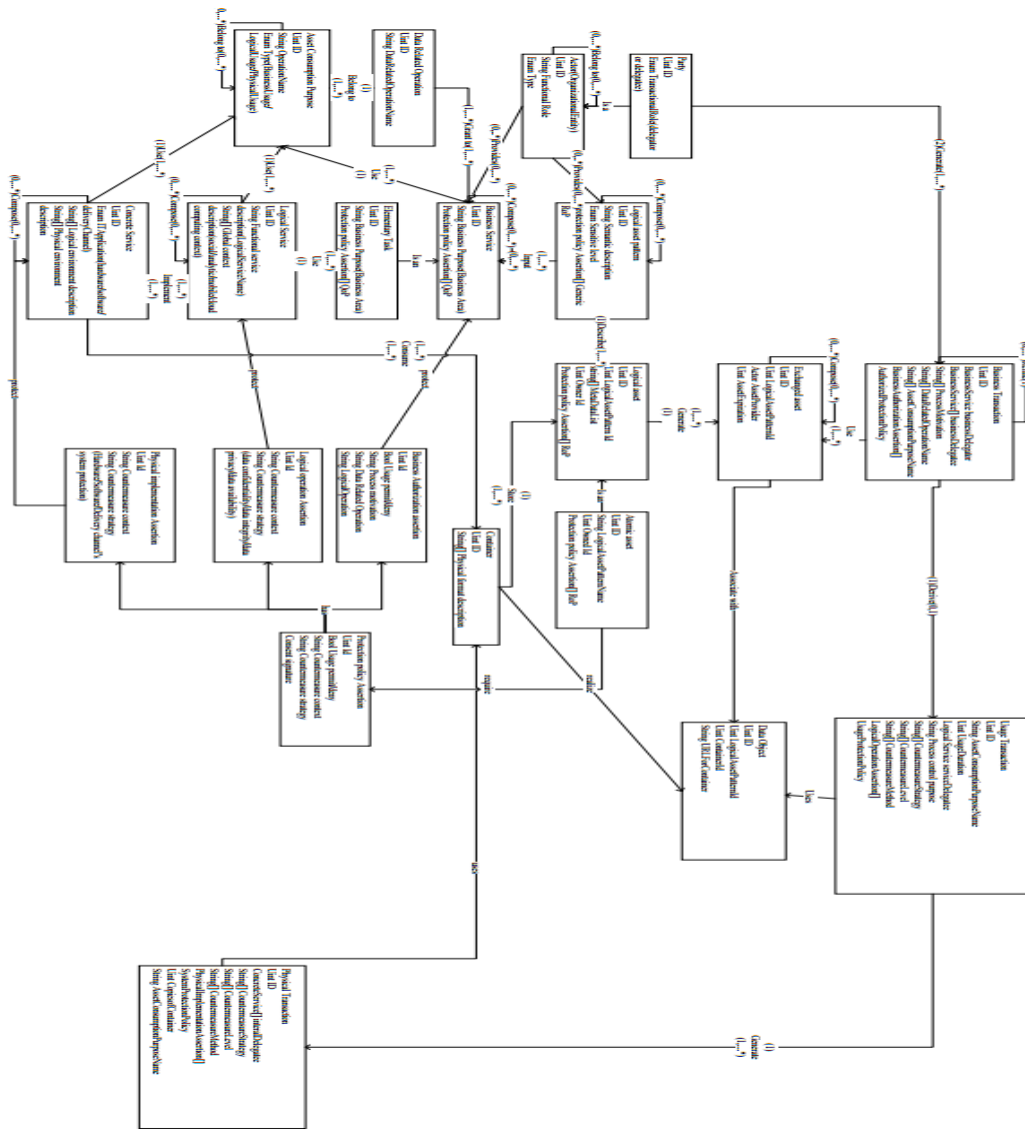


Figure 58 Global Class diagram of the Information System Description used in our prototype

To evaluate our DUP architecture, we designed an experiment based on the motivating example introduced in Chapter 3. This evaluation example integrates collaborative business and end-users interactions. It relies on an online shopping platform (called later Online Shopping), shared by different companies. Online Shopping proposes “manufactured on demand” products and services, from different suppliers (such as Company A and Company B) to clients. The different partners may share and exchange product information or client personal information depending on the business process requirements. To reduce the carbon footprint, Company A uses the 3D printers hosted by company B to “manufacture” the product as close as possible to the client. Online Shopping also shares data with MyAnalytics company which uses the customer data to establish recommendations and provide marketing analysis to Online Shopping company.

Alice browses the Online Shopping platform. Online shopping platform collects Alice's browsing history, connection information as well as other information related to the product she intends to buy. Consequently, Alice will have to "share" different personal information such as traces of her online activity, financial information, her address, and what she has bought with Online Shopping platform, MyAnalytics (which can also mix this information with other sources), and Company A and B (the product suppliers).

Alice also interacts with other online platforms, Personal Information Management Systems (PIMs), social networks, etc. using her own computer or smartphone. Protecting consistently her personal information is difficult for Alice as she interacts with different systems, providing their own Terms of Service (ToS for short). Moreover, she cannot get any information to check whereas her personal information is really protected and if it is used according to the ToS she accepted.

The online Shopping platform is responsible for Alice's personal information protection. According to the GDPR, Online Shopping may also have to prove that it uses and protects this information according to the exact ToS. While exchanging data with its partners (MyAnalytics, Company A or B), Online shopping must check the business purpose of the external service requesting information to verify if this is allowed according to Alice's consent. Online Shopping has also to transfer this ToS to the service provider.

Based on this simple example, different requirements are taken into account: first, Alice needs to define protection requirements associated with her data assets, including protection requirements for each logical asset. Alice must also identify assets' replications, who store and use these different copies. Second, Alice and Online Shopping need to approve the exact ToS, specifying the business purpose for the operations occurring on Alice's data. Third, Online Shopping gets Alice's consent and has to manage operation authorization according to it. Fourth, Alice needs to track the containers she exchanges with Online Shopping.

Focusing on the Terms of Usage negotiation, two transactions must be detailed:

- OLS provides a ShoppingService to Alice which requires:
  - AccountInformation (including UserName and Password),
  - DeliveryInformation(including DeliveryReceiver, DeliveryAddress, DeliveryContact) and
  - OrderingInformation(includingProductOrderDescription, FinancialCertificate)
- CompanyB provides a DeliveryService to OLS which requires:
  - DeliveryInformation (including DeliveryName, DeliveryAddress, DeliveryContact) and
  - ProductOrderDescription.

Focusing on the data consumer side, OLS will collect different assets: account information, delivery information and ordering information. Figure 59 presents the different usages declared by the different services. In brief, during the Workflow execution, OLS will

- store OrderingInformation and DeliveryInformation

- extract OrderingInformation to have the ProductOrderDescription.
- transfer DeliveryInformation and ProductOrderDescription.

CompanyB will

- aggregate DeliveryInformation with ProductOrderDescription to have the PrintInformation.
- show&track PrintInformation.

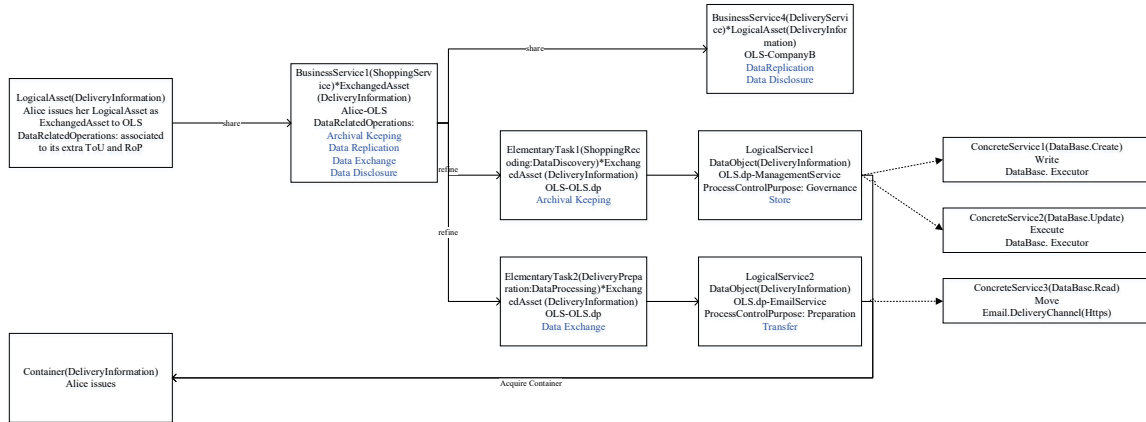


Figure 59 Object diagram describing the different usages associated to the services

Alice and OLS negotiate to generate TermsOfUsage1(ToU1) and OLS and CompanyB negotiate to generate TermsOfUsage2(ToU2).

We use our prototype to evaluate the Terms of Usage associated to the Order and delivery Business Service provided by OnLine Shopping. We assume that only the elementary task and external services are associated to business authorization assertions (see Figure 60 and Figure 61). Figure 62 presents the ToS generation process report and the ToS generation result is presented Figure 63.

AssetDescription	BusinessServiceDescription	ProcessMotivation	DataRelatedOperation	UsageName	Decision	CountermeasureContext	UsageStatus
ContactInformation	DeliveryProcessing	DataProcessing	DataReplication	Refine	high	UsageDelegation	Private
ProductRecord	DeliveryProcessing	DataProcessing	DataReplication	Share	medium	BusinessPurposeEnforcement	shared
ProductRecord	DeliveryProcessing	DataProcessing	DataExchange	Transfer	medium	BusinessPurposeEnforcement	shared
ProductRecord	DeliveryProcessing	DataProcessing	DataReplication	Share	low	UsageIntegrity	shared
ProductRecord	DeliveryProcessing	DataProcessing	DataExchange	Transfer	low	UsageIntegrity	shared
ProductOrder	OrderEstablishment	DataVisualization	InformationConstruction	Generate	notyet	unknown	Private
ContactInformation	OrderProcessing	DataDiscovery	DataExchange	transfer	notyet	unknown	conditioned
ProductRecord	OrderProcessing	DataDiscovery	DataExchange	transfer	notyet	unknown	conditioned
AccountInformation	OrderGovernance	DataVisualization	ArchivalKeeping	store	notyet	unknown	conditioned

Figure 60 Partial dump of the motivating example data base: Initial ToS evaluation

AssetDescription	BusinessServiceDescription	ProcessMotivation	DataRelatedOperation	UsageName	CountermeasureContext	Decision	UsageStatus
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Refine	UsageDelegation	high	conditioned
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	UsageDelegation	high	Private
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Share	UsageDelegation	high	shared
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	UsageDelegation	high	shared
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Refine	BusinessPurposeEnforcement	high	conditioned
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	BusinessPurposeEnforcement	high	Private
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Share	BusinessPurposeEnforcement	high	shared
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	BusinessPurposeEnforcement	high	shared

Figure 61 Partial dump of the motivating example: Basic usages

```

it doesn't has the business Authorization assertion
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
add the extra assertions
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
it doesn't has the business Authorization assertion
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
add the extra assertions
it doesn't has the business Authorization assertion
it doesn't has the business Authorization assertion
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
add the extra assertions
aggregate the subservice assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP
add the extra assertions
aggregate the subLogicalAssetPattern assertion to the business service
start to aggregate the assertions by the UsageName and UsageStatus and external QoP

```

Figure 62 Part of the ToS generation report from the motivating example

ContactInformation	OrderProcessing	DataDiscovery	DataExchange	transfer	notyet	unknown	conditioned
ProductRecord	OrderProcessing	DataDiscovery	DataExchange	transfer	notyet	unknown	conditioned
AccountInformation	OrderGovernance	DataVisualization	ArchivalKeeping	store	notyet	unknown	conditioned
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Refine	high	UsageDelegation	conditioned
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Refine	high	BusinessPurposeEnforcement	conditioned
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Share	high	UsageDelegation	shared
ContactInformation	ProductDelivery	DataProcessing	DataReplication	Share	high	BusinessPurposeEnforcement	shared
ContactInformation	ProductDelivery	DataDiscovery	DataExchange	transfer	high	UsageDelegation	conditioned
ContactInformation	ProductDelivery	DataDiscovery	DataExchange	transfer	high	BusinessPurposeEnforcement	conditioned
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	high	UsageDelegation	Private
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	low	UsageIntegrity	Private
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	medium	BusinessPurposeEnforcement	Private
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	high	UsageDelegation	shared
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	low	UsageIntegrity	shared
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	medium	BusinessPurposeEnforcement	shared
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Refine	high	BusinessPurposeEnforcement	Private
AccountInformation	ProductDelivery	DataProcessing	DataReplication	Share	high	BusinessPurposeEnforcement	shared
AccountInformation	ProductDelivery	DataVisualization	ArchivalKeeping	store	high	UsageDelegation	conditioned
AccountInformation	ProductDelivery	DataVisualization	ArchivalKeeping	store	low	UsageIntegrity	conditioned
AccountInformation	ProductDelivery	DataVisualization	ArchivalKeeping	store	medium	BusinessPurposeEnforcement	conditioned
AccountInformation	ProductDelivery	DataDiscovery	DataExchange	transfer	high	UsageDelegation	shared
AccountInformation	ProductDelivery	DataDiscovery	DataExchange	transfer	low	UsageIntegrity	shared

Figure 63 Partial dump of the database showing the results of the ToS generation process



```

start the root of the business service composition
build the business service composition
recursive to build the business service composition36ProductDelivery
The businessNode isNot Atomiccontrol.BusinessNode@1cd629b3[control.BusinessNode@1cd629b3]36
find SubBusinessService37OrderEstablishment
Start for the decendent of the business service
recursive to build the business service composition37OrderEstablishment
The businessNode is AtomicOrderEstablishment37
find SubBusinessService41OrderGovernance
Start for the decendent of the business service
recursive to build the business service composition41OrderGovernance
The businessNode is AtomicOrderGovernance41
find SubBusinessService40Order-Processing
Start for the decendent of the business service
recursive to build the business service composition40Order-Processing
The businessNode is AtomicOrder-Processing40
find the logical service
find the logical service
business service is elementary task37OrderEstablishment
LogicalService isManufactureService13
find the logical service
business service is elementary task41OrderGovernance

```

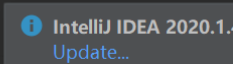


Figure 64 Result of the QoP generation process

AssetDescription	FunctionalDescription	CountermeasureContext	CountermeasureStrategy	CountermeasureLevel	UsageName	UsageDuration	CountermeasureMet
AccountInformation	RegistrationService	DataExistenceControl	DataLossControl	medium	store	permanent	Lock down the netw
AccountInformation	RegistrationService	UsageDelegation	BusinessPurposeEnforcement	high	store	permanent	realTimeExecutionO
ProductRecord	RecordStorageService	DataIdentification	DataAnonymization	high	Generate	1 day	DifferenyPrivacy
ProductRecord	RecordStorageService	UsageDelegation	BusinessPurposeEnforcement	high	Read	permanent	realTimeExecutionO
ProductRecord	RecordStorageService	UsageIntegrity	DataEncapsulation	high	Update	permanent	WaterMarking
ContactInformation	ManufactureService	DataIdentification	DataAnonymization	high	Share	20 days	DifferenyPrivacy
ContactInformation	ManufactureService	UsageIntegrity	DataEncapsulation	high	Transfer	1 day	WaterMarking
ProductRecord	ManufactureService	UsageIntegrity	DataEncapsulation	high	Transfer	1 day	WaterMarking
ProductRecord	ManufactureService	UsageIntegrity	DataEncapsulation	high	Transfer	1 day	WaterMarking

Figure 65 Partial dump of the data base showing the final Tos assertions generated from the motivating example

Then the Quality of Protection is evaluated. Figure 66 and Figure 67 present the initial QoP of services stored in OLS IS description Data Base whereas Figure 68 and Figure 69 present the QoP generation process report and result.

AssetDescription	FunctionalDescription	countermeasurecontext	CountermeasureStrategy	CountermeasureLevel	CountermeasureMethod	lid
ProductRecord	ManufactureService	DataIdentification	DataEncryption	medium	AES	13
ContactInformation	RecordStorageService	UsageIntegrity	DataEncapsulation	high	RSA	12
ProductRecord	RecordStorageService	UsageIntegrity	DataEncapsulation	high	RSA	12
AccountInformation	RegistrationService	DataExistenceManagement	DataLossControl	high	EndpointSecurity	11

Figure 66 Partial dump of the data base showing the initial Quality of Protection of Logical Services

AssetDescription	FunctionalDescription	CountermeasureContext	CountermeasureStrategy	CountermeasureLevel	CountermeasureMethod	cid
ProductRecord	RecordTransform	HardwareProtection	MemoryProtection	high	segmentation	10

Figure 67 Partial dump of the data base showing the initial Quality of Protection of Physical concrete service

```

aggregate the QoP of sub service=DeliveryProcessingand metaData=ContactInformat
The qop generation of meta data=ContactInformationand service =DeliveryProcessi
it doesn't have QoP assertion
aggregate the qop with sub qop by CountermeasureContext
aggregate the assertios to a uniifed countermeasureContext
aggregate the assertios to a uniifed countermeasureContext
The qop generation of meta data=AccountInformationand service =ProductDelivery3
it doesn't have QoP assertion
aggregate the QoP of sub service=OrderGovernanceand metaData=AccountInformation
The qop generation of meta data=AccountInformationand service =OrderGovernance4
it doesn't have QoP assertion
service is an elementary task
it will generate the QoP from logical service and concrete service
aggregate the qop with sub qop by CountermeasureContext
aggregate the assertios to a uniifed countermeasureContext
aggregate the assertios to a uniifed countermeasureContext
add the extra QoP
add the extra QoP
aggregate the QoP of sub LogicalAssetPattern=ProductRecordand service=Produc
The qop generation of meta data=ProductRecordand service =ProductDelivery36

```

Figure 68 Part of the QoP generation report

AssetDescription	qid	ServiceDescription	countermeasurecontext	CountermeasureLevel
ContactInformation	40	Order-Processing	UsageIntegrity	high
ContactInformation	36	ProductDelivery	UsageIntegrity	high
AccountInformation	41	OrderGovernance	DataExistenceManagement	high
ProductRecord	37	OrderEstablishment	DataIdentification	high
ProductRecord	37	OrderEstablishment	HardwareProtection	high
ProductRecord	40	OrderProcessing	UsageIntegrity	high
AccountInformation	36	ProductDelivery	DataExistenceManagement	high
AccountInformation	36	ProductDelivery	UsageIntegrity	high
AccountInformation	36	ProductDelivery	DataIdentification	high
AccountInformation	36	ProductDelivery	HardwareProtection	high
ContactInformation	40	OrderProcessing	UsageIntegrity	high

Figure 69 Partial dump of the data base showing the final Quality of Protection Evaluation

Lastly, the ToS and QoP are aggregated (see execution report ) to set the Terms of Usage (see ).

```

get the QoP of Internal policy
get the ToS and QoP OF extra policy
Aggregate for each service
Aggregate for each service
Aggregate for each service
Aggregate for each service

```

Figure 70 Terms of Usage generation report

AssetDescription	qid	ServiceDescription	CountermeasureContext	CountermeasureLevel	UsageName	ProcessMotivation	DataRelatedOperation
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageDelegation	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageIntegrity	low	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataIdentification	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	HardwareProtection	high	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	Refine	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageDelegation	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	UsageIntegrity	low	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataIdentification	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	HardwareProtection	high	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	Share	DataProcessing	DataReplication
ContactInformation	36	ProductDelivery	DataExistenceManagement	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	UsageDelegation	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	UsageIntegrity	low	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	DataIdentification	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	HardwareProtection	high	transfer	DataDiscovery	DataExchange
ContactInformation	36	ProductDelivery	BusinessPurposeEnforcement	medium	transfer	DataDiscovery	DataExchange
AccountInformation	36	ProductDelivery	DataExistenceManagement	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	UsageDelegation	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	UsageIntegrity	low	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	DataIdentification	high	Refine	DataProcessing	DataReplication
AccountInformation	36	ProductDelivery	HardwareProtection	high	Refine	DataProcessing	DataReplication

Figure 71 Partial dump of the data base showing the final Terms of Usage Policy

Before approving this Terms of Usage, Alice has to evaluate if it matches or not with her Requirements of Protection. Focusing on our motivating example, Alice has to manage different Requirements of protection (see Figure 72) regarding her own contact information used for the billing process and her brother Bob’s contact information used for the delivery process. Bob as allowed Alice using his data, provided that she follows his requirements of protection. After launching the process, the RoP assertions are aggregated (see Figure 73) and the final RoP policy is generated (see Figure 74).

1. The basic RoP policy defined by the Alice:

id	metaData	CountermeasureLevel	CountermeasureContext	lid
1	ContactInformation	medium	UsageDelegation	(NULL)
2	PaymentInformation	medium	DataIdentification	(NULL)
3	OrderHistory	medium	DataIdentification	(NULL)
4	OrderEstablishment	low	DataExistenceManagement	(NULL)
5	ContactInformation	low	DataIntegrity	(NULL)
6	Address	medium	UsageDelegation	(NULL)
7	Address	low	DataConfidentiality	-1

2. The extra RoP policy defined by Bob

id	metaData	CountermeasureLevel	CountermeasureContext	lid
1	Address	medium	UsageDelegation	2
2	Telephone	medium	UsageDelegation	3
3	Name	low	UsageDelegation	4
4	Address	high	DataIntegrity	2

Figure 72 Partial dump of the data base showing the part of Alice’s Information System description related to RoP policies

```

Find whether the logical asset has the final RoP1ContactInformation
No, Create its Logical final RoP
Create RoP of logical AssetContactInformationContactInformation
Logical asset is not AtomicContactInformation1
It has2basic protection assertions
It has0extra protection assertions
Aggregate both to generate RoP assertion
aggregate the RoP
It has extra protection policy
It has extra protection policy
recursive to aggregate the subLogicalAsset's assertionAddress2to the composed logical assetContactInformation1
Find whether the logical asset has the final RoP2Address
No, Create its Logical final RoP
Create RoP of logical AssetAddressAddress
Logical asset is AtomicAddress2
It has2basic protection assertions
It has2extra protection assertions
Aggregate both to generate RoP assertion
aggregate the RoP
Compare the countermeasureLevel in the negative strategy
medium
It has extra protection policy

```

Figure 73 RoP evaluation on our motivating example: Delivery contact information RoP management on Alice side

id	metaData	CountermeasureLevel	CountermeasureContext	lid
54	Address	medium	UsageDelegation	2
55	Address	high	DataIntegrity	2
56	Address	low	DataConfidentiality	2
57	Telephone	medium	UsageDelegation	3
58	Name	low	UsageDelegation	4
59	ContactInformation	medium	UsageDelegation	1
60	ContactInformation	high	DataIntegrity	1
61	ContactInformation	low	DataConfidentiality	1

Figure 74 Partial dump of the data base showing the consolidated RoP stored in Alice's Information System description

Then the matching process is launched to validate the ToU.

Once Alice and OLS approved ToU1, the different Business transactions are refined (see Figure 75) and more precise Terms of usage are generated:

- ToU11(AccountInformation in the ShoppingService),
- ToU12(DeliveryInformation in the ShoppingService)

ToU13(OrderingInformation in the ShoppingService).

Similarly, ToU2(DeliveryService) between OLS and CompanyB is refined and ToU21 and ToU22 are generated, providing more precise usage descriptions.

Then Usage Transactions and Physical Transactions are generated (see Figure 76 and Figure 77), delegating the initial consent formalized in the ToU to the different services using the data assets.

```

Create BusinessTransaction of businessServiceProductDelivery36
Select all the Logical Operations used by businessService
initialize the business transaction from the business service
Set the business transaction's provider and consumer
For each LogicalOperation to update the BusinessTransaction
find the LogicalAssetPattern
find the DataRelatedOperatin and ProcessMotivation and BusinessUsage
Find the BusinessAuthorizationAssertion for sub BusinessService: it will be DRO,PM AND UsageName
Find the businessAuthorizationAssertion of this businessService to the subBusinessService
Find the ToS of each subBusinessService
create the exchanged asset of this business transaction
determine whether the exchangedAsset is existed
the business transaction is the initial business transaction, register the Logical asset
Associate LogicalAsset to the exchanged asset
Store the Exchanged asset in the business transaction
Store the metaDataList of Exchangedasset in the business transaction
Associate the ExchangedAsset to the DataRelatedOperation and Terms of Usage assertion
Find the AuthenticationPolicyAssertion for this BusinessService: it will be BusinessUsage,BS
From a list of metaData to select the most similar metaData with Lap
From a list of metaData to select the most similar metaData with Lap
Find the approved ToU assertion to certify BusinessAuthorizationQoP and AuthenticationPolicyAssertion
find the LogicalAssetPattern

```

*Figure 75 Part of the Business Transaction refinement report Process*

```

Set the business transaction's provider and consumer
Set the business transaction's provider and consumer
UsageDerivation
determine whether the business service is elementary task
Select the Logical Service of the business service
Select all Usage Operation used by Logical Service
Get the DataObject from the uo
set the DataObject into the Ut
Get the LogicalAssetPattern from the DataObject
Select the Exchanged Asset descrbing using LAP and involved in BT
Get the DataRelatedOperation and UsageName
Get the ToU assertion a authorizing DRO for LAP from ToU Policy P
set the QoPLogicalService of LAP in the LogicalService
set the business transaction
set the LogicalService
set party of this usage transaction
Store the UsageTransaction into the DataBase
Start the physical transaction generation

```

*Figure 76 Usage Transaction Generation Process*

```

Start the physical transaction generation
Get the LogicalService involved in the usageTransaction
Get all concrete service implementing LS
For each concrete service to generate physical transaction
Set the party of physicalTransaction
associate to the usage transaction and concrete service
select the physical operation from concrete service
Get the dataObject from the physical operation
Get the physical Usage from this PhysicalOperation
Get the LogicalAsset from ut
Get the Qop assertion from logical asset
Set the ToU assertion
Select the container by data object and logical asset

```

*Figure 77 Physical Transaction Generation Process*

The Smart Contract Factory uses these transactions to generate the associated Smart Contract, extracting Smart Contract parameters from the Information System Data Base (see Figure 78). The Smart Contract Factory communicate with the REMIX and the blockchain environment using Web3 Provider and JSON. Then the smart contract is created in the Blockchain and parameters are set using the different configuration function invocation. In the example proposed Figure 79, Online-shopping provides Alice's data to its shopping's department. This figure shows the smart contract creation and its configuration thanks to the dedicated configuration function invocation, using three accounts:

- Alice is the DataOwner and original DataProvider: (0xE3124464a94A73e78A35C79C0305ef4caF00D78d)
- Online-shopping is the DataDelegator and the SmartContractGenerator: (0xEA5b5cf63828dFAE6B97Ed247B78A6fD6728B435)
- Online-shopping department is the DataDelegatee: 0x8BECB7769369B611626A43BB23685bA6E3470331

```

determine whether it is the original transaction
find the Original business transaction
get the business service associated to the business transaction
get the sub businessService which will generate future business transaction
Get the business purpose of this business service
the exchangedAsset provider such as Alice
Get thisServiceProvider who will be the delegator
Get thisService's next delegates
Select all the exchangedAsset in this BusinessService
For each ExchangedAsset to start the delegation
manage authorization of the Delegator's authorization to the delegatee for the sub businessService
create the ExchangedSC
Select the usage Transactions
manage authorization of the Delegator's authorization to the delegatee for the sub businessService
create the ExchangedSC
Select the usage Transactions
manage authorization of the Delegator's authorization to the delegatee for the sub businessService
create the ExchangedSC
Select the usage Transactions

```

*Figure 78 Smart Contract Factory Execution Report*

## 1. Create ExchangeSmartContract

TX HASH		CONTRACT CREATION	
0xea494815f4d171e7c5c5d0a29410ed14c64378653d27b5c45775bb0f6afbe8a9			
FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	4691107	0

## 2. Create ExchangedAsset of the AccountInformation

TX HASH		CONTRACT CALL	
0xa8d62ac7bbb5604e9e3a12ce33568989b561af2b2646b1bb502ab7e4920fcb4a			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	151451	0

## 3. Add metadata of the AccountInformation and ProductRecord(AccountInformation compose ProductRecord)

TX HASH		CONTRACT CALL	
0x376315fb237684567520d4792f219cd182d536c20b57d6f77157386d13eeb848			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	25522	0

## 4. Add the subBusinessService request (give the delegatee's account):

TX HASH		CONTRACT CALL	
0xdf494fb98f8ebc7454205637a33a205b35b6d3737b4d26451902440888d019a1			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	51078	0

## 5. Add the businessAuthorization:

TX HASH		CONTRACT CALL	
0x76e24a3fe5cd9844315c64096efde92f01b1960dd3eb0ad91453d655da5607a4			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	135827	0

## 6. Add the request

TX HASH		CONTRACT CALL	
0x6711897f447d1846033538e3776370920a1e5b152cc23df7284ad22fba32f1a7			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	134241	0

## 7. Authorization

TX HASH		CONTRACT CALL	
0x6812bc00bf8fad9cf027e7f4b9dd43cf00f28eaaab2b977437968adeb1f5ccd4			
FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0xeA5b5cf63828dFAE6B97Ed247B78A6fD6728B435	0x2a965B72D24DF06e4fDa5482733667EB2981D4c7	59216	0

Figure 79 Smart Contract deployment

Thanks to this Smart Contract generation, OLS can operate the ShoppingService based on the ToU1 consent. While OLS can share/refine the exchangedAssets which are AccountInformation, DeliveryInformation, and OrderingInformation from Alice and can require and use the container of exchangedAssets from Alice. While this workflow is launched, different smart contracts are generated from OLS to the CompanyB to prove their usages on the exchangedAssets (see the dependencies Figure 80):

- Alice generates TrackingSmartContract1(issue of Container and Data Object) that Alice will certify to transfer the Container (AccountInformation) to OLS
- OLS generates ExchangeSmartContract1 to declare the BusinessService1 with ExchangedAsset and authorized Usage from the ToU1 signed by Alice

- OLS invokes ExchangeSmartContract1 to declare its usages. OLS will select the BusinessService2 to send the ExchangedAsset and delegate its authorized usages to CompanyB.
  - OLS invokes ExchangeSmartContract1 to declare its usages. OLS will select the ElementaryTask1 to send the ExchangedAsset and delegate its authorized usages to OLS.dp.
  - OLS invokes ExchangeSmartContract1 to declare its usages. OLS will select the ElementaryTask2 to send the ExchangedAsset and delegate its authorized usages to OLS.dp.
  - OLS.dp generates ExchangeSmartContract2 to declare the ElementaryTask1 with ExchangedAsset and authorized Usage from the output of ExchangeSmartContract1
  - OLS.dp invokes ExchangeSmartContract2 to declare its usage. OLS.dp will select the LogicalService1 to send the ExchangedAsset and delegate its authorized usages to its employee who is in charge of ManagementService.
  - ManagementService employee generates UsageSmartContract1 to declare the LogicalService with DataObject and authorized Usage from the output of ExchangedSmartContract2 and declare the required Countermeasure efficiencies.
  - ManagementService employee invokes UsageSmartContract1 to declare its usage. ManagementService employee will update the LogicalService with provided Countermeasure efficiencies.
  - ManagementService employee invokes TrackingSmartContract to verify Alice's business authorization assertion and logical operation assertion of exchangedAsset and acquire the container certification for the Data Object.
  - ManagementService employee will generate PhysicalSmartContract1 from the data object belonging to the UsageSmartContract1 in its private blockchain to declare its usage. ManagementService employee will select the concrete services to send container with required Countermeasure efficiencies.
  - ManagementService employee will invoke PhysicalSmartContract1 in its private blockchain to declare its usage. ManagementService employee will update with provided Countermeasure efficiencies.
- At last, concrete Service can get the authorization to access the container by the output of the PhysicalSmartContract1.



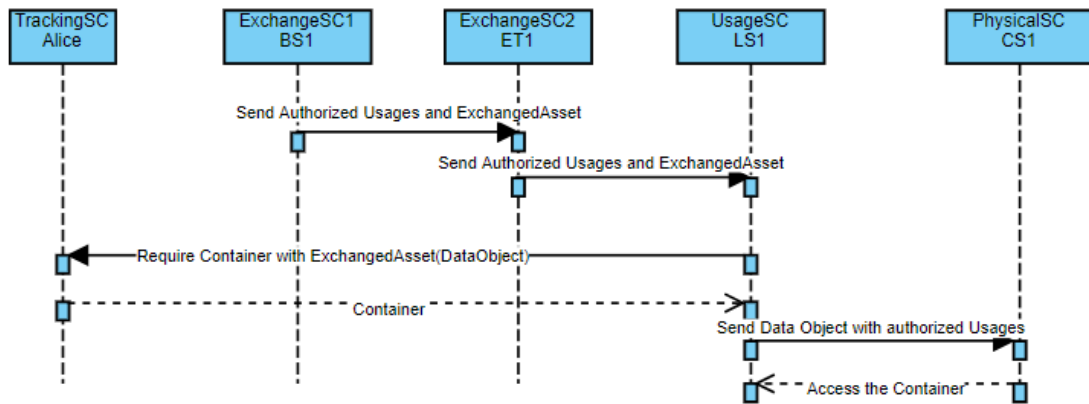


Figure 80 Sequence diagram showing the Smart-Contract dependencies

Alice is associated to a single Actor Account and has a unique key (DataOwner) to govern the BusinessService. It manages the ToUNumber and ExchangedAssetNumber and ContainerNumber. Alice can issue the ExchangedAsset from the its LogicalAsset for BusinessService1 thanks to ToU1. OLS can issue the ExchangedAsset for the BusinessService2 by ToU2. The ExchangedAsset is associated to a logical asset pattern, a logical asset, data object, and a container. States are associated to these different assets described in the section 3.2.3 with events description.

Focusing on the Blockchain part, our Proof of Concept evaluation is focused on smart contract generation and deployment. Our solution, which includes several smart contract invocations from dedicated actors, provides a loosely coupled connection to the Blockchain. We use Ganache to support the Blockchain deployment and Remix to manage solidity code. As the blockchain deployment is based on a sandbox and not on a real highly distributed system, we do not provide any performance measures and only focus on the cost of the different smart contracts. To manage this experiment, we invoke several times functions and get the gas cost. From this experiment, gas cost remains stable. We first compare the total deployment cost for each SC (see Figure 81). It shows that the smart contract deployment cost is heavily related to its complexity (in terms of parametric functions it provides). Then we compare the deployment cost with the different function invocation cost for each type of smart contract. It shows that the invocation of the parametric function can be neglected compared to the deployment cost, mostly (see Figure 82, Figure 83, Figure 84, and Figure 85).

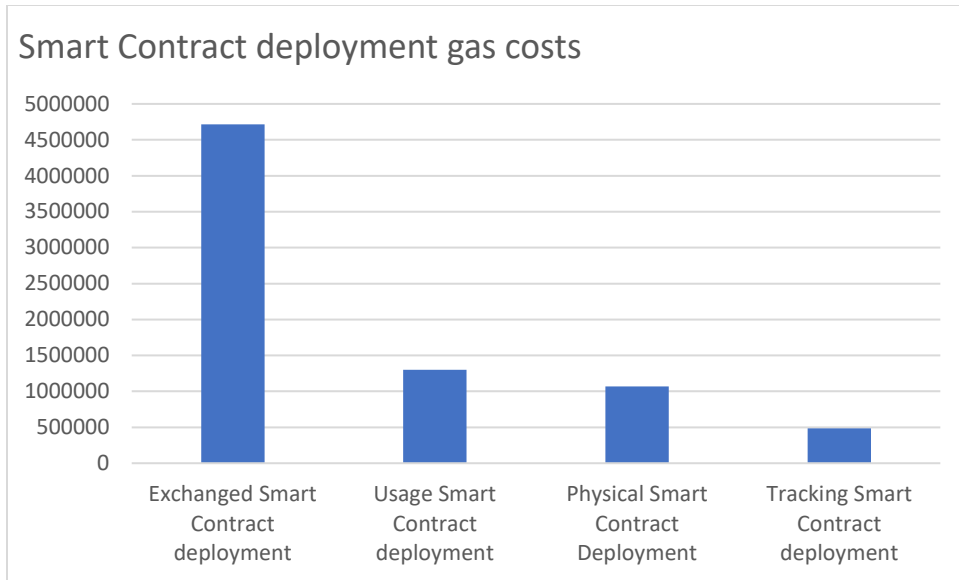


Figure 81 Different Smart Contract deployment costs

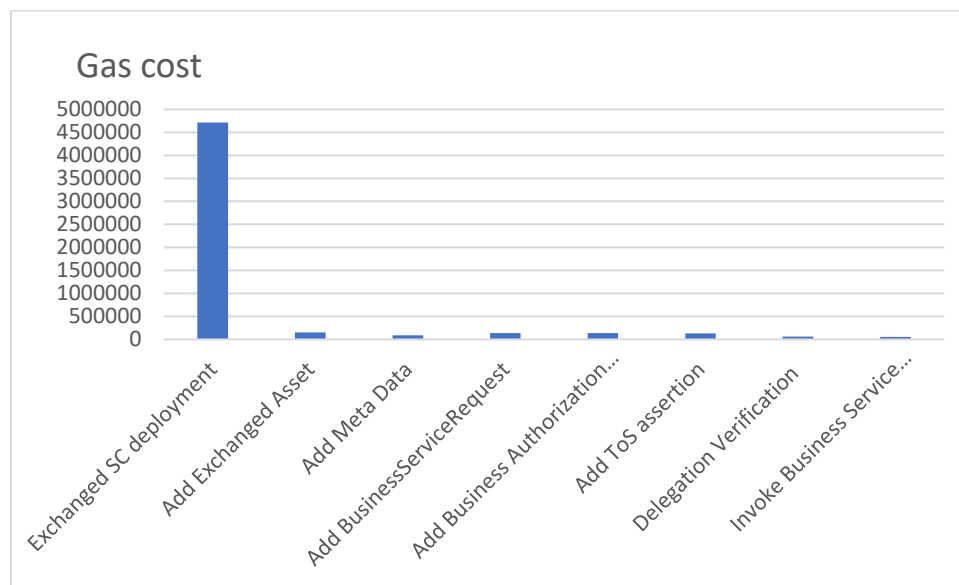


Figure 82 Comparison of the exchange smart contract deployment and invocation function costs

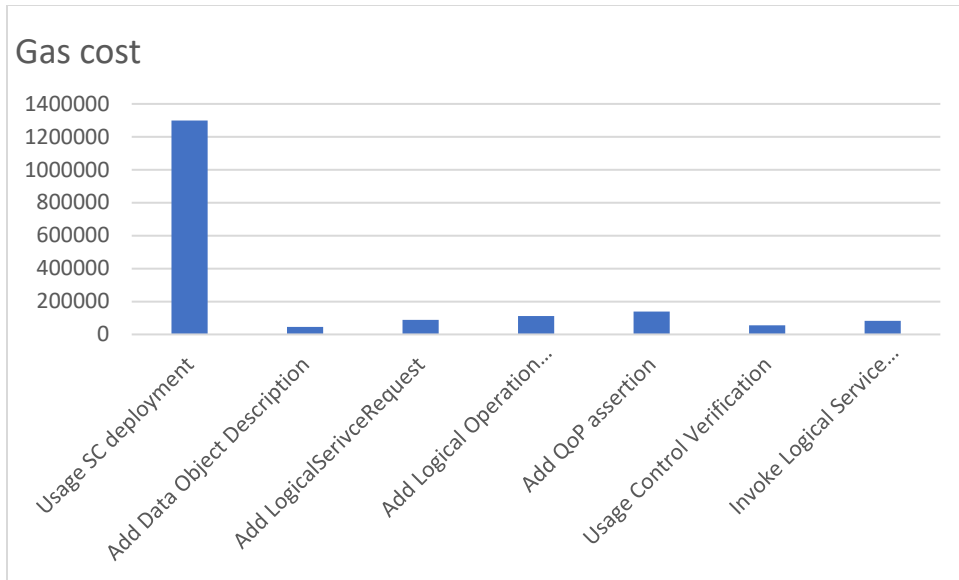


Figure 83 Comparison of the usage smart contract deployment and invocation function costs

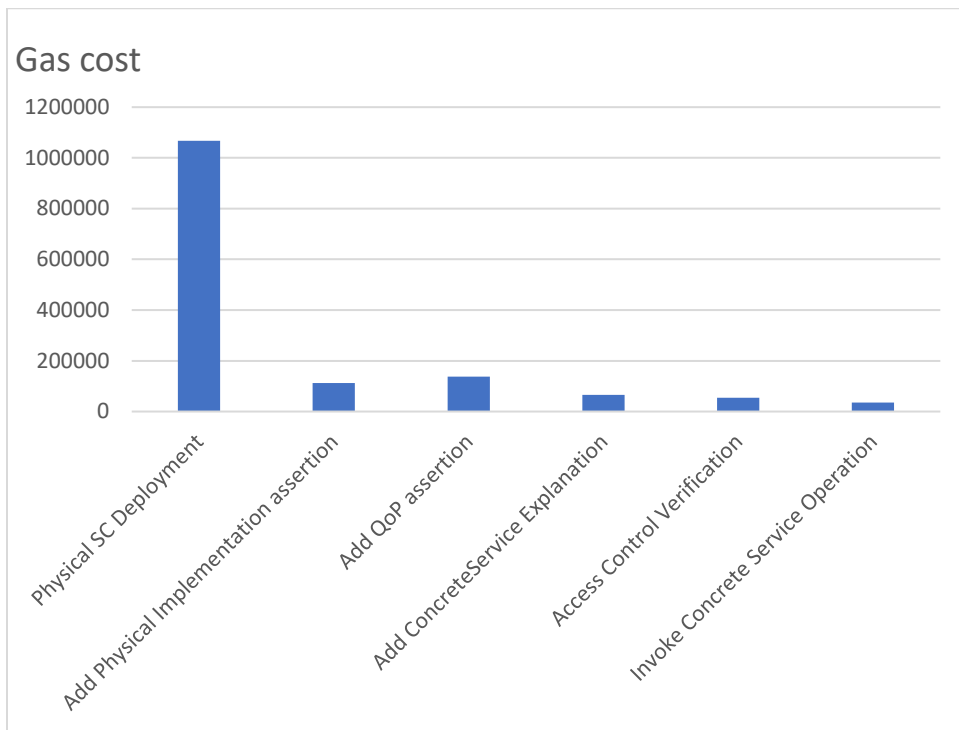
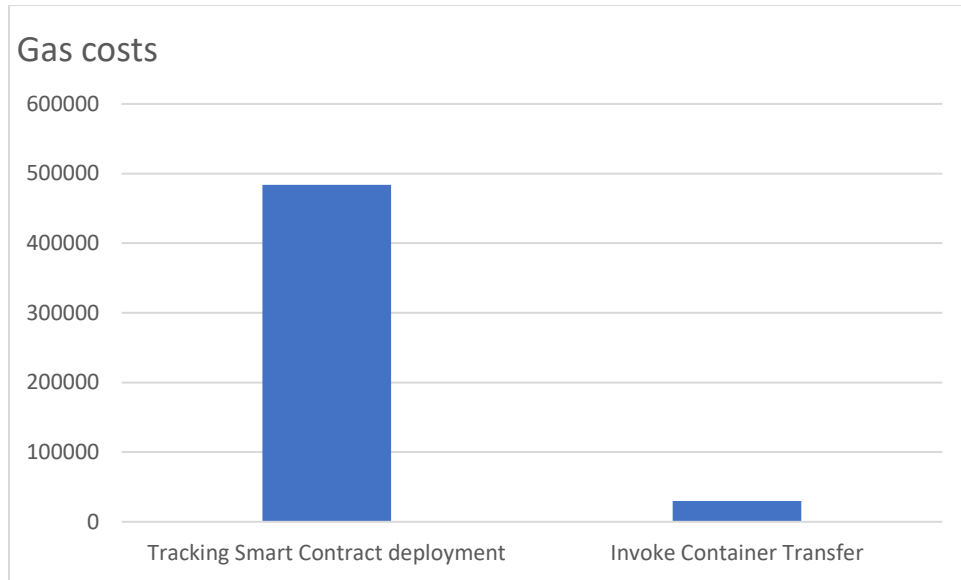


Figure 84 Comparison of the physical smart contract deployment and invocation function costs



*Figure 85 Comparison of the tracking smart contract deployment and invocation function costs*

#### 4.4.2 Evaluation of DUP

After this simple experiment, we evaluate our DUP proposal compared to other related works.

Based on this experiment, we first evaluate our Terms of Usage ontology by comparing it with others. To this end, we identify 3 main comparison criteria:

- subject attribute defines the attributes of the party requiring the access,
- control objective defines the attributes which are used to describe ‘Rights’, ‘Obligation’ and ‘Condition’.
- countermeasure scope includes infrastructure security, communication security, data storage and access control.

As far as the subject attribute is concerned, [72], [79], [78] and [82] define generic roles allowing to capture part of organizational knowledge. Nevertheless, they do not integrate usage-related role (such as data owner, data consumer). This makes harder the definition of subject associated to sharing usages, mostly when collaborative B2B processes are concerned. Focusing on organizational and social knowledge, although [74] extends roles definition by integrating reputation and [80] integrates social relationships, these works only allow managing (trusted) links between actors. Our extended ToU ontology extends the subject description to manage both individual and organizational entities. It also couples with usage-related roles and real subject identity. By this way, it can be used to identify exactly the actors involved in a particular usage. Moreover, it provides a delegation mechanism and a hierarchy of organizational entity allowing defining more or less precisely groups of allowed subjects.

Focusing on the control objective, [73], [78] and [82] consider either the service or the trust level associated to the stakeholder or the asset while [81], [12] and [109] propose rules associated to the semantic value of the asset. These ontologies do not support a

synthetic definition of the business context, making harder to restrict data usage according to business purpose (in our motivating example, Bob's address can be used only for delivery process). Our ToU ontology designed according to our multi-layer model extends the control object to define precisely contextual usage information associated to logical data and physical copies, including archival keeping, data portability, data sharing, CRUD operations.... This allows propagating the usage conditions even when the usage right is delegated. For example; the "restricted to delivery purpose" usage condition on Bob's address can be propagated to the copy shared with with company A.

Focusing on countermeasure scope, [72] and [80] focus on access control whereas [109] even integrates infrastructure condition with access control, allowing integrating the secured exchange channel constraint. Our ToU ontology integrates infrastructure, communication, data-protection and access control means by extending access control and operational service to "business purpose", i.e. generic operations fitting a business goal and "collaboration operations". By this way, the deletion constraints can be taken into account as other protection means (storing encrypted payment token, exchanging data through SSL-based channels...). Compared to other ontologies, our ToU integrates all the necessary elements to describe usage and protection features, including data sharing and usage delegation. By this way, constraints on life-long usage control and protection features can be described using a single ontology. Moreover, the usage-related roles allow integrating the collaborative context (i.e. the relationships between stakeholders) in the fine-grained policy rules.

Our usage-based protection enriches the UCON<sub>ABC</sub> model [15] and the Collaborative Usage Control model [85] by integrating business context and protection countermeasures in the usage condition. Thanks to the organizational knowledge, it also allows couplin CUCON with RBAC access control [13] strategies. This will ease the adaptation of already deployed access control features.

Then, as our usage model extends the Collaborative Usage Control and ToSDR<sup>12</sup> based policies used in the PICS project described in [112], we integrate our ontology in this prototype and extract Requirements of Protection and Quality of Protection policies from the Information System description data base to build XML policy files stored in .rop and .qop files. This prototype uses JENA API and Jena-arq API to model policy files to model and query these .rop and .qop files. This prototype is deployed on top of a HP machine, 2.7 GHz with 8 GB memory, running Windows 10. The application allows to generate .rop files, to aggregate two .qop files of the same service using the negative aggregation, to aggregate two .qop files of already used services, and to match user's requirements with targeted service's quality of protection. SPARQL queries are generated and launched on the .rop and .qop files in order to recover descriptions of services and user's requirements in the form of java objects. illustrates the execution time of the whole process including. RoP generation, QoP aggregation, and .RoP and QoP matching, varying the complexity of the aggregation of sharing data descriptions. To do that, we exploit the running scenario as follows: (1) data sharing assertion is set in only the ToS part, (2) data sharing appears in both RoP and ToS for the same context, (3) data sharing appears in both

---

<sup>12</sup> Terms of Service Didn't Read see <https://tosdr.org/>

RoP and QoP in a different context and (4) includes the previous cases. We observe a small increase in aggregation time as the complexity of the latter increases. Matching time remains stable. These results are similar to those evaluated for the PICS project and are presented in Figure 86

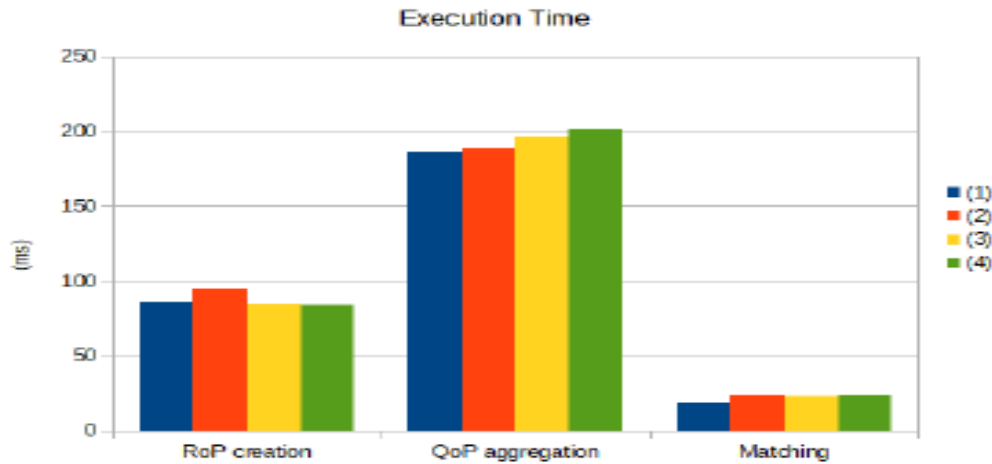


Figure 86 RoP, QoP and matching function execution time based on the PICS prototype

Then, we evaluate our Data-driven and Usage-based Protection architecture with other works integrating GDPR requirements, such as consent management, usage scope definition, operation tracking and life-long protection.

First, we identify that only [90] integrates the usage scope, i.e. business purpose. [104] and [108] refer to traditional consent management which doesn't consider usage scope and is only managed by the subject. [105] retrieves the consent "signature" from a blockchain. [106] and [107] do not integrate data origin to manage consent forwarding. Our system not only manages stand-alone consents, it also integrates consents provided in a collaborative context (i.e. when information is shared by different parties). Our Usage Governance architecture, allows monitoring and evaluating the real operations on the containers, paying attention to the business purpose. Based on the different assertions, our system stores the approved ToU in a Blockchain, data sharing consent can be managed and tracked. Moreover, the exchange smart contract allows certifying the data origin on the data consumer side.

Focusing on tracking abilities, [104] controls data encryption keys to track data access and usage whereas [106] tracks data forwarding and [107] tracks right transfer. Thanks to our governance architecture, our system tracks real operations on containers (i.e. copies of the logical data). As our system manages the rights delegation, the monitoring feature is also extended to other stakeholders getting a copy of a data. By this way, our system controls data usage operation achieved in collaborative and opened context. By this way, the life-long usage-based protection can be tracked and each party can prove that it has fulfilled its obligations.

To sum up this comparison, we identify the same 4 criteria (Table 10):

- *Usage scope*: identifies if the business purpose is considered or not
- *Consent management*: defines if the consent is stored or propagated
- *Tracking*: defines if operations and/or data provenance can be tracked
- *Data life-long protection* means that the data usage limitations and reporting can be achieved even after the data has been transmitted to another party

Ref.	Usage scope	Consent management	Tracking	Life-long protection
[107]	No	Yes	Partly for the right transfer	Partly: shared policy
[104]	No	Managed by the subject	Key exchange	No
[105]	No	Picked from the Blockchain	No	No
[106]	No	Yes	Data forwarding operations	No
[108]	No	Managed by the subject	Data operation	No
<b>Our DUP</b>	Yes	Yes	Data exchange and some operations	Yes

*Table 10 Comparison of our system with other Blockchain-based systems*

## 4.5 Conclusion

In this chapter we have defined our Data-centered and Usage-based Protection architecture. Our DUP is built on the Information System Description model, storing each party own Information System Organization, including the protection strategies. Transactions are associated to Business Processes execution, defining precisely the data assets and the way they are used. Our Terms of Usage generation process integrates the different operations used by the services and the deployed protection means. By this way, the data owner can evaluate precisely the risks and compares it with the Data Requirements of Protection so that the protection consistency can be checked.

Once the Terms of Usage are approved, this consent is derived in more specific usage authorization thanks to Transaction refinement processes. Our system takes advantage of the Blockchain immutability to store both the initial consent and these delegated consents. At runtime, the data consumer can invoke these precise delegated consent to register the usage operations. By this way, consent proofs and operations proofs can be provided.

Thanks to the integration of Business knowledge in this Usage Control process, usage governance and tracking means are set and allow both data provider and data consumer “proving” that assets are used and protected according to what has been approved, answering **question 3: How to manage the usage proofs to support usage and protection governance?**

## 5 Conclusion

Globalized market trends and fast-changing business conditions involved in the Social networks, Mobile computing environment, Big Data Analytics, Cloud, and Internet of Things technologies call for a new information-driven cyber-security management strategy. Traditional security engineering methods and cyber-security environments are built to protect well-perimetrized Information Systems. As they are mostly designed in a control-driven way, data may have different protections depending on the processes in which it is involved in. Moreover, they are designed to manage cyber-risks on the Information System but they do not integrate risks involved by the way data are processed. Lastly, they do not allow user consent management and govern security deployment according to the context. Focusing on SMACIT, risk engineering must be adapted to face opened and evolving environment, and particular attention must be paid to personal information protection. Moreover, SMACIT processes may also be considered as threats or unfair practices. For example, Big Data analytics relies on “cross processes” among several (Personal) Information sources leading to unpredictable privacy breaches. As such, securing consistently (Personal) Information in such opened context is a key challenge for both service providers and service consumers as information protection impacts trust levels between parties.

To fit this challenge, we have identified three main research questions:

- Question 1: Which security strategy can provide a consistent protection
- Question 2: How “fair usages” can be defined and integrated into protection policy
- Question 3: How usages proofs can be managed to support usage governance

To solve question one, we have chosen to promote a data-driven security strategy. To support this data-centric protection strategy, we proposed a multi-layer Information System Description model to capture business knowledge as well as the data and processes organization. In this way, data assets are defined logically and associated with a single requirement of protection policy that can be propagated to the different physical copies of this logical asset (called containers). Security policies describing both Requirements of Protection for the data asset and the quality of protection provided by services using these assets are integrated in this Information System Description model. To provide a simple definition of the requirements of protection, we have proposed a simple discrete scale rating the basic security services (confidentiality, integrity and availability).

As “unfair” usages can be seen as security breaches or threats, we have integrated the fair and due usages in our protection model, regulating the way data assets are consumed and used in different processes. To this end, we have proposed a multi-dimension protection ontology, coupling business knowledge to security ontologies and usage so that business context and security countermeasures can be gathered to define fine-grained usage rights. To fit the opened Information System constraint involved by the Collaborative Networked Organization and the SMACIT context, we have enriched the Collaborative Usage CONTROL ontology with dedicated operations, including data delegation.



By integrating the multi-layer information system description and the protection ontology, we have defined a contextual usage-based protection assertion to manage both security requirements and fair usages, answering question 2.

To answer question 3, we take advantage of this formal usage model to set our data-driven and usage-based protection architecture (DUP for short). This data-driven protection architecture addresses (i) data consumer's control enforcement over assets by a different type of services defined in the multi-layer IS Description model (ii) protection requirements and obligation compliance management between parties in the collaboration and (iii) data owner's usage governance analyzing and evaluating the enforcement. Built on our Information System Description Model, DUP integrates a transaction model to capture the dynamic data asset exchange and usage. Dedicated Transaction generation processes are proposed to derive elementary usage control assertions from the original consent. We take advantage of the Blockchain's immutability to manage consent and usage proofs. To this end, we have proposed different algorithms to generate smart contracts associated to consent and usage authorizations. This Blockchain deployment provides data exchange and usage proofs to data owners and data consumer allowing them to govern the asset usage. A small experiment, mixing the B2B and B2C context has been used to test the global deployment of our DUP prototype.

Our research does not claim to provide a perfect and indisputable answer to these thesis issues. There are still aspects that need to be further investigated:

- Our Information System Description model has been used to derive “concrete” IT usage from more generic business operations. Focusing on the GDPR requirements, security events must be notified to the data provider as soon as they can be identified. Taking advantage of the “concrete service deployment” described in the physical layer, security events may be propagated to the business layers, allowing to alert data providers more efficiently in case of security failures.
- Our ontology has been designed to capture business and IT usages. In this way, protection policies are generated focusing on the consequence of usages on the data. SMAC IT systems introduce that potential usages can be seen as threats. Although we have presented a table to introduce the usage impacts on the different representations of data, other usages may be defined. Text extraction techniques can be used to identify the impacts of these extra usage definitions and enrich our ontology.
- The blockchain-based proof of usage we have introduced relies on multiple elementary usages and involves different actors. We identify two main points that need further improvements
  - First, this solution has a great cost due to the high resource consumption, high memory and storage capabilities, and processing time in the blockchain. It means that it will increase the data provider's cost (gas cost in the blockchain) when the data provider wants to prove his operations with data consumers. To overcome this limit, more complex rights should be integrated into a single smart contract. This involves improving the smart contract generation process to allow these complex rights management in a single smart contract.
  - Second, our solution relies on different actors involved in the public blockchain, and more precisely functional actors are associated with the internal

organization of the data consumer. Blockchain analysis can possibly reveal associated business and usage transactions for an account. Although blockchain technology has masqueraded the social identity of the data consumer, we intend to incorporate the machine learning technique in the transactions to define the characteristics of an account to support the evaluation of the authorization and ownership of the functional actor.

- The last research focus is related to dynamic protection policy enforcement. Applying for data-driven protection in the SMAC IT environments is not straightforward due to several challenges including merging and analytics operations. Data consumers may generate new data assets according to various merging or analytics algorithms. The created data asset may include the intrinsic value of original data. To manage the data asset's consistent protection, these new data should be considered while evaluating the data asset protection. To allow capturing asset similarity, we intend to couple the knowledge graph with our multi-layer model to extend it with “value” relationships between assets and investigate the way these value relationships can be included in the global protection strategy.

## 6 References

- [1] ISO/IEC. Iso 17799, 2002
- [2] ISO/IEC. INTERNATIONAL STANDARD ISO/IEC 27002: 2005(e): information technology security techniques code of practice for information security management, 2005
- [3] Grance T, Hash J, Stevens M, et al. Guide to Information Technology Security Services[M]. National Institute of Standards and Technology, Technology Administration, US Department of Commerce, 2003.
- [4] EBIOS, Central Directorate for Information Systems Security, Version 2010 website. [Online]. Available: <http://www.ssi.gouv.fr>.
- [5] Salvi, Olivier, and Bruno Debray. "A global view on ARAMIS, a risk assessment methodology for industries in the framework of the SEVESO II directive." *Journal of hazardous materials* 130.3 (2006): 187-199.
- [6] Christopher. Alberts, Audrey. Dorofee, James. Stevens, and Carol. Woody. Introduction to the octave approach. Technical report 2003
- [7] Rebert J. Ellison and Andrew P. Moore. Architecture refinement for the design of survivable systems. Technical report, 2001
- [8] Xu Cuihua, Lin Jiajun, "An information systems security evaluation model based on AHP and GRAP", *IEEE Computer Science*, 2009, Vol.105, pp:493-496.
- [9] Xinlan, Zhang, et al. "Information security risk assessment methodology research: Group decision making and analytic hierarchy process." 2010 Second world congress on software engineering. Vol. 2. IEEE, 2010.
- [10] Duckworth, Holly A., and Rosemond Ann Moore. Social responsibility: Failure mode effects and analysis. CRC Press, 2010.
- [11] Gray, Gary, et al. "Assessing aeromedical risk: a three-dimensional risk matrix approach." *Heart* 105.Suppl 1 (2019): s9-s16.
- [12] Wang, L., Wijesekera, D., & Jajodia, S. (2004). A logic-based framework for attribute-based access control. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering* 45-55
- [13] Sandhu, Ravi S. "Role-based access control." *Advances in computers*. Vol. 46. Elsevier, 1998. 237-286.
- [14] Janicke, Helge, et al. "Dynamic access control policies: Specification and verification." *The Computer Journal* 56.4 (2012): 440-463.
- [15] Jaehong Park and Ravi Sandhu. "The UCONABC Usage Control Model". In: *ACM Transactions on Information and System Security* 7.1 (Feb. 2004), pages 128–174. ISSN: 1094-9224. DOI: 10.1145/984334.984339
- [16] Ouyang, Chun, et al. "Formal semantics and analysis of control flow in WS-BPEL." *Science of computer programming* 67.2-3 (2007): 162-198.
- [17] Lalouski, Aliaksandr, Fabio Martinelli, and Paolo Mori. "Usage control in computer security: A survey." *Computer Science Review* 4.2 (2010): 81-99.

- [18] Pretschner, Alexander, Enrico Lovat, and Matthias Büchler. "Representation-independent data usage control." *Data Privacy Management and Autonomous Spontaneous Security*. Springer, Berlin, Heidelberg, 2011. 122-140.
- [19] Pretschner, Alexander, Manuel Hilty, and David Basin. "Distributed usage control." *Communications of the ACM*. 2006.
- [20] Hilty, Manuel, et al. "A policy language for distributed usage control." *European Symposium on Research in Computer Security*. Springer, Berlin, Heidelberg, 2007.
- [21] Kumari, Prachi, et al. "Distributed data usage control for web applications: a social network implementation." *Proceedings of the first ACM conference on Data and application security and privacy*. ACM, 2011.
- [22] Kelbert, Florian, and Alexander Pretschner. "Data usage control enforcement in distributed systems." *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013.
- [23] Multimedia framework (MPEG-21) – Part 5: Rights Expression Language, 2004. ISO/IEC standard 21000-5:2004.
- [24] Open Mobile Alliance. DRM Rights Expression Language V2.1, 2008. [http://www.openmobilealliance.org/Technical/release\\_program/drm\\_v2\\_1.aspx](http://www.openmobilealliance.org/Technical/release_program/drm_v2_1.aspx)
- [25] Preda, Stere, et al. "Model-driven security policy deployment: property oriented approach." *International Symposium on Engineering Secure Software and Systems*. Springer, Berlin, Heidelberg, 2010.
- [26] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In *Workshop on Policies for Distributed Systems and Networks '95*.
- [27] Zhang, Xinwen, et al. "A logical specification for usage control." *Proceedings of the ninth ACM symposium on Access control models and technologies*. ACM, 2004.
- [28] Gupta, Brij Bhooshan, Shingo Yamaguchi, and Dharma P. Agrawal. "Advances in security and privacy of multimedia big data in mobile and cloud computing." *Multimedia Tools and Applications* 77.7 (2018): 9203-9208.
- [29] Feth D, Maier A, Polst S. A User-Centered Model for Usable Security and Privacy[C]//International Conference on Human Aspects of Information Security, Privacy, and Trust. Springer, Cham, 2017: 74-89.
- [30] Patsakis C, Zigomitos A, Papageorgiou A, et al. Distributing privacy policies over multimedia content across multiple online social networks[J]. *Computer Networks*, 2014, 75: 531-543.
- [31] Hilty M, Pretschner A, Schaefer C, et al. Usage control requirements in mobile and ubiquitous computing applications[C]//2006 International Conference on Systems and Networks Communications (ICSNC'06). IEEE, 2006: 27-27.
- [32] Jing X, Yan Z, Pedrycz W. Security Data Collection and Data Analytics in the Internet: A Survey[J]. *IEEE Communications Surveys & Tutorials*, 2018, 21(1): 586-618.
- [33] Cao L. Social security and social welfare data mining: An overview[J]. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012, 42(6): 837-853.
- [34] National Institute of Standards and Technology, The NIST Definition of Cloud Computing, Information Technology Laboratory, 2009.

- [35] NIST, NIST Cloud Computing Reference Architecture, 2011
- [36] Mell, P. & Grance, t. (2011) The NIST Definition of Cloud Computing, (Special Publication 800-145).
- [37] Chen D, Zhao H. Data security and privacy protection issues in cloud computing[C]//2012 International Conference on Computer Science and Electronics Engineering. IEEE, 2012, 1: 647-651.
- [38] <https://ccskguide.org/jericho-cloud-cube-model/>
- [39] Hashizume K, Rosado D G, Fernández-Medina E, et al. An analysis of security issues for cloud computing[J]. Journal of internet services and applications, 2013, 4(1): 5.
- [40] Modi C, Patel D, Borisaniya B, et al. A survey on security issues and solutions at different layers of Cloud computing[J]. The journal of supercomputing, 2013, 63(2): 561-592.
- [41] Hendre A, Joshi K P. A semantic approach to cloud security and compliance[C]//2015 IEEE 8th International Conference on Cloud Computing. IEEE, 2015: 1081-1084.
- [42] Betgé-Brezetz S, Kamga G B, Ghorbel M, et al. Privacy control in the cloud based on multilevel policy enforcement[C]//2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET). IEEE, 2012: 167-169.
- [43] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing[J]. Journal of network and computer applications, 2011, 34(1): 1-11.
- [44] Smith, Matthew, et al. "Big data privacy issues in public social media." 2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST). IEEE, 2012.
- [45] Jain, Priyank, Manasi Gyanchandani, and Nilay Khare. "Big data privacy: a technological perspective and review." Journal of Big Data 3.1 (2016): 25.
- [46] D'Acquisto, Giuseppe, et al. "Privacy by design in big data: an overview of privacy enhancing technologies in the era of big data analytics." arXiv preprint arXiv:1512.06000 (2015).
- [47] Feng, Deng-Guo, et al. "Study on cloud computing security." Journal of software 22.1 (2011): 71-83.
- [48] Suo, Hui, et al. "Security and privacy in mobile cloud computing." 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, 2013.
- [49] Jain, P., Gyanchandani, M. & Khare, N. Big data privacy: a technological perspective and review. *J Big Data* 3, 25 (2016). <https://doi.org/10.1186/s40537-016-0059-y>
- [50] Sagioglu, Seref, and Duygu Sinanc. "Big data: A review." 2013 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2013.
- [51] Terzi, Duygu Sinanc, Ramazan Terzi, and Seref Sagioglu. "A survey on security and privacy issues in big data." 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST). IEEE, 2015.
- [52] S. Curry, E. Kirda, E. Schwartz, W.H. Stewart and A. Yoran, "Big Data Fuels Intelligence Driven Security", RSA Security Brief, January 2013 <http://www.emc.com/collateral/industry-overview/big-data-fuelsintelligence-driven-security-io.pdf>

- [53] Cárdenas, Alvaro A., Pratyusa K. Manadhata, and Sreeranga P. Rajan. "Big data analytics for security." *IEEE Security & Privacy* 11.6 (2013): 74-76.
- [54] Feng Dengguo, Zhang Min, Li Hao. "Big data security and privacy protection" *Chinese Journal of computers* No.10 Vol.36 2013
- [55] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." (2003).
- [56] Borthakur, Dhruba. "HDFS architecture guide." *Hadoop Apache Project* 53 (2008): 1-13.
- [57] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.
- [58] Krasnow Waterman, K., and Paula J. Bruening. "Big Data analytics: risks and responsibilities." *International Data Privacy Law* 4.2 (2014): 89-95.
- [59] Lafuente G. The big data security challenge[J]. *Network security*, 2015, 2015(1): 12-14.
- [60] Zhang, Dongpo. "Big data security and privacy protection." 8th International Conference on Management and Computer Science (ICMCS 2018). Atlantis Press, 2018.
- [61] Inukollu, Venkata Narasimha, Sailaja Arsi, and Srinivasa Rao Ravuri. "Security issues associated with big data in cloud computing." *International Journal of Network Security & Its Applications* 6.3 (2014): 45.
- [62] Mehmood, Abid, et al. "Protection of big data privacy." *IEEE access* 4 (2016): 1821-1834.
- [63] Hind Benfenatki, Frédérique Biennier. "Business Process-Based Legitimacy of Data Access Framework for Enterprise Information Systems Protection" *International Conference on Research and Practical Issues of Enterprise Information Systems 2018* pp146-160
- [64] Jeff A. Estefan, Ken Laskey, et al. "Reference architecture for service-oriented architecture version 0.3" *OASIS standard* 2008
- [65] MacKenzie, C. Matthew, et al. "Reference model for service-oriented architecture 1.0." *OASIS standard* 12 (2006): 18.
- [66] Graa, Mariem, et al. "Using requirements engineering in an automatic security policy derivation process." *Data Privacy Management and Autonomous Spontaneous Security*. Springer, Berlin, Heidelberg, 2012. 155-172.
- [67] Autrel, Fabien, et al. "MotOrBAC 2: a security policy tool." 3rd Conference on Security in Network Architectures and Information Systems (SAR-SSI 2008), Loctudy, France. 2008.
- [68] Herrmann, Debra S. *Using the Common Criteria for IT security evaluation*. Auerbach Publications, 2002.
- [69] Masood, Rahat, Muhammad Awais Shibli, and Muhammad Bilal. "Usage control model specification in XACML policy language." *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, Berlin, Heidelberg, 2012.

- [70] Sirin, Evren, et al. "Pellet: A practical owl-dl reasoner." *Web Semantics: science, services and agents on the World Wide Web 5.2* (2007): 51-53.
- [71] Horridge, Matthew, et al. "A practical guide to building owl ontologies using protégé 4 and co-ode tools edition 1. 3." *The University of Manchester* 178 (2011).
- [72] Nejd, Wolfgang, et al. "Ontology-based policy specification and management." *European Semantic Web Conference*. Springer, Berlin, Heidelberg, 2005.
- [73] Choi, Chang, Junho Choi, and Pankoo Kim. "Ontology-based access control model for security policy reasoning in cloud computing." *The Journal of Supercomputing* 67.3 (2014): 711-722.
- [74] Chaari, Sodki, Youakim Badr, and Frédérique Biennier. "Enhancing web service selection by QoS-based ontology and WS-policy." *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008.
- [75] Horrocks, Ian, et al. "SWRL: A semantic web rule language combining OWL and RuleML." *W3C Member submission* 21.79 (2004).
- [76] Tonti, Gianluca, et al. "Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder." *International Semantic Web Conference*. Springer, Berlin, Heidelberg, 2003.
- [77] Finin, Tim, et al. "R OWL BAC: representing role-based access control in OWL." *Proceedings of the 13th ACM symposium on Access control models and technologies*. ACM, 2008.
- [78] Liu, Chi-Lun. "Cloud service access control system based on ontologies." *Advances in Engineering Software* 69 (2014): 26-36.
- [79] Hu, Yuh-Jong, Hong-Yi Guo, and Guang-De Lin. "Semantic enforcement of privacy protection policies via the combination of ontologies and rules." *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*. IEEE, 2008.
- [80] Tsai, Wei-Tek, and Qihong Shao. "Role-based access-control using reference ontology in clouds." *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE, 2011.
- [81] Masoumzadeh, Amirreza, and James Joshi. "Osnac: An ontology-based access control model for social networking systems." *2010 IEEE Second International Conference on Social Computing*. IEEE, 2010.
- [82] García, Roberto, et al. "Formalising ODRL semantics using web ontologies." *Proc. 2nd Intl. ODRL Workshop*. 2005.
- [83] Qu, Yuzhong, Xiang Zhang, and Huiying Li. "OREL: an ontology-based rights expression language." *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004.
- [84] Garcia, Diego, et al. "Towards a base ontology for privacy protection in service-oriented architecture." *2009 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 2009.
- [85] Su Z, Biennier F. End-to-end security policy description and management for collaborative system[C]//2010 Sixth International Conference on Information Assurance and Security. IEEE, 2010: 137-142.

- [86] Wei Wang, Hongwei Ding, Jin Dong, Changrui Ren, 2006. A comparison of Business Process modelling methods. IBM Research report RC23988 (C0606-008)
- [87] Azeem Lodhi, Veit Köppen, Gunter Saake: An Extension of BPMN Meta-model for Evaluation of Business Processes. *Sci. J. Riga Tech. Univ. Ser. Comput. Sci.* 43: 27-34 (2011)
- [88] OMG 2008. Business Process Definition Meta-model. Vol. 2 Process definitions. November 2008, 146 pages
- [89] Règlement européen sur la protection des données : ce qui change pour les professionnels, 10 juillet 2018. Available: <https://www.cnil.fr/fr/reglement-europeen-sur-la-protection-des-donnees-ce-qui-change-pour-les-professionnels>
- [90] Burmeister, F., Drews, P., & Schirmer, I. (2019). A Privacy-driven Enterprise Architecture Meta-Model for Supporting Compliance with the General Data Protection Regulation. In Proceedings of the 52nd Hawaii International Conference on System Sciences.
- [91] Cha, S. C., & Yeh, K. H. (2018). A data-driven security risk assessment scheme for personal data protection. *IEEE Access*, 50510-50517.
- [92] Tesfay W B, Hofmann P, Nakamura T, et al. I Read but Don't Agree: Privacy Policy Benchmarking using Machine Learning and the EU GDPR[C]//Companion of the The Web Conference 2018 on The Web Conference 2018. International World Wide Web Conferences Steering Committee, 2018: 163-166.
- [93] Pandit H J, Fatema K, O'Sullivan D, et al. GDPRtEXT-GDPR as a Linked Data Resource[C]//European Semantic Web Conference. Springer, Cham, 2018: 481-495.
- [94] Zarsky T Z. Incompatible: The GDPR in the age of big data[J]. *Seton Hall L. Rev.*, 2016, 47: 995.
- [95] Loukil F, Ghedira-Guegan C, Boukadi K, et al. Liopy: A legal compliant ontology to preserve privacy for the internet of things[C]//2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2018, 2: 701-706.
- [96] Sameer HAKIM, Ziyang LI, Yi PAN, Fabrice ZAUMSEIL, Huihui CHI, Wei ZHOU. The Impact of General Data Protection Regulation (GDPR) on Data Management Platforms (DMP): A Policy Perspective. *Management & Data Science* 2018.09
- [97] Bolognini L, Bistolfi C. Pseudonymization and impacts of Big (personal/anonymous) Data processing in the transition from the Directive 95/46/EC to the new EU General Data Protection Regulation[J]. *Computer law & security review*, 2017, 33(2): 171-181.
- [98] Anisetti M, Ardagna C, Bellandi V, et al. Privacy-aware Big Data Analytics as a service for public health policies in smart cities[J]. *Sustainable cities and society*, 2018, 39: 68-77.
- [99] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, Feb. 24, 2013. (<http://bitcoin.org/bitcoin.pdf>)
- [100] Li X, Jiang P, Chen T, et al. A survey on the security of blockchain systems[J]. *Future Generation Computer Systems*, 2017.
- [101] Hashemi, S.H., Faghri, F., Rausch, P., Campbell, R.H.: World of empowered IoT users. In: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 13–24. IEEE (2016)



- [102] Zyskind, G., Nathan, O., et al.: Decentralizing privacy: using blockchain to protect personal data. In: 2015 IEEE Security and Privacy Workshops (SPW), pp. 180–184. IEEE (2015)
- [103] Maesa D D F, Mori P, Ricci L. Blockchain based access control[C]//IFIP international conference on distributed applications and interoperable systems. Springer, Cham, 2017: 206-220.
- [104] Wirth C, Kolain M. Privacy by blockchain design: a blockchain-enabled GDPR-compliant approach for handling personal data[C]//Proceedings of 1st ERCIM Blockchain Workshop 2018. European Society for Socially Embedded Technologies (EUSSET), 2018.
- [105] Truong N B, Sun K, Lee G M, et al. Gdpr-compliant personal data management: A blockchain-based solution[J]. IEEE Transactions on Information Forensics and Security, 2019, 15: 1746-1761.
- [106] Neisse R, Steri G, Nai-Fovino I. A blockchain-based approach for data accountability and provenance tracking[C]//Proceedings of the 12th International Conference on Availability, Reliability and Security. 2017: 1-10.
- [107] Di Francesco Maesa, D., Mori, P., & Ricci, L. Distributed access control through Blockchain technology(2017).
- [108] Kaaniche, N., & Laurent, M. A Blockchain-based data usage auditing architecture with enhanced privacy and availability. In 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA) pp. 1-5. IEEE (2017, October).
- [109] Kim, A., Luo, J., & Kang, M. (2005). Security ontology for annotating resources. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems",1483-1499
- [110] Jingya Yuan, Frédérique Biennier & Aïcha-Nabila Benharkat (2020). « Data centered and Usage-based Security service ». 2nd Workshop on Smart Data Integration and Processing on Service Based Environments, 14 décembre 2020, Dubai (Émirats Arabes Unis). In H. Hacid et al. (Eds.): ICSOC 2020 Workshops, LNCS 12632, pp. 1–15, 2021. [https://doi.org/10.1007/978-3-030-76352-7\\_43](https://doi.org/10.1007/978-3-030-76352-7_43)
- [111] Jingya Yuan, Frédérique Biennier & Aïcha-Nabila Benharkat (2020). « A Data-centered Usage Governance: Providing Life-long Protection to Data Exchanged in Virtual Enterprises ». CEIS : 22nd International Conference on Enterprise Information Systems, 7 mai 2020, Prague (France), pp. 177-184. doi : 10.5220/0009349501770184.
- [112] Hind Benfenatki & Frédérique Biennier (2018). « Business Process-Based Legitimacy of Data Access Framework for Enterprise Information Systems Protection ». 12th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS), 19 septembre 2018, Poznan (Pologne), pp. 146-160. doi : 10.1007/978-3-319-99040-8\_12.

## 7 Annex: Smart Contract Patterns

### 7.1 Exchange Smart Contract pattern

```
contract ExchangeSmartContract{ //define a businessService managed by a delegator
    address public owner; //Data consumer who will link to the smart contract factory to give the
    authorization
    bool public isElementaryTask;
    constructor(bool _isElementaryTask)public{
        isElementaryTask=_isElementaryTask;
        owner=msg.sender;
    }
    mapping(uint=>ExchangedAsset) ExchangedAssetList;
    mapping(string=>uint) ExchangedAssetFind;
    uint public ExchangedAssetAccount;
    struct ExchangedAsset{
        address creator; //the previous address ExchangeSmartContract or address DataProvider;
        string LogicalAssetPattern; //the meaning of this LogicalAssetPattern;
        string ExchangedAssetstatus; // it will have the global(input) of the BusinessService or the internal
    ExchangedAsset;
        string[] metaDataList;
        uint ExpirationTime;
        mapping(string=>uint) metaDataToExchangedAsset;
    }
    modifier Onlyowner(address _account){
        require(_account==owner);
        _;
    }
    // Define the ExchangedAssetPart
    function CreateExchangedAsset(address _account, string memory _status, uint _Expiration, string
    memory _LogicalAssetPatternName)public Onlyowner(msg.sender)returns(uint){ //initial create the
    ExchangedAssets
        ExchangedAssetAccount++;
        ExchangedAsset storage currentAsset=ExchangedAssetList[ExchangedAssetAccount];
        currentAsset.creator=_account;
        currentAsset.ExchangedAssetstatus=_status;
        currentAsset.ExpirationTime=now+ _Expiration * 1 seconds;
        currentAsset.LogicalAssetPattern=_LogicalAssetPatternName;
        ExchangedAssetFind[_LogicalAssetPatternName]=ExchangedAssetAccount;
        return ExchangedAssetAccount;
    }
    function AddMetaData(uint _FatherExchangedAssetId, uint _ChildExchangedAssetId, string
    memory metaData)public Onlyowner(msg.sender)returns(bool){ //add the relationship of the
    ExchangedAssets
        bool success=false;

        if(keccak256(bytes(ExchangedAssetList[_ChildExchangedAssetId].LogicalAssetPattern))==keccak25
        6(bytes(metaData))) {
            ExchangedAssetList[_FatherExchangedAssetId].metaDataList.push(metaData);
```

```

ExchangedAssetList[_FatherExchangedAssetId].metaDataToExchangedAsset[metaData]=_ChildEx
changedAssetId;
    success=true;
    }
    return success;
}
//Define the Authorization from the ToU Assertion
mapping(string=>mapping(string=>BusinessPolicy)) BusinessAuthorization;
struct BusinessAssertion{
    string processMotivation;
    string DataRelatedOperation;
    string UsageName;
    bool permitted;
}
struct BusinessPolicy{
    mapping(uint=>BusinessAssertion) policy; //the detailed description of the assertion
    uint size; //the number of the assertion
}
function UpdateBusinessAuthorization(string memory _processMotivation, string memory
_DataRelatedOperation, string memory _UsageName, bool _permitted, string memory
_LogicalAssetPattern, string memory _BusinessPurpose)public Onlyowner(msg.sender)returns(uint){
    require(VerifyExchangedAsset(_LogicalAssetPattern)==true);
    BusinessPolicy storage
currentPolicy=BusinessAuthorization[_LogicalAssetPattern][_BusinessPurpose];
    currentPolicy.size++;
    uint number=currentPolicy.size;
    BusinessAssertion storage currentAssertion=currentPolicy.policy[currentPolicy.size];
    currentAssertion.processMotivation=_processMotivation;
    currentAssertion.DataRelatedOperation=_DataRelatedOperation;
    currentAssertion.UsageName=_UsageName;
    currentAssertion.permitted=_permitted;
    return number;
} //The delegator will know the assertionId of each assertion;
// define the ToS(QoP) of the DataConsumer(delegatee)
mapping(address=>Request) RequestGroup; //for different delegatee
struct Request{
    string LogicalAssetPattern; //the exchangedAsset
    string BusinessUsage; //the operation between Delegator and delegatee
    string BusinessPurpose; //Delegatee's business purpose
    string processMotivation;
    string DataRelatedOperation;
    uint OwnedAssetTime;
    mapping(string=>mapping(string=>RequiredToS)) RequiredDelegation; //
metaData=>BusinessArea; (Delegatee as the Delagtor to the subservice)
}
struct RequiredToS{
    uint AssertionAmout;
    mapping(uint=>ToS) Delegation;
}
struct ToS{
    string processMotivation;

```

```

    string DataRelatedOperation;
    string UsageName;
    bool permitted;
    bool isMatched;
}
function VerifyExchangedAsset(string memory _LogicalAssetPatternName)internal view
returns(bool){
    if(ExchangedAssetFind[_LogicalAssetPatternName]>0){
        return true;
    }else{
        return false;
    }
}
function subBusinessServiceRequest(string memory _LogicalAssetPatternName, string memory
_BusinessUsage, address _Delegatee, string memory _BusinessArea, string memory
_processMotivation, string memory _DataRelatedOperation)public Onlyowner(msg.sender){
    require(VerifyExchangedAsset(_LogicalAssetPatternName)==true);
    Request storage currentRequest=RequestGroup[_Delegatee];
    currentRequest.LogicalAssetPattern=_LogicalAssetPatternName;
    currentRequest.BusinessUsage=_BusinessUsage;
    currentRequest.BusinessPurpose=_BusinessArea;
    currentRequest.processMotivation=_processMotivation;
    currentRequest.DataRelatedOperation=_DataRelatedOperation;
}
function updateToS(string memory _processMotivation, string memory _DataRelatedOperation,
string memory _UsageName, bool _permitted, string memory _metaData, string memory
_BusinessPurpose, address _Delegatee)public Onlyowner(msg.sender){
    Request storage currentRequest=RequestGroup[_Delegatee];
    RequiredToS storage
currentTos=currentRequest.RequiredDelegation[_metaData][_BusinessPurpose];
    currentTos.AssertionAmout++;
    ToS storage usageToken=currentTos.Delegation[currentTos.AssertionAmout];
    usageToken.processMotivation=_processMotivation;
    usageToken.DataRelatedOperation=_DataRelatedOperation;
    usageToken.UsageName=_UsageName;
    usageToken.permitted=_permitted;
}
event BusinessUsageEvent(string LogicalAssetPattern,string purpose, string BusinessUsage, uint
Time, bool success, string OwnershipOrigin, string UsageRelatedOwnership);
function BusinessServiceOperation(address _delegatee, uint TokenId,uint _Time, string memory
OwnershipOrigin, string memory UsageRelatedOwnership )public Onlyowner(msg.sender){ //sign
the assetTransfer prove (DECLARE to assertionId(CREDENTIAL) AND MANAGE THE
TRANSFER)
    bool success=false;
    string memory usage= RequestGroup[_delegatee].BusinessUsage;
    string memory purpose=RequestGroup[_delegatee].BusinessPurpose;
    string memory asset=RequestGroup[_delegatee].LogicalAssetPattern;
    string memory processMotivation=RequestGroup[_delegatee].processMotivation;
    string memory DataRelatedOperation=RequestGroup[_delegatee].DataRelatedOperation;

require(keccak256(bytes(BusinessAuthorization[asset][purpose].policy[TokenId].processMotivation))
==keccak256(bytes(processMotivation)));

```

```

require(keccak256(bytes(BusinessAuthorization[asset][purpose].policy[TokenId].DataRelatedOperation))
==keccak256(bytes(DataRelatedOperation)));

require(keccak256(bytes(BusinessAuthorization[asset][purpose].policy[TokenId].UsageName))
==keccak256(bytes(usage)));
    require(!_Time<=ExchangedAssetList[ExchangedAssetFind[asset]].ExpirationTime);
    if(BusinessAuthorization[asset][purpose].policy[TokenId].permitted==true){
        success=true;
    }
    emit BusinessUsageEvent(asset,purpose,usage,_Time,success, OwnershipOrigin,
UsageRelatedOwnership);
}
function DelegationMatch(address _delegatee, uint AssertionId, uint ToSID, string memory
_metaData, string memory _BusinessArea)public Onlyowner(msg.sender){ // sign the UsageToken
prove
    Request storage request=RequestGroup[_delegatee];

require(keccak256(bytes(BusinessAuthorization[_metaData][_BusinessArea].policy[AssertionId].proc
essMotivation))
==keccak256(bytes(request.RequiredDelegation[_metaData][_BusinessArea].Delegati
on[ToSID].processMotivation)));

require(keccak256(bytes(BusinessAuthorization[_metaData][_BusinessArea].policy[AssertionId].Data
RelatedOperation))
==keccak256(bytes(request.RequiredDelegation[_metaData][_BusinessArea].Dele
gation[ToSID].DataRelatedOperation)));

require(keccak256(bytes(BusinessAuthorization[_metaData][_BusinessArea].policy[AssertionId].Usag
eName))
==keccak256(bytes(request.RequiredDelegation[_metaData][_BusinessArea].Delegati
on[ToSID].processMotivation)));

if(BusinessAuthorization[_metaData][_BusinessArea].policy[AssertionId].permitted==request.Requir
edDelegation[_metaData][_BusinessArea].Delegation[ToSID].permitted){
    request.RequiredDelegation[_metaData][_BusinessArea].Delegation[ToSID].isMatched=true;
}
}
// To generate the UsageSmartContract
function UsageSmartContractGeneration(uint AssertionId, string memory _metaData, string memory
_BusinessPurpose)public Onlyowner(msg.sender)returns(UsageSmartContract usageAddress){
    require(isElementaryTask==true);

require(BusinessAuthorization[_metaData][_BusinessPurpose].policy[AssertionId].permitted==true);
    string memory
usageName=BusinessAuthorization[_metaData][_BusinessPurpose].policy[AssertionId].UsageName;
    return new UsageSmartContract(usageName, _metaData);
}
}
}

```

## 7.2 Usage Smart Contract pattern

```

contract UsageSmartContract{
    address creator; // the exchangeSmartContract that creates this UsageSC.

```

```

address owner; //the external delegator that generate the UsageSC.
struct DataObject{
    string LogicalAssetPatternName;
    mapping(string=>string) ContaineroPERATIONDescription; //The containerDescription of
THIS uSAGE.
    string inputPattern; //the url/browser for logical service. It will send the container to the Logical
service's concrete Services
}
struct UsageofLogicalService{
    string UsageName;
    uint UsageDuration;
    string ProcessControlPurpose;
}
UsageofLogicalService LogicalOperation;
DataObject dataObject;
mapping(uint=>LogicalOperationAssertion) CountermeasureProtectionRequirement;
uint AssertionNumber;
constructor(string memory Usage, string memory _metaData )public{
    creator=msg.sender;
    owner=tx.origin;
    dataObject.LogicalAssetPatternName=_metaData;
    LogicalOperation.UsageName=Usage;
}
function addContainerDescripton(string memory _attribute, string memory _value)public
Onlyowner(msg.sender){ //add the description of the Container or add the generated
newLogicalAssetPATTERN
    dataObject.ContaineroPERATIONDescription[_attribute]=_value;
}
event LogicalAssetLifeCycleEvent(string LogicalAssetPattern,string UsageName, string
ProcessControlPurpose, string functionalDescription, uint UsageDuration, string LifeCycleState);
function LogicalServiceOperation(uint _Usageduration, string memory _ProcessControlPurpose,
string memory LifeCycleState)public Onlyowner(msg.sender){
    LogicalOperation.UsageDuration=_Usageduration;
    LogicalOperation.ProcessControlPurpose=_ProcessControlPurpose;
    string memory Usage=LogicalOperation.UsageName;
    string memory asset=dataObject.LogicalAssetPatternName;
    emit LogicalAssetLifeCycleEvent(asset,Usage,LogicalOperation.ProcessControlPurpose,
Service.functionalDescription, LogicalOperation.UsageDuration, LifeCycleState);
}
struct LogicalOperationAssertion{
    string LogicalServiceContextDescription; //such as ServiceType or some LogicalServiceContext
    string CountermeasureStrategy;
    string CountermeasureLevel;
}

```

```

}
function updatePolicy(string memory _LogicalServiceContextDescription, string memory
_CountermeasureStrategy, string memory _CountermeasureLevel)public
Onlyowner(msg.sender)returns(uint){
    AssertionNumber++;

CountermeasureProtectionRequirement[AssertionNumber].LogicalServiceContextDescription=_Log
icalServiceContextDescription;

CountermeasureProtectionRequirement[AssertionNumber].CountermeasureStrategy=_Countermeas
ureStrategy;

CountermeasureProtectionRequirement[AssertionNumber].CountermeasureLevel=_Countermeasure
Level;
    return AssertionNumber;
}
modifier Onlyowner(address _account){
    require(_account==owner);
    _;
}
struct LogicalServiceConsumer{
    address Gateway;
    mapping(uint=>QoP) GroupAssertion;
    uint size;
    string url;
    string functionalDescription; //equal to the businessArea
}
struct QoP{
    string LogicalServiceDescription;
    string CountermeasureStrategy;
    string CountermeasureLevel;
    string CountermeasureMethod;
}
LogicalServiceConsumer Service;
function updateLogicalService(string memory _url, address _ServiceDelegatee, string memory
LogicalServiceAPI)public Onlyowner(msg.sender){ //LogicalService toinvoke for container
    Service.Gateway=_ServiceDelegatee;
    Service.url=_url;
    Service.functionalDescription=LogicalServiceAPI;
}
mapping(uint=>bool) ComparisonConclusion;
function addAssertion(string memory _LogicalServiceContextDescription, string memory
_CountermeasureLevel, string memory _CountermeasureStrategy, string memory
_countermeasureMethod)public Onlyowner(msg.sender)returns(uint){

```

```

Service.size++;

Service.GroupAssertion[Service.size].LogicalServiceDescription=_LogicalServiceContextDescription;
Service.GroupAssertion[Service.size].CountermeasureStrategy=_CountermeasureStrategy;
Service.GroupAssertion[Service.size].CountermeasureLevel=_CountermeasureLevel;
Service.GroupAssertion[Service.size].CountermeasureMethod=_countermeasureMethod;
ComparationConclusion[Service.size]=false;
return Service.size;
}

function Comparation(uint ToSId, uint AssertionId)public Onlyowner(msg.sender){

require(keccak256(bytes(Service.GroupAssertion[ToSId].LogicalServiceDescription))==keccak256(b
ytes(CountermeasureProtectionRequirement[AssertionId].LogicalServiceContextDescription)));

require(keccak256(bytes(Service.GroupAssertion[ToSId].CountermeasureStrategy))==keccak256(byt
es(CountermeasureProtectionRequirement[AssertionId].CountermeasureStrategy)));

require(keccak256(bytes(Service.GroupAssertion[ToSId].CountermeasureLevel))==keccak256(bytes(
CountermeasureProtectionRequirement[AssertionId].CountermeasureLevel)));
    ComparationConclusion[ToSId]=true;
}
}

```

### 7.3 *Physical Smart Contract pattern*

```

contract PhysicalSmartContract{ //managed by the LogicalService to prove its concrete
service implementation
    address owner; // The Logical Service Account GateWay
    struct DataContainer{
        bytes32 ContainerHash; // the number of container
        bytes32 DataObjectHash; //link to the UsageSmartContract
    }
    struct Operation{
        string UsageName; //the physical Usage
        uint CopiesOfContainer; //the copies of number of generated container in this concrete
service
    }
    struct ConcreteServiceConsumer{
        string functionalDescription;
        address ConcreteService;
        mapping(uint=>QoP) GroupAssertion;
        uint size;
    }
    modifier Onlyowner(address _account){
        require(_account==owner);
    }
}

```



```

        mapping(uint=>PhysicalImplementationAssertion)
CountermeasureProtectionRequirement;
    struct QoP{
        string ConcreteServiceDescription;
        string CountermeasureStrategy;
        string CountermeasureLevel;
        string CountermeasureMethod;
    }
    struct PhysicalImplementationAssertion{
        string ConcreteServiceContextDescription;
        string CountermeasureStrategy;
        string CountermeasureLevel;
    }
    DataContainer dataContainer;
    Operation operation;
    uint AssertionNumber;
    constructor(bytes32 _ContainerHash, string memory Usage, bytes32
_DataObjectHash)public {
        owner=msg.sender;
        dataContainer.ContainerHash=_ContainerHash;
        dataContainer.DataObjectHash=_DataObjectHash;
        operation.UsageName=Usage;
    }
    function updatePolicy(string memory _ConcreteServiceContextDescription, string memory
_CountermeasureStrategy, string memory _CountermeasureLevel)public
Onlyowner(msg.sender)returns(uint) {
        AssertionNumber++;

CountermeasureProtectionRequirement[AssertionNumber].ConcreteServiceContextDescription=_C
oncreteServiceContextDescription;

CountermeasureProtectionRequirement[AssertionNumber].CountermeasureStrategy=_Countermeas
ureStrategy;

CountermeasureProtectionRequirement[AssertionNumber].CountermeasureLevel=_Countermeasure
Level;

        return AssertionNumber;
    }
    event PhysicalLifecycleEvent(bytes32 ContainerHash,string UsageName, uint Copies,
string functionalDescription, string PhysicalLifecycleState);
    function ConcreteServiceOperation(uint _Copies, string memory
PhysicalLifecycleState)public Onlyowner(msg.sender){
        operation.CopiesOfContainer=_Copies;
        emit
PhysicalLifecycleEvent(dataContainer.ContainerHash,operation.UsageName,operation.CopiesOfCo
ntainer,Service.functionalDescription,PhysicalLifecycleState);
    }
    function updateConcreteService(address _ServiceDelegatee, string memory
_ConcreteServiceDescription)public Onlyowner(msg.sender){ //LogicalService toinvoke for
container
        Service.ConcreteService=_ServiceDelegatee;
        Service.functionalDescription=_ConcreteServiceDescription;

```

```

    }
    ConcreteServiceConsumer Service;
    mapping(uint=>bool) ComparisonConclusion;
    function addAssertion(string memory _ConcreteServiceContextDescription, string memory
    _CountermeasureLevel, string memory _CountermeasureStrategy, string memory
    _countermeasureMethod)public Onlyowner(msg.sender)returns(uint) {
        Service.size++;

Service.GroupAssertion[Service.size].ConcreteServiceDescription=_ConcreteServiceContextDescript
ion;

Service.GroupAssertion[Service.size].CountermeasureStrategy=_CountermeasureStrategy;
    Service.GroupAssertion[Service.size].CountermeasureLevel=_CountermeasureLevel;
    Service.GroupAssertion[Service.size].CountermeasureMethd=_countermeasureMethod;
    ComparisonConclusion[Service.size]=false;
    return Service.size;
    }

    function Comparison(uint ToSId, uint AssertionId)public Onlyowner(msg.sender) {

require(keccak256(bytes(Service.GroupAssertion[ToSId].ConcreteServiceDescription))==keccak256(
bytes(CountermeasureProtectionRequirement[AssertionId].ConcreteServiceContextDescription)));

require(keccak256(bytes(Service.GroupAssertion[ToSId].CountermeasureStrategy))==keccak256(byt
es(CountermeasureProtectionRequirement[AssertionId].CountermeasureStrategy)));

require(keccak256(bytes(Service.GroupAssertion[ToSId].CountermeasureLevel))==keccak256(bytes(
CountermeasureProtectionRequirement[AssertionId].CountermeasureLevel)));
        ComparisonConclusion[ToSId]=true;
    }
}

```

#### 7.4 Tracking Smart Contract pattern

```

contract TrackingSmartContract { //managed by the container transfer from the Data
provider to the Logical Service: to prove
    address owner;
    bytes32 ContainerHash; //Description of the container
    constructor(bytes32 _ContainerHash)public { //Alice deploys the trackingSmatrContract
        owner=msg.sender;
        ContainerHash=_ContainerHash;
    }
    modifier Onlyowner(address _account) {
        require(_account==owner);
    }
    string Pk; //consumer's public key
    // for the Dataprovider to check the operation of the delegator
    function verifySignature(bytes32 hash, bytes memory signature) public view returns(bool)
    { //verify the transaction's signature
        bytes memory prefix = "\x19Ethereum Signed Message:\n32";

```

```

    bytes32 prefixedHash = keccak256(abi.encodePacked(prefix, hash)); //message
    if(recoverSigner(prefixedHash, signature) == owner){
        return true;
    }
}
function recoverSigner(bytes32 message, bytes memory sig)internal pure returns (address){
    (uint8 v, bytes32 r, bytes32 s) = splitSignature(sig);

    return ecrecover(message, v, r, s);
}
function splitSignature(bytes memory sig)internal pure returns (uint8 v, bytes32 r, bytes32
s){
    require(sig.length == 65);
    assembly {
        // first 32 bytes, after the length prefix.
        r := mload(add(sig, 32))
        // second 32 bytes.
        s := mload(add(sig, 64))
        // final byte (first byte of the next 32 bytes).
        v := byte(0, mload(add(sig, 96)))
    }
    return (v, r, s);
}

function ClaimsContainerTransfer(bytes32 hash, bytes memory RegistrationToken,string
memory _Pk)public returns(bytes32){ //LogicalService invoke to get the Container
    require(verifySignature(hash, RegistrationToken) ==true);
    Pk=_Pk;
    return ContainerHash;
}
function GetKey()public Onlyowner(msg.sender)returns(string memory){
    return Pk;
}
}
}

```

## 8 Annex: Prototype key functions

### 8.1 ToU generation process part

#### 8.1.1 ToS generation

```
public class TermsOfService { //
    private String UserName;
    private String BusinessArea;
    private String Role;
    private int id;
    private List<BusinessNode> businessNodes=new ArrayList<>();
    private BusinessNode Root;
    private List<LogicalNode> logicalNodes=new ArrayList<>();
    private List<ConcreteNode> concreteNodes=new ArrayList<>();
    BusinessServiceDao businessServiceDao=new BusinessServiceDao();
    ConcreteServiceDao concreteServiceDao=new ConcreteServiceDao();
    LogicalServiceDao logicalServiceDao=new LogicalServiceDao();
    LogicalAssetPatternDaoConsumer          logicalAssetPatternDao=new
LogicalAssetPatternDaoConsumer();
    QoPAssertionDaoNew qoPDao=new QoPAssertionDaoNew();
    private final Map<BusinessNode, List<BusinessAuthorizationQoP>> ToSpolicy=new
HashMap<>();
    private Map<Integer,LogicalNode> LogicalNodesList=new HashMap<>();
    private Map<Integer,ConcreteNode> ConcreteNodesList=new HashMap<>();
    // private String policyStrategy="positive Strategy";
    public TermsOfService(String userName, String businessArea, String role) throws
SQLException { //define for a business service
        businessNodes=new ArrayList<>();
        this.BusinessArea=businessArea;
        this.UserName=userName;
        this.Role=role;
        try {
            id =businessServiceDao.findbyName(UserName,Role,BusinessArea);
        } catch(SQLException throwables) {
            throwables.printStackTrace();
        }
        boolean atomicity=businessServiceDao.isAtomic(id); //determine whether the service is
an elementary task
        // System.out.println("start the root of the business service composition");
        Root=new BusinessNode(id,businessArea,userName,role,atomicity,null);
        // System.out.println("build the business service composition");
        businessNodes=Businessrecursion(Root, businessNodes); // this has established the
structure of business services
        // System.out.println("start to build the Terms of Service");
        DataToS(Root); //start the service from the root
        // build for the logical service
        for(int i=0; i<businessNodes.size();i++){
            LogicalService logicalService=null;
            BusinessNode node=businessNodes.get(i);
            //System.out.println("find the logical service");
```

```

        if(node.isIsleaf()==true){
            //          System.out.println("business service is elementary
task"+node.getId()+node.getBusinessArea());
            int LogicalId=businessServiceDao.getLogicalServiceId(node.getId());
            logicalService=logicalServiceDao.findbyId(LogicalId);
            //          System.out.println("LogicalService
is"+logicalService.getFunctionalDescription()+logicalService.getId());
            LogicalNode                                logicalNode=new
LogicalNode(logicalService.getId(),logicalService.getUserName(),logicalService.getRole(),logicalServic
e.getFunctionalDescription(),logicalService.getStatus(),node,logicalService.isAtomicity(),null);
            node.setLogicalTwin(logicalNode);
            LogicalNodesList.put(LogicalId,logicalNode);
            logicalNodes.add(logicalNode);
            businessNodes.set(i,node);
        }
    } //only set the businessServiceParent.
    //System.out.println("Start to the logical service composition");
    logicalNodes=LogicalRecursive(LogicalNodesList,logicalNodes);
    for (int i=0;i<logicalNodes.size();i++){
        LogicalNode node=logicalNodes.get(i);
        List<Integer> ConcreteIds=logicalServiceDao.getConceteServiceId(node.getId());
        //          System.out.println("find the concrete service of the logical
service"+node.getId()+node.getFunctionalDescription());
        for(Integer pid:ConcreteIds){
            ConcreteService concreteService=concreteServiceDao.findbyId(pid);
            ConcreteNode                                concreteNode=new
ConcreteNode(concreteService.getId(),concreteService.getGlobalDescription(),concreteService.isAto
micity(),node,null);
            //          System.out.println("LogicalService
is"+concreteService.getGlobalDescription()+concreteService.getId());
            concreteNodes.add(concreteNode);
            ConcreteNodesList.put(pid,concreteNode);
            node.setConcreteChildren(concreteNode);
        }
        logicalNodes.set(i,node);
    }
    // System.out.println("start the concrete service composition");
    concreteNodes=ConcreteRecursive(ConcreteNodesList,concreteNodes);
}
public List<BusinessNode> Businessrecursion(BusinessNode businessNode,
List<BusinessNode> businessNodes) throws SQLException {
    //          System.out.println("recusive to build the business service
composition"+businessNode.getId()+businessNode.getBusinessArea());
    businessNodes.add(businessNode);
    BusinessService businessService=null;
    if (businessNode.isIsleaf() == true) {
        //System.out.println("The          businessNode          is
Atomic"+businessNode.getBusinessArea()+businessNode.getId());
        return businessNodes;
    } else {
        //          System.out.println("The          businessNode          isNot
Atomic"+businessNode+getBusinessNodes()+businessNode.getId());

```

```

        List<Integer>                subBusinessServiceIds                =
businessServiceDao.findSubId(businessNode.getId());
    for (Integer bid : subBusinessServiceIds) {
        businessService = businessServiceDao.findbyId(bid);
        /* businessService.setId(businessService1.getId());
        businessService.setUserName(businessService1.getUserName());
        businessService.setAtomicity(businessService1.isAtomicity());
        businessService.setBusinessArea(businessService1.getBusinessArea());
        businessService.setRole(businessService1.getRole());*/
        BusinessNode childNode = new BusinessNode(businessService.getId(),
businessService.getBusinessArea(), businessService.getUserName(), businessService.getRole(),
businessService.isAtomicity(), businessNode);
        //                                System.out.println("find
SubBusinessService"+businessService.getId()+businessService.getBusinessArea());
        businessNode.setBusinesschildren(childNode);
        // System.out.println("Start for the decendent of the business service");
        Businessrecursion(childNode, businessNodes);
    }
}
return businessNodes;
}

    public void DataToS(BusinessNode businessNode) throws SQLException { //the
assertion of the root
        List<LogicalAssetPattern>
logicalAssetPatternList=businessServiceDao.getLogicalAssetPatternList(businessNode.getId()); //
determine all logical asset patternList of the businessService(JointSQL)
        businessNode.setMetaDataList(logicalAssetPatternList);
        //System.out.println("the                                business
service"+businessNode.getBusinessArea()+"consume"+                logicalAssetPatternList.size()+"logical
assets");
        for(LogicalAssetPattern lap: logicalAssetPatternList){
            if(ToSpolicy.containsKey(businessNode)==true) {
                // System.out.println("business service continue to add the businessAuthorization
policy");
                ToSpolicy.get(businessNode).addAll(ServiceRecursive(lap, businessNode));
            }else {
                // System.out.println("business service register to add the businessAuthorization
policy");
                List<BusinessAuthorizationQoP>
newAssertion=ServiceRecursive(lap,businessNode);
                ToSpolicy.put(businessNode,newAssertion);
                }//insert the QoP policy for the businessNode
            }
            if(businessNode.isIsleaf()==false){
                // System.out.println("business service is not the elementary
task"+businessNode.getBusinessArea()+businessNode.getId());
                //                                System.out.println("it
has"+businessNode.getBusinesschildren().size()+"subBusinessService");
                for(BusinessNode childnode:businessNode.getBusinesschildren()){
                    // System.out.println("Start to build the ToS of the subBusinessService");
                    DataToS(childnode);
                }
            }
        }
    }
}

```

```

    }
    }
}

public List<BusinessAuthorizationQoP> ServiceRecursive(LogicalAssetPattern
lap, BusinessNode businessNode) throws SQLException {
    BusinessService bs= businessServiceDao.findById(businessNode.getId());
    List<BusinessAuthorizationQoP> tosFinal = new ArrayList<>();
    // System.out.println("build the Tos of" + lap.getName() + "and" +
businessNode.getBusinessArea());
    boolean flag = false;
    if (qoPDao.isToSExist(lap, bs) == true) {
        // System.out.println("it already has the aggregated business Authorization
assertion");
        flag = true;
        tosFinal = (qoPDao.ToSfind(lap, bs));
    } else {
        //using the aggregation algorithm of ToS
        System.out.println("it doesn't has the business Authorization assertion");
        tosFinal=qoPDao.getBasicUsageAuthorization(lap,bs.getId());
        List<BusinessNode> child = businessNode.getBusinesschildren();
        for (BusinessNode sub : child) {
            // System.out.println("it has" + child.size() + "sub business service");
            BusinessService subservice = businessServiceDao.findById(sub.getId());
            //select the subBS consuming logicalAssetPattern
            if (qoPDao.whetherConsumeLAP(lap, subservice) == true) {
                // System.out.println("subBusinessService" + subservice.getBusinessArea() +
subservice.getId() + "consume metaData" + lap.getName());
                // System.out.println("find the business assertion of the subService");
                /*
                * For the internal business service, it will have the basic authorization for the
composed bs.
                * Such as business service(sub.getId()) will share the lap to its child bs.
                */

                List<BusinessAuthorizationQoP> assertions = ServiceRecursive(lap, sub);
                System.out.println("aggregate the subservice assertion to the business service");
                tosFinal = Aggregation(tosFinal, assertions, businessNode, lap);
                /**it need to aggregate the assertion with the existed assertion and insert to
the database.
                * 1. it will aggregate: process motivation, usagename, data related operation,
permission
                * 2. after the aggregation, it will change the logicalassetpattern and service
                * to generate a new businessauthorizationassertionQoP
                * and the new assertion will be inserted in the database.
                */
            }
        }
    }
    List<LogicalAssetPattern> subLogicalAssetList =
logicalAssetPatternDao.findSubLogicalAsset(lap);
    //System.out.println("logical asset pattern has" + subLogicalAssetList.size() + "sub
LogicalAsset");
}

```

```

        for (LogicalAssetPattern lp : subLogicalAssetList) {
            // System.out.println("find the BusinessAssertion of" + lp.getName() + "and" +
businessNode.getBusinessArea());
            List<BusinessAuthorizationQoP> assertions = ServiceRecursive(lp,
businessNode);
            System.out.println("aggregate the subLogicalAssetPattern assertion to the
business service");
            tosFinal = Aggregation(tosFinal, assertions, businessNode, lp);
        }
    }
    if(qoPDao.whetherConsumeLAP(lap, bs) == true) {
        if (flag == false) {
            for (int i = 0; i < tosFinal.size(); i++) {
                BusinessAuthorizationQoP assertion = tosFinal.get(i);
                if(qoPDao.isExistToSAssertion(assertion)!=true){
                    int assertionId = qoPDao.insertToSPolicy(assertion);
                    assertion.setId(assertionId);
                    tosFinal.set(i, assertion);
                }else{
                    tosFinal.remove(i);
                }
            }
        }
    }
    }
    return tosFinal;
}

public List<BusinessNode> getBusinessNodes() {
    return businessNodes;
}

public List<LogicalNode> getLogicalNodes() {
    return logicalNodes;
}

public List<ConcreteNode> getConcreteNodes() {
    return concreteNodes;
}

public BusinessNode getRoot(){
    return Root;
}

public List<BusinessAuthorizationQoP> Aggregation(List<BusinessAuthorizationQoP>
root,List<BusinessAuthorizationQoP> sub,BusinessNode businessNode,LogicalAssetPattern lp)
throws SQLException {
    System.out.println("start to aggregate the assertions by the UsageName and UsageStatus
and external QoP");
    for(int i=0;i<root.size();i++){
        BusinessAuthorizationQoP assertion1 = root.get(i);
        String UsageName1 = assertion1.getUsageName();
        for(int j=0;j<sub.size();j++){
            BusinessAuthorizationQoP assertion2 = sub.get(j);

```



```

        String UsageName2=assertion2.getUsageName();
        if(UsageName1.equalsIgnoreCase(UsageName2)) {
            String
usageStatusAggregated=Compare1(assertion1.getAuthorizedStatus(),assertion2.getAuthorizedStatus()
);
            BusinessAuthorizationQoP modifiedAssertion=new
BusinessAuthorizationQoP(assertion1.getId(),assertion1.getBid(),assertion1.getAssetDescription(),ass
ertion1.getBusinessServiceDescription(),assertion1.getProcessMotivation(),assertion1.getDataRelated
Operation(),assertion1.getUsageName(),assertion1.getDecision(),assertion1.getCountermeasureConte
xt(),usageStatusAggregated);
            root.set(i,modifiedAssertion);
        }
    }
}
boolean external=false;
for(int i=0;i<sub.size();i++) {
    BusinessAuthorizationQoP assertion1 = sub.get(i);
    String UsageName = assertion1.getUsageName();
    String CountermeasureContext=assertion1.getCountermeasureContext();
    if(CountermeasureContext.equalsIgnoreCase("unknown")!=true) {
        external=true;
    }
    boolean q = false;
    for (int j = 0; j < root.size(); j++) {
        BusinessAuthorizationQoP assertion2 = root.get(j);
        String UsageName2 = assertion2.getUsageName();
        if (UsageName.equalsIgnoreCase(UsageName2)) {
            q = true;
        }
    }
    if (q == false) { //the extra countermeasureContext that only owned by sub
policutext(),id,assertion2.getStatus());
        System.out.println("add the extra assertions");
        BusinessAuthorizationQoP newAssertion = new
BusinessAuthorizationQoP(root.size() + 1, businessNode.getId(), lp.getName(),
businessNode.getBusinessArea(), assertion1.getProcessMotivation(),
assertion1.getDataRelatedOperation(), assertion1.getUsageName(), assertion1.getDecision(),
assertion1.getCountermeasureContext(), assertion1.getAuthorizedStatus());
        root.add(newAssertion);
    }
}
if(external!=false) {
    root = ExtraAggregation(root, sub);
}
return root;
}
public List<BusinessAuthorizationQoP>
ExtraAggregation(List<BusinessAuthorizationQoP> root, List<BusinessAuthorizationQoP> sub){
    Map<String, String> qopLevel1=selfAggregate(root);
    Map<String, String> qopLevel2=selfAggregate(sub);
    Map<String, String> finalSecurity=new HashMap<>();
    List<BusinessAuthorizationQoP> finalRoot=new ArrayList<>();

```

```

for(String countermeasureContext: qopLevel1.keySet()){
    if(countermeasureContext.equalsIgnoreCase("unknown")){break;}
    else if(qopLevel2.containsKey(countermeasureContext)){
        String countermeasureLevel1=qopLevel1.get(countermeasureContext);
        String countermeasureLevel2=qopLevel2.get(countermeasureContext);
        String finalLevel=Compare(countermeasureLevel1,countermeasureLevel2);
        finalSecurity.put(countermeasureContext,finalLevel);
    }
}

for(String countermeasureContext:qopLevel2.keySet()){
    if(countermeasureContext.equalsIgnoreCase("unknown")){
        break;
    } else if(finalSecurity.containsKey(countermeasureContext)!=true){
        finalSecurity.put(countermeasureContext,qopLevel2.get(countermeasureContext));
    }
}
for(String countermeasureContext:qopLevel1.keySet()){
    if(countermeasureContext.equalsIgnoreCase("unknown")){
        break;
    } else if(finalSecurity.containsKey(countermeasureContext)!=true){
        finalSecurity.put(countermeasureContext,qopLevel1.get(countermeasureContext));
    }
}
for(BusinessAuthorizationQoP assertion:root){
    for(String counteremasureContext:finalSecurity.keySet()){

if(assertion.getCountermeasureContext().equalsIgnoreCase(counteremasureContext)){
        assertion.setStatusSet(finalSecurity.get(counteremasureContext));
        finalRoot.add(assertion);
    } else {
        BusinessAuthorizationQoP assertion1=new
BusinessAuthorizationQoP(assertion.getId(),assertion.getBid(),assertion.getAssetDescription(),asserti
on.getBusinessServiceDescription(),assertion.getProcessMotivation(),assertion.getDataRelatedOperat
ion(),assertion.getUsageName(),finalSecurity.get(counteremasureContext),counteremasureContext,ass
ertion.getAuthorizedStatus());
        finalRoot.add(assertion1);
    }
}
}
return finalRoot;
}
public Map<String, String> selfAggregate(List<BusinessAuthorizationQoP> policy){
    Map<String, String> countermeasure=new HashMap<>();
    for(BusinessAuthorizationQoP assertion: policy){
        String CountermeasureContext=assertion.getCountermeasureContext();
        String CountermeasureLevel=assertion.getDecision();
        countermeasure.put(CountermeasureContext,CountermeasureLevel);
    }
    return countermeasure;
}
public String Compare1(String v1, String v2){

```

```

        int value1=transform1(v1);
        int value2=transform1(v2);
        if(value1>value2){
            return v1;
        }else{
            return v2;
        }
    }
    public int transform1(String value){
        if(value.equalsIgnoreCase("public")){
            return 4;
        }else if(value.equalsIgnoreCase("shared")){
            return 3;
        }else if(value.equalsIgnoreCase("conditioned")){
            return 2;
        }else{
            return 1;
        }
    }
}
public String Compare(String v1, String v2){
    int value1=transform(v1);
    int value2=transform(v2);
    if(value1>value2){
        return v2;
    }else{
        return v1;
    }
}
public int transform(String value){
    if(value.equalsIgnoreCase("high")){
        return 2;
    }else if(value.equalsIgnoreCase("medium")){
        return 1;
    }else if(value.equalsIgnoreCase("low")){
        return 0;
    }else{
        return 4;
    }
}
}

//only consider the existed LogicalNodes
public
LogicalRecursive(Map<Integer,LogicalNode>nodeList,List<LogicalNode>
original) throws
SQLException {
    // System.out.println("logicalServiceRecursive");
    for(int i=0; i<original.size();i++){
        LogicalNode node1=original.get(i);
        List<Integer> subChildId=logicalServiceDao.getSubLogicalServiceId(node1.getId());
        //
        System.out.println("LogicalService"+node1.getFunctionalDescription()+node1.getId()+"has"+subC
hildId.size()+"subLogicalSERVICE");
        for(Integer id: subChildId){

```

```

        if(nodeList.containsKey(id)) {
            node1.setLogicalChildren(nodeList.get(id));
            nodeList.get(id).setLogicalParent(node1);
        }
    }
    original.set(i,node1);
}
return original;
}

public Map<BusinessNode, List<BusinessAuthorizationQoP>> getToSpolicy() {
    return ToSpolicy;
}

public List<ConcreteNode> ConcreteRecursive(Map<Integer,ConcreteNode> nodeList,
List<ConcreteNode> original) throws SQLException {
    // System.out.println("ConcreteServiceRecursive");
    for(int i=0; i<original.size();i++) {
        ConcreteNode node1=original.get(i);
        List<Integer>
subChildId=concreteServiceDao.getSubConcreteServiceId(node1.getId());
        //
System.out.println("ConcreteService"+node1.getId()+"has"+subChildId.size()+"subConcreteService
");
        for(Integer id: subChildId) {
            if(nodeList.containsKey(id)) {
                node1.setConcreteChildren(nodeList.get(id));
                nodeList.get(id).setPhysicalparent(node1);
            }
        }
        original.set(i,node1);
    }
    return original;
}
}
}

```

### 8.1.2 QoP evaluation

```

package control;

import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.*;

public class QoPAggregation2 {
    //aggregate the LogicalOperation and PhysicalImplementation, Evaluate QoP
    private String userName;
    private String role;
    private String businessArea;
    private TermsOfService tos;
}

```

```

QoPAssertionDaoNew qoPDao = new QoPAssertionDaoNew();
private Map<BusinessNode, List<QoPAssertionNew>> QoPpolicy=new HashMap<>();

public TermsOfService getTos() {
    return tos;
}

BusinessServiceDao businessServiceDao = new BusinessServiceDao();
LogicalAssetPatternDaoConsumer logicalAssetPatternDao=new
LogicalAssetPatternDaoConsumer();
private String policyStrategy="positive Strategy";
//descendents of the businessNode and its related Qop
public QoPAggregation2(String userName, String role, String businessArea) throws
SQLException { //determine the business service
    this.userName = userName;
    this.role = role;
    this.businessArea = businessArea;
    tos = new TermsOfService(userName, businessArea, role);
    QoPbusinessService(tos.getRoot());
}
public void QoPbusinessService(BusinessNode businessNode) throws SQLException { //this
will be the recursive part of the business service
    //1. determine whether it has QoP by the database
    //2. For the service, it will have the subService until the service is atomic
    //3. For each atomic service, it will have a logical service using QoPData
    // System.out.println("Start the QoP
generation"+businessNode.getId()+businessNode.getBusinessArea());
    List<LogicalAssetPattern>
logicalAssetPatternList=businessServiceDao.getLogicalAssetPatternList(businessNode.getId());
    // System.out.println("it has"+logicalAssetPatternList.size()+"LOGICALASSETPATTERN");
    for(LogicalAssetPattern lap:logicalAssetPatternList){
        if(QoPpolicy.containsKey(businessNode)==true) {
            QoPpolicy.get(businessNode).addAll(ServiceRecursive(lap, businessNode));
        } else {
            List<QoPAssertionNew> newAssertion=ServiceRecursive(lap,businessNode);
            QoPpolicy.put(businessNode,newAssertion);
        }
    }
    if(!businessNode.isIsleaf()){
        // System.out.println("Start the QoP of child service");
        for(BusinessNode childnode:businessNode.getBusinesschildren()){
            QoPbusinessService(childnode);
        }
    }
}

public List<QoPAssertionNew> ServiceRecursive(LogicalAssetPattern lap, BusinessNode
bsNode) throws SQLException {
    BusinessService bs= businessServiceDao.findById(bsNode.getId());
    List<QoPAssertionNew> policy = new ArrayList<>();
    // System.out.println("The qop generation of meta data="+lap.getName()+"and service
="+bs.getBusinessArea()+bs.getId());
    boolean flag = false;
    if (qoPDao.isQoPExist(lap, bs) == true) { //this is for the BusinessQoPAssertion
        // System.out.println("it has the QoP assertion");
        flag = true;
    }
}

```

```

        policy = qoPDao.QoPfind(lap, bs); //find the BusinessQoPAssertion
    } else {
        // System.out.println("it doesn't have QoP assertion");
        if (bsNode.isIsleaf()) {
            // System.out.println("service is an elementary task");
            LogicalNode logicalNode = bsNode.getLogicalTwin();
            // System.out.println("it will generate the QoP from logical service and concrete
service");
            List<QoPAssertionNew> QoP = QoPData(logicalNode, lap, bsNode); //QoP policy
of logical asset pattern and business service
            policy = QoP;
        } else {
            List<BusinessNode> child = bsNode.getBusinesschildren();
            for (BusinessNode sub : child) {
                //select the subBS consuming logicalAssetPattern
                BusinessService subservice = businessServiceDao.findById(sub.getId());
                if (qoPDao.whetherConsumeLAP(lap, subservice) == true) {
                    // System.out.println("aggregate the QoP of sub service="+
sub.getBusinessArea()+"and metaData="+lap.getName());
                    policy = AggregationQoP(policy, ServiceRecursive(lap, sub), bsNode, lap);
                }
            }
            List<LogicalAssetPattern> subLogicalAssetList =
logicalAssetPatternDao.findSubLogicalAsset(lap);
            for (LogicalAssetPattern lp : subLogicalAssetList) {
                // System.out.println("aggregate the QoP of sub
LogicalAssetPattern="+lp.getName()+"and service="+bs.getBusinessArea());
                policy = AggregationQoP(policy, ServiceRecursive(lp, bsNode), bsNode, lap);
            }
        }
    }
    if (qoPDao.whetherConsumeLAP(lap, bs) == true) {
        if (flag == false) {
            for (int i = 0; i < policy.size(); i++) {
                QoPAssertionNew assertion = policy.get(i);
                int assertionId = qoPDao.InsertQoPAssertion(assertion);
                assertion.setId(assertionId);
                policy.set(i, assertion);
            }
        }
    }
    return policy;
}
public Map<BusinessNode,List<QoPAssertionNew>> getQoPpolicy(){
    return QoPpolicy;
}
public List<QoPAssertionNew> QoPData(LogicalNode logicalNode, LogicalAssetPattern lap,
BusinessNode bs) throws SQLException { //this will be the dataObject to the Logical service and concrete
service
    List<QoPAssertionNew> policy=new ArrayList<>();
    // One logicalNode may consume multiple dataObject because One businessNode may
consume multiple logicalAssetPattern
    List<LogicalOperationQoP>
logicalOperationQoPList=qoPDao.findLogicalOperationAssertion(logicalNode.getId(),lap);
    for(int i=0;i<logicalOperationQoPList.size();i++){
        String FunctionalDescription=logicalOperationQoPList.get(i).getFunctionalDescription();

```

```

        String
CountermeasureContext=logicalOperationQoPList.get(i).getCountermeasureContext();
        //String
CountermeasureStrategy=logicalOperationQoPList.get(i).getCountermeasureStrategy();
        //String UsageName=logicalOperationQoPList.get(i).getUsageName();
        String CountermeasureLevel=logicalOperationQoPList.get(i).getStatus();
        QoPAssertionNew newAssertion=new QoPAssertionNew(-
1,lap.getName(),bs.getBusinessArea(),CountermeasureContext,CountermeasureLevel,bs.getId());
        // int id=qoPDao.InsertQoPAssertion(newAssertion);
        // newAssertion.setId(id);
        policy.add(newAssertion);
    }
    for(ConcreteNode child:logicalNode.getConcreteChildren()){
        List<PhysicalImplementationQoP>
physicalImplementationQoPList=qoPDao.findPhysicalImplementationAssertion(child.getId(),lap);
        for(int i=0;i<physicalImplementationQoPList.size();i++){
            String
FunctionalDescription=physicalImplementationQoPList.get(i).getFunctionalDescription();
            String
CountermeasureContext=physicalImplementationQoPList.get(i).getCountermeasureContext();
            //String
CountermeasureStrategy=physicalImplementationQoPList.get(i).getCountermeasureStrategy();
            // String UsageName=physicalImplementationQoPList.get(i).getUsageName();
            String CountermeasureLevel=physicalImplementationQoPList.get(i).getStatus();
            QoPAssertionNew newAssertion=new QoPAssertionNew(-
1,lap.getName(),bs.getBusinessArea(),CountermeasureContext,CountermeasureLevel,bs.getId());
            // int id=qoPDao.InsertQoPAssertion(newAssertion);
            // newAssertion.setId(id);
            policy.add(newAssertion);
        }
    }
    policy=selfAggregate(policy);
    return policy;
}
public List<QoPAssertionNew>AggregationQoP(List<QoPAssertionNew>
root,List<QoPAssertionNew> sub, BusinessNode bs, LogicalAssetPattern lp) throws SQLException {
    // System.out.println("aggregate the qop with sub qop by CountermeasureContext");
    root=selfAggregate(root);
    sub=selfAggregate(sub);
    for (int i = 0; i < root.size(); i++) {
        QoPAssertionNew assertion1 = root.get(i);
        String CountermeasureContext1=assertion1.getCountermeasureContext();
        for (int j = 0; j < sub.size(); j++) {
            QoPAssertionNew assertion2 = sub.get(j);
            String CountermeasureContext2 = assertion2.getCountermeasureContext();
            if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){
                // System.out.println("compare the countermeasureLevel in the positive strategy");
                String countermeasureLevel = Compare(assertion2.getStatus(), assertion1.getStatus());
                QoPAssertionNew newAssertion=new
QoPAssertionNew(assertion1.getId(),lp.getName(),bs.getBusinessArea(),assertion1.getCountermeasureContext(
),countermeasureLevel,bs.getId());
                root.set(i,newAssertion);//modify the assertion1
            }
        }
    }
}
for(int i=0;i<sub.size();i++){
    QoPAssertionNew assertion1 = sub.get(i);

```

```

String CountermeasureContext1 = assertion1.getCountermeasureContext();
boolean q=false;
for(int j = 0; j < root.size(); j++){
    QoPAssertionNew assertion2 = root.get(j);
    String CountermeasureContext2=assertion2.getCountermeasureContext();
    if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){
        q=true;
    }
}
if(q==false){ //the extra countermeasureContext that only owned by sub policu
    System.out.println("add the extra QoP");
    QoPAssertionNew newAssertion=new
QoPAssertionNew(root.size()+1,lp.getName(),bs.getBusinessArea(),assertion1.getCountermeasureContext(),ass
ertion1.getStatus(),bs.getId());
    root.add(newAssertion);
}
}
root=selfAggregate(root);
return root;
}
public List<QoPAssertionNew> selfAggregate(List<QoPAssertionNew> root){
    // System.out.println("aggregate the assertios to a uniifed countermeasureContext");
    List<QoPAssertionNew> finalROOT=new ArrayList<>();
    Map<String, QoPAssertionNew> flag = new HashMap<>();
    for (int i = 0; i < root.size(); i++) {
        QoPAssertionNew assertion1 = root.get(i);
        String CountermeasureContext = assertion1.getCountermeasureContext();
        if (flag.containsKey(CountermeasureContext)) {
            String usageStatusAggregated=Compare(flag.get(CountermeasureContext).getStatus(),
assertion1.getStatus());
            flag.get(CountermeasureContext).statusSet(usageStatusAggregated);
        } else {
            flag.put(CountermeasureContext, assertion1);
        }
    }
    for (String key: flag.keySet()){
        finalROOT.add(flag.get(key));
    }
    return finalROOT;
}
public String Compare(String v1, String v2){
    int value1=transform(v1);
    int value2=transform(v2);
    if(value1>value2){
        return v2;
    }else{
        return v1;
    }
}
public int transform(String value){
    if (value.equalsIgnoreCase("NOTYET")){
        return 3;
    }
    else if(value.equalsIgnoreCase("high")){
        return 2;
    }
    else if(value.equalsIgnoreCase("medium")){
        return 1;
    }
}

```



```

    }else{
        return 0;
    }
}

/** It will have three steps: QoPservice=QoPbusinessService
 * 1. determine whether it has the QoP(businessService).
 * 2. The QoP service includes all the logical asset patterns that composes.
 * (It means that it will create all QoPservices and insert to the database one by one)
 *
 * **/
}

```

### 8.1.3 RoP generation

```

package control;

import dao.LogicalAssetDao;
import dao.LogicalAssetPatternDao;
import dao.RoPAssertionDao;
import pojo.LogicalAsset;
import pojo.LogicalAssetPattern;
import pojo.RoPAssertionNew2;
import pojo.UsageManagementPolicy;

import java.sql.SQLException;
import java.util.*;

public class RoPAggregation2 {
    private String policyStrategy = "negative aggregation";
    RoPAssertionDao rOp = new RoPAssertionDao();
    LogicalAssetDao ldo = new LogicalAssetDao();
    LogicalAssetPatternDao ldpo=new LogicalAssetPatternDao();
    private Map<LogicalAsset, List<RoPAssertionNew2>> Globalpolicy=new
HashMap<>();
    public RoPAggregation2(){}
    public RoPAggregation2(TosQoPNew UMP, String UserName) throws SQLException {
        Map<BusinessNode, List<UsageManagementPolicy>> DraftToU =
UMP.getFinalToS();
        Set<BusinessNode> BusinessNodesList = DraftToU.keySet();
        for (BusinessNode node : BusinessNodesList) {
            List<LogicalAssetPattern> logicalAssetPatternList = node.getMetaDataList();
            for (LogicalAssetPattern lap : logicalAssetPatternList) {
                String metaData = lap.getName();
                LogicalAsset logicalAsset = ldo.findLogicalAsset(metaData, UserName);
                if(logicalAsset !=null)
                { Globalpolicy.put(logicalAsset, Find(logicalAsset));}
            }
        }
    }

    public List<RoPAssertionNew2> Find(LogicalAsset lap) throws SQLException {

```

```

        System.out.println("Find whether the logical asset has the final
RoP"+lap.getId()+lap.getMetaData());
        if (rOp.isRoPExist(lap) == true) {
            System.out.println("check it has the final RoP");
            return rOp.RoPLogicalAssetfind(lap); //the aggregated RoP(final RoP policy)
        } else {
            System.out.println("No, Create its Logical final RoP");
            return Create(lap);
        }
    }
}

/* The final RoP policy will be considered in 3 parts:
1. initial LogicalAssetPattern's RoP
2. Extra LogicalAsset's RoP when the data provider is not the original data owner
3. the subLogicalAsset's RoP
In case, subLA1 is low, subLA2 is low but LA3 composing LA1 and LA2 is high protection
subLA1 is high, subLA2 is low but LA3 will be high.
Both of them will be considered in the aggregation part
**/
public List<RoPAssertionNew2> Create(LogicalAsset La) throws SQLException {
    System.out.println("Create RoP of logical Asset"+La.getMetaData()+La.getMetaData());
    List<RoPAssertionNew2> policy=new ArrayList<>();
    if (ldo.isAtomic(La) == true) {
        System.out.println("Logical asset is Atomic"+ La.getMetaData()+La.getId());
        String metaData = La.getMetaData();
        LogicalAssetPattern logicalAssetPattern = ldpo.getPattern(metaData);
        List<RoPAssertionNew2> Initialpolicy =
rOp.RoPLogicalAssetPatternfind(logicalAssetPattern);
        System.out.println("It has"+ Initialpolicy.size()+"basic protection assertions");
        List<RoPAssertionNew2> ExtraPolicy = rOp.RoPLogicalAssetfind(La);
        System.out.println("It has"+ ExtraPolicy.size()+"extra protection assertions");
        System.out.println("Aggregate both to generate RoP assertion");
        policy.addAll(Aggregation(La,ExtraPolicy, Initialpolicy));
    } else {
        System.out.println("Logical asset is not Atomic"+ La.getMetaData()+La.getId());
        List<LogicalAsset> subLogicalAssetList = ldo.findSubLogicalAssetbyId(La.getId());
        String metaData = La.getMetaData();
        LogicalAssetPattern logicalAssetPattern = ldpo.getPattern(metaData);
        List<RoPAssertionNew2> Initialpolicy =
rOp.RoPLogicalAssetPatternfind(logicalAssetPattern);
        System.out.println("It has"+ Initialpolicy.size()+"basic protection assertions");
        List<RoPAssertionNew2> ExtraPolicy = rOp.RoPLogicalAssetfind(La);
        System.out.println("It has"+ ExtraPolicy.size()+"extra protection assertions");
        System.out.println("Aggregate both to generate RoP assertion");
        policy=Aggregation(La,ExtraPolicy,Initialpolicy);
        for (LogicalAsset sub : subLogicalAssetList) {
            System.out.println("recursive to aggregate the subLogicalAsset's
assertion"+sub.getMetaData()+sub.getId()+"to the composed logical
asset"+La.getMetaData()+La.getId());
            policy=Aggregation(La,policy, Find(sub));
        }
    }
}

```

```

        for(int i=0;i<policy.size();i++){
            RoPAssertionNew2 assertion=policy.get(i);
            int assertionId=rOp.insertFinalPolicy(assertion);
            assertion.setId(assertionId);
            policy.set(i,assertion);

System.out.println("metaData"+assertion.getMetaData()+"requires"+assertion.getCountermeasureC
ontext()+"has"+assertion.getStatus());
        }
        return policy;

    }

    public List<RoPAssertionNew2> Aggregation(LogicalAsset
asset,List<RoPAssertionNew2> root, List<RoPAssertionNew2> sub) {
        System.out.println("aggregate the RoP");
        int id=asset.getId();
        for (int i = 0; i < root.size(); i++) {
            RoPAssertionNew2 assertion1 = root.get(i);
            String CountermeasureContext1=assertion1.getCountermeasureContext();
            for (int j = 0; j < sub.size(); j++) {
                RoPAssertionNew2 assertion2 = sub.get(j);
                String CountermeasureContext2 = assertion2.getCountermeasureContext();
                if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){
                    // System.out.println(assertion1.getMetaData());
                    // System.out.println(CountermeasureContext1);
                    // System.out.println(assertion1.getStatus());
                    // System.out.println(assertion2.getMetaData());
                    // System.out.println(assertion2.getStatus());
                    System.out.println("Compare the countermeasureLevel in the negative strategy");
                    String countermeasureLevel = Compare(assertion2.getStatus(),
assertion1.getStatus());
                    System.out.println(countermeasureLevel);
                    RoPAssertionNew2 newAssertion=new
RoPAssertionNew2(assertion1.getId(),asset.getMetaData(),assertion1.getCountermeasureContext(),id
,countermeasureLevel);
                    root.set(i,newAssertion);//modify the assertion1
                }
            }
        }
        for(int i=0;i<sub.size();i++){
            RoPAssertionNew2 assertion2 = sub.get(i);
            String CountermeasureContext2 = assertion2.getCountermeasureContext();
            boolean q=false;
            for(int j = 0; j < root.size(); j++){
                RoPAssertionNew2 assertion1 = root.get(j);
                String CountermeasureContext1=assertion1.getCountermeasureContext();
                if(CountermeasureContext1.equalsIgnoreCase(CountermeasureContext2)){
                    q=true;
                }
            }
        }
        if(q==false){ //the extra countermeasureContext that only owned by sub policu

```

```

        System.out.println("It has extra protection policy");
        RoPAssertionNew2 newAssertion=new
RoPAssertionNew2(root.size()+1,asset.getMetaData(),assertion2.getCountermeasureContext(),id,asse
rtion2.getStatus());
        root.add(newAssertion);
    }
}
return root;
}
public String Compare(String v1, String v2) {
    int value1=transform(v1);
    int value2=transform(v2);
    if(value1>value2){
        return v1;
    }else{
        return v2;
    }
}
public int transform(String value) {
    if(value.equalsIgnoreCase("high")){
        return 2;
    }else if(value.equalsIgnoreCase("medium")){
        return 1;
    }else{
        return 0;
    }
}

public Map<LogicalAsset, List<RoPAssertionNew2>> getGlobalpolicy() {
    return Globalpolicy;
}
}

```

#### 8.1.4 ToU evaluation

```

package control;

import dao.QoPAssertionDaoNew;
import pojo.*;

import java.sql.SQLException;
import java.util.*;

public class TosQoPNew {
    private String userName;
    private String BusinessArea;
    private String Role;
    private Map<BusinessNode, List<UsageManagementPolicy>> FinalToS=new
HashMap<>();
    private QoPAssertionDaoNew qop1=new QoPAssertionDaoNew();
    private List<BusinessAuthorizationQoP> tospolicy = new ArrayList<>();
    private List<QoPAssertionNew> qopPolicy=new ArrayList<>();
}

```

```

    public TosQoPNew(String userName, String businessArea, String role) throws
SQLException {
    this.userName = userName;
    BusinessArea = businessArea;
    Role = role;
    aggregation(userName, BusinessArea, Role);
}

```

```

    public void aggregation(String userName, String businessArea, String role) throws
SQLException {
    System.out.println("get the QoP of Internal policy");
    QoPAggregation2 qop = new QoPAggregation2(userName, role, businessArea);
    System.out.println("get the ToS and QoP OF extra policy");
    TermsOfService tos =qop.getTos();
    List<BusinessNode> businessNodeList = tos.getBusinessNodes();
    for (BusinessNode node : businessNodeList) {
        tospolicy=tos.getToSpolicy().get(node);
        qopPolicy= qop.getQoPpolicy().get(node);
        if(qopPolicy.size(>0&&qopPolicy!=null) {
            FinalToS.put(node, ExtraAggregation(tospolicy, qopPolicy));
        }else{
            List<UsageManagementPolicy> PP=new ArrayList<>();
            for(int i=0;i<tospolicy.size();i++){ //the external ToS
                BusinessAuthorizationQoP assertion1=tospolicy.get(i);
                UsageManagementPolicy assertion=new UsageManagementPolicy(-
1,assertion1.getBid(),assertion1.getAssetDescription(),assertion1.getBusinessServiceDescription(),asse
rtion1.getProcessMotivation(),assertion1.getDataRelatedOperation(),assertion1.getUsageName(),asser
tion1.getCountermeasureContext(),assertion1.getDecision());
                if(qop1.isExistToUAssertion(assertion)!=true) {
                    int id = qop1.InsertUMP(assertion);
                    assertion.setId(id);
                    PP.add(assertion);
                }
            }
            FinalToS.put(node,PP);
        }
    }
}

```

```

    public List<UsageManagementPolicy>
ExtraAggregation(List<BusinessAuthorizationQoP> root, List<QoPAssertionNew> sub) throws
SQLException {
    System.out.println("Aggregate for each service");
    Map<String, String> qopLevel1=selfAggregate(root);
    Map<String, String> qopLevel2=selfAggregateQop(sub);
    Map<String, String> finalSecurity=new HashMap<>();
    List<UsageManagementPolicy> finalRoot=new ArrayList<>();
    for(String countermeasureContext: qopLevel1.keySet()){
        if(countermeasureContext.equalsIgnoreCase("unknown")){break;}
        else if(qopLevel2.containsKey(countermeasureContext)) {
            String countermeasureLevel1=qopLevel1.get(countermeasureContext);
            String countermeasureLevel2=qopLevel2.get(countermeasureContext);

```

```

        String finalLevel=Compare(countermeasureLevel1,countermeasureLevel2);
        finalSecurity.put(countermeasureContext,finalLevel);
    }
}

for(String countermeasureContext:qopLevel2.keySet()){
    if(countermeasureContext.equalsIgnoreCase("unknown")){
        break;
    }else if(finalSecurity.containsKey(countermeasureContext)!=true){
        finalSecurity.put(countermeasureContext,qopLevel2.get(countermeasureContext));
    }
}
for(String countermeasureContext:qopLevel1.keySet()){
    if(countermeasureContext.equalsIgnoreCase("unknown")){
        break;
    }else if(finalSecurity.containsKey(countermeasureContext)!=true){
        finalSecurity.put(countermeasureContext,qopLevel1.get(countermeasureContext));
    }
}
//然后针对 root 的每一条, 如果他有这个 CountermeasureContex, 那么就改变
level, 否则就是将其添加新的 CountermeasureContext 和 CountermeasureLevel
for(BusinessAuthorizationQoP assertion:root){
    for(String counteremasureContext:finalSecurity.keySet()){

if(assertion.getCountermeasureContext().equalsIgnoreCase(counteremasureContext)){
        assertion.setStatusSet(finalSecurity.get(counteremasureContext));
        UsageManagementPolicy assertionfinal = new UsageManagementPolicy(-1,
assertion.getBid(), assertion.getAssetDescription(), assertion.getBusinessServiceDescription(),
assertion.getProcessMotivation(), assertion.getDataRelatedOperation(), assertion.getUsageName(),
assertion.getCountermeasureContext(), assertion.getDecision());
        finalRoot.add(assertionfinal);
    }else{
        UsageManagementPolicy assertionfinal = new UsageManagementPolicy(-1,
assertion.getBid(), assertion.getAssetDescription(), assertion.getBusinessServiceDescription(),
assertion.getProcessMotivation(), assertion.getDataRelatedOperation(), assertion.getUsageName(),
counteremasureContext, finalSecurity.get(counteremasureContext));
        finalRoot.add(assertionfinal);
    }
}
}
for(int i=0;i<finalRoot.size();i++){
    UsageManagementPolicy assertion=finalRoot.get(i);
    if(qop1.isExistToUAssertion(assertion)!=true){
        int id=qop1.InsertUMP(assertion);
        assertion.setId(id);
        finalRoot.set(i,assertion);}else{
        finalRoot.remove(i);
    }
}
}
return finalRoot;

```

```

}
public Map<String, String> selfAggregate(List<BusinessAuthorizationQoP> policy) {
    Map<String, String> countermeasure=new HashMap<>();
    for(BusinessAuthorizationQoP assertion: policy) {
        String CountermeasureContext=assertion.getCountermeasureContext();
        String CountermeasureLevel=assertion.getDecision();
        countermeasure.put(CountermeasureContext,CountermeasureLevel);
    }
    return countermeasure;
}
public Map<String, String> selfAggregateQop(List<QoPAssertionNew> policy) {
    Map<String, String> countermeasure=new HashMap<>();
    for(QoPAssertionNew assertion: policy) {
        String CountermeasureContext=assertion.getCountermeasureContext();
        String CountermeasureLevel=assertion.getStatus();
        countermeasure.put(CountermeasureContext,CountermeasureLevel);
    }
    return countermeasure;
}
public String Compare(String v1, String v2) {
    int value1=transform(v1);
    int value2=transform(v2);
    if(value1>value2) {
        return v2;
    } else {
        return v1;
    }
}
public int transform(String value) {
    if(value.equalsIgnoreCase("notyet")) {
        return 3;
    } else if(value.equalsIgnoreCase("high")) {
        return 2;
    } else if(value.equalsIgnoreCase("medium")) {
        return 1;
    } else {
        return 0;
    }
}

public Map<BusinessNode, List<UsageManagementPolicy>> getFinalToS() {
    return FinalToS;
}
}

```

## 8.2 Transaction generation part

### 8.2.1 Business Transaction

```
package Transaction;
```

```

import control.BusinessNode;
import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.*;

public class BusinessTransaction {
    private int id;
    // private BusinessTransaction parent;
    private int bid;
    private BusinessTransaction parent;
    private BusinessService businessService; //BSi
    private Map<String,ExchangedAsset> assets=new HashMap<>();
    private List<BusinessService> subBusinessService=new ArrayList<>();
    private List<LogicalOperation> LO=new ArrayList<>();
    Map<ExchangedAsset,String> AssetExpiration=new HashMap<>(); //the shared
exchanged asset already asset expiration in this business service
    Map<ExchangedAsset,List<LogicalAssetPattern>> subAssets=new HashMap<>();

    Map<ExchangedAsset,List<BusinessAuthorizationQoP>>ownedAuthorizationPolicy=new
HashMap<>();

    Map<ExchangedAsset,List<BusinessAuthorizationQoP>>ownedAuthenticationPolicy=new
HashMap<>();
    //each ExchangedAsset's ToU assertion includes the its subAssets and
subBusinessService's ToU assertion.
    List<UsageManagementPolicy> ownCertifiedPolicy=new ArrayList<>();

    public List<UsageManagementPolicy> getOwnCertifiedPolicy() {
        return ownCertifiedPolicy;
    }

    public BusinessTransaction getParent() {
        return parent;
    }

    public void setOwnCertifiedPolicy(List<UsageManagementPolicy> ownCertifiedPolicy) {
        this.ownCertifiedPolicy = ownCertifiedPolicy;
    }
    private LogicalAssetPatternDaoConsumer lapo=new LogicalAssetPatternDaoConsumer();

    private OrganizationalEntity consumer;
    private OrganizationalEntity provider;
    OrganizationalEntityDao dao=new OrganizationalEntityDao();
    BusinessTransactionDao bto=new BusinessTransactionDao();
    public void setLogicalAssetPatternList(ExchangedAsset asset) throws SQLException {
        LogicalAssetPattern metaData=asset.getMetaData();
        asset.setSubmetaDatList(sresusiveFind(metaData));
    }
    public List<LogicalAssetPattern> sresusiveFind(LogicalAssetPattern metaData) throws
SQLException {

```



```

        List<LogicalAssetPattern> ls=new ArrayList<>();
        if(lapo.isAtomicity(metaData.getName())!=true){
            List<LogicalAssetPattern>
subLogicalAssetPatternList=lapo.findSubLogicalAsset(metaData);
            for(LogicalAssetPattern each: subLogicalAssetPatternList){
                ls.addAll(sresusiveFind(each));
            }
        }
        return ls;
    }
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public boolean hasExchangedAsset(LogicalAssetPattern lap) {
        if(assets.containsKey(lap.getName())){
            return true;
        }else{
            return false;
        }
    }

    public void setAssets(String metaData,ExchangedAsset asset) {
        this.assets.put(metaData,asset);
    }

    public LogicalAsset getLogicalAsset(LogicalAssetPattern Lap) throws SQLException {
        LogicalAsset logicalAsset=null;
        if(hasExchangedAsset(Lap)==true){ //The parent and child consume the same Lap or
the exchanged asset has been established by a LO
            logicalAsset=getExchangedAsset(Lap).getAsset();
        }else{ //the parent will search the LogicalAsset
            for (String metaData: assets.keySet()){
                ExchangedAsset asset=assets.get(metaData);
                if(lapo.isAtomicity(asset.getMetaData().getName())!=true){ //the exchanged asset
is not atomic
                    if(recursiveCheck(asset.getAsset(),Lap)==null){
                        continue;
                    }else{
                        return recursiveCheck(asset.getAsset(),Lap);
                    }
                }
            }
        }
        return logicalAsset;
    }
    private LogicalAssetConsumerDao lado=new LogicalAssetConsumerDao();

```

```

        public LogicalAsset recursiveCheck(LogicalAsset asset, LogicalAssetPattern ll) throws
SQLException {
    LogicalAsset finalAsset=null;
    if(asset.getMetaData().equalsIgnoreCase(ll.getName())) {
        return asset;
    }else{
        if(!lpo.isAtomicity(asset.getMetaData())==true){
            return null;
        }else{
            List<LogicalAsset> subAssetList=lado.findSubLogicalAssetbyId(asset.getId());
            for(LogicalAsset subAsset:subAssetList){
                if(recursiveCheck(subAsset,ll)==null){
                    continue;
                }else{
                    return recursiveCheck(subAsset,ll);
                }
            }
        }
    }
    return finalAsset;
}
public ExchangedAsset getExchangedAsset(LogicalAssetPattern Lap){
    return assets.get(Lap.getName());
}
}

```

```

public BusinessServiceDao bsdo=new BusinessServiceDao();

```

```

        public BusinessTransaction(int id, int bid, BusinessTransaction parent) throws
SQLException {
    this.bid=bid;
    this.id=id;
    this.parent=parent;
    businessService=bsdo.findById(bid);
    String role=businessService.getRole();
    String username=businessService.getUserName();
    System.out.println("Set the business transaction's provider and consumer");
    consumer=dao.getUser(role,username);
    if(parent==null){ //initialize the business transaction
        provider=dao.getUser("customer","Alice");
    }else{
        provider=parent.getConsumer();
    }
    List<Integer> subList=bsdo.findSubId(bid);
    for(int SUBid: subList) {
        BusinessService subservice = bsdo.findById(SUBid);
        subBusinessService.add(subservice);
    }
}
public void addLogicalOperation(LogicalOperation lo){
    LO.add(lo);
}
}

```

```

public OrganizationalEntity getProvider() {
    return provider;
}
private Map<String,Set<String>> BusinessUsageComposition=new HashMap<>();

public Map<String, Set<String>> getBusinessUsageComposition() {
    return BusinessUsageComposition;
}

public void setBusinessUsageComposition(Map<String, Set<String>>
businessUsageComposition) {
    BusinessUsageComposition = businessUsageComposition;
}

public OrganizationalEntity getConsumer() {
    return consumer;
}
public void setOwnedToUPolicy(ExchangedAsset asset,List<BusinessAuthorizationQoP>
tou){
    if(ownedAuthorizationPolicy.containsKey(asset)!=true){
        ownedAuthorizationPolicy.put(asset,tou);
    }else {
        ownedAuthorizationPolicy.get(asset).addAll(tou);
    }
}

public BusinessService getBusinessService() {
    return businessService;
}

public List<BusinessService> getSubBusinessService() {
    return subBusinessService;
}

public void setOwnedAuthenticationPolicy(ExchangedAsset asset,
List<BusinessAuthorizationQoP> aa) {
    if(ownedAuthenticationPolicy.containsKey(asset)!=true){
        ownedAuthenticationPolicy.put(asset,aa);
    }else {
        ownedAuthenticationPolicy.get(asset).addAll(aa);
    }
}
}
}

```

## 8.2.2 Business Transaction generation

```

package Transaction;

import Transaction.BusinessTransaction;
import Transaction.LogicalOperation;
import control.BusinessNode;

```

```

import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.*;

public class GenerateBusinessTransaction {
    private BusinessServiceDao bsdao=new BusinessServiceDao();
    private BusinessTransaction parent;
    private BusinessService businessService;
    private Map<BusinessNode,List<UsageManagementPolicy>> TouPolicy=new
HashMap<>();
    private List<BusinessAuthorizationQoP> inputToSpolicy=new ArrayList<>();
    private LogicalAssetPatternDaoConsumer lapdao=new
LogicalAssetPatternDaoConsumer();
    /* 1. The most important thing for the input policy and output policy for a
businessTransaction(bs)
    1. the input policy is related to the Tos of this bs.
    It means that OLS (Usage) in the bs with the input asset.
    input asset, bs, Usage, DataRelatedOperation, ProcessMotivation
    2. the output policy is related to the LogicalOperations of this bs which associate to the
subBs's tos.
    It means that OLS share/delegate (Usage) with the output asset for subBs.
    output asset(exchanged asset), BusinessUsage, DataRelatedOperation,
ProcessMotivation, Usage.
    => It means that the businessAuthorizationToken:
    DataRelatedOperation,ProcessMotivation, exchangedAsset(from Alice) for the subBS.
    Select the LAP,DataRelatedOperation,ProcessMotivation from subBS's Tos
    => It means that the AuthenticationProtection:
    BusinessUsage, exchangedAsset in the bs.
    => The output asset compose input asset. It means that input asset can be
contactInformation.
    But the exchanged asset can have two. One is the address and the other is the
contactInformation for different subBusinessservices.
    */
    OrganizationalEntityDao odo=new OrganizationalEntityDao();
    BusinessTransactionDao bto=new BusinessTransactionDao();
    public GenerateBusinessTransaction(BusinessTransaction parent, BusinessService
businessService,List<BusinessAuthorizationQoP>
tosPolicy,Map<BusinessNode,List<UsageManagementPolicy>> tou) throws SQLException {
        this.parent = parent;
        this.businessService = businessService;
        this.TouPolicy=tou;
        inputToSpolicy=tosPolicy;
        System.out.println("Create BusinessTransaction of
businessService"+businessService.getBusinessArea()+businessService.getId());
        BusinessTransaction bs1=Create(businessService,tosPolicy,parent,TouPolicy);
        System.out.println("Insert the created BusinessTransaction into the Database");
        int id=bto.InsertBusinessTransaction(bs1);
        bs1.setId(id);
        System.out.println("Register the logicalAsset to the exchanged asset");
        System.out.println("Refine the BusinessTransaction");
    }
}

```

```

        if(businessService.isAtomicity()!=true) {
            if(businessService.getBusinessArea()!="deliveryProcessing") {
                RefinementBusinessTransaction refined=new
RefinementBusinessTransaction(bs1,tou);}
            }
            System.out.println("UsageDerivation");
            //DerivationUsageTransaction usageTransaction=new
DerivationUsageTransaction(bs1);
        }
        public BusinessTransaction Create(BusinessService bs,List<BusinessAuthorizationQoP>
tos,BusinessTransaction parent,Map<BusinessNode,List<UsageManagementPolicy>> tou) throws
SQLException {
            System.out.println("Select all the Logical Operations used by businessService");
            Map<LogicalOperation,String> LO=bto.selectLogicalOperation(bs);
            System.out.println("initialize the business transaction from the business service");
            BusinessTransaction bs1=new BusinessTransaction(-1,bs.getId(),parent);
            System.out.println("For each LogicalOperation to update the BusinessTransaction");
            Map<String,Set<String>> Usage=new HashMap<>();
            bs1.setBusinessUsageComposition(Usage);
            for(LogicalOperation lo:LO.keySet()){
                String AssetExpiration=LO.get(lo);
                bs1=Find(tos,lo,bs1,parent,AssetExpiration,tou);
            }
            return bs1;
        }
        public BusinessTransaction Find(List<BusinessAuthorizationQoP> tos,LogicalOperation
lo,BusinessTransaction bt, BusinessTransaction parent,String
AssetExpiration,Map<BusinessNode,List<UsageManagementPolicy>> tou) throws SQLException {
            Map<String,Set<String>> Usage=bt.getBusinessUsageComposition();
            System.out.println("find the LogicalAssetPattern");
            LogicalAssetPattern Lap=lo.getLAP(); // this is the input asset of the
subBusinessServices
            // System.out.println(Lap.getName());
            System.out.println("find the DataRelatedOperatin and ProcessMotivation and
BusinessUsage");
            String DataRelatedOperation=lo.getDataRelatedOperation();
            String ProcessMotivation=lo.getProcessMotivation();
            String UsageName=lo.getUsageName(); //this is the business usage
            System.out.println("Find the BusinessAuthorizationAssertion for sub BusinessService: it
will be DRO,PM AND UsageName");
            List<BusinessAuthorizationQoP>
authorizedAssertion=FindBusinessAuthorizationAssertion(Lap,DataRelatedOperation,ProcessMotiv
ation,bt);
            System.out.println("create the exchanged asset of this business transaction ");
            System.out.println("determine whether the exchangedAsset is existed");
            ExchangedAsset usedAsset=null;
            if(bt.hasExchangedAsset(Lap)!=true){
                usedAsset=new ExchangedAsset(Lap);
                LogicalAsset asset=null;
                if(parent==null){
                    System.out.println("the business transaction is the initial business transaction,
register the Logical asset");

```

```

        asset=new LogicalAsset();
        asset.setMetaData(Lap.getName());
        asset.setRole("Customer");
        asset.setUserName("Alice");
        OrganizationalEntity entity=odo.getUser("Customer","Alice");
        usedAsset.setEntity(entity);
    }else{
        System.out.println("from parent transaction's exchanged asset to get the Logical
asset");
        asset=parent.getLogicalAsset(Lap);
        OrganizationalEntity entity=parent.getConsumer();
        usedAsset.setEntity(entity);
    }
    System.out.println("Associate LogicalAsset to the exchanged asset");
    usedAsset.setUserId(bt.getProvider().getId());
    usedAsset.setAsset(asset);
    System.out.println("Store the Exchanged asset in the business transaction");
    bt.setAssets(Lap.getName(),usedAsset);
    System.out.println("Store the metaDataList of Exchangedasset in the business
transaction");
    bt.setLogicalAssetPatternList(usedAsset);
} else{
    usedAsset=bt.getExchangedAsset(Lap);
}
usedAsset.setAssetExpiration(AssetExpiration);
System.out.println("Associate the ExchangedAsset to the DataRelatedOperation and
Terms of Usage assertion");
bt.setOwnedToUPolicy(usedAsset,authorizedAssertion);
System.out.println("Find the AuthenticationPolicyAssertion for this BusinessService: it
will be BusinessUsage,BS");
// LogicalAssetPattern a=new LogicalAssetPattern("productRecord");
BusinessService service=bt.getBusinessService();
List<LogicalAssetPattern>
inputDataList=bsdao.getLogicalAssetPatternList(service.getId());
LogicalAssetPattern tosPattern=FindToSPattern(Lap,inputDataList);
if(Usage.containsKey(tosPattern.getName())) {
    Set<String> usageNameList=Usage.get(tosPattern.getName());
    if (usageNameList.add(UsageName)) { //UsageName
        List<BusinessAuthorizationQoP> authenticationAssertion =
FindAuthenticationProtectionAssertion(Lap, UsageName, tos, bt);
        bt.setOwnedAuthenticationPolicy(usedAsset, authenticationAssertion);
        Usage.put(tosPattern.getName(), usageNameList);
        bt.setBusinessUsageComposition(Usage);
    }
} else{
    Set<String> usageNameList=new HashSet<>();
    usageNameList.add(UsageName);
    Usage.put(tosPattern.getName(), usageNameList);
    List<BusinessAuthorizationQoP> authenticationAssertion =
FindAuthenticationProtectionAssertion(Lap, UsageName, tos, bt);
    bt.setOwnedAuthenticationPolicy(usedAsset, authenticationAssertion);
    bt.setBusinessUsageComposition(Usage);
}

```

```

    }
    System.out.println("Find the approved ToU assertion to certify
BusinessAuthorizationQoP and AuthenticationPolicyAssertion");
    if(bt.getOwnCertifiedPolicy().size()<=0) {
        List<UsageManagementPolicy> approvedToken = FindApprovedToU(bt,tou);
        bt.setOwnCertifiedPolicy(approvedToken);
    }
    return bt;
}
public List<UsageManagementPolicy>FindApprovedToU(BusinessTransaction
bt,Map<BusinessNode,List<UsageManagementPolicy>> tou){
    List<UsageManagementPolicy> outputtou=getToUpolicy(bt,tou);
    while(outputtou.size()==0| (outputtou==null)){
        bt=bt.getParent();
        if(bt!=null) {
            outputtou = getToUpolicy(bt,tou);
        }else{
            for(BusinessNode node:tou.keySet()){
                if(node.getBusinessArea().equalsIgnoreCase("ProductDelivery")){
                    return tou.get(node);
                }
            }
        }
    }
    return outputtou;
}
public List<UsageManagementPolicy> getToUpolicy(BusinessTransaction
bt,Map<BusinessNode,List<UsageManagementPolicy>> tou){
    List<UsageManagementPolicy> outputtou=new ArrayList<>();
    for(BusinessNode node:tou.keySet()){
        if(node.getBusinessArea().equalsIgnoreCase(bt.getBusinessService().getBusinessArea())){
            return tou.get(node);
        }
    }
    return outputtou;
}
public List<BusinessAuthorizationQoP>
FindBusinessAuthorizationAssertion(LogicalAssetPattern Lap,String DRO,String
PM,BusinessTransaction bt) throws SQLException {
    BusinessService service=bt.getBusinessService(); //find the business service id
    if(service.isAtomicity()==true){
        return null;
    }
    System.out.println("Find the businessAuthorizationAssertion of this businessService to
the subBusinessService");
    //the business authorization aims to all the sub business service will share the
dataRelatedOperation
    List<Integer> subIdList=bsdao.findSubId(service.getId());
    List<BusinessService> subBusinessServiceList=new ArrayList<>();
    for(Integer ids: subIdList){
        subBusinessServiceList.add(bsdao.findbyId(ids));
    }
}

```

```

    }
    System.out.println("Find the ToS of each subBusinessService");
    List<BusinessAuthorizationQoP> outputtos=new ArrayList<>();
    for(BusinessService child: subBusinessServiceList) {
        List<BusinessAuthorizationQoP>
childTos=bto.getDedicatedToS(child,Lap,DRO,PM); //from the PM,DRO,LP and BS to get the tos
of the BS
        outputtos.addAll(childTos);
    }
    return outputtos;
}
public List<BusinessAuthorizationQoP>
FindAuthenticationProtectionAssertion(LogicalAssetPattern Lap, String
UsageName,List<BusinessAuthorizationQoP> tos,BusinessTransaction bt) throws SQLException {
    BusinessService service=bt.getBusinessService();
    if(service.isAtomicity()==true) {
        return null;
    }
    List<BusinessAuthorizationQoP> listToS=new ArrayList<>();
    List<LogicalAssetPattern>
inputDataList=bsdao.getLogicalAssetPatternList(service.getId()); //find the input asset which is
related to the tos
    LogicalAssetPattern tosPattern=FindToSPattern(Lap,inputDataList); //Alice allow
OLS share contactIF(tos) in Service1. OLS can share address(Lap) from Service1 to Service2
    for(BusinessAuthorizationQoP assertion: tos) {

if(assertion.getAssetDescription().equalsIgnoreCase(tosPattern.getName())&&assertion.getUsageName()
.equalsIgnoreCase(UsageName)) {
        listToS.add(assertion);
    }
    }
    return listToS;
}
public LogicalAssetPattern FindToSPattern(LogicalAssetPattern
Lap,List<LogicalAssetPattern>metaDataList) throws SQLException {
    System.out.println("From a list of metaData to select the most similar metaData with
Lap");
    LogicalAssetPattern initial=null;
    for(LogicalAssetPattern inputData:metaDataList) {
        if(inputData.getName().equalsIgnoreCase(Lap.getName())) {
            return inputData;
        }else if(lapdao.isAtomicity(inputData.getName())!=true) { //the LogicalAssetPattern
is not atomic
            if(recursiveCheck(inputData,Lap)==null) {
                continue;
            }else {
                return inputData;
            }
        }
    }
    return initial;
}
}

```



```

    public LogicalAssetPattern recursiveCheck(LogicalAssetPattern asset, LogicalAssetPattern
ll) throws SQLException {
        LogicalAssetPattern finalAsset=null;
        if(asset.getName().equalsIgnoreCase(ll.getName())) {
            return asset;
        }else{
            if(lapdao.isAtomicity(asset.getName())==true){
                return null;
            }else{
                List<LogicalAssetPattern> subAssetList=lapdao.findSubLogicalAsset(asset);
                for(LogicalAssetPattern subAsset:subAssetList){
                    if(recursiveCheck(subAsset,ll)==null){
                        continue;
                    }else{
                        return recursiveCheck(subAsset,ll);
                    }
                }
            }
        }
        return finalAsset;
    }
}

```

### 8.2.3 Usage Transaction

```

package Transaction;

import Transaction.BusinessTransaction;
import Transaction.UsageOperation;
import Transaction.UsageTransaction;
import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class GenerateUsageTransaction {
    private BusinessTransactionDao bto=new BusinessTransactionDao();
    private LogicalServiceDao lsdo=new LogicalServiceDao();
    private BusinessServiceDao bsdo=new BusinessServiceDao();
    private List<UsageTransaction> transaction=new ArrayList<>();
    private OrganizationalEntityDao odo=new OrganizationalEntityDao();
    private QoPAssertionDaoNew qopdao=new QoPAssertionDaoNew();
    public GenerateUsageTransaction(BusinessTransaction bt) throws SQLException {
        System.out.println("determine whether the business service is elementary task");
        if(bt.getSubBusinessService()!=null){
            System.out.println("Select the Logical Service of the business service");
            BusinessService service= bt.getBusinessService();

```

```

        int lsId=bsdo.getLogicalServiceId(service.getId());
        LogicalService ls=lsdo.findById(lsId);
        String ProcessControlPurpose=bto.getBusinessServiceToLogicalService(service);
        System.out.println("Select all Usage Operation used by Logical Service");
        Map<UsageOperation,String> UOS=bto.SelectUsageOperations(ls);
        //System.out.println("Get the LogicalService's QoP policy assertion");
        //List<LogicalOperationQoP>
qop=qopdao.findLogicalOperationAssertion(service.getId());
        // System.out.println("For each UsageOperation to create its Usage Transaction");
        for(UsageOperation uo:UOS.keySet()) {
            String UsageDuration=UOS.get(uo);
            UsageTransaction ut=Create(bt,uo,UsageDuration,ls);
            System.out.println("set the business transaction");
            ut.setParent(bt);
            System.out.println("set the LogicalService");
            ut.setService(ls);
            ut.setProcessControlPurpose(ProcessControlPurpose);
            System.out.println("set party of this usage transaction");
            OrganizationalEntity
consumer=odo.getUser(service.getRole(),service.getUserName());
            ut.setConsumer(consumer);
            ut.setProvider(bt.getConsumer());
            ut.setProcessControlPurpose(ProcessControlPurpose);
            transaction.add(ut);
        }
        System.out.println("Store the UsageTransaction into the DataBase");
        for(UsageTransaction ut:transaction) {
            int id=bto.InsertUsageTransaction(ut);
            ut.setId(id);
            System.out.println("Start the physical transaction generation");
            GeneratePhysicalTransaction p=new GeneratePhysicalTransaction(ut);
        }
    }
}
}
}
}

public UsageTransaction Create(BusinessTransaction bt,UsageOperation uo,String
UsageDuration,LogicalService ls) throws SQLException {
    UsageTransaction ut=new UsageTransaction();
    ut.setUsageDuration(UsageDuration);
    System.out.println("Get the DataObject from the uo");
    DataObject ob=uo.getDO();
    System.out.println("set the DataObject into the Ut");
    ut.setObject(ob);
    System.out.println("Get the LogicalAssetPattern from the DataObject");
    LogicalAssetPattern lap=bto.getLogicalAssetPattern(ob);
    System.out.println("Select the Exchanged Asset descrbing using LAP and involved in
BT");
    ExchangedAsset asset= bt.getExchangedAsset(lap);
    LogicalAsset realAsset=asset.getAsset();
    ob.setRealasset(realAsset);
    System.out.println("Get the DataRelatedOperation and UsageName");
    String DRO=uo.getDataRelatedOperation();
    ut.setDataRelatedOperation(DRO);

```

```

        String UsageName=uo.getUsageName();
        ut.setUsageName(UsageName);
        System.out.println("Get the ToU assertion a authorizing DRO for LAP from ToU Policy
P");
        ut.setProofToken(bt.getOwnCertifiedPolicy());
        System.out.println("set the QoPLogicalService of LAP in the LogicalService");
        List<LogicalOperationQoP>
QOP=qopdao.findLogicalOperationAssertion(ls.getId(),lap);
        ut.setAssertion(QOP);

        return ut;
    }
}

```

## 8.2.4 Usage Transaction generation

```

package Transaction;

import Transaction.BusinessTransaction;
import Transaction.UsageOperation;
import Transaction.UsageTransaction;
import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class GenerateUsageTransaction {
    private BusinessTransactionDao bto=new BusinessTransactionDao();
    private LogicalServiceDao lsdo=new LogicalServiceDao();
    private BusinessServiceDao bsdo=new BusinessServiceDao();
    private List<UsageTransaction> transaction=new ArrayList<>();
    private OrganizationalEntityDao odo=new OrganizationalEntityDao();
    private QoPAssertionDaoNew qopdao=new QoPAssertionDaoNew();
    public GenerateUsageTransaction(BusinessTransaction bt) throws SQLException {
        System.out.println("determine whether the business service is elementary task");
        if(bt.getSubBusinessService()!=null) {
            System.out.println("Select the Logical Service of the business service");
            BusinessService service= bt.getBusinessService();
            int lsId=bsdo.getLogicalServiceId(service.getId());
            LogicalService ls=lsdo.findbyId(lsId);
            String ProcessControlPurpose=bto.getBusinessServiceToLogicalService(service);
            System.out.println("Select all Usage Operation used by Logical Service");
            Map<UsageOperation,String> UOS=bto.SelectUsageOperations(ls);
            //System.out.println("Get the LogicalService's QoP policy assertion");
            //List<LogicalOperationQoP>
qop=qopdao.findLogicalOperationAssertion(service.getId(),);

```

```

// System.out.println("For each UsageOperation to create its Usage Transaction");
for(UsageOperation uo:UOS.keySet()){
    String UsageDuration=UOS.get(uo);
    UsageTransaction ut=Create(bt,uo,UsageDuration,ls);
    System.out.println("set the business transaction");
    ut.setParent(bt);
    System.out.println("set the LogicalService");
    ut.setService(ls);
    ut.setProcessControlPurpose(ProcessControlPurpose);
    System.out.println("set party of this usage transaction");
    OrganizationalEntity
consumer=odo.getUser(service.getRole(),service.getUserName());
    ut.setConsumer(consumer);
    ut.setProvider(bt.getConsumer());
    ut.setProcessControlPurpose(ProcessControlPurpose);
    transaction.add(ut);
}
System.out.println("Store the UsageTransaction into the DataBase");
for(UsageTransaction ut:transaction){
    int id=bto.InsertUsageTransaction(ut);
    ut.setId(id);
    System.out.println("Start the physical transaction generation");
    GeneratePhysicalTransaction p=new GeneratePhysicalTransaction(ut);
}
}
}
private LogicalAssetPatternDaoConsumer lado=new LogicalAssetPatternDaoConsumer();
public UsageTransaction Create(BusinessTransaction bt,UsageOperation uo,String
UsageDuration,LogicalService ls) throws SQLException {
    UsageTransaction ut=new UsageTransaction();
    ut.setUsageDuration(UsageDuration);
    System.out.println("Get the DataObject from the uo");
    DataObject ob=uo.getDO();
    String metaData=ob.getMetaData();
    LogicalAssetPattern logicalAssetPattern=lado.getPattern(metaData);
    ExchangedAsset givenAsset=bt.getExchangedAsset(logicalAssetPattern);
    ob.setAuthorizedasset(givenAsset);
    System.out.println("set the DataObject into the Ut");
    ut.setObject(ob);
    System.out.println("Get the LogicalAssetPattern from the DataObject");
    LogicalAssetPattern lap=bto.getLogicaAssetPattern(ob);
    System.out.println("Select the Exchanged Asset descrbing using LAP and involved in
BT");
    ExchangedAsset asset= bt.getExchangedAsset(lap);
    LogicalAsset realAsset=asset.getAsset();
    ob.setRealasset(realAsset);
    System.out.println("Get the DataRelatedOperation and UsageName");
    String DRO=uo.getDataRelatedOperation();
    ut.setDataRelatedOperation(DRO);
    String UsageName=uo.getUsageName();
    ut.setUsageName(UsageName);

```

```

        System.out.println("Get the ToU assertion a authorizing DRO for LAP from ToU Policy
P");
        ut.setProofToken(bt.getOwnCertifiedPolicy());
        System.out.println("set the QoPLogicalService of LAP in the LogicalService");
        List<LogicalOperationQoP>
QOP=qopdao.findLogicalOperationAssertion(ls.getId(),lap);
        ut.setAssertion(QOP);

        return ut;
    }
}

```

### 8.2.5 *Physical Transaction*

```

package Transaction;

import pojo.*;

import java.util.List;

public class PhysicalTransaction {
    private int id;
    private Container container;
    private UsageTransaction parent;
    private int containerCopies;
    private String UsageName;
    private DataObject dataObject;
    List<PhysicalImplementationQoP> qop;
    private OrganizationalEntity consumer;
    private OrganizationalEntity provider;

    public ConcreteService getService() {
        return service;
    }

    public void setId(int id) {
        this.id = id;
    }

    public OrganizationalEntity getConsumer() {
        return consumer;
    }

    public void setConsumer(OrganizationalEntity consumer) {
        this.consumer = consumer;
    }

    public OrganizationalEntity getProvider() {
        return provider;
    }
}

```

```

public void setProvider(OrganizationalEntity provider) {
    this.provider = provider;
}

public void setService(ConcreteService service) {
    this.service = service;
}

private ConcreteService service;

public Container getContainer() {
    return container;
}

public void setContainer(Container container) {
    this.container = container;
}

public UsageTransaction getParent() {
    return parent;
}

public void setParent(UsageTransaction parent) {
    this.parent = parent;
}

public int getContainerCopies() {
    return containerCopies;
}

public void setContainerCopies(int containerCopies) {
    this.containerCopies = containerCopies;
}

public String getUsageName() {
    return UsageName;
}

public void setUsageName(String usageName) {
    UsageName = usageName;
}

public DataObject getDataObject() {
    return dataObject;
}

public void setDataObject(DataObject logicalAsset) {
    this.dataObject = logicalAsset;
}

public List<PhysicalImplementationQoP> getQop() {
    return qop;
}

```

```

    }

    public void setQop(List<PhysicalImplementationQoP> qop) {
        this.qop = qop;
    }
}

```

### 8.2.6 *Physical Transaction generation*

```

package Transaction;

import Transaction.PhysicalOperation;
import Transaction.PhysicalTransaction;
import Transaction.UsageTransaction;
import dao.*;
import pojo.*;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class GeneratePhysicalTransaction {
    private BusinessTransactionDao bto=new BusinessTransactionDao();
    private OrganizationalEntityDao odo=new OrganizationalEntityDao();
    private LogicalServiceDao lsdo=new LogicalServiceDao();
    private ConcreteServiceDao csdo=new ConcreteServiceDao();
    private QoPAssertionDaoNew qopdao=new QoPAssertionDaoNew();
    public GeneratePhysicalTransaction(UsageTransaction UT) throws SQLException {
        List<PhysicalTransaction> physicalList=new ArrayList<>();
        System.out.println("Get the LogicalService involved in the usageTransaction");
        LogicalService service=UT.getService();
        System.out.println("Get all concrete service implementing LS");
        List<Integer> csListid=lsdo.getConceteServiceId(service.getId());
        List<ConcreteService> csList=new ArrayList<>();
        for(int csid:csListid){
            csList.add(csdo.findbyId(csid));
        }
        System.out.println("For each concrete service to generate physical transaction");
        for(ConcreteService cs: csList){
            PhysicalTransaction pt=Create(cs,UT);
            physicalList.add(pt);
        }
        System.out.println("Insert the physicalTransactions into the database");
        for(PhysicalTransaction ps: physicalList){
            int id= bto.InsertPhysicalTransaction(ps);
            ps.setId(id);
        }
    }
    private LogicalAssetPatternDaoConsumer lapdo=new
LogicalAssetPatternDaoConsumer();

```

```

    public PhysicalTransaction Create(ConcreteService cs, UsageTransaction ut) throws
SQLException {
    PhysicalTransaction pt=new PhysicalTransaction();
    System.out.println("Set the party of physicalTransaction");
    pt.setConsumer(odo.getUser(cs.getRole(),cs.getUserName()));
    pt.setProvider(ut.getConsumer());
    System.out.println("associate to the usage transaction and concrete service");
    pt.setParent(ut);
    pt.setService(cs);
    System.out.println("select the physical operation from concrete service");
    PhysicalOperation op=bto.selectPhysicalOperation(cs);
    int ContainerCopies=bto.selectContainerCopies(cs);
    pt.setContainerCopies(ContainerCopies);
    System.out.println("Get the dataObject from the physical operation");
    DataObject dataObject=op.getOb();
    System.out.println("Get the physical Usage from this PhysicalOperation");
    String UsageName=op.getPhysicalUsageName();
    pt.setUsageName(UsageName);
    System.out.println("Get the LogicalAsset from ut");
    LogicalAsset ls=ut.getAsset();
    System.out.println("Get the Qop assertion from logical asset");
    // String metaData=ls.getMetaData();
    List<PhysicalImplementationQoP>
qop=qopdao.findPhysicalImplementationAssertion(cs.getId(),lapdo.getPattern(ls.getMetaData()));
    System.out.println("Set the ToU assertion ");
    pt.setQop(qop);
    System.out.println("Select the container by data object and logical asset");
    /*LogicalService receive a format type of LogicalAsset but the concrete services may
transform and
    * used different format type of LogicalAsset such as pdf->text*/
    Container container=bto.SelectContainer(dataObject,ls);
    if(container==null) {
        container=new Container(dataObject,ls);
    }
    pt.setContainer(container);
    return pt;
}
}

```

### 8.3 Smart Contract generation

```

public class SmartContractGeneration {
    OrganizationalEntityDao userFind=new OrganizationalEntityDao();
    public SmartContractGeneration(BusinessTransaction bt) throws SQLException {
        boolean Original=false;
        System.out.println("determine whether it is the original transaction");
        if (bt.getParent()==null) {
            System.out.println("find the Original business transaction");
            Original=true;
        }
    }
}

```



```

    }else{
        String delegatorAccount=bt.getDelegator();
        Original=false;
    }
    System.out.println("get the business service associated to the business transaction");
    BusinessService thisService=bt.getBusinessService();
    System.out.println("get the sub businessService which will generate future business
transaction");
    List<BusinessService> request=bt.getSubBusinessService();
    System.out.println("Get the business purpose of this business service");
    String businessPurpose=thisService.getBusinessArea();
    System.out.println("the exchangedAsset provider such as Alice");
    Party DataProvider=new Party(bt.getProvider());
    System.out.println("Get thisServiceProvider who will be the delegator");
    Party Delegator=new Party(bt.getConsumer());
    System.out.println("Get thisService's next delegates");
    HashSet<Party> delegateeList=new HashSet<>();
    for(BusinessService subService:request) {
        Party
                                                                    delegatee=new
Party(userFind.getUser(subService.getRole(),subService.getUserName()));
        delegateeList.add(delegatee);
    }
    System.out.println("Select all the exchangedAsset in this BusinessService");
    List<ExchangedAsset> ExchangedAssetList=bt.getFinalExchangedAsset();
    System.out.println("For each ExchangedAsset to start the delegation");
    for(ExchangedAsset givenAsset:ExchangedAssetList) {
        System.out.println("manage authorization of the Delegator's authorization to the
delegatee for the sub businessService ");
        CreateToken(givenAsset,bt,Original,delegateeList,DataProvider);
    }
}
BusinessTransactionDao bto=new BusinessTransactionDao();
public void CreateToken(ExchangedAsset asset, BusinessTransaction bt,boolean
Original,HashSet<Party> delegateeList,Party dataProvider) throws SQLException {
    List<BusinessAuthorizationQoP>
UsageAuthorizationTokens=bt.getOwnedAuthorizedPolicy(asset);
    List<UsageManagementPolicy> tou=bt.getOwnCertifiedPolicy();
    if(Original==false) {
        System.out.println("Invoke the AskforDelegation function to get the token by the
prepared parameters");
        JSONObject DelegatorSC= new JSONObject(dataProvider);
        List<BusinessAuthorizationQoP> ToS=bto.getToS(bt.getBusinessService());
        List<BusinessAuthorizationQoP> token=UsageAuthorizationTokens;
        JSONObject Certificator=new JSONObject();
        Certificator=DelegatorSC;
    }else {
        List<UsageManagementPolicy> Consent=tou;
        List<BusinessAuthorizationQoP> ToS=bto.getToS(bt.getBusinessService());
        JSONObject Certificator=new JSONObject(dataProvider);
    }
    System.out.println("create the ExchangedSC");
    System.out.println("Select the usage Transactions");

```

```

List<UsageTransaction> UT=new ArrayList<>();
if(bt.getBusinessService().isAtomicity()==true){
    UT=bto.SelectUsageTransaction(bt.getId());
}
for(UsageTransaction ut:UT){
    System.out.println("get the parameters used for the UsageSmartContract");
    DataObject object=ut.getObject();
    LogicalService service=ut.getService();
    Party
                                serviceDelegate=new
Party(userFind.getUser(service.getRole(),service.getUserName()));
    List<LogicalOperationQoP> usageOperationtokenQoP=ut.getAssertion();
    JSONObject usageauthorizationtoken=new JSONObject();
    usageauthorizationtoken.setProcessMotivation(ut.getProcessControlPurpose());
    usageauthorizationtoken.setDataRelatedOperation(ut.getDataRelatedOperation());
    usageauthorizationtoken.setUsageDuration(ut.getUsageDuration());

usageauthorizationtoken.setProof(ut.getParent().getOwnedAuthorizedPolicy(object.getAuthorizedasset()));
    System.out.println("generate usage operation token and create usage smart contract");
    PhysicalSmartContractGeneration psc=new PhysicalSmartContractGeneration(ut);
}
}
}

```



## FOLIO ADMINISTRATIF

### THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : YUAN  
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 8 Juillet 2021

Prénoms : Jingya

TITRE : Security and Privacy as a Service for Social, Mobile, Analytics and Cloud Information Technologies: Data driven and usage based protection architecture

NATURE : Doctorat

Numéro d'ordre : 2021YSEI044

Ecole doctorale : InfoMaths

Spécialité : Informatique

RESUME :

Protecting Information Systems (IS) relies traditionally on security risk analysis methods. Designed for well-perimetrised environments, these methods rely on a systematic identification of threats and vulnerabilities to identify efficient control-centered protection countermeasures. Unfortunately, this does not fit security challenges carried out by the opened and agile organizations provided by the Social, Mobile, big data Analytics, Cloud and Internet of Things (SMACIT) environment. Due to their inherently collaborative and distributed organization, such multi-tenancy systems require the integration of contextual vulnerabilities, depending on the a priori unknown way of using, storing and exchanging data in opened cloud environment. Moreover, as data can be associated to multiple copies, different protection requirements can be set for each of these copies, which may lead the initial data owner lose control on the data protection. To overcome these limits, we propose a Data centered Usage based Protection model relying on an IS description model to set a consistent protection for data assets. Protection means are defined according to both organizational and technical risks. To this end, we propose a GDPR compliant security and extended usage ontology which is used to define usage-control assertions coupling usage rights to security countermeasures so that data assets can be efficiently protected according to both organizational and technical dimensions. Thanks to a Blockchain-based usage control, our Data centered and Usage based Protection architecture also allows tracking the way assets are used so their life-long protection can be checked.

MOTS-CLÉS : Sécurité, Protection des données personnelles, RGPD, ontologie, Blockchain, Modèle de droits basé sur les usages, SMACIT

Laboratoire (s) de recherche : LIRIS – UMR 5205

Directeur de thèse: Pr. Frédérique Biennier

Président de jury :

Composition du jury : Pr. Marco WINCKLER  
Pr Michael MARISSA  
Dr Khalid. BENALI  
Dr Genoveva VARGAS-SOLAR  
Pr. Frédérique BIENNIER  
Dr Nabila BENHARKAT



**INSA**