Deliverable D3.2

Privacy enhancing techniques in the Smart City applications

Editor:	Henrich C. Pöhls, Ralf C. Staudemeyer, UNI PASSAU
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31. August 2015
Actual delivery date:	2. September 2015
Suggested readers:	Privacy researchers, Policy decision makers, IERC, IoT application developers, IoT system administrators, security experts
Version:	1.0
Total number of pages:	292
Keywords:	RERUM, Internet of Things, smart cities, applications, use-cases, smart transportation, home energy management, environmental monitoring, comfort quality, privacy, location privacy, redactable signatures, anonymity, pseudonymity

Abstract

Privacy must be a major concern in any IoT related project, ever more so in Europe which values data privacy protection and thus privacy-by-design becomes a legal obligation. Legal issues aside, as RERUM is focussed on the technology, achieving a high level of privacy is of paramount importance in the smart cities domain to initially win —and keep— citizens' active participation. D2.3 included components for a privacy-aware architecture of the RERUM platform and D3.2 gives the details on the design of the privacy components. To enable privacy D3.2 builds on top of RERUM's baseline security concepts described in D3.1, especially confidentiality protection and authentication capabilities are required to build privacy. Privacy-by-design has to cover the whole IoT, its a truly cross-cutting topic: not only does it need to be considered on all architectural layers, i.e. its vertical to the ISO/OSI layers, it is also crossing the towards the socio-technical domain. D3.2 focusses on the Privacay Enhancing Technologies (PETs): A *Consent Manager* and a *Privacy Dashboard* to check and set *Privacy Policies* following the *Sticky-Policy*-approach, a *Privacy Policy Enforcement Point*, components to minimise data (*pseudonym* related components or a special PET for *geo-location privacy*), and an enhanced integrity component using *Malleable Signatures*. Hence, in this deliverable we describes RERUM's steps towards allowing the IoT to adhere to privacy-by-design.

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 609094

Impressum

Full project title	Reliable, resilient and secure IoT for smart city applications
Short project title	RERUM
Number and title of work-package	WP3 - System & Information Security and Trust
Number and title of task	T3.3 - Information Security and Privacy Enhancing Technologies
Document title	Privacy enhancing techniques in the Smart City applications
Editor: Name, company	Henrich C. Pöhls,Ralf C. Staudemeyer, UNI PASSAU
Work-package leader: Name, company	Henrich C. Pöhls, UNI PASSAU
Estimation of person months (PMs) spent on the Deliverable	

Executive summary

This deliverable presents the core results of the RERUM project in the area of Privacy in the Internet of Things (IoT). Up until now, there were only few EU projects that had dealt with the issue of privacy in this area, because their focus was mainly on developing the foundation technologies for making IoT a reality. However, RERUM, being a project working in the area of Smart City applications, acknowledges the fact that users' private data can be at major risk in smart city deployments, where thousands (or even millions) of smart devices are monitoring their everyday life activities. It is understandable though, that not every person has the same sensitivity considering the privacy of their data and many citizens do not have a clear view on which data are considered private or not. Nevertheless, the EU has set specific Directives for handling private user data and these should be followed at any given deployment of smart city applications.

RERUM, acknowledging the EU Directives and requests for a safer and more secure IoT, aims to build an architecture based on the concepts of security and privacy by design. The technologies for making IoT more secure were presented in RERUM Deliverable D3.1 that was delivered in March 2015. This deliverable aims to provide the reader with a detailed overview of the privacy issues in the IoT and the proposed technologies to protect the privacy of the citizens' sensitive information in smart city applications. As it is described in the RERUM System Architecture in Deliverable D2.5, RERUM has defined a large set of Privacy Components that are closely coupled with the RERUM Middleware, for ensuring that whenever required, the information that is passed from the RERUM System to the applications will be cleared of any information that could allow the tracking of individuals or the linking of data with users. Furthermore, privacy enhancing components are also installed on the devices to provide a first low-level step of privacy when data are gathered and transmitted to the gateways. Of course, the ultimate decision on handling their personal data should be taken by the users and for this reason, the RERUM Privacy architecture gives the power to the users, allowing them to set their own policies for handling different types of data gathered by the devices. When such policies do not exist, the users are being asked to provide their consent to applications that are requesting more information. This dynamicity of the RERUM architecture when handing personal user information is a key point for making the future Smart City deployments privacy-preserving by design.

Due to the very technical nature of the deliverable, for easing the understanding of the proposed techniques, an introductory section discussing the requirements of "Privacy by Design" and the existing methodologies is included with the document, followed by a glossary to provide the explanations for the different terms that are used throughout the document. Then, the following components and techniques are described and analysed:

- **Consent Manager**, which is the main component of the architecture that connects the "data subject" with the applications that are requesting their private information and asks the "data subject" for its consent, when required.
- **Privacy Policy Enforcement Point**, which shows how RERUM deals with the enforcement of the privacy policies that are set either by the administrator of the system or by the user himself.
- **Deactivator/Activator of Data Collection**, which is closely connected with the RERUM Middleware component that gathers the data from the RERUM Devices and can either de-activate the collection of data from some devices when there is a need to protect the user privacy or re-activate the collection when these data are requested by approved applications.

- **Privacy Dashboard**, which is the main component of the architecture that gives power to the users for handling the policies for their private information.
- **Anonymising, Pseudonymising Management and De-Pseudonymiser**, which are the components that are hiding the identities of the users from the applications, not allowing third parties to track down individuals through their identities.
- **Geo-Location PET**, which enables the system to not allow the disclosure of the location of individuals in applications like the traffic monitoring where the location of the users is being gathered by the devices.
- **Security Techniques for Enhancing Privacy**, discussing how some of the security techniques described in D3.1 can indeed be used for enhancing the privacy of users. These techniques are the data encryption, D2D authentication, credential bootstrapping, and integrity generation and verification.
- **Privacy Policy Checker and Attribute Need Reported**, which are closely connected with the access control mechanism and check the privacy policies each time an application requests to access the user attributes, as well as they renew the set of attributes that need to be checked, enhancing the dynamicity of the system.
- **Sticky Policies**, which are basically privacy policies that are stuck to data as they are transmitted all the way in the system, helping to promote the awareness of allowed actions and consent obligations for them.
- **Malleable Signatures**, which allow the signer to control authorised changes to signed data for enhancing the data privacy.
- **Data Perturbation with Integrity Preservation**, which allows intermediate authorised nodes to modify the data that are transmitted by the devices in order to enhance the system privacy by wiping out identifiable information, without unduly affecting the integrity of the data.
- **Leakage Resilient MAC**, which are message authentication codes that are preventing the leakage of sensitive information via side channels.

After presenting the entire list of RERUM privacy enhancing techniques, a discussion on the application of those techniques on the RERUM use cases follows. This chapter provides an excellent approach on the application of the various privacy techniques on the use cases, discussing the requirements of those use cases in terms of privacy and how each of the developed technique helps to address these requirements. This could be also quite interesting to service providers and system administrators in order to understand which of those techniques can be utilised for other applications that they provide. Finally, some open research items that have been identified are briefly described in order to stimulate future research in the very interesting area of privacy in the IoT.

List of Authors

Company	Author	Contribution
ATOS	Dario Ruiz	Privacy Policy Enforcement Point—design— (3.2), Privacy Policy Checker and Attribute Need Reporter (3.10), Privacy Policy Enforcement Point—concept— (4.4), Enhanced Privacy for User Information Retrieval (4.5), User-friendly ways to generate privacy policies (6.3) component analysis in use cases (5.3.3, 5.4.1, 5.4.3)
SAG	Santiago Suppan Ricarda Weber Jorge Cuellar	TOC and structure, Introduction (1), Privacy by Design (2), Introductions to chapters 3, 4, and 5, User Consent Manager (3.1), Deactivator / Activator of Data Collection (3.3), Privacy Dashboard (3.4), Anonymising and Pseudonymising Management (3.5), De-Pseudonymiser (3.6), PET Geo-Location (3.7), technical descriptions of these components (4.1, 4.6, 4.7 and 4.8), their analysis in use cases (5.1.1, 5.1.2, 5.1.3, 5.3.4, 5.3.1, 5.4.2), Introduction to use cases (5.1, 5.2, 5.3, 5.4), Summary to chapter 5, Measuring and visualising privacy (6.4), Appendix (A)
UNIVBRIS	Marcin Wojcik	Leakage Resilient MAC (4.10 and 5.3.5)
UNI PASSAU	Henrich C. Pöhls, Ralf C. Staudemeyer, Johannes Bauer, Benedikt Petschkuhn	Deliverable template in LaTeX, ToC and structure, editorial of deliverable, Integrity Generator / Verifier (3.9), Malleable Signatures (4.2, 5.3.2), Data Perturbation with Integrity Preservation on the Gateway (4.3), Combined UC-I2 and UC-O2 (5.2.1), Improving privacy with unobservable communication (6.2), Summaries to chapters 3 and 4, internal reviewing
FORTH	Alexandros Fragkiadakis Elias Tragos	Executive Summary, Data encrypter/decrypter for privacy (3.8.1), Compressive sensing encryption (4.9, 5.2.2), Conclusions (6.1), Traffic anonymisation (6.5)

Table of Contents

List of Figures

List of Tables

Abbreviations

ANR	Attribute Need Reporter (RERUM term)
BMW	a German Car Manufacturer
BSD	Berkeley Software Distribution (various UNIX flavours)
BSI	Bundesamt für Sicherheit in der Informationstechnik
CCTV	Closed Circuit Television, video surveillance
CH	Context Handler (PEP)
cid	Consent IDentifier (RERUM term)
CLS	Controllable Local Systems
CMU	Carnegie Mellon University
CPA	Chosen Plaintext Attack
CS	Compressive Sensing
DLC	Direct Load Control
DoS	Denial of Service
DR	Demand Response
DSM	Demand Side Manager
DSO	Distribution System Operator
EC	European Commission
ENISA	European Union Agency for Network and Information Security
EPSRC	Engineering and Physical Sciences Research Council (UK)
EU	European Union
EULA	End User License Agreement
GPL	Gnu Public License
GPS	Global Positioning System
GUI	Graphical User Interface
GVO	Global Virtual Object (RERUM term)
HAN	Home Area Network
HSM	Hardware Security Module
ICT	information and communication technology
IdA	Identity Agent (RERUM term)

IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet-of-Things
ISO	International Standardisation Organisation
LINDDUN	KU Leuven Acronym: Linkability, Identifiability, Non-repudiation, Detectability, Information Disclosure, Unawareness, Non-compliance
LMN	Local Metrological Network
MCDE	UK project: Meaningful Consent in the Digital Economy
MPC	secure Multi-Party Computation
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OAUTH	Open AUTHentication
OECD	Organisation for Economic Co-operation and Development
OWASP	Open Web Application Security Project
P3P	Privacy policies language: Platform for Privacy Preferences
PAP	Policy Administration Point
PbD	Privacy-by-Design
PDF	Probability Density Function
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PET	Privacy Enhancing Technology
PGW	Privacy Gateway
PIA	Privacy Impact Assessment
PIAF	EU project: A Privacy Impact Assessment Framework for data protection and privacy rights assessment
PII	US Term, Personal Identifiable Information
PIP	Policy Information Point
PPC	Privacy Policy Checker (RERUM term)
pPDP	Privacy Policy Decision Point (RERUM term)
pPEP	Privacy Policy Enforcement Point (RERUM term)

pPIP	Privacy Policy Information Point (RERUM term)
pPRep	Privacy Policy Repository
pPRP	Privacy Policy Enforcement Point (RERUM term)
PRC	Policy Retrieval Point (RERUM term)
PRep	Policy Repository
PRESCIENT	EU project: privacy and emerging fields of science and technology
PRIPARE	EU Project PReparing Industry to Privacy-by-design by supporting its Application in REsearch
PRP	Policy Retrieval Point
PSS	Probabilistic Signature Scheme (Bellare and Rogaway)
RD	Rerum Device
RFID	Radio Frequency IDentification
RSA	Rivest, Shamir und Adleman
RSS	Redactable Signature Scheme
S4P	Privacy policies language: A Generic Language for Specifying Privacy Preferences and Policies
SAML	Security Assertion Markup Language
SAPIENT	EU project: Surveillance, Privacy and Ethics
SG	Smart Grid
sid	Session IDentifier
SIMPL	Privacy policies language: a SIMple Privacy Language
SM	Smart Meter
SMGW	Smart Meter Gateway
SMO	Smart Metering Operator
sPEP	Security Policy Enforcement Point (RERUM term)
S&P&T Mechanisms	RERUM's Security, Privacy and Trust Mechanisms
STRIDE	Microsoft Acronym: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TC	Technical Committee
TOS	Terms of Service
TPM	Trusted Platform Module

UC	Use Case (RERUM term)
uid	User IDentifier
UN	United Nations
URL	Uniform Resource Locator
US	United States (of America)
USA	United States of America
VE	Virtual Entity
VO	Virtual RERUM Object
VRD	Virtual RERUM Device
W3C	World Wide Web Consortium
WD	Working Draft
WSDL	Web Services Description Language
XACML	eXtended Access Control Markup Language
XML	eXtensible Markup Language

1 Introduction

What constitute each individual's privacy might be different among individuals. It might be different due to social or cultural norms. However when you give away data about yourself and it is used for a different purpose than for which you understood it being used at the time of release, than this is a breach of privacy. As this deliverable by a comparative study on privacy definitions will show, RERUM is clearly rooted in the European Union's understanding of data protection: by EU law privacy starts with three key words *informed* and *voluntary consent*. RERUM's goal was to bring privacy to the Internet-of-Things 'by-design'. Such that individuals can understand and control, individually, how more and more monitoring devices collect data and use collected data.

Privacy matters —not only legally—. Citizen's increasingly become uncomfortable when actually shown clearly that they are being watched and what deductions are made from their behaviour, as the New York Times reported in July 2013 [56] when US stores started monitoring their physical customers's in shop behaviour. Privacy usually gets bad press only; and the Internet of Things currently is no exception as a look at privacy related headlines in Section 1.2 reveals. Thus, privacy concerns must be taken into account for legal compliance and —more importantly— for citizen's appropriation of the new technology. Because the best technology will not help, if its unaccepted and citizens will try to circumvent it, like DRM copy protection.

Data helps cities making better informed decisions. Good quality data allows improving or even enables informed decisions. This data is about the citizens and it can be voluntarily provided by the citizens. If available, it has been shown that it empowers not only the municipalities administration but every citizen to make more informed and thus better decisions. Examples are crowd-sourced traffic estimation like waze[1].

Possibilities for future Applications are endless, but application's details matter for privacy. According to the vision of the "Smart Cities Council" [215], a body founded by industry to promote smart city applications and use cases with governments and citizens, smart cities gather data from smart devices and sensors embedded in its roadways, power grids, buildings and other assets. RERUM does not know what the future will hold in store in terms of possible data as well as applications. Hence, we want our framework to be general to support a number of technical mechanisms; we want those mechanisms that are designed and developed by RERUM to be flexible to support different data structures, deployments and data flows. While being general, privacy —may be even more than security— can not be protected without a concrete system and application scenario. Thus this document, like RERUM, will always refer back to scenarios and the use cases to explain and highlight privacy problems and RERUM's solutions.

RERUM embraces privacy-by-design. The RERUM use cases where chosen to address several of the smart city scenarios mentioned above. RERUM views the use of IoT in the smart city context with a European mind-set, we want to —and are legally obliged to— address privacy topics from the very beginning of the design of "smart things" and applications. RERUM does not view the citizen's privacy as a luxurious after-thought (see Section 1.3). RERUM's design for an IoT framework, the resulting IoT infrastructure components and finally the use case implementations are capable of respecting and preserving the privacy of any concerned individual. Noteworthy to point out is that privacy-by-design, and as such also the RERUM framework, does not withhold the collection of data. RERUM brings all the tools to collect, send, store and process data

[1]https://www.waze.com/en/

1.1 Objective of this document

The main objective of this deliverable is to explain and document the designed and to-be-implemented privacy components that are facilitated for an increased privacy within the RERUM framework.

Deliverable D3.2[183] consolidates the output of task T3.3 in the work package on System & Information Security and Trust. The output of this task consists of conceptual work on privacy components and initial prototypes of some of those components. D3.2 presents the design of the privacy components defined previously in the D2.3[219], following D3.1[201] on security, this deliverable is focussing on privacy.

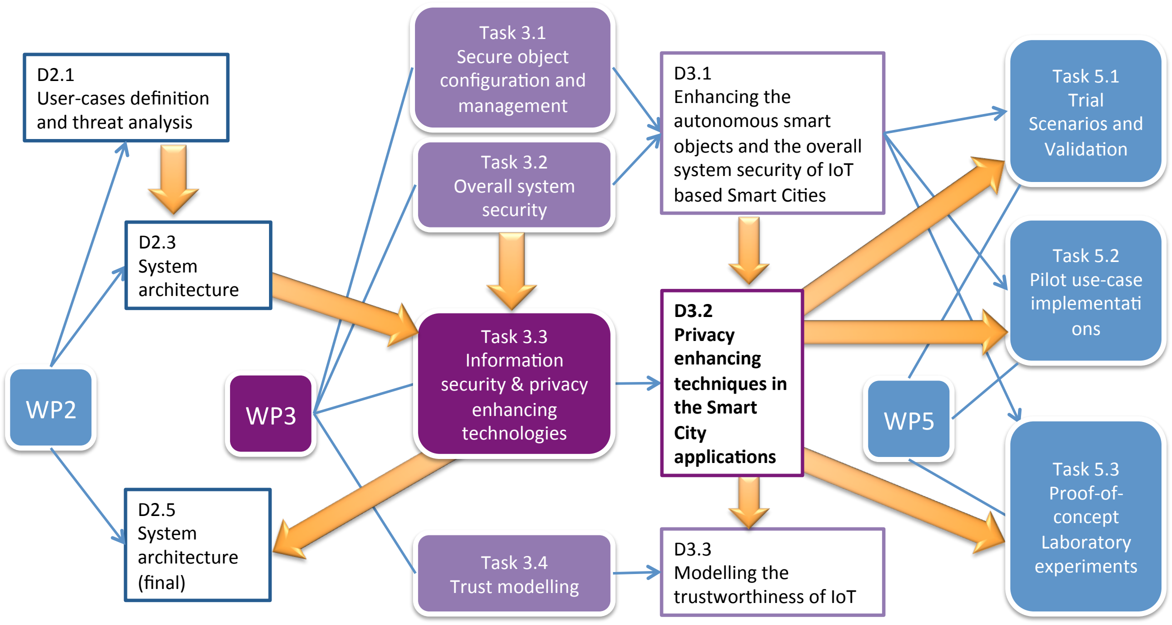

D2.1 User-cases definition and threat analysis
D2.3 System architecture
D2.5 System architecture (final)
WP2
WP3
Task 3.1 Secure object configuration and management
Task 3.2 Overall system security
Task 3.3 Information security & privacy enhancing technologies
Task 3.4 Trust modelling
D3.1 Enhancing the autonomous smart objects and the overall system security of IoT based Smart Cities
D3.2 Privacy enhancing techniques in the Smart City applications
D3.3 Modelling the trustworthiness of IoT
WP5
Task 5.1 Trial Scenarios and Validation
Task 5.2 Pilot use-case implementati ons
Task 5.3 Proof-of-concept Laboratory experiments

Figure 1: Overview of tasks and deliverables in WP3 and the most important links of D3.2

1.2 Privacy in IoT---current headlines

Privacy consciousness is increasing, as are the attempts to infringe on individuals' privacy. Privacy violation, consumer tracking and remote surveillance (also sometimes termed "Dataveillance") is quite some news topic today. As the Austrian consumer magazine "Konsument" phrases it in its issue 1/2015: "Who doesn't pay for their services on the Internet, in all probability is more of a product than a customer."

However even paying large sums of money for products and services may not ensure consumer privacy. "Konsument" in its issue 1/2015 describes "BMW Tele Services" of German car manufacturer BMW. There automatically and repeatedly car data are being transmitted to BMW. These functions are present in nearly all BMW cars from about April 2014 on "free of charge". "BMW Tele Services" are being enabled per default on car delivery to the customer. As BMW describe it themselves [24], technical car data are being transmitted to and evaluated by BMW both regularly and on demand. Additionally with "BMW Floating Car Data" [25] time-correlated location and other sensor data collected during vehicle operation are being transmitted to the "BMW Connected Drive" centre and contracted third parties to provide traffic information services, of course "free of charge" and "completely anonymous". German computer magazine “Heise Online” and a German automobile club recently found security flaws [112] in BMW’s online systems that facilitated data and car theft.

Many consumer electronic products, like TV sets, smart phones, e-book readers, digital cameras, play stations, media centres and such like are capable of establishing a network / an internet connection and reporting consumer behaviour sensitive data to the device manufacturer or other service provider. For instance the German newspaper “tz” [221] reports about a new Barbie doll that includes a microphone and reports the talk of the child “owner” back to the manufacturer Mattel for analysis and instructions on how to talk to the child.

Seemingly familiar street commodities, like the garbage can case reported by BBC in August 2013 [161], may unobtrusively watch pedestrians [214]. More and more insurance companies are bent on monitoring individuals to individualise their premium calculations, in front of all car and health insurance companies. The German road toll system prefers monitoring of road use of individual cars to a low-tech solution like a pay and display sticker on the windscreen. The European Union aims for establishing ”smart metering” in private households [208]. The dangers of such approaches are being discussed in the press (for instance [76]).

1.3 Privacy---human right or luxury?

Privacy traditionally is regarded quite differently in Europe and the USA. In Europe people tend to sense privacy as a right, in the USA it is commonly seen as a commodity that may be bought and sold. Regarding the future of privacy in an ever-growing internet-of-things, therefore there is quite some spectrum of opinions to be found:

- ENISA [64], the European Union Agency for Network and Information Security states: “Privacy is a fundamental human right, acknowledged by Article 8 EU Convention on Human Rights (respect for one’s ‘private and family life, home and correspondence’), EU Charter of Fundamental Rights Article 7 and 8, also the UN Declaration of Human Rights, Article 12. Privacy protection must be regarded as an individual value, also as an essential element in the functioning of democratic societies.”
- Lee Rainie [194], of the US-based Pew Research Centre predicts: “Few individuals will have the energy, interest, or resources to protect themselves from ‘dataveillance’; privacy will become a ‘luxury’. …Individuals will get used to the fact that mass surveillance exists and will not expect privacy by 2025. …The situation will worsen as the Internet of Things arises and people’s homes, workplaces, and the objects around them will ’tattle’ on them.”
- Hal Varian [194], a Google manager is confident that: “People will be comfortable sharing personal information with organisations. …Everyone will expect to be tracked and monitored, since the advantages, in terms of convenience, safety, and services, will be so great. …Continuous monitoring will be the norm.”
- Mathias Döpfner [70], CEO of the German Axel Springer Group in 2014 addressed an open letter to a Google manager, containing the statement: “Forget Big Brother - Google is better!”

Even if one regarded privacy as a mere commodity, the current prices paid for personal data are inadequate. A comment [36] in the German newspaper “Süddeutsche Zeitung” uses the analogy of the Spanish Conquistadores trading glass beads for gold in the 16th century. Modern Conquistadores like Facebook and Google collect big data, systematically exploit them, and turn them into big money. And the naive natives of the new digital dominions are guilelessly acquiescing to that, accepting mere pittances like permission to use a search engine or sending photos as remuneration for their personal data and their most intimate privacy. Data subjects today quite often are blissfully unaware of the true value of their data and the risks and potential negative consequences of forfeiting their privacy. They need to be made aware of the true value of their personal data and privacy. They must be provided with adequate instruments to protect their privacy in the ever growing Internet of things.

In October 2014 Glenn Greenwald, who was one of the first reporters to see the Edward Snowden files, gave a noted speech [104] in 07.10.2014 in Rio de Janeiro, talking about "Why privacy matters". He argued about the fallacy of the common statement: "I don't really worry about invasions of privacy because I don't have anything to hide." Greenwald argues that people, who seek privacy, are by no means per definition bad people and how free a society really is can be derived from how it treats its dissidents and those who resist orthodoxy. He points out that when humans are in a state where they can be monitored, their behaviour changes dramatically. The range of behavioural options that persons consider when they think they're being watched severely reduce. They become vastly more conformist and compliant. The potential of constant monitoring and surveillance is an instrument of control that suppresses human freedom. That this is not too far fetched, shows Figure 2, where you see from energy consumption data when someone is at home. In the figure the use of a steam iron (2000 Watt) gives a noticeable peak.

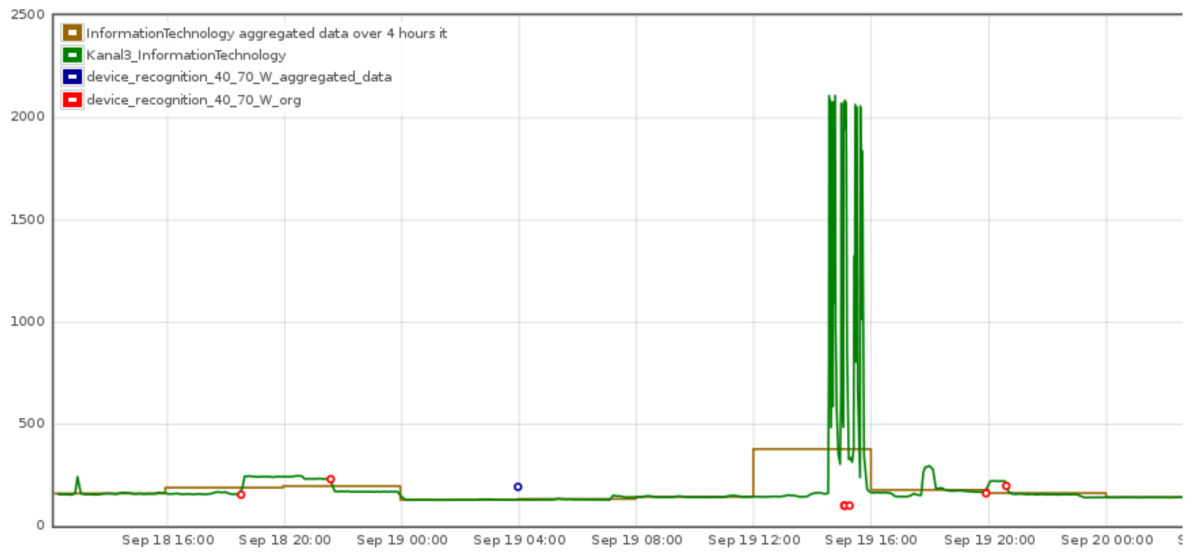


Figure 2: Energy consumption profiles can be quite unique; above peaks stem from the use of a 2000W steam iron

The UN General Assembly in its Draft of 19 November 2014 (A/C.3/69/L.26/Rev.1 [222]) affirms the right to privacy also in the digital age. It states: "…no one shall be subjected to arbitrary or unlawful interference with his or her privacy, family, home or correspondence, and the right to the protection of the law …as set out in article 12 of the Universal Declaration of Human Rights." [222] The body is of the opinion that the same rights that people have offline must also be protected online, including the right to privacy. Consequently the UN General Assembly calls upon all states to:

- respect and protect the right to privacy, including digital communication.
- take measures to put an end to violations of those rights and to …prevent such violations.
- review their procedures, practices and legislation regarding (mass) surveillance.
- establish or maintain …mechanisms capable of ensuring transparency …and accountability.
- provide individuals whose privacy has been violated with access to an effective remedy.

1.4 Structure of the document

In Chapter 2 we look at Privacy-by-Design issues. We investigate human privacy aspects and analyse which IoT data require privacy protection. We compare the different traditional privacy principles recommended by relevant bodies and summarise recent developments following the publication of RERUM Deliverable D2.1 [167] regarding privacy issues arising especially in the IoT context. We present the LINDDUN privacy threat analysis method and the PRIPARE overall privacy engineering process. Finally we update our RERUM Privacy-by-Design requirements specified in D2.2 (Section 2.6.3 [62]) and provide a privacy glossary.

Chapter 3 specifies in detail the seven privacy related functional components from D2.3 [219], namely User Consent Manager, Privacy Policy Enforcement Point, Deactivator / Activator of Data Collection, Privacy Dashboard, Anonymizing and Pseudonymising Management, De-Pseudonymiser, and Privacy Enhancing Technologies for Geo-Location. We also summarise several security components from D3.1 [201] needed as privacy basis, specifically Data Encrypter / Decrypter, Device-to-Device Authenticator, and Credential Bootstrapping Client / Authority. Finally we introduce two newly conceived privacy components, Integrity Generator / Verifier, and Privacy Policy Checker / Attribute Need Reporter.

Chapter 4 is dedicated to an in-depth description of the RERUM privacy enhancing protocols and mechanisms specifically developed for or adapted to and improved for RERUM needs, and to the elaboration on relevant aspects of certain RERUM privacy enhancing components. We address sticky policies, malleable signatures on devices, details of the privacy policy enforcement point's implementation, specific aspects of enhanced privacy for user information retrieval, an efficient pseudonym generation and management mechanism, a RERUM specific concept for privacy-enhanced tokens for authorisation in constrained environments, GeoLocation position hiding mechanisms, a more secure compressive sensing encryption method, and a practically deployable leakage resilient MAC.

In Chapter 5 we explain the how the RERUM privacy functional components facilitate selected enhancements of the citizen's privacy in several situations. These situations are derived from RERUM's four use cases, smart transportation, environmental monitoring, home energy management, and comfort quality management. After summarising the overall use case goal and highlighting typical privacy problems that these use cases might bring to the citizens, we show how selected functional components of RERUM will enhance privacy, while still allowing the goals of the use case to be achieved.

Chapter 6 concludes this document and addresses additional relevant privacy topics, points out open issues, like the need for regulatory action, and indicates open issues for future research, like end-user-friendly ways to generate privacy policies.

2 Privacy-by-Design

The support of "Privacy-by-Design" (PbD) is one of RERUM's main project objectives (see e.g. RERUM Deliverable D2.1, Section 4.1 [167]). Privacy is to be taken into account from the very conception of "smart things", corresponding infrastructures, and applications. Further, the RERUM platform will provide a set of tools and components that can be used as Privacy-Enhancing-Technologies (PETs) in other IoT contexts as well. But it is important to notice that PbD (as well as any privacy methodology) is a process, which can not be simply reduced to the use of a set of PETs. When using the RERUM toolbox, the RERUM PETs cannot replace the necessary PbD process that should be part of any project that collects or uses personal data.

Which IoT data require privacy protection? Sensors usually do not collect personally identifiable information like typically associated with names and addresses. Then which IoT data, if any, are personal data and thus privacy sensitive? In Section 2.1 we investigate this question and the aspects of human privacy in general.

What is the meaning of "Privacy-by-Design"? Which privacy principles should be observed? RERUM needs not only to take traditional information and communication scenarios into account, but also issues arising especially in the IoT context. There have been quite a lot of recent developments following the publication of RERUM Deliverable D2.1. To ensure RERUM takes all relevant privacy aspects into account, in this chapter we review both traditional Privacy-by-Design principles in Section 2.2 and recent IoT-specific "Privacy-by-Design" issues in Section 2.3.

When developing and operating IoT devices, systems and applications handling privacy sensitive data, privacy engineering must be defined and integrated into the traditional systems and software engineering life cycle, similar to security engineering. Already in early phases of conception some privacy threat analysis should be conducted equivalent to a traditional security threat and risk analysis. A method for this is offered by LINDDUN [237] (already deployed by RERUM in D2.1), which we summarise in Section 2.4. It can be used as part of an overall privacy engineering process, like the one described by the EU project PRIPARE [220], which we outline in Section 2.5. Privacy sensitive data needs to be protected by appropriate privacy protection measures. We talk about "hard" and "soft" privacy controls in Section 2.6.

In Section 2.7 we update the RERUM Privacy-by-Design requirements specified in D2.2 (Section 2.6.3 [62]). We finish this subsection with a RERUM privacy glossary in Section 2.8 for reference of the terms used in this deliverable.

2.1 Personal data in the IoT

Human privacy has many different aspects. In 2013 Finn, Wright and Friedewald [87][2] identified seven "types of privacy":

1. Privacy of **the person**: Refers to the right to keep body functions and body characteristics (such as genetic codes and biometrics) private.

[2]Their works were in the context of the EU project PRESCIENT (which stands for privacy and emerging fields of science and technology: Towards a common framework for privacy and ethical assessment) [117].

2. Privacy of **behaviour and action**: Includes sensitive issues such as sexual preferences and habits, political activities and religious practices in public, as well as private space.
3. Privacy of **communication**: Aims to avoid the interception of communications, including mail, telephone, wireless, et cetera.
4. Privacy of **data and image**: Makes sure that individuals data and images are not automatically available to others and that data subjects are given a substantial degree of control over that data and its use.
5. Privacy of **thoughts and feelings**: Defines the right not to share their thoughts or feelings or to have those thoughts or feelings revealed.
6. Privacy of **location and space**: Specifies the right to move about in public or semi-public space without being identified, tracked or monitored; also right to solitude and a right to privacy in spaces such as the home, the car or the office.
7. Privacy of **association (including group privacy)**: Declares the right to associate with whomever a person wishes without being monitored.

Personal data means data which relates to a (living) individual who can be identified (even without a name associated with it) either directly from those data, or when fused with other information (potentially) available to the data controller. This includes opinions about and intentions for the data subject, like performance assessments or a health conditions. Sensitive personal data needs to be treated with greater care than other personal data and comprises for instance racial or ethnic origin of the data subject, political opinions, religion, health conditions, and criminal record.

Personal data are much more then just name and address of a person, even in traditional context, as explained by the different types of privacy above. This still holds even more in IoT environments. In October 2014 the 36th International Privacy Conference of the Data Protection and Privacy Commissioners was held [173]. Consensus was that connectivity is going to be ubiquitous and big money is in new services and IoT data. The data protection and privacy commissioners recommend regarding and treating **all IoT sensor data** as personal data. The Data Protection and Privacy Commissioners demand that privacy protection must start when IoT data are collected, not only when advanced data processing begins. Additionally in IoT scenarios initially impersonal data may in the course of sensor, data source and knowledge fusion become privacy sensitive data.

Personal data collected in our RERUM IoT use cases could be related to e.g. physical location, energy consumption, ambient room conditions, CO_2 production, et cetera. They could be revealing individual behaviour, e.g. actions, habits and lifestyle, feelings, mood, …of the persons who contributed to the data sensed and processed (see figure 3). Personal data status may change, e.g. via aggregation, if personal data are used for statistics, the result might be not personal any longer. Or via sensor / data fusion, where initially non-personal data may become personal, if fused with other data sources, e.g. knowledge, which family member is at home, or who possesses a certain smart phone. Especially via sensor fusion, a meaningful summary result may be extracted from seemingly meaningless individual source data. Sophisticated algorithms can be used to extract sensitive data from various seemingly innocent non-personal sources. Sensor data can be combined with other sources like CCTV and internet logs. IoT data may preclude real anonymous use, re-identification attacks via data fusion may become possible.

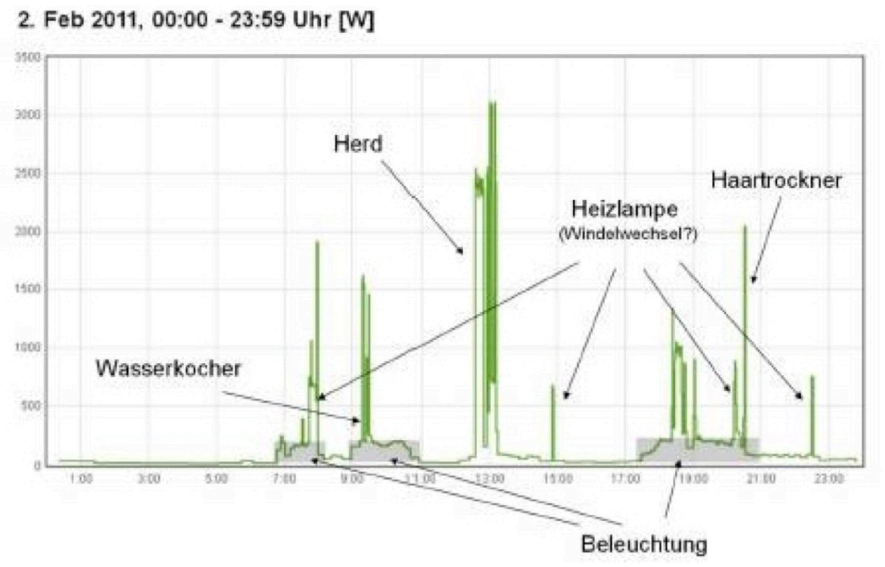


Figure 3: Example household power consumption profile [126]

Characterising a planned system preparatory to in-depth privacy protection activities involves identifying personal data and their flows. IEC/ISO 29100 [119] recommends to specify the personal data collected, created, communicated, processed or stored within privacy domains or systems and to classify personal data in terms of its identifiability and sensitivity. *Sensitive personal data* may involve stricter regulation.

2.2 Traditional Privacy-by-Design principles

Adequate privacy protection needs observation of fundamental Privacy-by-Design principles at every stage of the system and application development process. "Privacy-by-Design" is usually defined as a number of principles that designers can apply from the very beginning of system development. This ensures that privacy is addressed correctly including proof of data protection compliance. There are many initiatives proposing principles relevant in this context. In this section, we summarise a relevant subset of traditional "Privacy-by-design" concepts focusing mainly on classical internet commerce and transactions.

2.2.1 OECD privacy principles (09/1980)

The OECD (Organisation for Economic Co-operation and Development) data privacy principles [95] were released initially in September 1980 and substantially revised in year 2013 [172]. They aim to take both European data protection legislation and (as they term it) "cultural expectations" into account. They are presented in the Annex to the 2013 OECD Privacy Guidelines [172] (see Part Two, paragraphs 7 though 14). The principles are:

1. **Collection limitation:** There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.

2. **Data quality:** Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.
3. **Purpose specification:** The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.
4. **Use limitation:** Personal data should not be disclosed, made available or otherwise used for purposes other than those specified except with the consent of the data subject; or by the authority of law.
5. **Security safeguards:** Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data.
6. **Openness and transparency:** Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller. There should be a general policy of openness about developments, practices and policies with respect to personal data.
7. **Individual participation and control:** An individual should have the right to obtain from a data controller the data relating to him, and the right to have incorrect or illegally obtained data erased, rectified, completed or amended.
8. **Accountability:** A data controller should be accountable for complying with measures which give effect to the principles stated above.

US “Safe Harbour” [223] is a cross-border data transfer option for organisations in the US that conduct business in the EU, particularly for handling customer data. The United States Department of Commerce developed the “Safe Harbour self certifying legal framework” to instruct US organisations to comply with the EC Data Protection Directive. Because of the purpose, the framework’s principles follow closely with OECD’s.

2.2.2 ISO/IEC 29100 privacy principles (11/2011)

ISO/IEC 29100 specifies general privacy principles and was published in December 2011. RERUM has derived an initial set of Privacy-by-Design principles from this standard in D2.2 (Section 2.6.3). ISO/IEC 29100 is publicly available [119]. **Personal data** this standard terms as “personally identifiable information” (PII). A **data subject** is the person the personal data are about. A **data controller** determines the purposes for which and the manner in which any personal data are, or are to be processed. A **data processor** processes personal data on behalf of the data controller. The eleven ISO/IEC 29100 principles are:

1. **Consent and choice:** The data subject needs to be given choice whether or not to permit personal data processing. Consent must be given freely, specific and on a knowledgeable basis. The data subject may withdraw consent. Means for choice and consent need to be offered at the time of collection, first use, or as soon as practicable.

2. **Purpose legitimacy and specification:** The purpose of data collection and processing must comply with the rule of law. The data subject is to be informed of the purpose with sufficient explanations, in an unambiguous manner, and in advance.
3. **Collection limitation:** The collection of personal data should be limited to what is legal and strictly necessary for the specified purpose. The type of personal data collected and its justification should be documented. The data subject should be clearly informed of optional data requests.
4. **Data minimisation:** This principle goes beyond mere data collection; data processors should use procedures to minimise processing of personal data; they should also minimise the number of parties personal data are disclosed to ("need-to-know"). Non-personal and unlinked data processing should be preferred.
5. **Use, retention and disclosure limitation:** The data controller should limit use, retention and disclosure including the transfer to purpose and legal compliance. Personal data should be deleted or de-personalised as soon as possible. National or local requirements specific to cross-border transfers need to be observed.
6. **Accuracy and quality:** Personal data should be accurate, complete, up-to-date, and relevant for the purpose; especially where data could be used to grant a benefit or result in harm to a natural person.
7. **Openness, transparency and notice:** Data subjects should be provided with clear and accessible information about the data controller and its purpose, policies, practices, and processing. This includes means open to data subject for influencing processing, and notice about major processing changes. It may include transparency of processing logic.
8. **Individual participation and access:** Data subjects should be enabled to access and review their own personal data, and to request correction and removal of these data, as appropriate.
9. **Accountability:** Processing of personal data requires responsibility for their adequate protection. This includes implementation and documentation of policies and practices, and responsibility for compliance of third party recipients, also privacy breach notifications, and complaint handling and redress procedures for data subjects.
10. **Information security:** Data controllers must protect personal data under its authority with adequate information security controls throughout the complete data life cycle. This includes careful selection of data processors.
11. **Privacy compliance:** Data controllers must be able to prove that processing meets data protection and privacy requirements by periodical audits. This also includes privacy risk assessments.

To supplement aspects not covered by OECD principles, ISO/IEC 29100 added the principle of "data minimisation" to cover data processing. Arguably one could subsume "collection limitation", as well as "use, retention and disclosure limitation" as being aspects of "data minimisation". One aspect of OECD principle 3 "purpose specification" addressing "secondary use" is not mentioned explicitly in ISO/IEC 29100. Neither Canadian principle 2 "privacy-as-the-default" nor principle 7 "respect for user privacy" are mentioned in ISO/IEC 29100. Principles "Data minimisation" and "Use, retention and disclosure limitation" emphasise the advisability of using non-personal, unlinked and anonymized data. This indicates a preference for *"de-personalised data"* use.

Other ISO/IEC standards base themselves on this standard for domain-specific profiling, like ISO/IEC 27018 [120] ("Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processor"). This standard adapts ISO/IEC 29100 for the public cloud computing environment.

2.2.3 Canadian Privacy-by-Design principles (01/2009)

Former Canadian data protection officer Ann Cavoukian has compiled seven rather general and high-level "Privacy-by-Design" principles [49], the most recently published specification dating from January 2009:

1. **Proactive not Reactive:** Privacy should be protected preventative not remedial. One should anticipate and prevent privacy-invasive events before they happen.
2. **Privacy as the Default:** Default Settings and default rules should protect personal data automatically even if an individual does nothing.
3. **Privacy Embedded into Design:** Privacy is an essential component of the core functionality.
4. **Full Functionality:** Integrating privacy and security and other requirements should result in A combination of security and privacy should result in a positive-sum benefit, not a zero-sum one. Legitimate interests should be taken into account in "win-win" manner, not e.g. privacy vs. security; we need both.
5. **Full Life Cycle Protection:** End-to-end security and privacy protection assures that privacy is included prior to data collection, and extends to secure data destruction.
6. **Visibility and Transparency:** One should keep the data collected and the manners of using them open, and assure that system is operating according to the stated promises and objectives. One should seek independent verification (like specified in IEC 29100 / chapter 11. Compliance).
7. **Respect for User Privacy:** One should keep the privacy protection user-centric, and adhere to the interests of the individual data subject / user uppermost. Strong privacy defaults, appropriate notice, user-friendly options are just some issues here.

Notable are the Canadian requests for privacy by default and respect for user privacy interests. These are not in focus of the OECD and IEC/ISO 29100.

2.2.4 ENISA - Privacy-by-Design recommendations (12/2014)

ENISA, the European Union Agency for Network and Information Security in this context has issued a report called "Privacy and Data Protection by Design - from policy to engineering" in December 2014 [64]. ENISA derives privacy requirements from existing and currently discussed EU data protection laws in the mind-set of EU terms of privacy:

1. **Lawfulness:** Currently there are (a) unambiguous consent of data subject, (b) part of contract, (c) legal basis, (d) medical emergency, (e) public interest, or (f) legitimate interests not overridden by fundamental rights and freedoms of the data subject. This however does not mean, that any legitimate business interest whatsoever is sufficient to justify any collection of personal data.
2. **Transparency:** Data subjects get sufficient information about data collection and use, understands risks and control actions they can take.
3. **Consent:** Data subjects need to grant specific, informed, explicit, and voluntary indication of their intentions with respect to the processing of their data.
4. **Purpose binding:** Personal data obtained for one purpose must not be processed for other purposes that are not compatible with the original purpose.
5. **Data minimisation:** Only personal data necessary for a specific purpose may be processed, and must be deleted or anonymized as soon as possible.
6. **Control rights:** Data subjects require the right to rectify, block, and erase personal data, withdraw consent for the future.
7. **Information security:** Calls for technical and organisational safe-guards.
8. **Accountability:** Ensure and be able to demonstrate the compliance with privacy and data protection principles or legal requirements.
9. **Privacy-by-Design:** Consider full data life cycle from system design on. Default settings must protect user privacy in full.
10. **Privacy by default:** Data subjects must enable explicitly extended functionality with potentially reduced privacy protection.

ISO/IEC 29100 principles 3 “collection limitation”, 4 “data minimisation”, 5 “Use, retention and disclosure limitation”, as well as maybe 6 “accuracy and quality” are subsumed under “data minimisation” here. ISO/IEC 29100 principle 11 “privacy compliance” has been included in the definition of “accountability”. The prohibition of “unrelated secondary use” specified in OECD principle 3 “purpose specification” here is emphasised in a separate principle “Purpose binding” as appropriate in an European mind-set. Principle “data minimisation” also promotes to use of *“de-personalised data”*. Canadian principle 2 “privacy as the default” is explicitly addressed by ENISA in “privacy by default”, as is the general aim of the Canadian principles in “Privacy-by-Design”. The need for a specific purpose seems to be implied by ENISA.

ENISA’s report contains a list of high-level recommendations to various bodies to improve general digital privacy:

- Politics, Legislation, and Data Protection Authorities should
  - support development of mechanisms for privacy-friendly services,
  - fund investigation in privacy engineering, incl. multidisciplinary approaches,
  - promote privacy and data protection in their norms, and
  - provide independent guidance and assess modules and tools for privacy engineering.
- Developers and Research should
  - offer tools that enable the intuitive implementation of privacy properties and

  - support infrastructure projects and privacy-supporting components, such as key servers and anonymizing relays.
- Standardization Bodies should
  - include privacy considerations in their process and
  - develop standards for interoperability of privacy.

There is an upcoming data protection regulation of the European Parliament and of the Council on the "protection of individuals with regard to the processing of personal data and on the free movement of such data" (General Data Protection Regulation, draft from March 2014 [83]. Article 23 and reason 61 in its preamble require data protection via technical means ("data protection by design") and data protection friendly default settings ("data protection by default"). Data processors need to develop suitable strategies and controls for data protection. The regulation for instance names strategies like data minimisation, early pseudonymization, transparency, and the data subject supervising data processing. The regulation requires product developers to take into account data protection, so data processors fulfil their data protection obligations.

2.2.5 Mapping of traditional Privacy-by-Design principles

This subsection provides an approximative mapping of traditional "Privacy-by-Design" principles (see Table 1) of the initiatives presented in the previous subsections as well as of the PRIPARE initiative (see Section 2.5).

The mapping in Table 1 may only be understood as quite approximatively, as the meaning of the principles is different from initiative to initiative. The reader should not think of the single privacy principles as clearly distinct and independent. Even within the same framework they are overlapping. For details please refer to the subsections describing the individual initiatives.

Notable are the Canadian requirements for " 1, 3, 5, 7 Proactive User-friendly privacy covering the complete life cycle", thus defining "Privacy-by-Design", and " 2 Privacy as default setting.". These have not been in focus of the OECD and are also not listed by ISO/IEC 29100. They however are explicitly addressed by ENISA in principles " 9 Privacy-by-Design" and " 10 Privacy by default", and also by PRIPARE in their principles " 13 Privacy and data protection by design" and " 14 Privacy and data protection by default".

ISO/IEC 29100, in comparison to OECD, added the principle of " 4 Data minimisation" to cover data processing specifically. The "purpose binding" of OECD principle " 3 purpose specification" is not in focus of ISO/IEC 29100, but is taken up again by ENISA in principle " 4 purpose binding" and also comprised in PRIPARE principle " 3 purpose specification". ISO/IEC 29100 in principle " 11 Privacy compliance" introduced the need to proof compliance in audits, which also is stressed by the Canadian principle " 6 Visibility and transparency", and comprised in ENISA principle " 8Accountability" and PRIPARE principle " 11 Accountability" respectively.

ISO/IEC 29100 principles " 3 Collection limitation", " 4 Data minimisation", and" 5 Use, retention and disclosure limitation", as well as parts of " 6 Accuracy and quality" were subsumed under ENISA principle " 5 Data minimisation". ISO/IEC 29100 principle " 11 privacy compliance" has been included in the definition of ENISA principle " 8 accountability".

OECD	ISO 29100	Canadian	ENISA	PRIPARE
(1) Collection limitation	(1) Consent and choice	(7) Respect User Privacy	(3) Consent, (6) Control rights	(2) Data minimisation and proportionality
(3) Purpose specification	(2) Purpose legitimacy and specification	(5) Full Life Cycle Protection, (6) Visibility and Transparency	(1) Lawfulness, (4) Purpose binding	(3) & (4) Purpose specification and limitation
(1) Collection limitation	(3) Collection limitation		(5) Data minimisation	(2) Data minimisation and proportionality
(3) Purpose specification	(4) Data minimisation		(5) Data minimisation	(2) Data minimisation and proportionality
(4) Use limitation	(5) Use, retention and disclosure limitation		(5) Data minimisation	(10) Limited conservation and retention
(2) Data quality	(6) Accuracy and quality			(1) Data quality
(6) Openness and transparency	(7) Openness, transparency and notice		(2) Transparency	(5) Transparency and openness
(7) Individual participation	(8) Individual participation and access		(6) Control rights	(6) Right of access (7) Right to object (12) Right to erasure
(8) Accountability	(9) Accountability		(8) Accountability	(11) Accountability
(5) Security safeguards	(10) Information security	(4) Full functionality	(7) Information security	(8) Confidentiality and security
	(11) Privacy compliance	(6) Visibility and transparency	(8) Accountability	(9) Compliance with notification requirements
		(1), (3), (5), (7) Proactive User-friendly privacy	(9) Privacy-by-Design	(13) Privacy and data protection by design
		(2) Privacy as default setting	(10) Privacy by default	(14) Privacy and data protection by default

Table 1: Approximative mapping of traditional privacy principles

“Data quality” is not in focus of ENISA, as it is for OECD with “ 2 Data quality”, ISO/IEC 29100 with “ 6 Accuracy and quality”, and PRIPARE with “ 1 Data quality”. ENISA principle “ 5 Data minimisation” promotes to use of *“de-personalised data”*, as do ISO/IEC 29100 principles “ 4 Data minimisation” and “ 5 Use, retention and disclosure limitation”.

2.3 IoT-specific Privacy-by-Design aspects

IoT specific privacy aspects have not yet fully or at all addressed by traditional “Privacy-by-Design” principles. Recently however quite a few initiatives have addressed privacy issues in IoT-specific contexts, both in the US and in Europe. There are substantial discrepancies between the US and European mindset regarding privacy as already pointed out in the introduction of this deliverable.

2.3.1 The Future of Privacy Forum, an US body (11/2013)

According to US American understanding, privacy often is not so much a human right, but rather a commodity to be sold, given away, and exploited as profitable as possible. The “Future of Privacy Forum” in November 2013 released an updated privacy paradigm for the IoT [233]. They are aware many connected IoT devices will be invisible to users, devices might even be shared, and that it might be unclear of whom to obtain consent (operator, user, data subject …).

The general tenor of the white-paper is that ‘privacy sucks’. The frequent need for explicit consent would be cumbersome to data subjects and detrimental to data collector business interests. Purpose specification, use limitation, and data minimisation would limit development. Rigidly and narrowly specifying context would trap knowledge and hamper progress. As an example the paper states that the US via total mobile phone surveillance was able to monitor post-earthquake migration in Haiti.

Recommendations regarding privacy preservation in the IoT are promotional to business interests. The Forum suggests that IoT data controllers and processors should generally develop codes of conduct for privacy handling in IoT contexts and specifically observe the following recommendations:

- **De-personalisation:** Use anonymized data when practical. The data subject should not fear re-identification. However data and sensor fusion and analysis technologies may already or eventually allow for re-identification.
- **Purpose binding:** Respect context where personal data are collected. Consumers expect the worst anyway. Few secondary uses of personal data will surprise them.
- **Transparency:** Be transparent about data use. This may involve stating that personal data are used in whatever way pleases the data collector.
- **Accountability:** Use automated accountability mechanisms to monitor and log data transfers and uses.
- **Individual participation:** Provide reasonable access for data subjects to their personal data.

These recommendations essentially consist of a subset of traditional privacy principles, while many other traditional privacy principles are classified as detrimental to business interests. Obtaining meaningful and high-quality “consent” in IoT context in a manner that involves a high degree of usage comfort for the data subject has been recognised as an open issue. “De-personalisation” of sensor data as well as meta-data is particularly hard to achieve in practice. This subject may require more technical attention and should involve less data subject trust than assumed by the “Future of Privacy Forum”.

2.3.2 Law Prof. Peppet, University of Colorado, USA (08/2014)

Prof. S. Peppet [178] of the University of Colorado, US, issued a white-paper [177] about first steps towards legally regulating the IoT. Focused use cases are

- health and fitness,
- automobile sensors,
- smart home, smart grid,
- employee sensors, and
- smart mobile sensors.

As a challenge to IoT regulation in the US (besides the general legal US system) the author regards big data analytics and sensor fusion. IoT data may *reveal more than intended* by the data subject and sensor data are *difficult to de-identify*. Many *security flaws* weaken IoT devices, giving third-party abusers access to personal data. It is unclear how to obtain *valid high-quality consent* (=notice and choice) on IoT devices.

Legislation in the US (and elsewhere) is unprepared for the abuse potential opened by the IoT (e.g. anti-discrimination, consumer protection, privacy). The author stresses the urgent need for legal regulation, before IoT is (even more) established. First steps proposed by him include use of existing constraints by defining **personal data in IoT** to fall under them. He recommends expanding data breach notification laws so they include IoT data as well.

Also he recommends to improve on "consent practices". Especially he recommends to consider carefully what the data subject should know to allow for an informed decision. For this the author proposes some topics of interest, and analyses existing personal smart devices regarding how well they address these topics.

Like the "Future of Privacy Forum", also Prof. Peppet regards "consent" in IoT a subject leaving much room for improvement. For him the same applies to effective "de-personalisation" of IoT sensor data and IoT "information security". The author would like to define IoT "sensor data as personal data" as recommended by the 36th International Privacy Conference.

2.3.3 EU Article 29 Data Protection Working Party on IoT (08/2014)

The EU Article 29 Data protection working party issued its opinion on the recent developments in the IoT [82] in 08/2014. In there, the working party refers to its opinion 02/2013 on apps on smart devices. Main IoT use cases taken into account are

- wearable computing and quantified self (which to RERUM understanding implies conscious users coinciding with data subjects) and
- domotics (that is, IT- and automation technologies applied to the smart home; related to our RERUM indoor use cases).

As **data controllers** (or data collectors, as they term it), device manufactures, social platforms, app developers, and other parties are identified. **Data subjects** taken into account are subscribers, users, and non-users.

The EU Article 29 Data protection working party identified the following privacy challenges especially in the context of their use cases:

- **Lack of data subject control:** In combination with a substantial information asymmetry this gives a disproportional advantage to the data controller and processor.
- **Data subject consent quality:** Regarding EU law this requires informed and voluntary consent. However, this consent quality on the Internet rarely is achieved for traditional IT applications, much less for IoT-based ones. It is still unclear how to obtain high-quality consent from IoT data subjects, especially non-user ones.
- **Sensor data and data fusion:** The legal notion of *"Personal Data"* is still not sufficiently defined in the case of IoT. In traditional approaches the legal term refers to name and address data correlated with further data. This clearly is insufficient in the face of IoT sensors and as well sensor and data fusion:

  - Inferences can be derived from data, aggravated by sensor and data fusion, and secondary use, which at the time of data collection may not even have been conceived or been conceivable. Frequently the potential of such interferences and the possible corresponding negative impacts are quite unclear to the data subject.
  - Data and sensor fusion allow for privacy intrusive behaviour pattern modelling and profiling, resulting in permanent surveillance in public and at home. This is commonly regarded as incompatible with human dignity.
  - IoT sensor data often allow to identify individual data subjects, which implies a limitation of anonymous service use.
- **Information security:** In some cases security is traded versus energy- and other efficiency requirements, resulting in poor system security allowing for attacks that may also result in privacy breaches.

Facing such challenges the working party issues the following initial recommendations for IoT-specific Privacy-by-Design. The requirements are classified as incomplete and require augmentation over time.

- **Privacy-by-design:** A rigourous privacy impact analysis must come first.
- **Privacy-by-default:** Privacy settings must involve good defaults, without data subject intervention, privacy must hold, setting changes should rather result in lessening privacy than be required to protect it.
- **Data minimisation and de-personalisation:** Data should be evaluated and aggregated as early as possible in the communication chain. Local processing is to be preferred. One should delete raw data soon, also to prevent inappropriate secondary abuse. De-personalisation should good even after attempts at data fusion, aggregation and advanced reasoning. This is a *paramount* requirement.
- **Individual participation:** There must be a reasonable subject-control on their personal or personalizable data.
- **Consent and alternatives:** There should be an aim to obtain high-quality voluntary consent. For true voluntariness, there may be no economic penalties for withholding consent and insisting on privacy and non-observation. There may also be no degraded capability access. For instance the use of a TV, heating, power source, fridge, watch, etc. must also be possible without the "smart".
- **Notice and awareness:** The data subject's awareness must be supported by clear and repeated announcing/broadcasting/reminding of data collection.

As a final technical recommendation the working party considers investigation of *"(personal) privacy proxies"* and *"sticky policies"* an interesting approach for privacy enhancing technologies in the IoT.

The working party regards IoT "sensor data as personal data" as recommended by the 36th International Privacy Conference. "Data minimisation" and the need for effective "de-personalisation" of sensor data is stressed, the latter also is to include specifically early evaluation and aggregation, i.e. a preference for local processing, as well as deletion of raw data to prevent "secondary abuse" and ensure "purpose binding" as proposed by ENISA in its principle 4. A "preference of local processing" is also recommended by the data protection and privacy commissioners in their 36th International Privacy Conference. That the IoT-relevant aspect of effective "de-personalisation" should good even after attempts at data fusion, aggregation and advanced reasoning is emphasised by the EU Article 29 Data Protection Working Party.

High-quality “consent” practices are pointed out as required in IoT contexts by the working party as well, together with the need for “viable alternatives” for non-consenting data subjects. “Notice” and special “awareness” of ongoing data collection is indicated as particularly relevant in IoT contexts to prevent data subjects from forgetting they are being watched, particularly as an IoT infrastructure may be collecting data in an unobtrusive manner. IoT-specific privacy issues are currently being addressed by several EU-funded projects [13].

2.3.4 Summary of IoT Privacy-by-Design aspects

In summary, the initiatives presented in this subsection, especially the by the EU Article 29 Data Protection Working Party, but also the previously mentioned data protection and privacy commissioners in their 36th International Privacy Conference in October 2014 have emphasised or raised the following IoT-specific privacy aspects:

- Sensor data should be regarded as personal data. In the presence of data fusion and advanced data analytics these data may even turn out to be highly sensitive.
- Data minimisation is recognised as paramount for privacy protection.
- De-personalisation of sensor data and information blurring appear promising privacy-preserving approaches.
- Local processing is to be preferred to minimise personal data propagation and to improve data subject control, as is early evaluation and aggregation, and raw data deletion.
- Obtaining meaningful and high-quality “consent” in IoT context in a manner that involves a high degree of usage comfort for the data subject has been recognised as an open issue.
- IoT things and applications should offer viable alternatives that provide some core functionality even if data subjects refuse consent for external sensor data transmission, e.g. wrist watch functionality displays at least time without “smart”.
- Awareness of ongoing data collection should be raised to prevent users from forgetting or ignoring unobtrusive ongoing IoT data collection.

2.4 Privacy threat analysis: LINDDUN model

An important early phase of an IT security engineering live cycle deals with the analysis of threats to the system. There are various threat analysis methods. One involves threat categories like Microsoft STRIDE [160]. The analyst views the application from various angles and answers “what, if” and “how” questions. For instance in the category “Spoofing identity” a question could be “How can an attacker change authentication data?”, and “What could an attacker achieve, if that attacker could impersonate this and that legitimate user?” resulting in a list of potential threats and impacts. Predefined threat trees can be helpful to build customised threat trees. Goal is to elicit suitable security requirements.

KU Leuven has developed a corresponding privacy threat analysis model [237] they call LINDDUN. It is used to elicit potential privacy threats and corresponding privacy protection requirements. It addresses part of a more comprehensive Privacy-by-Design process by supporting privacy impact analysis. First a data flow diagram of the system is created using four major types of building blocks: entities, data stores, data flows, and processes. For each building block the threats of the corresponding threat categories have to be examined.

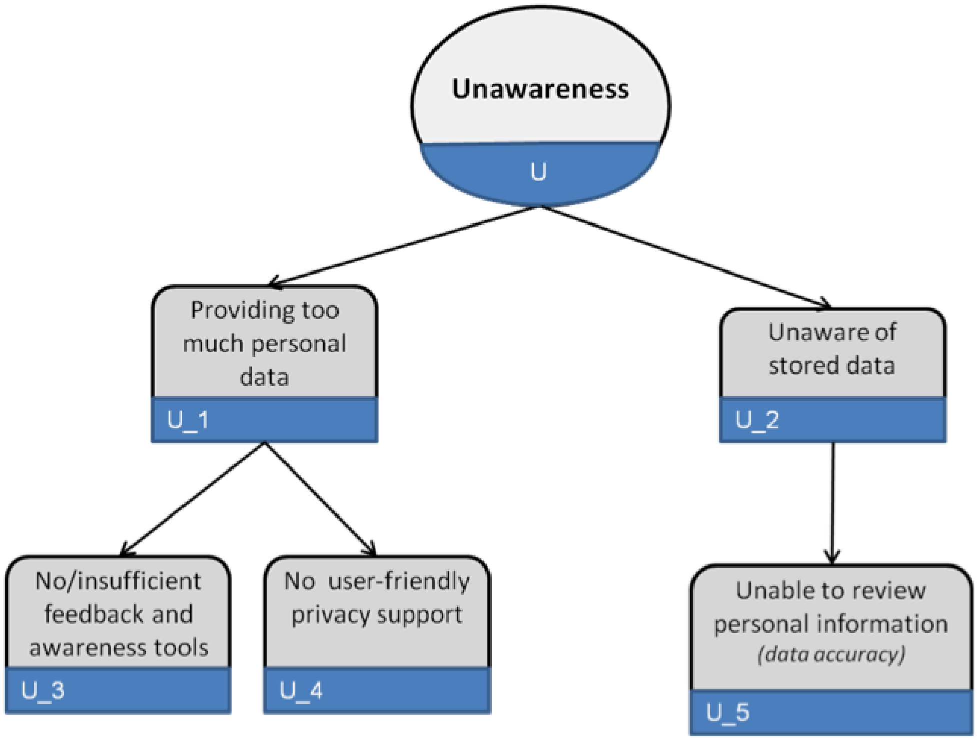

Unawareness
U
Providing too much personal data
U_1
Unaware of stored data
U_2
No/insufficient feedback and awareness tools
U_3
No user-friendly privacy support
U_4
Unable to review personal information (data accuracy)
U_5

Figure 4: LINDDUN unawareness of entity threat tree [236]

The LINDDUN privacy threat categories are:

Linkability: Not being able to hide the link between two or more actions/identities/pieces of information.

Identifiability: Not being able to hide the link between the identity and an action or information.

Non-repudiation: Not being able to deny a claim.

Detectability: Being able to distinguish sufficiently whether an item of interest exists or not.

Information Disclosure: Same as in Microsoft STRIDE (see above).

Unawareness: Being unaware of the consequences of sharing information.

Non-compliance: Not being compliant with legislation, regulations, and corporate policies.

RERUM has deployed LINDDUN privacy threat categories as part of a privacy impact analysis in the context of our use cases (see Deliverable 2.1, Section 3.8) and found it very useful to discuss potential threats in our use case scenarios.

A set of threat trees (example see Figure 4) is provided which describe the most common attack paths for each possible combination of a threat type and a data flow diagram of a building block type. The analyst with the help of misuse case scenarios describes the possible attacks in detail. The identified privacy threats then need to be rated and translated into privacy requirements.

Privacy threat catalogues may also help to identify and refine threats, as well as to detail attacks and to rate the probability of threats materialising. For instance there is one catalogue being compiled by the OWASP (Open Web Application Security Project) initiated privacy project [217] dating from 2014. This project uses the OECD Privacy Guidelines. That threat catalogue for instance comprises software vulnerabilities, operator-side data leakage, insufficient data breach response, insufficient data deletion, non-transparent terms and conditions, collection of non-necessary data, sharing with third parties, outdated data, insufficient session expiration and insecure data transfer.

Besides LINDDUN, there are many other initiatives for the development of a privacy impact assessment methodology in Europe, like in the EC-funded PIAF [68], PRESCIENT [117], and SAPIENT [89]). These methods currently focus traditional client-server applications. They assume voluntary and explicit data disclosure by data subjects who are application users. IoT scenarios however may involve surveillance of (potentially involuntary) non-user data subjects (e.g. by non-personal sensors). Data analytics and data fusion facilities are getting more sophisticated, and application-external data and knowledge sources are to be taken into consideration. There may be sensor data that in certain circumstances can (post-collection) be aligned to individuals. In LINDDUN, e.g. threat category "identifiability" may cover such scenarios.

2.5 Privacy engineering: The PRIPARE methodology

The EU Project PRIPARE [220] (PReparing Industry to Privacy-by-design by supporting its Application in REsearch) specifies a privacy and security-by-design systems engineering methodology. An important aspect is taking into account the European dimension, like to achieve compliance with the upcoming EU general data protection resolution. They have published a first version of their methodology [92] in 11/2013.

Their notion of Privacy-by-Design also involves applying a set of privacy principles from the earliest conception phases of an information and communication technology (ICT) system in order to mitigate security and privacy concerns throughout the development and operation of that system. The PRIPARE privacy principles are

1. **Data quality:** Safeguarding the quality of personal data. Data should be accurate and, where necessary, kept up to date.
2. **Data minimisation and proportionality:** Limit the processing data and ensure data avoidance. Only adequate and relevant personal data is processed.
3. **Purpose specification and limitation:** Personal data must be collected for specified, explicit and legitimate purposes, and not further processed in a way incompatible with those purposes. This is referred to as the "finality principle".
4. **Purpose specification and limitation specific for sensitive data:** Legitimacy of processing sensitive personal data must be ensured either by basing data processing on explicit consent, or a special legal basis.
5. **Transparency and openness:** Compliance with the data subject's right to be informed.
6. **Right of access:** It must be ensured that the data subject's wish to access, rectify, erase and block his/her data is fulfilled in a timely manner.

7. **Right to object:** Facilitating the objection to the processing of personal data, direct marketing activities, and disclosure of data to third parties.
8. **Confidentiality and security of processing:** Preventing unauthorised access, logging of data processing, network and transport security and preventing accidental loss of data.
9. **Notification obligations of the supervisory authority:** Notification about data processing, prior compliance checking and documentation needs to be ensured.
10. **Limited retention:** Retention of data should be for the minimum period of time consistent with the purpose of the retention or other legal requirements.
11. **Accountability:** Demonstrable acknowledgement and responsibility for having in place appropriate policies and procedures, including correction and remedy for failures and misconduct.
12. **Right to erasure:** Require the data controller to take all reasonable steps to have individuals' data erased, including by third parties without delay, for the personal data that the controller has made public without legal justification.
13. **Privacy-by-Design:** Data protection is to be embedded within the entire life cycle of the technology, from very early design stage, right through to its ultimate deployment, use and final disposal.
14. **Privacy by default:** Requires data subjects' control on the distribution of their personal data and explicit consent each time personal data processing is intended. Preferences by default must be set to their most privacy-preserving configuration.

PRIPARE references the EU data protection directive and the upcoming EU general data protection resolution principles, complemented with some of the security principles identified by OWASP. The PRIPARE set of principles is open for supplementation. The ISO 29100 idea of guiding the transformation of high level privacy principles into privacy controls is not only followed by the OASIS Privacy Management Reference Model [202], but also recommended by PRIPARE, which provides a step by step methodology to allow transforming these high-level principles into an actual system implementation and operation.

The privacy engineering process proposed by this methodology involves loops and feedback cycles just like traditional system and security engineering life cycles. This may well require re-defining and re-engineering the system in case of detection of a violation of privacy principles. The horizontal iterative approach involves to start with an initial architecture and follow an iterative process refining it stepwise order to achieve the desired privacy objectives, while taking the other (potentially conflicting) system requirements into account, carefully considering the trade-offs and cost-benefit on each alteration.

Among many other things the PRIPARE methodology strongly advocates various types of privacy impact analysis and assessment throughout the system's life cycle to identify the privacy requirements, vulnerabilities, risks and to define the measures to prevent those risks becoming a reality. The methodology covers the complete life cycle of the system, before its inception (covering organisational aspects) and until its decommission, including accountability aspects.

The PRIPARE Privacy-by-Design engineering methodology (see Figure 5) involves a much broader process than a privacy threat analysis model (like LINDDUN), a privacy impact analysis methodology as proposed by the EU project PIAF [68] (Privacy Impact Assessment Frameworks) or the ISO/IEC WD 29134 [121] (Privacy Impact Assessment Methodology). It includes those steps, and others, like practices for selecting

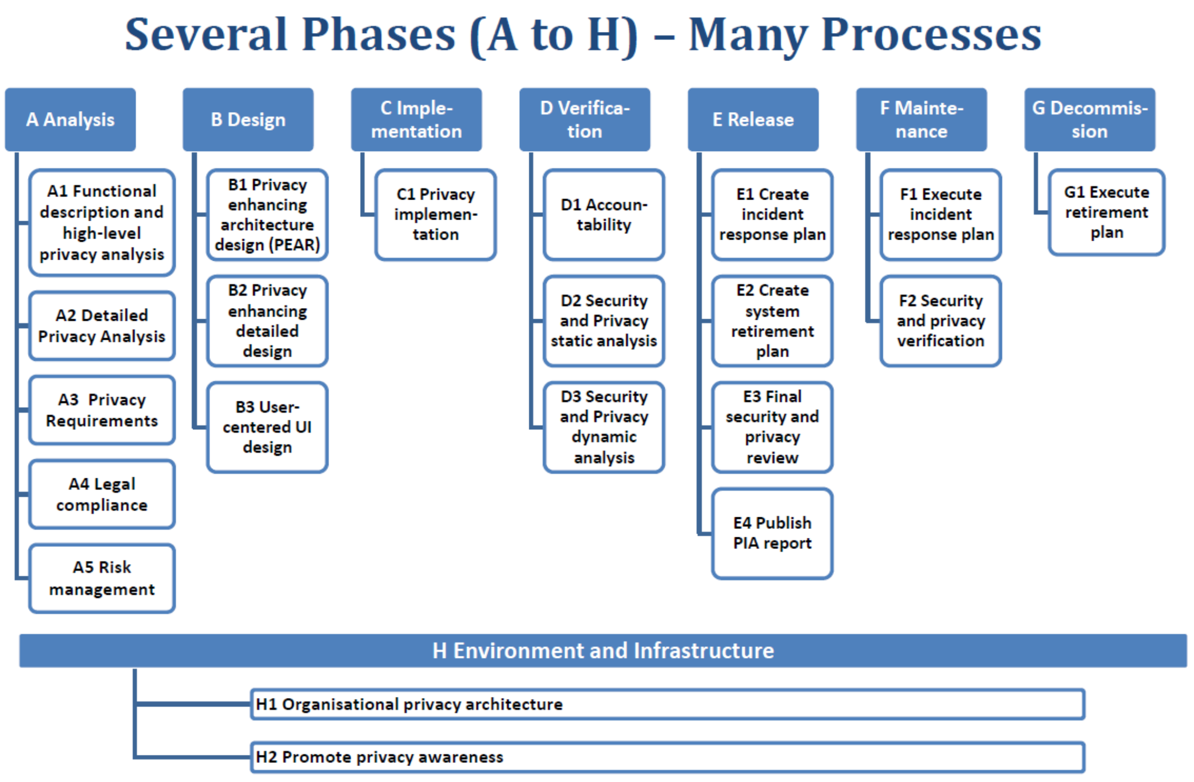

Several Phases (A to H) – Many Processes
A Analysis
A1 Functional description and high-level privacy analysis
A2 Detailed Privacy Analysis
A3 Privacy Requirements
A4 Legal compliance
A5 Risk management
B Design
B1 Privacy enhancing architecture design (PEAR)
B2 Privacy enhancing detailed design
B3 User-centered UI design
C Implementation
C1 Privacy implementation
D Verification
D1 Accountability
D2 Security and Privacy static analysis
D3 Security and Privacy dynamic analysis
E Release
E1 Create incident response plan
E2 Create system retirement plan
E3 Final security and privacy review
E4 Publish PIA report
F Maintenance
F1 Execute incident response plan
F2 Security and privacy verification
G Decommission
G1 Execute retirement plan
H Environment and Infrastructure
H1 Organisational privacy architecture
H2 Promote privacy awareness

Figure 5: PRIPARE privacy engineering methodology

privacy controls as proposed by the ISO 29151 [122] (Code of Practice for Personally Identifiable Information Protection; extends ISO 27002).

Being applicable to all types and sizes of organisations is one of the aims of the PRIPARE methodology. RERUM has started to cooperate with PRIPARE to discuss the usability of their privacy engineering methods within RERUM. However their complexity and time and resource requirements render them impractical for full application within the implementation of the RERUM use cases. Still, RERUM not only shares many of PRIPARE's privacy principles as well as their principle-to-control Privacy-by-Design approach. Also their approach of iteratively improving the initial privacy concept and architecture design is naturally applied in RERUM, where in D2.1, D2.2, and D2.3 we have given an initial privacy design, which is improved and refined in this deliverable based on our RERUM use cases.

2.6 ``Hard'' and ``soft'' privacy controls

Privacy enhancing technologies (PETs) enable data subjects to preserve their privacy in traditional eCommerce as well as in various IoT contexts such as smart metering, electronic traffic pricing, ubiquitous computing or location based services. Data subjects may for instance want to avoid mass data collection and linkability. Against whom may data subjects require protection of their privacy? Data subjects may wish for protection of their privacy against third parties, but still be willing to place trust in their data controller/processor (**“soft privacy”**, personal data management, privacy-supporting controls). Data subjects can also wish for privacy protection against the data controller/processor (**”hard privacy”**, privacy-enforcing controls).

2.6.1 ``Hard'' privacy controls

Privacy can be protected by “hard” measures that allow the data subject to determine which personal data are collected and allowed beyond the data subject’s sphere of control. Data minimisation, local processing, and blocking of data transmission are basic controls here. They prevent abuse by preventing disclosure. Their goal is to enable users with means to enforce their privacy preferences. Local data blockers (ads, pop-ups, ...) may be helpful, as may be the use of trusted hardware, like Trusted Platform Module (TPM) and Hardware Security Module (HSM), or secure Multi-Party Computation (MPC) to process data.

2.6.1.1 Data minimisation

PETs aim at reducing the amount and quality of data disclosed, like by reducing the granularity and adding noise to the sensor data. Location privacy in particular refers to blurring and hiding the exact location of the data subject. Processing and using the data locally in sensor-actor configurations avoids the need for data disclosure, so operation without keeping log and history data reduces the surveillance potential to a great extent.

2.6.1.2 Data anonymization

This refers to the removal of identifiers, adding noise,et cetera. Data anonymization aims to allow for de-personalisation of personal data. But in certain situations and with the aid of data fusion, re-personalisation and linking often may still be possible. This especially applies to meta-data. Often inferring information about individuals remains possible despite anonymization of the various raw source data.

2.6.1.3 Anonymous credentials

This measure should provide completeness and soundness (be convincing and reliable), and involve zero-knowledge and unlinkability. Optionally anonymous credentials could allow for revocation, linkability, partial shows, and re-identification (e.g. in the case of fraud). There is other privacy-preserving cryptography, like blind or redactable signatures.

There of course are many other types and aspects of hard privacy preserving controls. Truly anonymous communication requires also protection against traffic analysis, like via mixing (e.g. onion routing). Privacy preserving access control for instance may involve attribute certificates and private authentication.

2.6.2 ``Soft'' privacy controls

In many cases privacy protection must rely on the compliance and cooperation of the data controller and processor, i.e. the services receiving the data subject's personal data. Privacy compliant behaviour can here be improved by "soft" measures that require trust in integrity and honesty of the data recipient, the organisation that holds the personal data. These controls cannot guarantee privacy, they do not offer protection against misbehaving data recipients. Aim is to allow for compliant behaviour.

2.6.2.1 Data management

PETs for data management can be applied before, but mainly once personal data has been disclosed. Data subjects should be offered access, modification and deletion rights for their personal data. This may involve policy, feedback, and data removal tools.

2.6.2.2 Decision support

PETs for decision support enable data subjects to form well-informed decisions. Data subjects must be provided oversight over the collection, processing, and use of their personal data. This information helps the data subject understand and decide. Examples from traditional eCommerce are:

- Google Dashboard [102]: what personal data is stored and who has access
- Firefox Lightbeam [224]: list of entities tracking users
- Mozilla Privacy Icons [195]: simple visual language to make privacy policies more understandable
- IE Privacy Bird [58]: shows user whether web page complies with preferred policy based on images

2.6.2.3 Consent support

PETs for consent support provide users with means to express their privacy preferences and give consent. The data subject may define appropriate data usage and privacy preferences. Data controllers and data subjects may proclaim privacy policies which even may be machine processable. Privacy policies can be attached to personal data. This allows honest recipients to adhere to the data subject's preferences. Knowing the data subject's privacy preferences helps the data controller/processor to act compliant and responsible. "Sticky policies" associated to personal data may ask trusted third parties to disclose encryption keys only in certain cases. Consent supportive examples from traditional eCommerce are:

- **Privacy policies languages (P3P, S4P, SIMPL):** Automated or semi-automated processing and comparison with users preferences
- **Do-not-track browser options and plugins:** Anti-tracking declarations

2.6.2.4 Accountability support

PETs for accountability support improve data controllers' ability to demonstrate compliance. Personal data needs to be confidential and may not be disclosed to unauthorised parties. Data controllers may wish to prove, that they, for instance, have not communicated personal data to external recipients. Audits and certification can be helpful to inspire data subject confidence. Logs need to be non-repudiable (backuped, distributed, ...). This applies to all logs, and includes forward integrity, as well as tools for log audits.

2.7 RERUM Privacy-by-Design requirements

In this subsection we define our understanding of "Privacy-by-Design" in RERUM. As also proclaimed by the Canadian Privacy-by-Design principles, we share the opinion that any Privacy-by-Design approach must offer *end-to-end coverage* from system inception until final destruction of all raw and processed personal data. The citizen is at the center of RERUM's attention. As demanded in the Canadian Privacy-by-Design principles, primary focus of the privacy protection efforts must be the data subject and the *data subject's interests*. In IoT scenarios often data subjects (whose personal data are recorded by IoT sensors) are not (conscious, known, and authenticated) users of the application or system.

The eight RERUM IoT privacy requirements specified in D2.2 (Section 2.6.3) have been based on the terminology and the eleven privacy principles proposed in ISO 29100 (see Section 2.2.2 and (see Table 1). They were complemented by the terminology and privacy principles used in the EU Directives on conventional personal data, like the European Directive 95/46/EC on the protection of individuals with regard to the processing of personal data, the European Directive 2002/58/EC / 2009/13/6/EC concerning processing of personal data and protection of privacy in the electronic communications sector. They were as follows:

1. Consent and choice (also possibility of subsequent withdrawal)
2. Purpose legitimacy and specification
3. Collection limitation (adequate, relevant and not excessive)
4. Data minimisation
5. Accuracy and quality (delete or rectify incorrect data)
6. Notice and access (of/to collected and processed data)
7. Individual participation & transparency (user can activate/deactivate collection)
8. Accountability (of the person responsible for privacy breaches)

Resulting from the analysis in this section, we propose to add/refine the following privacy principles to our RERUM Privacy-by-Design understanding:

- **Privacy-by-default:** Privacy-friendly default settings, as introduced in the Canadian principles and repeated by ENISA and PRIPARE.
- **Data minimisation** This principle comprises many different aspects. We want to explicitly consider the following ones:
  - Pseudonymous and, better, anonymous application and system use is preferred.
  - Granularity of recorded sensor data are to be kept as coarse as possible. If the city quarter is enough, it is not necessary to collect/transmit or process a more exact location.
  - Earliest possible data aggregation, de-personalisation and anonymization of sensor data, obscuring the data subject relationship as early and as much as possible. Local ephemeral sensor-actor constellations without external communication are preferable.

2.8 RERUM privacy glossary

In RERUM, as EU project, we mainly use European privacy terminology. The following glossary is to ensure a consistent use of privacy terms throughout this deliverable. As basis we use the traditional privacy glossary provided by the European Data Protection Supervisor [40]. RERUM has aligned also with PRIPARE's use of privacy terms [93]. We enhance these definitions with IoT specific aspects as well as additional terms arising in the IoT context.

Accountability: Accountability is a basic privacy principle. The entity collecting or storing personal data must explicitly acknowledge and be able to demonstrate the privacy effort and assume the responsibility for having in place appropriate policies and procedures, and promotion of good practices that include correction and remedy for failures and misconduct. Accountability requires that data controllers put in place internal mechanisms and control systems that ensure compliance and provide evidence – such as audit reports – to demonstrate compliance to external stakeholders, including supervisory authorities.

Anonymity: Anonymity is the characteristic of information that does not permit a data subject to be identified directly or indirectly. Anonymous data cannot be related to a specific person and are consequently normally not regarded as personal data, so controllers and processors are exempt from applying the principles of personal data protection. However there are de-anonymization risks, especially in IoT contexts and with sensor data and data fusion.

ANR: ANR stands for Attribute Need Reporter. It is a component that constructs an initial list of attributes needed by the security and privacy policies so it is possible to ask for all of them in a single operation. It works jointly with the IdA and PPC to cache these values and ensure that only granted attributes are really accessed.

Authoriser: In RERUM, an authoriser is a SW component that is responsible for evaluating whether a given request is granted to be executed or not

IdA: IdA stands for Identity Agent. It is a component defined in D3.1 responsible for gathering the information on the user that will be used in the authorisation process. It works jointly with the ANR and the PPC to ensure that only relevant and granted attributes of the user are retrieved.

Identity Provider: An Identity Provider ia a piece of SW, normally hosted by an external trusted party, which is responsible for verifying the identity of the RERUM registered user and providing the value of the attributes associated to him.

Interceptor: A piece of software that intercepts incoming requests. It is normally used jointly with some other component, such an authoriser, to let it make operations on the request, such as accepting or rejecting it.

Choice: Consent needs to be voluntary and informed. True voluntary consent prerequisites viable alternatives to choose from.

(Privacy) Compliance: Data controller must ensure and be able prove processing meets data protection and privacy requirements by periodical audits; includes privacy risk assessments.

Consent: refers to any freely and unambiguously given, specific, and well informed indication of the wishes of a data subject, by which he/she agrees to personal data relating to him/her being processed. It is one of the conditions that can legitimise processing of personal data, according to the EU Data Protection Directive. The obtained consent can only be used for the specific processing operation for which it was collected, and may in principle be withdrawn without retroactive effect.

Data controller: is the institution or body that determines the purposes and means of the processing of personal data. In particular, the controller has the duties of ensuring the quality of data and, in the case of the EU institutions and bodies, of notifying the processing operation to the data protection officer. It is responsible for the security measures protecting the data and receives requests from data subjects to exercise their rights.

Data minimisation: is a basic privacy principle principle; it means that a data controller should limit the collection and processing of personal information to what is directly relevant and necessary to accomplish a specified purpose. Controllers should also retain the data only for as long as is necessary to fulfil that purpose. The EU directive states that personal data must be "collected for specified, explicit and legitimate purposes" and must be "adequate, relevant and not excessive in relation to the purposes for which they are collected and/or further processed".

Data quality: involves a set of principles and several different aspects. Originally it was implying that personal data must be accurate and where necessary kept up to date and processed lawfully, collected for specified, explicit and legitimate purposes only. The data subject has the right to request correction of incorrect personal data. Other quality aspects include that data must be kept in a form which does not allow identification of data subjects, if possible, or permits identification of data subjects for no longer than is necessary for the original purpose. Furthermore, high data quality may not always be preferable from privacy considerations at all. By reducing data quality, i.e. by lowering the resolution, privacy may be improved.

Data source: The entity mechanisms, or process where potentially privacy sensitive data are generated or stored, and can be retrieved from.

Data subject: human person whose personal data are collected, held or processed.

Data transfer: transmission / communication of data to a recipient in whatever way; should according to EU legislation be necessary for the legitimate performance of the purpose; subject to specific safeguards when the recipient is located in a country outside the EU (e.g. Safe Harbour scheme).

Detectability: Being able to distinguish sufficiently whether an item of interest exists or not (LINDDUN).

Further processing: involves personal data initially collected for an explicit purpose and re-used at a later time for purposes that are incompatible with the initial purpose (secondary use).

Hard privacy: If data subjects require privacy protection against the data controller/processor, they need measures that allow the data subject to determine which personal data are collected and allowed beyond the data subject's sphere of control. They prevent abuse by preventing disclosure. Data minimisation and blocking of data transmission are examples here.

Identifiability: Not being able to hide the link between the identity and an action or information (LINDDUN).

Information Disclosure: Same as in Microsoft STRIDE (LINDDUN).

Lawfulness: (a) unambiguous consent of data subject, (b) part of contract, (c) legal basis, (d) medical emergency, (e) public interest, or (f) legitimate interests not overridden by fundamental rights and freedoms of the data subject. (ENISA)

Legitimate interest: when the controller's interest in processing the data outweighs the data subject's interest in not processing the data.

Linkability: Not being able to hide the link between two or more actions/identities/pieces of information (LINDDUN).

Notice: Notice requires the data subject being given timely and clear notice of all relevant facts pertaining to the intended data processing and disclosure.

Notification: A notification is an action carried out by controllers to inform the data subject and/or Privacy Commission that they will be processing data; mainly consists of a description of the data processing operation.

Non-repudiation: Not being able to deny a claim (LINDDUN).

Non-compliance: Not being compliant with legislation, regulations, and corporate policies (LINDDUN).

Opt in (consent): Prior explicit consent is required before any data collection and processing.

Opt out (consent): Allows data subject to object to data processing. "withdrawal" of previously "assumed" only consent.

Personal data: any information relating to an identified or identifiable natural person, referred to as "data subject" - an identifiable person is someone who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his or her physical, physiological, mental, economic, cultural or social identity. In IoT contexts sensor data may also qualify as personal data and there may be data, that will not be identifiable as personal data at the time of collection, but only after data fusion.

PEP: Stands from Policy Enforcement Point: It is a piece of software that gathers the information contained on a request and let pass or reject the request. It works together with a PRP for holding the information gathered and a PDP for deciding whether to grant or not the request.

PDP: Stands for Policy Decision Point: It is a piece of component that decides whether a concrete request with its specific content should be granted or not. It works jointly with a PRP to obtain the criteria applicable for taking the decision

Policy Store: It is a software artifact for storing the security criteria for accessing the system

PPEP: Stands for Privacy Policy Enforcement Point: specific type of PEP that evaluates only privacy policies

PPC: Stands for Privacy Policy Checker is an Authoriser that runs against a specific set of privacy policies that are specialized on checking privacy for those user attributes that are used in the authorisation decisions. It works jointly with the IdA to ensure that only granted user attributes are retrieved.

PRP: Stands for Policy Retrieval Point: Is the component responsible retrieval for selecting those policies that are applicable for a given request. It is used jointly with the PDP to decide whether to grant or reject access to the that request.

Privacy: ability of an individual to be left alone, out of public view, and in control of information about oneself. The concept of privacy overlaps, but does not coincide, with the concept of data protection. The right to privacy is protected in the Universal Declaration of Human Rights (Article 12) as well as in the European Convention of Human Rights (Article 8). Finn, Wright and Friedewald distinguish seven "types of privacy": physical person, behaviour and action, communication, data and image, thoughts and feelings, location and space, and association (including group privacy).

Privacy-by-Design: aims at building privacy and data protection up front, into the design specifications and architecture of information and communication systems and technologies, in order to facilitate compliance with privacy and data protection principles.

Privacy by default: aims at delivering system the default settings of which are privacy respecting. So the data subject ideally does not have to take any explicit configuration steps to ensure privacy protection.

PETs (Privacy Enhancing Technologies) : refers to a coherent system of ICT measures that protect privacy by eliminating or reducing personal data or by preventing unnecessary and/or undesired processing of personal data, all without losing the functionality of the information system. The use of PETs can help to design information and communication systems and services in a way that minimizes the collection and use of personal data and facilitates compliance with data protection rules.

Privacy Impact Assessment: An analysis of how information is handled: (i) to ensure handling conforms to applicable legal, regulatory and policy requirements regarding privacy; (ii) to determine the risks and effects of collecting, maintaining and disseminating information in identifiable form in an electronic information system, and (iii) to examine and evaluate protections and alternative processes for handling information to mitigate potential privacy risks.

Privacy Policy: overall intention and direction, rules and commitment, as formally expressed by the data controller related to the processing of personal data in a particular setting; advises employees on the collection and the use of the data, as well as data subjects on any specific rights they may have.

Privacy principles: set of shared values governing the privacy protection of personal data when processed in information and communication technology systems.

Privacy preferences: specific choices made by a data subject about how their personal data should be processed for a particular purpose.

Processing (of personal data): any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, etc.

(Data) Processor: natural person, legal person, organisation or public authority processing data on behalf of the controller, except for individuals who are under the direct authority of the controller and who have been authorised to process the data. Transfers of personal data from a data controller to a data processor must be secured by a data processor agreement.

Pseudonymity: ensures that a user may use a resource or service without disclosing its identity, but can still be accountable for that use. It uses pseudonyms as identifiers being another than the subject's real name.

Purpose: specific reason why the personal data are collected and processed. Personal data can only be collected for a specific, explicitly stated purpose for which the user has provided consent.

Purpose binding: Personal data obtained for one purpose must not be processed for other purposes that are not compatible with the original purpose.

Safe Harbour Principles: in consultation with the European Commission, the American Department of Commerce has elaborated the Safe Harbour Principles, intended to facilitate the transfer of personal data from the European Union to the United States. If companies declare to respect these principles in a statement to the American Department of Commerce, they are considered as companies ensuring adequate safeguards for data protection.

Right of access, information, rectification, deletion, and objection: right of access for any data subject to obtain from the controller of a processing operation the confirmation that data related to him/her are being processed, the purpose(s) for which they are processed, as well as the logic involved in any automated decision process concerning him or her.Everyone has the right to know that their personal data are processed and for which purpose. The right to be informed is essential because it determines the exercise of other rights. The right of information refers to the information which shall be provided to a data subject whether or not the data have been obtained from the data subject. The right of rectification is the right to obtain from the controller the rectification without delay of inaccurate or incomplete personal data. The right to object has two meanings. First, it is the basic right of any data subject to object to the processing of data relating to him or her. Second, it is the specific right of any data subject to be informed, free of charge, before personal data are first disclosed to third parties or before they are used on their behalf for the purposes of direct marketing, and to object to such use without justification.

request: The act of a piece of sotware contacting an external service to execute its associated functionality.

RERUM registered user: Any entity registered in RERUM to identify the requester whom the request is issued.

Sensitive data: Certain personal data are more sensitive than others. An individual's name and address are rather innocent data, but this does not hold true for his political opinions, sexual preferences or judicial past. The EU Privacy Act regulates registration and use of those sensitive data more strictly in comparison with other personal data. Sensitive data traditionally relate to race, political opinions, religious or philosophical beliefs, trade union membership, health, sex life, suspicions, persecutions and criminal or administrative convictions. In principle, processing such data is prohibited. In IoT contexts for instance sensor data and subsequent data fusion and analytics may be used to derive equivalent data about the data subject. Thus there may be reasons to regard sensor data not only as personal data, but even as sensitive data.

Security Policy: A file containing access criteria for RERUM

Session: A particular time period that starts when a given entity tries to access a system till it is considered ot have left it

Soft privacy: Data subjects may want to request protection of their privacy against third parties, but still be willing place trust in their data controller/processor. Privacy-supporting controls require compliance and cooperation of the data controller. They aim to allow for and support compliant behavior, but they do not offer protection against misbehaving. Example technologies are policy, feedback, and data removal tools as well as privacy icons and dashboards.

SPEP: Stands for Security Policy Enforcement Point. It is a PEP that is specialized in evaluating access criteria that do not have to do with Privacy.

Traffic data: Traffic data are data processed for the purpose of the conveyance of a communication on an electronic communications network. According to the means of communication used, the data needed to convey the communication will vary, but may typically include contact details, time and location data. Although such traffic data are to be distinguished from content data, both are quite sensitive as they give insight in confidential communications. These data therefore enjoy special protection in Articles 5 and 6 of the E-privacy Directive 2009/136/EC and Articles 36 and 37 of Regulation (EC) No 45/2001.

Transparency: Data subjects need sufficient information about data collection and use, understands risks and control actions they can take. They require means to find out the existence and nature of personal data, and the main purposes of their use, identity and residence of the data controller, options for influencing processing, and information about major processing changes.

Unawareness: Being unaware of the consequences of sharing information (LINDDUN).

Unlinkability: ISO 15408 defines that unlinkability ensures a user may make multiple uses of resources or services without others being able to link these uses together.

Use limitation: Personal data should not be disclosed, made available or otherwise used for purposes other than those specified except with the consent of the data subject; or by the authority of law (OECD).

Unobservability: (or Undetectability) ISO 15408 defines that unobservability ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used.

XACML: Stands for eXtended Access Control Markup Language: It is a language defined by OASIS standardisation body to define formally access criteria in the internet in a file named policy

3 RERUM privacy enhancing components

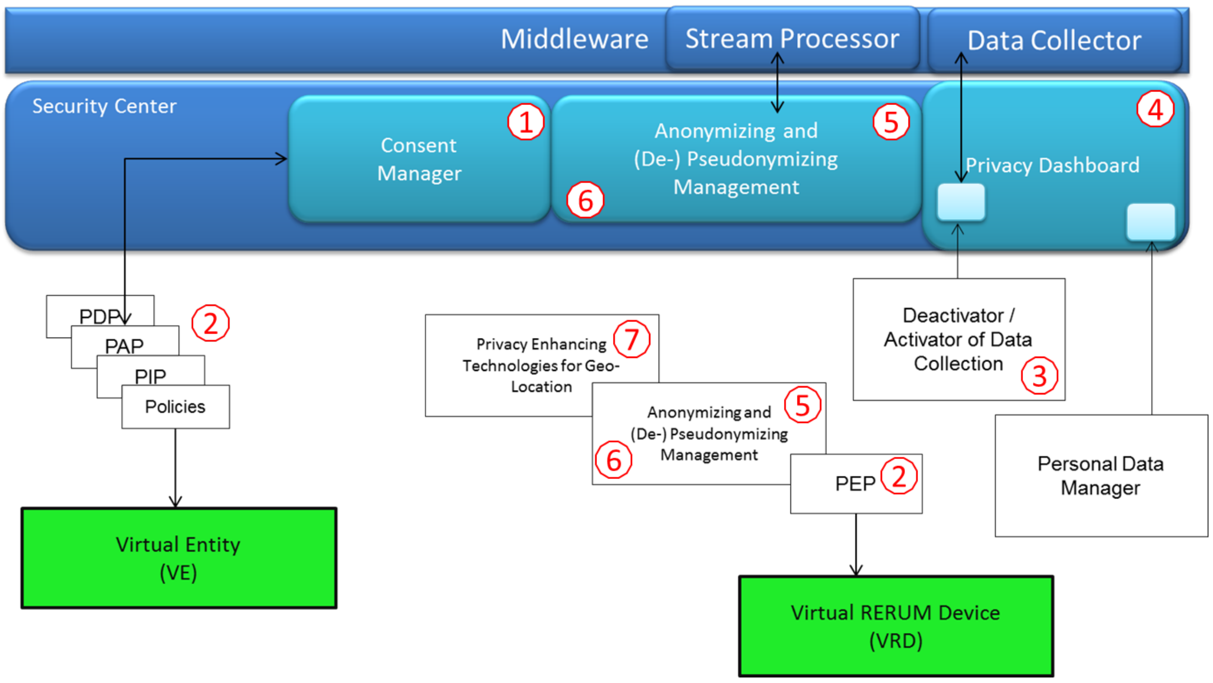

Middleware
Stream Processor
Data Collector
Security Center
Consent Manager
1
Anonymizing and (De-) Pseudonymizing Management
5
6
Privacy Dashboard
4
PDP
PAP
PIP
Policies
2
Privacy Enhancing Technologies for Geo-Location
7
Anonymizing and (De-) Pseudonymizing Management
5
6
Deactivator / Activator of Data Collection
3
PEP
2
Personal Data Manager
Virtual Entity (VE)
Virtual RERUM Device (VRD)

Figure 6: RERUM Privacy Functional Components (from D2.3)

This chapter is dedicated to the detailed specification of the seven privacy related functional components from D2.3 [219] (see Figure 6). We also summarise several security components from D3.1 [201] needed as privacy basis. Finally we introduce two newly conceived privacy components.

First we specify in detail all those privacy components briefly sketched in D2.3 (see Figure 6):

(1) User Consent Manager (from D2.3, Section 6.11.2.1): Section 3.1.

(2) Privacy Policy Enforcement Point: (pPEP, from D2.3, Section 6.11.2.2): Section 3.2; this section details the Privacy PEP (D2.3/6.11.2.2), PEP (D2.3/6.11.1.5), PDP (D2.3/6.11.1.6), and PRP (D2.3/6.11.1.7) and their interworking. This involves reuse of the access control authorisation components (from D2.3, Section 4.3) as detailed in Section 4.4.

(3) Deactivator / Activator Data Collection (from D2.3, Section 6.11.2.3): Section 3.3.

(4) Privacy Dashboard (from D2.3, Section 6.11.2.4): Section 3.4.

(5) Anonymizing and Pseudonymizing Management (from D2.3, Section 6.11.2.5): Section 3.5.

(6) De-Pseudonymizer (from D2.3, Section 6.11.2.6): Section 3.6.

(7) PET for Geo-Location (from D2.3, Section 6.11.2.7): Section 3.7; special component to deal with certain privacy problems from collection of location information in Section 4.8.

In Section 3.8 we summarise three security components described in RERUM Deliverable D3.1 [201] together with a short description of how they may enhance privacy:

(8.1) Data Encrypter / Decrypter (from D2.3, Section 6.11.1.2): Section 3.8.1.

(8.2) D2D Authenticator (from D2.3 Section, 6.11.1.3): Section 3.8.2.

(8.3) Credential Bootstrapping Client / Authority (from D2.3, Section 6.11.1.4) and **Trusted Credential Storage** (from D2.3, Section 6.11.1.8): Section 3.8.3; cryptographic components need to protect key material (like secret key confidentiality and access control).

Finally we propose and specify two additional privacy components not mentioned already in D2.3, partially derived from security components sketched in D2.3 and detailed in D3.1 and D2.5:

(9) Integrity Generator / Verifier (from D2.3, Section 6.11.1.1 with details in D3.1 [201]): Section 3.9; this component benefits from Trusted Credential Storage build on Secure Storage (from D2.3, Section 6.11.1.8) and needs Credential Bootstrapping Client / Authority (from D2.3, Section 6.11.1.4).

(10) Privacy Policy Checker and Attribute Need Reporter (from D2.5, [157]): Section 3.10; we explain how the **Attribute Need Reporter** (computes the user attributes potentially needed) and **Privacy Policy Checker** (ensures access control to those user attributes) work jointly with the IdA from D3.1 [201] to enrich the authorisation process with privacy features.

Table 2: Privacy component novelty and technical readiness summary template

–Name of functional component–		
Technical level	Level 1,2,3	–short description of components state justifying the indicated level–
Suggested Method(s) for Implementation	–Name of suggested method(s)–	
	–Link to method(s) (extern or within deliverable)–	
Technical readiness of implementation within RERUM	design	–yes/no–
	experiments	–yes/no–
	trial	–yes/no–

Component Classification Scheme: Using the template shown in Table 2, at the end of each component description we discuss potential mechanisms to achieve this component's functionality and indicate the current and planned technical level of implementation within RERUM and the novelty of this component. We define three novelty 'levels':

Level 1: *An implementation for this component already exists and can be integrated without modification.* Here, we offer a link and explain at least one existing technology and how it can be integrated into the RERUM framework to achieve the privacy functionality.

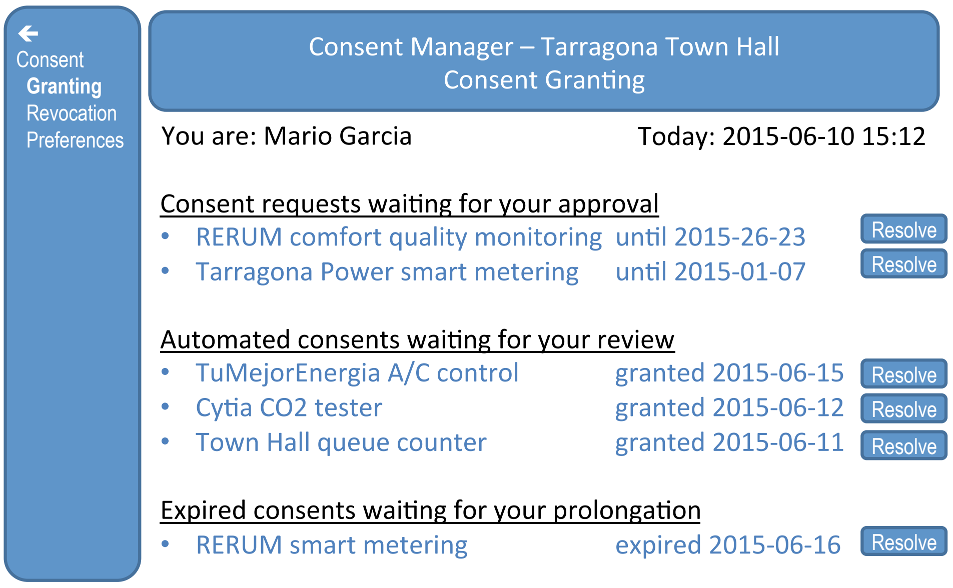
Level 2: *This component already exists but RERUM adapted it to be integrated technically into the RERUM Framework*. For more details the reader is referred to to a dedicated section in Chapter 4 of this deliverable.

Level 3: *Such a component did not exist as such for the IoT-domain*. For more details the reader is referred to a dedicated section in Chapter 4 of this deliverable.

3.1 User Consent Manager

When personal data are collected, generated, stored, and processed, a preceding consent from the data subject is needed, for instance because the law (in the EU) or fairness and good business conduct (in the US) require it. Service providers (data controllers) and other data processing parties (data processors) must clearly and lawfully explain the data collection purpose to the user (data subject). After a successful consent confirmation by the data subject (user), the application (data controller) may collect the specified personal data and to process them as described in the consent content. A request for consent may involve negotiations with the data controller and selection of options by the data subject to concretise the consent content. In IoT a data controller may rely on services coupled with a physical entity, e.g. manufacturer services of smart wear, as well as services using own custom-setup or even pre-existing infrastructures of smart things. This must be described clearly in the request for consent.

The RERUM "Consent Manager", as sketched in D2.3 (Section 6.11.2.1), is responsible to interact with the data subject (the user) to display the application's request for consent to the user, if need be, to assist with negotiations and option selection, and to obtain the user's consent (or refusal to grant consent). The data controller must explain the purpose of data collection to the data subject clearly and lawfully. Therefore the RERUM "Consent Manager" must be accessed on a device with advanced graphical user interface or with audio capabilities to display the consent content and to interact with the user. At the consent manager data controllers can register their requests for consent and users can give their informed consent in form of a mouse-click, a touch on their smartphone or otherwise (see Figure 7). The "Consent Manager" is a centralised point (per RERUM IoT infrastructure / middleware installation). Consent content and the sets of consents of a data subject usually represent privacy sensitive information. This is one of the reasons not to have a central single private consent manager per data subject that might be compromised and misused.



Consent Manager – Tarragona Town Hall
Consent Granting

You are: Mario Garcia Today: 2015-06-10 15:12

Consent requests waiting for your approval
- RERUM comfort quality monitoring until 2015-26-23
- Tarragona Power smart metering until 2015-01-07

Automated consents waiting for your review
- TuMejorEnergia A/C control granted 2015-06-15
- Cytia CO2 tester granted 2015-06-12
- Town Hall queue counter granted 2015-06-11

Expired consents waiting for your prolongation
- RERUM smart metering expired 2015-06-16

Figure 7: Consent Manager: sample consent granting screen

Note: In eCommerce scenarios the data subject normally is a conscious user of the data controller's application. In IoT scenarios numerous sensors are present in the data subject's personal environment and devices; data collection often is very unobtrusive. It may happen that IoT applications collect personal sensor data about humans that are not conscious IoT users. They have not consented to the collection of their sensitive personal data. If the data subject cannot be asked for consent (not even post-collection), one should assume non-consent and abandon the data. A work-around for dealing with non-users could be to place well visible disclaimers on physical premises equipped with IoT devices / sensors, like *"Attention: Your conversation is being recorded. Cameras are in use throughout the building. Any of your movements in this building are recorded. Your respiratory rate and eye movements are monitored. If you don't agree to this, please leave these premises now!"*, maybe together with red flash lights to focus attention, and frequent reminders of ongoing surveillance. The RERUM consent manager is intended for the use of **conscious and authenticated IoT users** only. For instance the RERUM consent manager can be deployed in the RERUM "Traffic Shaping" use case "O1". There the city council is requesting the user's consent for collecting readings of the user's smart phone sensors. The user can grant or deny consent, and temporarily or permanently suspend data collection, as well as unsharpen their precise location, acceleration, orientation and other readings. Users participate by installing and operating a specific "O1" application on their smart phone. Before this application starts to gather and transmit the smart phone's sensors' data, permission of the smart phone owner must be obtained. This is where the RERUM consent manager comes into action.

We first clarify the meaning of consent in the European mind set in Section 3.1.1, outline topics that may need to be addressed in an IoT request for consent in Section 3.1.2, and summarise briefly existing PETs to ease the burden of giving meaningful consent in Section 3.1.3. What machine readable parts an IoT consent should contain, we sketch in 3.1.4. After this we specify the main as well as auxiliary functionality of the RERUM consent manager in Section 3.1.5. We describe the processes of requesting (see Section 3.1.6) and granting (see Section 3.1.7) consent in more detail, as well as of deriving and deploying privacy policies (see Section 3.1.8) and specifying consent handling preferences to allow for (partial) consent automation (see Section 3.1.9). After this we address the topics of keeping consent history and allowing for consent revocation in Section 3.1.10 and elaborate on the relationship of the consent manager and the RERUM privacy dashboard as well as methods to resolve potential conflicts between requests for consents and privacy preferences in Section 3.1.11. We propose how the consent manager interacts with the other RERUM privacy components to deploy and enforce privacy policies in Section 3.1.12. We conclude this section briefly listing existing technologies and standards that may be used to implement the RERUM consent manager in Section 3.1.13.

3.1.1 High-quality consent

The European Data Protection Directive defines an individual's consent as any freely given specific and informed indication of their wishes by which the data subject signifies their agreement to personal data relating to them being collected and processed. Voluntary and informed consent however is not trivial to obtain [108]. A valid consent in European mind set should be

Prior: Obtain consent prior to any data collection. However sometimes a data subject can be identified only after data were recorded, maybe fused, and evaluated. When can / should one ask consent in such cases?

Informed: There needs to be a precise and easily understandable description of the matter, and an outline of the consequences of consenting or not. The data subject must be informed e.g. about the purpose of data collection, processing and use, transfer to third parties (if any), the possibility to deny consent and consequences of such a denial, the possibility to revoke consent for the future and where to do this. Information also should include details about the benefits and harms that might reasonably be expected from the action under consideration. Information should address the important values, needs, and interests of the individual. How does one ensure that all data subjects understand this correctly? How do we know the data subject has comprehended the matter fully? Exam to be passed, like reiterate in own words or apply to a set of hypothetical events? Currently, requests for consent frequently are characterised by obfuscating marketing language without an honest desire actually to inform the data subject.

Specific: The object of consent must be specific and well defined. The data controller should explicitly state the purpose or reason for undertaking the action, and avoid unnecessary technical detail. What are limits and scope of a particular purpose? Especially in the US not every further processing for a different purpose is necessarily incompatible. What is a compatible purpose? Reasonable data subject expectations, context, nature of data, impact on data subject, fairness, technical safeguards like anonymization, data subject benefits, et cetera. If processing or operations change or are added, a new consent is needed. This is desirable, but is it possible to get a new consent for every secondary use?

Voluntary: For a truly free and voluntary consent there must be a real choice and no risk of deception, coercion, intimidation, and (substantial) negative consequences when withholding consent. Consent may not be coerced or overly manipulated. In many cases consent is not really free. What are substantial negative consequences: no wrist watch, no mobile phone, no electricity, no light, no heating, no television, no photos, no job, no health-/life-/car-insurance, no …?

Explicit: There must be some active communication between the parties, so an individual can "signify" agreement. Granting of consent should be performed in writing. Explicit consent may be achieved in some cases in other forms than in writing, but organisations should not infer consent, especially not from non-reaction of the data subject. Online an option should need to be checked actively. Pre-selection is not a desirable option. No implicit consent should be assumed, opt-in is clearly to be preferred to opt-out. Are opportunities to accept or decline visible and readily accessible? Purchase and use of an IoT device should not imply consent to uploading sensor data about human data subjects to the manufacturers or other parties' servers.

Documented: Consent must be documented and the person concerned must be able to review the consent anytime. Documentation should include circumstances of granting consent.

Revokable: Consent withdrawal must be possible anytime; at least with respect for the future. Any consent under EU legislation can be revoked. Revoking may not be more complicated than granting that consent, and needs to be free of charge. Withdrawing a consent subsequently, like withholding a consent initially, may not be followed by (substantial) negative consequences that would refrain the data subject from exercising their right to withdraw their consent.

As said already, a valid, meaningful, high-quality consent is not easy to obtain. This is true especially in IoT settings. What held in traditional eCommerce scenarios (even if it doesn't work very well there either [199] [155]), cannot be transferred to the IoT world 1:1. In traditional eCommerce a data subject knows they are about to give details about their address and bank account. When they consent to their use, it is possible that they can judge the implications and potential consequences of giving these personal data. In IoT scenarios with their numerous (mostly not very visible) sensors and actors, data subjects frequently aren't even aware of personal data being recorded, nor do they understand the potential deductions that can be made from the gathered data.

Above mentioned criteria should be met by a valid consent in European mindset. However are they sufficient to guarantee a meaningful consent? Friedman, Lin and Miller [90] list six components of a meaningful informed consent.

1. **Disclosure:** Providing adequate information, which is required for "informed" consent.
2. **Comprehension:** The data subject having sufficient understanding of the provided information, also an issue of "informed" consent.
3. **Voluntariness:** Ability of the data subject to resist participation in a reasonable manner is essential for informed "consent", which must be voluntary.
4. **Competence:** The data subject possessing the requisite mental, emotional, and physical capabilities to decide and -if need be- to resist, are also relevant for informed "consent" to be truly voluntary.
5. **Agreement:** A reasonably clear opportunity to resist participation is also an essential part of informed "consent", which must be voluntary and in European mindset given prior and explicit. In the mindset of the authors from the USA also notice and opt-out procedures might be acceptable.
6. **Minimal Distraction:** The consent process may not be so overwhelming, as to cause the data subject to disengage from the process. This is a very relevant issue of "informed consent". The activities of being informed and giving consent should happen with minimal distraction, without diverting data subjects from their primary task or overwhelming them with a lot of nuisance. One needs to get the data subject's attention in order to disclose information. In consent processes of today, the data subject frequently accepts any "Terms and Conditions" without even looking at them, being keen on getting at whatever service is behind them, and incapable and unwilling of wading to a huge pit of small print legal jargon. This criterion is challenging to implement, because process of informing and obtaining consent necessarily distract data subjects from their primary task. Just imagine an IoT scenario and for every sensor being passed a new consent dialogue has to be completed. Here consent support, for instance in (semi-) automated consent procedures based on the data subject's consent handling preferences may be needed to get a meaningful consent.

Note: The Engineering and Physical Sciences Research Council (EPSRC) is an UK agency for funding research in engineering and the physical sciences. It has recently funded a project called "MCDE" (Meaningful Consent in the Digital Economy) [207]. This ongoing project is located at the University of Southampton (Partners: Baxi, Centre for Science and Policy, eBay Research Labs, Information Commissioners Office, Madano Partnership, Massachusetts Institute of Technology, Microsoft, Nokia, Stanford University), started February 2014, and is going to end August 2016. Though not IoT specific, their progress of work may be worth monitoring.

3.1.2 Informed consent in IoT

A consent template recommended by German data protection officers for signature-on-paper scenarios is the following: *With your permission, your data will be collected, processed, and used for the following purposes: (…). Your personal data will be collected, processed, and used in the context of the aforementioned objectives in accordance with the (…) Data Protection Act. The collection, processing, and use of your data take place on a voluntary basis. Furthermore, you can revoke your consent at any time without any adverse consequences / with the consequence that (…). Please send any notice of cancellation to: (…). In the event of cancellation, your data will be deleted upon receipt of your notice.* [17]. This template can quickly be read and easily be understood in simple everyday form-filling situations.

In eCommerce scenarios the data subject normally is a conscious user of the data controller's application (as opposed to many IoT scenarios) and can decide to enter and submit personal data or not, including browsing a web site using for instance cookies. Still, digitally given consent currently usually is rather meaningless. Terms and conditions are not read or understood by the "consenter" in the digital world. Consequences of consent are not clear, e.g. how personal data is being used. Chosen "request for consent" narrative affects user action. Often, the consent requesting party seems not really interested in making the data subject comprehend the request implications. In IoT scenarios with numerous sensors being placed in the data subject's personal environment and / or personal devises, data collection becomes both more unobtrusive and data expressiveness much more intensive and detailed. Potential deductions are becoming more powerful with intelligent processing and data fusion. Possible implications are hard to determine, explain and understand, as often is the IoT application itself.

In the request for consent there should be sufficient information for the data subject to come to a well-informed conclusion whether or not to grant consent. In IoT scenarios an interested data subject may look in the request of consent for collection of personal data for instance for at least the following information:

Purpose: What will the information be used for? What is the purpose of the IoT device and the IoT application respectively? What is the benefit for the data subject, what that of the data controller (device manufacturer, application operator, ...)? What secondary use the data is intended to, what could be put to? Are there cross-context use-constraints for secondary use? Is some in-context secondary use permitted? However what if the actual service purpose does not yet exist when granting consent for data collection? In such cases there needs to be a subsequent consent for secondary use.

Data Controller: Who will have access to the information? Who owns/possesses/controls/uses the device/sensor/application (including their postal address)? Who controls the raw, aggregated and processed data? Where can the written privacy policy be found? Are there additional separate data controllers and privacy policies for individual IoT devices?

Assessments, Audits and Seals: Are the main privacy principles observed? Especially data minimisation? Has the IoT device/application been assessed by a trusted third party? What privacy trust seals has the IoT device/application been awarded? What is the legislative background of these seal programs?

Data Recording: What information will be collected? What data does the device/application record? What sensors are used to collect the data? Are the gathered sensor values transmitted in encrypted form (including details)?

Data Processing and Fusing: What other data sources (also personal knowledge) the sensor data are/can be correlated with? Where are these data stored (IoT device, user's computer, controller's servers, ...)?

Data Storing: How long will the information be archived? Where and how long are the data stored? What logs, backups and history data are kept? Are the data stored in encrypted form (including details)? There should be reliable date/time for (raw) data destruction.

Data Sharing: Whom will the manufacturer/operator share the data with and in what form? Are the data transmitted in encrypted form (including details)?

Identification of the data subject: How will the identity of the individual be protected? Is the information is stored in a de-identified form? Is the manufacturer/operator able to re-identify data? How easy is it to identify the data subject and track its behaviour?

Privacy Impact on the data subject: What are risks and side effects for data subject from the data collection and processing as well as the conclusions drawn from these data? What is the abuse potential of these data?

Rights and Controls of the data subject: What privacy controls are available to the data subject? Can the user access (raw) sensor data, export them to another service/device? Can the user view, edit, or delete sensor data from the device and/or the manufacturer's/operator's servers? What rights has the user to opt out of data uses and disclosures? How can the data subject limit/stop data recording/transmission, disconnect device? Note there is an indication of a paradoxon of privacy "control": A study found that data subjects who are given an explicit option to publish their data feel less privacy concerned and thus become more likely to not just answer, but also allow the publication of their answers [142].

Technical Details: Description of the IoT device/application, its system architecture, trust boundaries and main data flows, as well as of the IT security architecture may be of interest for the technically minded and privacy conscious data subject, as well as for any privacy assessment.

Clearly, reading one of today's eCommerce privacy policies, or terms and conditions, or end user license agreements online is very tedious and hardly ever done. The need for information is even greater in IoT situations, as would be the corresponding "request for consent" documents. And these situations will occur much more frequently than eCommerce situations before. In the next section we are going to take a look at strategies to reduce consent complexity.

3.1.3 Reducing consent complexity

If an organisation deliberately makes it easy to "consent" without reading and understanding the terms and conditions, should they be able to rely on that "consent" in a court of law? Does failure to insist on meaningful consent really indicate that data subjects "don't care" about their privacy? But how can a meaningful consent be obtained from human data subjects without overwhelming them? What support is there for data subjects to make their consent mean something?

Primarily, IoT device manufacturers and application providers should implement privacy by design, especially observing the data minimisation and collection limitation principles. Data that are not available cannot be abused. If for instance secondary use is excluded, no hidden agenda is being followed behind the official purpose, personal data are secured at all times, and an adequate value-add is granted to the data subject, there is no need to hide ugly things behind a smokescreen of legal phrases. This will make the privacy policy easier for data subjects to read, comprehend and consent to.

There are numerous projects that aim to track, parse, analyse, distil, and better communicate privacy policies, terms of service, and end-user license agreements [15] [140]. Many deal with the question on how to make the information presented in the request for consent easier intelligible, like

Simple Language: Privacy agreements should be written in plain language. As opposed to huge documents containing a lot of legal language in small print, the "simple language" approach tries to support users in understanding privacy policies better by rephrasing the gist of the individual clauses briefly in easier-to-understand language. "500px.com Terms" [107] is an example of how agreements can be made easier to read using simple language. Tools and templates allow to create simply and clearly worded privacy policies. "Iubenda" [96] is an example of a privacy policy generator tool where one page is optimised for reading and simplicity, a second one uses legal language. However one can only simplify to a certain extent before losing precision. Especially in legal texts it may imply a certain risk. Courts of law might regard the simplified policy as legally binding and disregard the detailed policy in legal language.

Standardised Terms: To improve readability without undue loss of precision, standardisation of terms is a good option. Like in Mathematics, there should be a set of well-defined privacy terms that are used in the same way in every request for consent expressing a complex privacy issue concisely. "CommonTerms" [139] for instance has compiled a database of common terms in online Terms and Conditions [140]. However it will not be possible to standardise everything. Some terms will remain application-proprietary.

Standardised Policies: The use of standardised privacy policies like those well known from the open source software movement like GPL, BSD, and Apache may be helpful for the data subject. They learn how to rate certain well-known agreements. Docracy [229] for instance is providing a set of standardised "Terms and Services" texts. However also here it is not possible to cover everything with standard agreements.

Icons: Promote the use of standard symbols. What "Creative Commons" license icons [159] did for copyright, other initiatives applied to privacy. For instance a group of Yale students [110] designed a set of privacy icons to visualise compliance of a privacy policy with a user's privacy preferences. There are many other examples like [156] or [174] (see Figure 8), proposing icons for privacy policies. However it is virtually impossible to create icons for all terms, so it is challenging which ones and how many to select to improve readability and to avoid confusion.

Standardised Templates: Similar to medical package inserts that are modelled to a common standard template, one can also standardise a privacy package insert template, including unified presentation and order. This reduces reading distraction and eases the data subject's orientation.

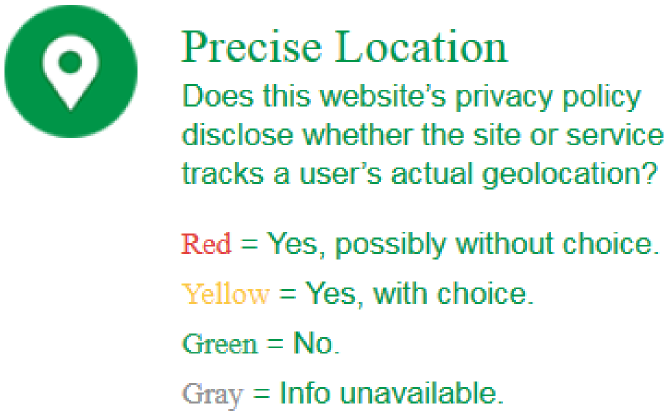
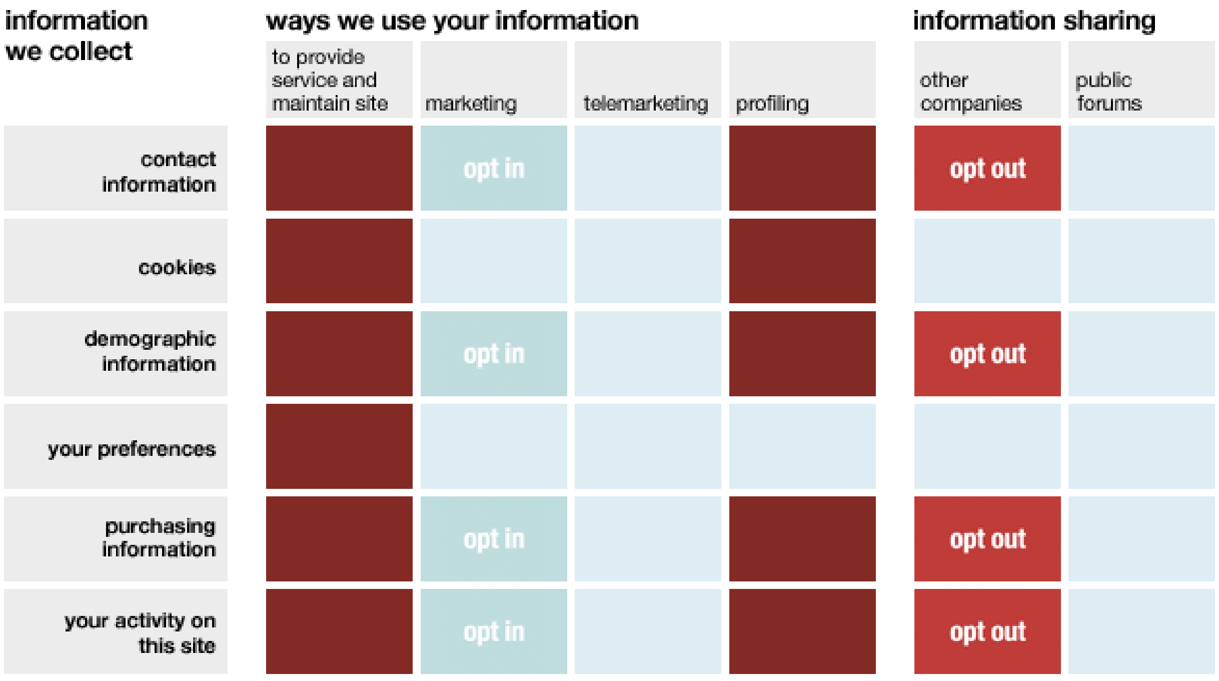

Precise Location
Does this website's privacy policy disclose whether the site or service tracks a user's actual geolocation?
Red = Yes, possibly without choice.
Yellow = Yes, with choice.
Green = No.
Gray = Info unavailable.

Figure 8: DisconnectMe precise location privacy icon [174]



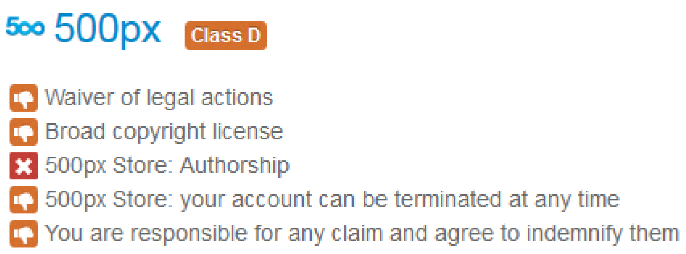
information we collect	ways we use your information				information sharing	
	to provide service and maintain site	marketing	telemarketing	profiling	other companies	public forums
contact information	🟥	opt in		🟥	opt out	
cookies	🟥			🟥		
demographic information	🟥	opt in		🟥	opt out	
your preferences	🟥					
purchasing information	🟥	opt in		🟥	opt out	
your activity on this site	🟥	opt in		🟥	opt out	

Figure 9: CMU privacy nutrition label example

Standardised Summaries: Like nutrition labels for food, standardised summaries may be helpful for data subjects to gain a quick grasp on the gist of a privacy policy, as shown by several initiatives [4] [131] (see Figure 9).

Trust and Score: A strong simplification would be privacy traffic lights or a privacy school grade type of rating. PrivacyScore [230] was doing this for privacy: Calculating a number to represent the overall privacy score of a web site. Trust-e [10] has a similar but binary approach: Either you qualify for the trust mark or you don't. TOS-DR ("Terms of Service; Didn't Read") [200] have begun grading

(A-F) and commenting on TOS documents (see Figure 10). However it is not possible to rate many terms, e.g. terms that aren't inherently good or bad (e.g. jurisdiction). This requires assessment by trusted third party. There could also be an application and device provider trust rating, and a privacy certification ("Privacy seals") by trusted third parties. Also consent procedures should be subject to "consent audits" and "good consent practices seals".


500px Class D

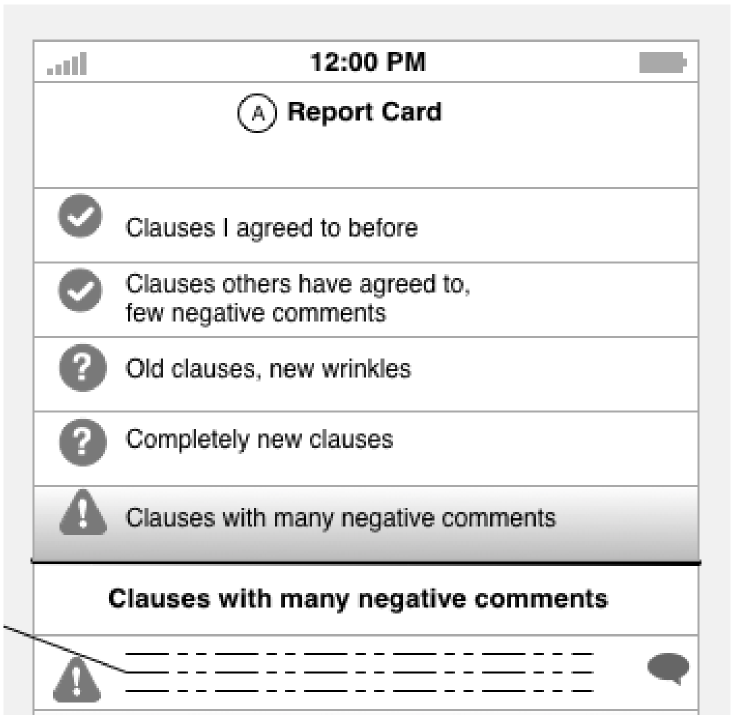
- Waiver of legal actions
- Broad copyright license
- 500px Store: Authorship
- 500px Store: your account can be terminated at any time
- You are responsible for any claim and agree to indemnify them

Figure 10: TOS-DR rating of 500px

Tracking: Initiatives and services tracking privacy policies aim to collect, analyse and compare many privacy policies (e.g. Youluh [216]). Find details below in this section. Central monitoring was for instance also attempted by TosBack [78] of the EFF which automatically is harvesting and tracking changes to TOS documents. CMU for instance has coded and compared a set of bank privacy policies [60]. Such central monitoring helps with change tracking and comparisons. It will not necessarily make the content more accessible. There are several initiatives reading and analysing privacy or general TOS policies and pointing out interesting features, like "digitaltrends" [57] or "KnowPrivacy" [100]. Such human expert analysis may be very helpful, but certainly is expensive. There are also automated tools, like EULAlyzer [31], which scans EULAs trying to find out whether they contain hints that the software intends e.g. install displays pop-up ads, transmit personally identifiable information, or use unique identifiers to track users, listing potentially interesting words and phrases.

Preferences and Negotiation: IoT increases the possibility of users being asked to make consent decisions on numerous occasions everyday, e.g. when walking through a smart office building. Under certain circumstances, consent might be (partially) negotiated and automated using agents and data subject preferences, like for instance originally suggested by the Platform for Privacy Preferences Project (P3P) initiative [59]. P3P enables web sites to express their privacy practices in a standard format that can be retrieved and interpreted by user agents. These inform their users of site practices (in both machine- and human-readable formats) and may automate decision-making based on these practices when appropriate. Another project in this context was "EmanciTerms" by Harvard law school [209], where vendors and customers use corresponding terms for privacy preferences to allow the process of arriving at agreements to be (partially) automated. Support some degree of consent automation based on data subject preferences may be complemented by manual review and adjustment of automatically generated consents, maybe in the manner sketched in [99].

Awareness and Education: There are numerous persons and groups, trying to educate about privacy protection and to raise awareness for privacy issues, like "biggestLie" [141], "ZeroKnowledgePrivacy" [85], as well as many privacy protection officers and experts.


12:00 PM

Ⓐ Report Card

Clauses I agreed to before

Clauses others have agreed to, few negative comments

Old clauses, new wrinkles

Completely new clauses

Clauses with many negative comments

Clauses with many negative comments

Figure 11: Youluh report card screenshot

We look at one initiative in more detail. The Youluh [216] service allows its users to set up pseudonymous accounts or accounts associated with email addresses. Each time a user is confronted with the need to consent to a EULA or terms of service or similar, the user can submit the text of the request for consent to Youluh. The service stores the text associated with the user's pseudonym, analyses the text and returns a "Report Card" (see Figure 11) whose intention is to help the user prioritise what to read first. For example, it indicates which of the clauses are (almost) identical to previously accepted clauses, which are new clauses, which clauses many other users have accepted without a lot of negative feedback, which clauses have received much negative user feedback, which clauses are modified old clauses, and which clauses are completely new clauses. Users may leave comments on clauses. Any submitted text of a request for consent becomes part of that user's library of EULAs in that user's Youluh account. There the user can re-read any of them.

In summary, data controllers should write their privacy policies and requests for consent using standardised symbols, phrases, layouts, and/or plain and simple language. There should be machine readable parts of the policy to enable computer processing, and maybe also user configuration. The data controller should offer the data subject participation (feedback) in shaping the wording of the request for consent. Trust may be promoted by third party ratings and comments. A promising approach seems to be the development of a set of standard requests for consent, similar to GPL, BSD and other open source licenses. Automated analysis of privacy policies and tracking of changes can support decision making, also regarding consent revocation, as do repositories of already accepted agreements for future reference and comparison (like the Youluh concept). Regarding the principle of "minimal distraction"[90], there should be support of consent automation based on data subject preferences with possibly a manual review and adjustment of automatically computed settings by the data subject.

3.1.4 IoT request for consent

A request for consent contains a human readable part. The human readable part describes the purpose of the application and other details. It also should display any machine processable part of that request for consent in a human readable manner.

In IoT environments, one can precisely specify the installed sensors, actors, data services, and other services available, as well as their meta data and capabilities [228]. Data controllers may wish for access to them to obtain potentially personal data and trigger actions. This access may require consent of the data subject. The RERUM GVO Registry (see D2.3, Section 6.6.2) represents an IoT device and service registry. All IoT device (sensor, actor, service, ...) specifications of the IoT infrastructure are contained in it. These specifications can be referenced by the security and privacy policies as well. They may also be referred to in requests for consents that later on may be transformed into data subject consented privacy policies. Additionally to a list of sensors and actors (VRD) the GVO Registry should also contain a list of of services (=VE), to allow the use of aggregated, locally processed data additionally to raw sensor data.

In IoT a request for consent therefore usually should contain some machine processable part. This part among other things would at least detail all the sensors, actors, and other data sources and services. The data controller asks the data subject's consent for them including all details (like the data rate or resolution). Sensors for instance may read temperature (degree Celsius), pressure (kiloPascal), humidity (percentage), light (Lux), noise (dbA), proximity (cm), speed (km/h), acceleration (metre per second squared), and orientation (gyroscope dimensions in degrees). A request for consent in IoT can reference a sensor/actor/service or even a list of such components and corresponding read/write specifications. A data controller not always may require access to raw sensor data or native actor commands, but instead frequently may be content to ask for access to composite services offering locally pre-processed or aggregated data (which would be desirable from a privacy point of view).

In addition to these IoT specific details, generally a request for consent should offer administrative details in machine readable format for ease of processing. Of interest is for instance the specific purpose for requesting personal data, how the data will be processed and which data sources are to be tapped specifically. The IoT infrastructure should specify those fields that need to be provided, for instance at least:

Company details: meta data about the data controller (company details, "impressum"). These, besides name and address, should contain a data controller's id that allows to retrieve any trust seal, endorsements and other information about the data controller, for instance additionally retrieved from a pPIP and added to the request for consent.

Consent number: for mutual reference, also for revocation. The data controller and the data subject should share a common consent number/uri, so the data controller can refer to it at the PEP and the data subject can refer to it for changes and revocation.

Purpose: the purpose of the data collection must be explained to the data subject very well and for reference the purpose should be retrievable easily for future reference. It maybe could include an application name and the data types permitted for it.

Validity period: It might be useful to time-limit the consent, in addition to the legally required possibility to revoke the consent.

List of sensors, actors, and services: (VRD/VE) and specifications of values to be collected and functions to be called. Device/Sensor meta data could for instance comprise location, owner, recorded data type, ...

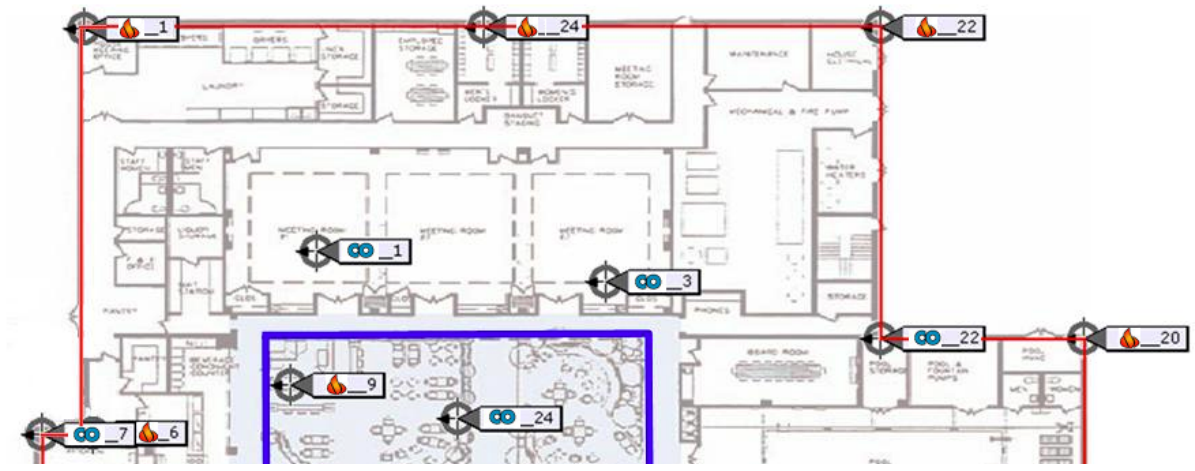


Figure 12: Sample sensor placement map for a fictional indoor use case

Sensor and actor specifications of an IoT situation the consent manager may visualise (e.g. see map in Figure 12), eventually also those of services. For this the data in the RERUM GVO Registry (see D2.3, Section 6.6.2) are needed. The consent manager could highlight the requested-for sensors, actors, and services. By clicking on an item, the data subject would be shown the details of the request as well as details about the component and potential settings permitted by the data controller.

Requests for consent may contain wildcards or ranges, like when an application wants to access all data of all sensors within the IoT infrastructure.

```
application: "traffic shaping"
sensor:      any
type:        any
resolution:  any
data rate:   any
duration:    any
```

However usually a request for consent hopefully should be more specific, maybe like this:

```
application: "traffic shaping"
sensor:      08.15
type:        location-xy
resolution:  street level
{
   data rate: every 30 minutes
   duration: may-15-15 to aug-01-15 daytime
} {
   data rate: every 5 minutes
   duration: may-15-15 to aug-01-15 nighttime
}
```

The consent manager should provide a **"request-for-consent" template** for its domain as guideline for data controllers to shape their consent in a more uniform manner, as recommended in Section 3.1.3. Data subjects might even be permitted to adjust values in the consent ("negotiation"). This could be effected via options being present in the request for consent explicitly offered by the data controller. Setting aside direct modification facilities for requests of consent, data subjects should be encouraged to and supported in giving feedback about the request for consent to the data controller.

3.1.5 Consent Manager functionality

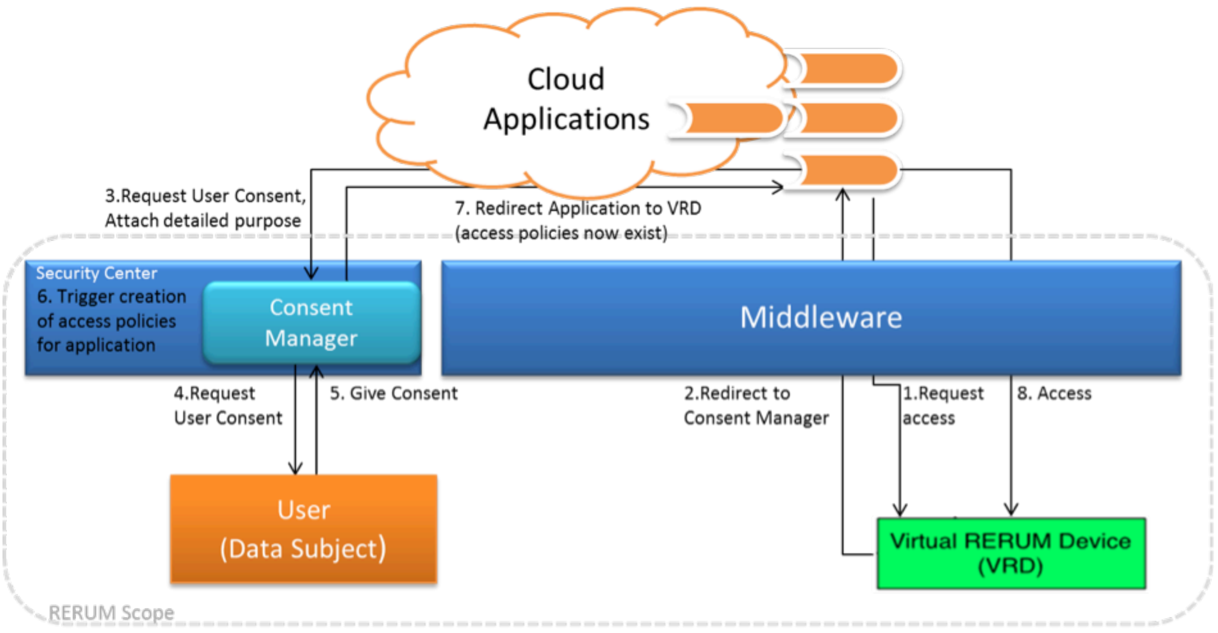

Cloud Applications
3.Request User Consent, Attach detailed purpose
7. Redirect Application to VRD (access policies now exist)
Security Center
6. Trigger creation of access policies for application
Consent Manager
Middleware
4.Request User Consent
5. Give Consent
2.Redirect to Consent Manager
1.Request access
8. Access
User (Data Subject)
Virtual RERUM Device (VRD)
RERUM Scope

Figure 13: RERUM Consent Manager interaction (from D2.3)

We assume a shared RERUM "Consent Manager" being provided by every IoT infrastructure (RERUM "Security Centre" for all of its data subjects (see Figure 13 from D2.3). The operator of the local IoT infrastructure most likely also operates the consent manager. It needs to be trusted by the data subjects and the data controllers. The RERUM "Consent Manager" is located conceptually at the RERUM "Security Centre", and interacts with the data processing parties above the RERUM "Middleware" as depicted in Figure 13 taken from D2.3. In that figure a new data controller (application) (1) requests access to personal data residing in the VRD (or Virtual RERUM Object - VO), which (2) requires consent of the data subject. The consent manager supports both (3) the data controller and the (4) data subject in obtaining and (5) granting that consent, eventually offering (semi-)automated consent based on the data subject's consent handling preferences. It (6) generates and deploys the corresponding privacy policies of the data subject and (7) informs the data controller of the consent grant, eventually providing OAUTH-style credentials and access point information. The data controller may then successfully (8) access the personal data at the VRD/VO.

The core consent manager functionality is:

Requesting Consent: accept and process requests for consent by the data controller.

Granting Consent: support the data subject in reviewing and deciding about requests for consent.

Policy Derivation: generate and deploy privacy policies from requests for consent that the data subject has decided on. The consent manager generates an appropriate privacy policy and places it in the privacy policy repository (pPRep).

Consent Automation: To relieve the data subject from the burden of granting consent and to observe the principle of "minimal distraction" the consent manager should allow the data subject to specify consent handling preferences that allow for some degree of automated consent handling, provided the data subject is able to review and endorse the automated decisions afterwards.

History: keep a consent history, including grants, denials, expiries and revocation. The consent manager stores the request for consent, the granted consent and the consent context details for future reference by the data subject. The consent repository is not a policy repository but serves for keeping track of consent history.

Revoking Consent: support the data subject in consent revocation. This includes notification of the data controller and retraction of the corresponding privacy policies from the pPRep. It may include directions on how to delete or request deletion of personal data.

In addition to this, for instance the following features may enhance the consent manager's functionality:

Consent Reading Guidance: It would be of advantage, if the consent manager were to offer Youluh-style functionality.

SPAM Prevention: We desire SPAM protection from requests of consent from misbehaving data controllers. Data controllers pestering data subjects with flooding by requests for consent can be black-listed. Then, as long as this blacklisting is in effect, the consent manager will not accept any consent requests any more from this data controller for that data subject. If a request for consent is pending, or as long as a granted or denied consent is still in effect, the data controller may not ask for that consent again.

Update Option: Data controllers may ask for an update to a previously granted or denied consent. For this the data controller needs to ask for a new consent referring to the "cid" of the old consent request (additional input parameter) that this one is to replace. The consent manager should clearly highlight the new sections compared to the previous version. This may ease the consent decision for the data subject. The consent manager can revoke the prior consent, once the new edition has been decided on and the newly generated privacy policy super-seeds the old version.

Feedback: The consent manager should allow data subjects to give feedback to the data controller on the request for consent. This may happen rather anonymously, e.g. by the consent manager calling back to the data controller on the "cid" submitting the feedback. Or by the data subject agreeing to providing contact details to the data controller. This will help the data controller to improve on shaping future requests for consent.

Review Recommender: A request for consent has been declined or granted. A mechanism might detect and visualise equivalent clauses contained in this and prior approved or declined requests for consent, that appear contradictory to the current decision. These could be indicated to the data subject. Maybe the data subject wishes to revise some of these clauses. Another cause to consider review of consents might be alteration's for instance in the data controllers reputation, feedback to individual clauses by other users, or a change of the data subject's privacy preferences.

The following functionality should also be provided, though not (solely) by the consent manager:

OAUTH: An access control layer is in effect. The accessing service sometimes must present particular credentials of the user on behalf of which he is accessing the data. The consent manager indicates the data subject's consent to the data controller. If need be, the consent manager can also mediate the necessary credentials (OAUTH token) to the data controller and provide service end point information to gather sensed data and to process them as described in the consent content. Those OAUTH tokens are evaluated by a privacy or other policy enforcement point. However OAUTH tokens need to be issued by the data subject. Details of the necessary infrastructure for this need to be specified.

Sticky Policies: User consent is an agreement between a data subject and a processing party on a purpose, which describes why personal data are collected and processed. This purpose needs to hold at all times, whenever personal data are processed. The privacy policy resulting from the consent (and the consent or a reference to it) may be attached to the data in transmission and at rest ("sticky policies"). There should be a functionality the retrieve a policy (set) for a given data set and stick it to that data set. It most likely will be needed by the pPDP and refer to the consent "cid" that was quoted by the data controller when asking for that data set.

3.1.6 Requesting consent

When a data controller tries to access a data source or other resource involving personal data without sufficient consent, it must be told to obtain consent first. A possible request consent work flow is depicted in the sequence chart in Figure 14. We assume, the data controller has authenticated at a suitable identity provider first and is bringing along a suitable set of credentials. This e.g. may be a SAML assertion together with assertions for any required attributes. We further assume that multiple data subjects need to consent for that particular resource and purpose. We want to hide from the data controller which of the data subjects declines or lets expire a request for consent. As long as the data subject does not choose to grant a request for consent or otherwise to interact with the data controller explicitly (e.g. via non-anonymous feedback), the identity of the user is shielded from the data controller by the consent manager.

The data controller (1) initiates a session (session id: "SId") with the RERUM IoT component containing the privacy policy enforcement point (pPEP). It requests data from the restricted data source. The pPEP

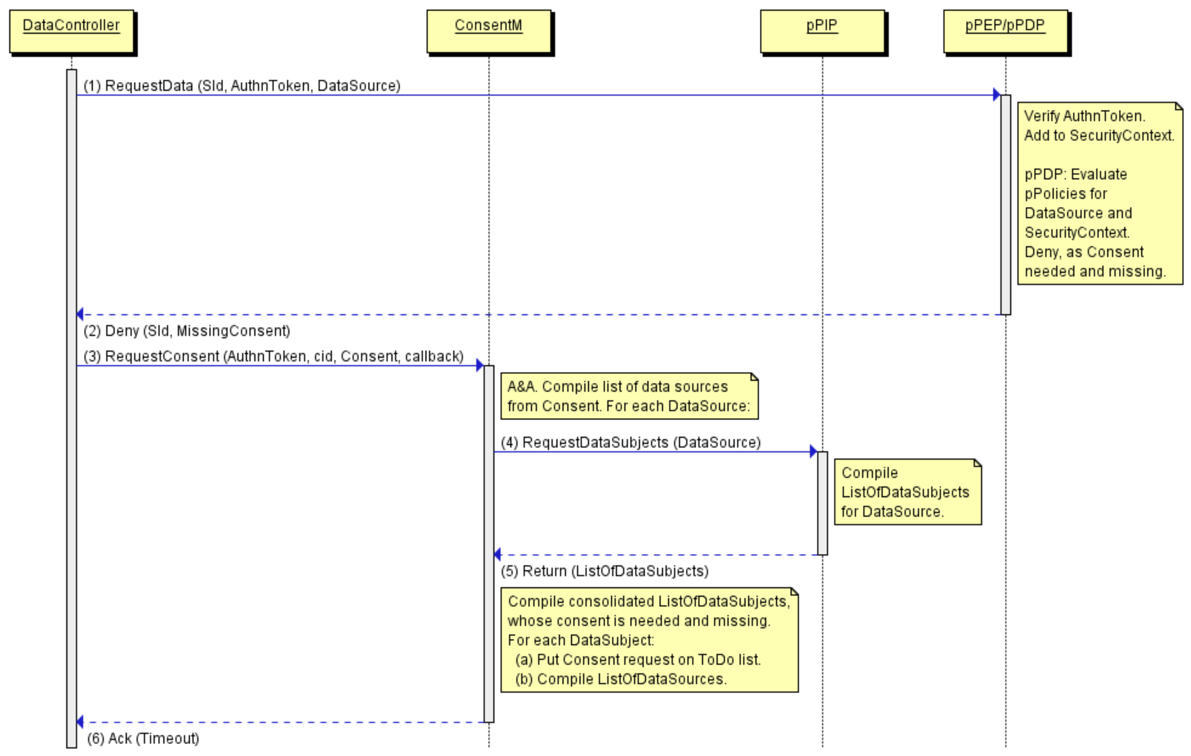

DataController
ConsentM
pPIP
pPEP/pPDP
(1) RequestData (SId, AuthnToken, DataSource)
Verify AuthnToken. Add to SecurityContext.
pPDP: Evaluate pPolicies for DataSource and SecurityContext. Deny, as Consent needed and missing.
(2) Deny (SId, MissingConsent)
(3) RequestConsent (AuthnToken, cid, Consent, callback)
A&A. Compile list of data sources from Consent. For each DataSource:
(4) RequestDataSubjects (DataSource)
Compile ListOfDataSubjects for DataSource.
(5) Return (ListOfDataSubjects)
Compile consolidated ListOfDataSubjects, whose consent is needed and missing. For each DataSubject: (a) Put Consent request on ToDo list. (b) Compile ListOfDataSources.
(6) Ack (Timeout)

Figure 14: Consent Manager: sample need for consent sequence diagram

verifies the authentication token and adds it to its security context. Then it asks (no extra message depicted, sketched in the comment field only) the privacy policy decision point (pPDP) for permission to serve that data controller with the data source. The pPDP (also no extra message depicted) consults its privacy policy repositories and various privacy policy information points. It needs to figure out, if a consent is needed for that data source, and whether all required consents are present (and valid!). If (what is not the case in our sample work flow here) the data controller quoted a consent id “cid” (or a list of them), the pPDP could check if that consent were (at least partially) present already (e.g. in an indoor use case an office gets occupied with two more persons). The pPDP denies the access request and instructs the pPEP of the obligation that must be (2) passed on to the data controller.

The data controller prepares a suitable request for consent (at best including the minimum set of data sources needed for the specific purpose at hand). The data controller should ask each data subject for consent for the complete IoT application (not individual data sources) and explain the connections between the requested data sources. No data subject would like to be flooded by tiny consents for each data source. Rather they would most likely prefer to be asked consent once for the complete IoT application. This helps to keep the number of requests for consent down and the burden of giving consent at an acceptable level. It also helps the data subject to get a bigger picture than would be possible from a multitude of tiny consents for individual data sources. The data controller should avoid a “trial-and-error” approach and not ask for separate consent for every data source. This is confusing, inefficient, time-consuming and tedious to the data subjects; especially if we assume manual consent granting; and an IoT application that uses many different data sources associated with different data subjects.

The data controller can just as well omit steps (1) and (2) and directly approach the consent manager, if it wants a new consent to be processed. It may include an "uid" in the request (additional input parameter), if it requires consent of just one specific previously known user (not in our example work flow). The data controller (3) submits the authenticated request for Consent to the consent manager, stating a unique consent id "cid" for reference and specifies a callback function. There the consent manager can leave a notification on the progress of the request-for-consent processing. The consent manager analyses the machine processable part of the request for consent, that details the required data sources and their specifics. Then for each data source on that list the consent manager (4) requests from the privacy policy information point (pPIP) a list of data subjects associated with that data source, whose consent "cid" is still missing. The pPIP compiles such a list and (5) returns it to the consent manager.

The consent manager evaluates all the data sources for associated data subjects. Then it prepares a consolidated list of data subjects and (a) puts the request for consent "cid" on their to-do-list. Additionally it (b) compiles a list of data sources associated with each data subject. This list will be needed to indicate to each data subject the relevant data sources. Later on, after consent has been granted or denied, it serves to derive a suitable (minimised) privacy policy for that data subject. The data controller (6) is informed that the request for consent was submitted to the appropriate parties. It is also told, when that request for consent will expire, should it not have been resolved completely by then. Up to now, the data controller is not aware which data subjects are involved with what data source. Consent granting in many cases requires manual user interaction. It does not make much sense to keep the data controller waiting online for completion of the process. Therefore we have opted for the asynchronous callback method. Only in cases where a complete request for consent of all concerned data subjects can be granted automatically (e.g. based on consent handling preferences), a synchronous answer would make sense. Still the same functionality can be implemented by instant callback as well.

3.1.7 Granting consent

When a data subject (who is a registered RERUM user) in their ToDo-List finds a request for consent awaiting their approval, they can select to review it. We imply that the data subject so far has not indicated consent handling preferences that in this case would allow for a fully automated consent handling. The data subject / user interacts with the consent manager via a graphical user interface (GUI) provided by a suitable user agent[3].

A possible manual consent work flow is depicted in the sequence chart in Figure 15, where the actual data subject in question is "user1", whose user agent is depicted. We assume, that "user1" brings a suitable set of authentication and authorisation credentials. We omit sketching the introductory dialogues, where the data subject selects a "request for consent" to be reviewed from the menu (as sketched in Figure 7).

The data subject (1) initiates a session (session id: "SId") with the consent manager and requests a certain consent ("cid") for review. The consent manager authenticates and authorises the data subject,

[3]The user plane in this case consists of the data subject and the user agent, which in turn may be a browser with or without a browser-based app or (frequently in the case of smart mobiles) a pure app. For reasons of simplifying the sequence charts we omitted depicting the user and just drew the user's agent.

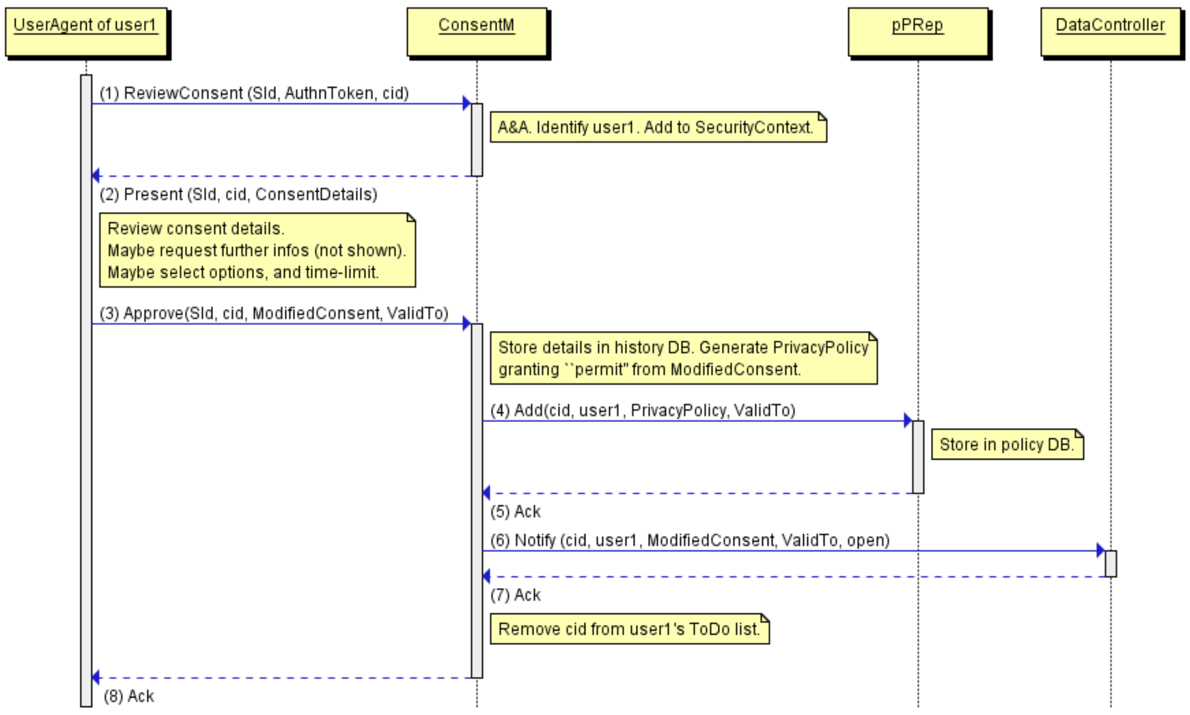

UserAgent of user1
ConsentM
pPRep
DataController
(1) ReviewConsent (SId, AuthnToken, cid)
A&A. Identify user1. Add to SecurityContext.
(2) Present (SId, cid, ConsentDetails)
Review consent details.
Maybe request further infos (not shown).
Maybe select options, and time-limit.
(3) Approve(SId, cid, ModifiedConsent, ValidTo)
Store details in history DB. Generate PrivacyPolicy granting ``permit" from ModifiedConsent.
(4) Add(cid, user1, PrivacyPolicy, ValidTo)
Store in policy DB.
(5) Ack
(6) Notify (cid, user1, ModifiedConsent, ValidTo, open)
(7) Ack
Remove cid from user1's ToDo list.
(8) Ack

Figure 15: Consent Manager: sample manual consent granting sequence diagram

whose identity is "user1". Then it retrieves the request for consent "cid", and prepares a suitably enhanced presentation of it, e.g. by using one or more of the techniques mentioned in Sections 3.1.3 and 3.1.4. Among other things the consent manager should indicate to the data subject, which of the data sources indicated in the request for consent are actually associated with the data source. The consent manager (2) submits the presentation to the data subject for review and approval.

The presentation should include clear indications which of the data subject's current privacy preferences would be in conflict with the request for consent. If a data subjects strongly desires a certain service or favours a certain data controller, they may be willing to grant exceptions from their privacy preferences for the corresponding request for consent. Details regarding this issue are being discussed in Section 3.1.11. The matter is not represented in Figure 15.

The data subject reviews the consent details and maybe requests further information on various aspects of the request for consent. If the request for consent contains options to select from, the data subject can elect to do so. Eventually the consent might be time-limited. Once the data subject has finished the review, (3) approval to the (maybe concretised and time-limited) request for consent can be sent to the consent manager. The consent manager stores the consent decision in its history data base. This includes the original request for consent, the presentation provided to the data subject, the modifications made to the request for consent and all other relevant circumstances and context data.

3.1.8 Privacy policy derivation and deployment

After the data subject has decided to grant or deny the request for consent, the consent manager (as depicted in Figure 15) transforms the (modified, time-limited) request for consent into an XACML-compliant privacy policy. That policy is applicable for that data subject, that data controller, and that "cid", and addresses only those data sources with which the data subject is actually associated with (see step 5.b of Figure 14). The policy may eventually result in a "permit" statement. If consent was denied, it would be a "deny" statement. This privacy policy is (4) added to the privacy policy repository's (pPRep) data base. It needs to be clarified, whether the PDP prefers a separate policy per data source (VRD, VE, service), or as few policies as possible. The pPRep (5) acknowledges the receipt and the new policy is in effect by then (or will be from the start date indicated in the privacy policy, if any).

User initiated changes to the consent content the consent manager may report to the data controller when informing it via callback. The consent manager (6) calls back the data controller regarding the "cid" on the newly obtained consent grant including the specific details. If more consent approvals are pending from the same "cid", this is indicated by the status "open". The consent manager may set the status to "final", if every data subject on the list has either consented or refused before the expiry date. Otherwise the status of the "cid" can be set to "expired". This means that at least one data subject has not reacted in time. Alternatively one can elect to have only two status values. Before time-out the status is "open", afterwards it is "expired".

There are at least two possible approaches for the consent manager to deal with a denied request for consent. (a) It can record the denial in its own history database and reject any future re-requests for that "cid" for that "uid". Here it is not generating a privacy policy. The "default-deny" principle would prevent the pPEP from granting access anyway. However the pPEP would then advise the data controller of "missing" consents. This leads to confusion and communication overhead. Thus we recommend (b) to generate and deploy a privacy policy for denied requests for consent just as for granted ones.

3.1.9 Consent handling preferences and consent automation

The consent manager should support some degree of automation of consent granting. For this the consent manager requires user-defined consent handling preferences. With these preferences the data subject authorises the consent manager to grant consent to certain applications (data controllers) on behalf of the data subject. If the consent manager has been supplied with sufficient authority via consent handling preferences, it can decide autonomously on granting a given request for consent. However also in these cases the data subject should be encouraged to review such automated decision afterwards (as sketched in Figure 7).

When consent requests can only partially be resolved in an automated manner, the work flow described in the previous subsubsections remains applicable. Some clauses of the request for consent will be resolved already. These clauses and the corresponding preferences are highlighted to the data subject. The data subject can review them during the manual resolution process. This eases the complexity of giving consent.

Such a model of "Semi-Autonomous Interactions for Ubiquitous Consent" based on **"Consenting Agents"** has been suggested in [99]. Applying the model described there, the RERUM consent manager acts as consenting agent on behalf of the data subject. The "Semi-autonomous consent (SAC)" model of [99] allows for preference elicitation being decoupled from the act of consenting itself. In a first phase, the data subject is setting the consent handling preferences for the given IoT scenario. There is "minimal distraction" for the data subject (see principles in Section 3.1.1), as primary objective of the data subject is exactly completing the task of instructing the consenting agent. In the IoT situation, the data subject does not need to interact with the consent manager anymore for the time being. After having left the IoT situation, the data subject is at liberty to concentrate on privacy protection issues once more. The data subject can log into the consent manager and review the automatically granted consents (as sketched in Figure 7), and revoke unsatisfactory consent decisions. Data subjects can adjust the consent handling preferences having lead to the unwelcome decisions. Thus gradually they may arrive at a working set of privacy preferences. Of course, the options made available to the data subject and the way it is presented to the data subject requires careful user interface design and needs to be tailored to the actual IoT situation very carefully to avoid misunderstandings and misconfigurations as far as possible. This issue requires further in-depth investigation beyond the scope of RERUM.

How can we elicit **consent handling preferences** from the data subject? What questions should the data subject be asked? A characterisation / classification of data controllers (e.g. based on trust or reputation) enables the data subject to formulate consent handling rules, just as would a classification of the available data sources, and a taxonomy or ontology of purposes common to data controller and data subject. The consent manager might offer a **preferences assistant** to guide the data subject through the process of making useful settings. More research is required on this topic, involving user interface design, and cognitive psychology aspects.

Coarse-grained high-level settings promise both to be comprehensible to and manageable by data subjects. For instance, the user could state that access is granted only to certain statistics of the user's data, or very coarse-grained location. Regarding the data controller, the data subject could specify that only data controllers with reputation ranking of "high" or above and with a certificate from a certain trust provider are allowed access to personal data. With respect of the type of purpose of data collection and processing the data subject may be more comfortable with some than with another. For instance population statistics may be acceptable, but individual behaviour analysis may be not.

The consent manager could offer setting of more detailed consent handling preferences, for instance on a per-data-source and per data-controller basis, and for a given specific purpose. The consent manager could visualise the available data sources (services, VEs and VRDs, see for instance Figure 12) together with their capabilities. Then the data subject can specify the data source, the degree of detail available to a given type of or a particular data controller for a specific purpose. Such a degree of detail however may overwhelm many data subjects. It may be hard for them to comprehend the consequences of such fine-grained settings, and to maintain and revise them later on.

Requests for consent sometimes may be in conflict with the data subject's privacy preferences. If a data subjects strongly desires a certain service or favours a certain data controller, they may be willing to grant exceptions from their privacy preferences for the corresponding request for consent. Rules for such exceptions may (with caution!) be set automatically based on preferences. The user may supply authorisation tokens (e.g. OAUTH tokens) to the consent manager to grant such exceptions at the privacy dashboard. For more details see Section 3.1.11.

3.1.10 Changes, revocation, and history

IoT infrastructures may expand, or shrink, or otherwise alter. For instance new data subjects are associated with data sources and others are dissociated from a data source. New consents become necessary and others obsolete. To be aware of changes, the consent manager needs synchronise with the respective registries (like the GVO registry and the user registry). The consent manager is responsible for generating and deploying privacy policies from consents. It also needs to clean up obsolete consents or clauses of consents. It must update or remove the policies derived from such consents.

Adding: In a given IoT infrastructure there may be new data subjects, new data sources, and new associations between a data subject and a data source. We want to avoid the user granting permissions relating to obsolete consents. We don't want unnecessary privacy policies being in effect when they are not really required. We also dislike requesting too many (different) consents. We consider the following cases:

- If a new data subject is participating in an IoT infrastructure, they may become associated with one or more data sources. Before an already existing data controller for a known purpose now can access one of those affected data sources, it needs to obtain consent from the new data subject. Here the complete request for consent needs to be reviewed by that new data subject.
- If an IoT infrastructure is expanding by adding further data sources (e.g. sensors), then consent needs to be asked for such a (non-exclusive multi-data-subject) data source of all (pre-existing) data subjects associated with it. If the new data source is to be accessed by a known data controller for a known purpose, then an existing request for consent can be updated for it. After the update has been consented to by the data subjects (that already had granted a previous version of that consent without the new data source), the newly derived privacy policy version super-seeds the old one.
- If an existing data subject is newly associated with an existing data source for the same purpose that it already has granted consent to, the proceeding would be the same as for the previous case. It would involve a consent update, for instance when a data subject is starting to frequent an additional office in RERUM UC-I2. If the purpose is a new one, a complete request for consent becomes necessary.
- In RERUM UC-O1, smart phone owners join the IoT by registering their smart phone and donating its sensors as data sources. A new data subject and its new single-user sensors are becoming part of an existing IoT infrastructure. Before the data controller can to access one of those sensors, it must request consent for it. The only data subject concerned is the smart phone owner, who will once grant the RERUM UC-O1 consent for all own sensors, like other smart phone participants before (see Subsection 5.1.1).

Revoking: A *data subject* can not only withhold consent initially, but also revoke a consent at any time, whether it had been granted automatically or manually. The data controller must be informed of this decision to allow for compliant behaviour. For revocation of consent the data subject in the consent manager accesses a list granted consents in the consent history database. The data subject can select the consent(s) to revoke. For revoked, expired, superseded, and otherwise obsolete consents the consent manager must remove the corresponding privacy policies and inform the data controller. This includes cancelling exceptions granted from a data subject's privacy preferences for this consent in the privacy dashboard. For details on this topic see Section 3.1.11. Also *data controllers* may decide (with effect for the future) they do not need a previously granted consent anymore. Of this decision they should inform the consent manager which can automatically revoke that consent and inform the users concerned.

Removing: A (part of a) consent may become obsolete for instance if a data subject is no longer associated with a data source or if a data source or a data subject permanently leave the IoT infrastructure. It is important to remove obsolete policies. For instance, if a data subject had denied access to a data source, but all other data subjects associated with that resource had allowed it, access would become possible for the data controller to that data source, once the blocking data subject had been dissociated from the data source. In the case of withdrawn data sources still for matters of efficiency and good management it is advisable to clean up obsolete consent and policy clauses immediately.

Reconsidering: In case of an initially declined request for consent, the data subject could wish to revoke such a decision. Care should be taken to inform the data controller of such a decision. The situation can be handled in various ways.

- One approach is to revoke the derived "deny" policies and to allow the data controller implicitly to request that consent again. This would happen, if the data controller tried to access a desired data source again. Then the pPEP would then indicate a "missing" consent, and the data controller then could approach the consent manager again to obtain that consent.
- Otherwise the consent manager on behalf of the data subject could explicitly inform the data controller on the data subject's wish to reconsider a previous denial and invite it to re-submit that request for consent. This way we ensure the data controller is aware of a potentially granted consent and still desires to have it.
- If the timeout for granting consent has not yet expired at the time the data subject reconsiders an initial denial (which we assume has not been indicated to to data controller), it is sufficient to alter the decision. The consent manager removes the previous "deny" policy and installs the new "grant" one. Then it calls back the data controller to inform it of the newly granted consent.

All these decisions and actions need to be recorded in the consent manager's *history database* for future reference by the data subject (or auditors). For each consent request, the data subject can look up the complete history, and re-read the request for consent and its circumstances. The history of consent decisions also forms a basis for future decisions.

3.1.11 Relationship to the Privacy Dashboard

In RERUM data subjects can adjust the privacy settings of individual sensors, actors, and services in the privacy dashboard (see Figure 16). The can also block and unblock data transmission with the help of the deactivator / activator of data collection (see Figure 18), whose GUI is provided by the privacy dashboard. Both these ways are independent of any consent granted by the data subject. Granting of consents however is a third method for the data subject to express their privacy protection needs.

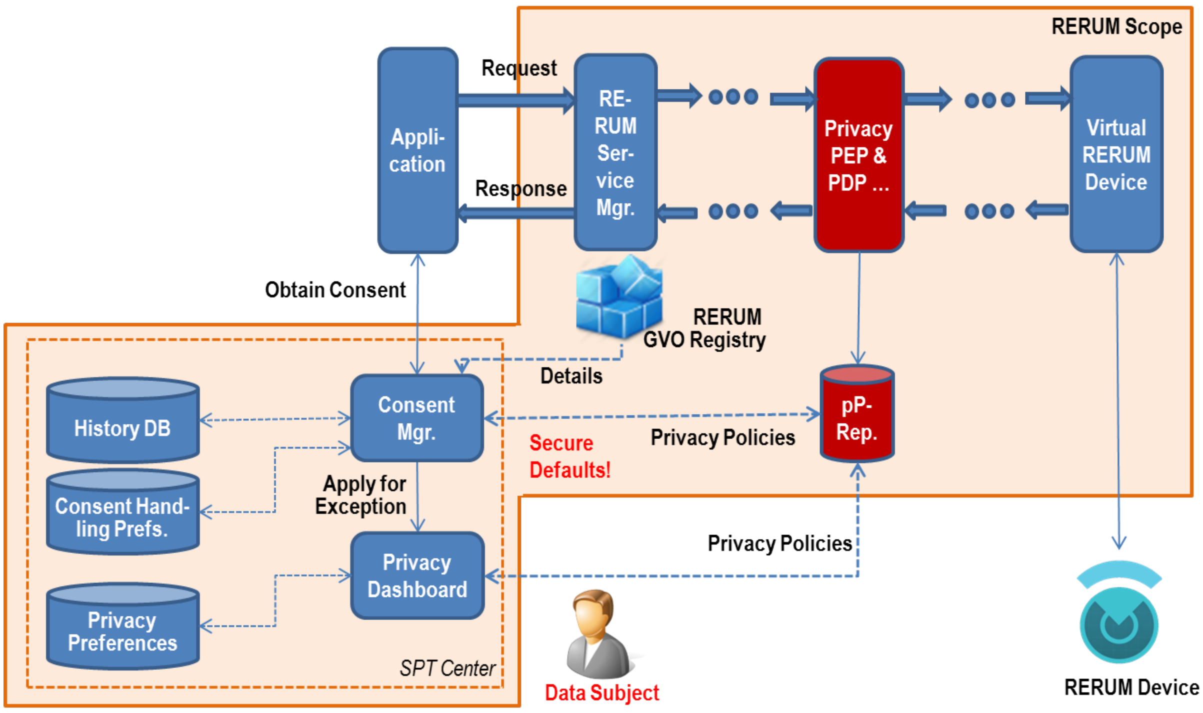


Figure 16: Interworking between Consent Manager and Privacy Dashboard

The data subject alters privacy settings, requests activation or deactivation of data collection, and grants or denies requests for consent at their will. However what will happen when there are conflicts? How are conflicting directions in the various privacy protective components be reconciled? We address this issue by defining a default precedence strategy.

- The settings in the activator/deactivator of data collection have priority over any privacy policy, because this way the data subject can temporarily suspend surveillance without needing the revoke privacy policies.
- Privacy policies generated from privacy preferences set in the privacy dashboard take precedence over privacy policies derived from consents, because data subjects have a right to define and enforce their own privacy well-being environment.
- If a data source associated with a data subject is addressed by no privacy policy of that data subject, access to it is denied per default for any data controller. This observes a fundamental IT security and privacy principle ("secure defaults").

The privacy dashboard and the consent manager need to indicate clearly to the data subject which privacy preferences and (parts of) consents are overruled by which activator/deactivator settings and/or privacy settings respectively. The activator/deactivator settings overrule any privacy policy at least temporarily. Regarding conflicts between privacy settings and consents, the data subject may resolve the conflict in several ways:

Accept the precedence: The data subject is agreeable to privacy preferences overruling some parts of a consent. If privacy policies based on preferences obscure privacy policies based on consents, for instance an opaque (or even transparent) XACML rewrite may allow the data controller to get at least some data (discussion see below).

Grant exceptions to preferences: If a data controller is not getting the quality of data they need, the data subject may not be getting the quality of service they desire. In such cases the data subject may be willing to grant an exception to certain privacy preferences for a given consent. This may be part of the dialogue the consent manager is conducting with the data subject when handling a request for consent. The data subject is made aware of conflicts. They can decide to grant exceptions to the conflicting privacy preferences (see Figure 16). These exceptions must be listed clearly in the privacy dashboard (and indicated in the consent manager history). Exceptions need to be removed automatically once the consent becomes obsolete.

Feedback on request for consent: Via feedback mechanisms, a data subject can inform a data controller of discrepancies between the data subject's desire for privacy and the data controller's wish for input. They can ask for adaption of subsequent releases of requests for consent.

One may consider automated granting of exemptions is to be included in the consent handling preferences of a data subject. In this case we propose the data subject supplies OAUTH-style authorisation credentials to the consent manager, which then can set necessary exceptions in the privacy dashboard on behalf of the data subject (see also end of Section 3.1.9 and towards the end of Section 3.1.13).

Figure 17 shows a message sequence where a potential conflict between a data subject's privacy preferences and a new request for consent is resolved by granting exemptions. The work flow starts as described in Section 3.1.7 with data subject "user 1" desiring to (1) review the request for consent "cid". The consent manger authenticates the user. The consent manager starts compiling the presentation of the request for consent "cid". It needs information whether current privacy policies based on privacy settings would shadow policies based on that request for consent.

To this end the consent manager creates a hypothetical security context (HSecurityContext). This pretends the data controller is placing a request for action. The consent manager analyses the request for consent and lists all basic actions referenced in it. For instance the data controller wants to read the data from sensor 1 with a certain precision. The consent manager then (2) interacts with the pPDP. It wants to find out, which of the actions the pPDP decides about on basis of policies created from privacy preferences, and what these are (reason=yes). The pPDP decides this hypothetical question and indicates blocking preference(s) in its (3) "deny(reason)" response. Step by step the consent manager compiles the ConflictInfos. These the consent manager includes in the (4) presentation of the request for consent. As before, the data subject starts reviewing the information provided. In the course of the review the data subject is informed about the policy conflicts to be expected and about the privacy preferences involved. The data subject decides to (5) grant exceptions to some preferences for this consent "cid". For this the data subject may log into the privacy dashboard. After (6) successful granting of exceptions the data subject can (7) approve of the request for consent and the work flow proceeds as described before in Section 3.1.7.

3.1.12 Deploying and Enforcing Privacy Policies

There are several sources of privacy policies in RERUM.

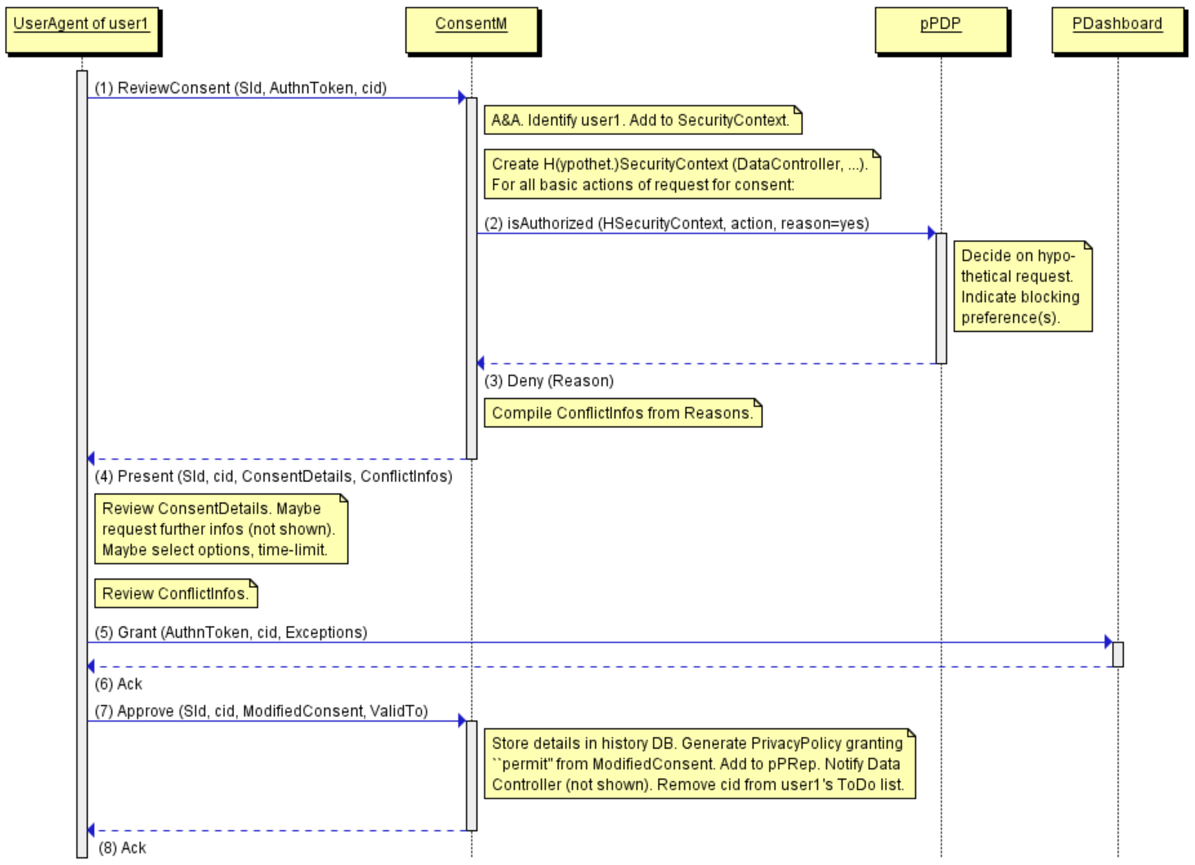

UserAgent of user1
ConsentM
pPDP
PDashboard
(1) ReviewConsent (SId, AuthnToken, cid)
A&A. Identify user1. Add to SecurityContext.
Create H(ypothet.)SecurityContext (DataController, ...). For all basic actions of request for consent:
(2) isAuthorized (HSecurityContext, action, reason=yes)
Decide on hypothetical request. Indicate blocking preference(s).
(3) Deny (Reason)
Compile ConflictInfos from Reasons.
(4) Present (SId, cid, ConsentDetails, ConflictInfos)
Review ConsentDetails. Maybe request further infos (not shown). Maybe select options, time-limit.
Review ConflictInfos.
(5) Grant (AuthnToken, cid, Exceptions)
(6) Ack
(7) Approve (SId, cid, ModifiedConsent, ValidTo)
Store details in history DB. Generate PrivacyPolicy granting ``permit" from ModifiedConsent. Add to pPRep. Notify Data Controller (not shown). Remove cid from user1's ToDo list.
(8) Ack

Figure 17: Conflict resolution during consent granting sequence diagram

- The RERUM Consent Manager is a shared component for all data subject users of the local IoT and needs to be trusted both by data subjects and data controllers. The Consent Manager creates and deploys the *privacy policies based on consents* (see Figure 18) in the privacy policy repository (pPRep).
- These consents (see previous subsubsection) are prioritised lower than the *privacy policies based on personal privacy preferences* of the data subject. The RERUM Privacy Dashboard (see Section 3.4) allows the data subjects to view and alter their privacy settings regarding the local IoT. The data subject's personal privacy settings are translated into privacy policies and stored in the pPRep as well.

Privacy policies need to be deployed and enforced to take effect.

We recommend a dedicated privacy policy enforcement infrastructure (see red boxes in Figure 18), separate from the security policy enforcement infrastructure (green boxes). The privacy policies and settings are controlled by the data subject and submitted to and observed by the privacy policy enforcement infrastructure. The security access control policies are controlled by the administrator of the RERUM IoT infrastructure and submitted to and executed by the security enforcement infrastructure. The privacy policy enforcement infrastructure could even potentially be installed by or on behalf of the data

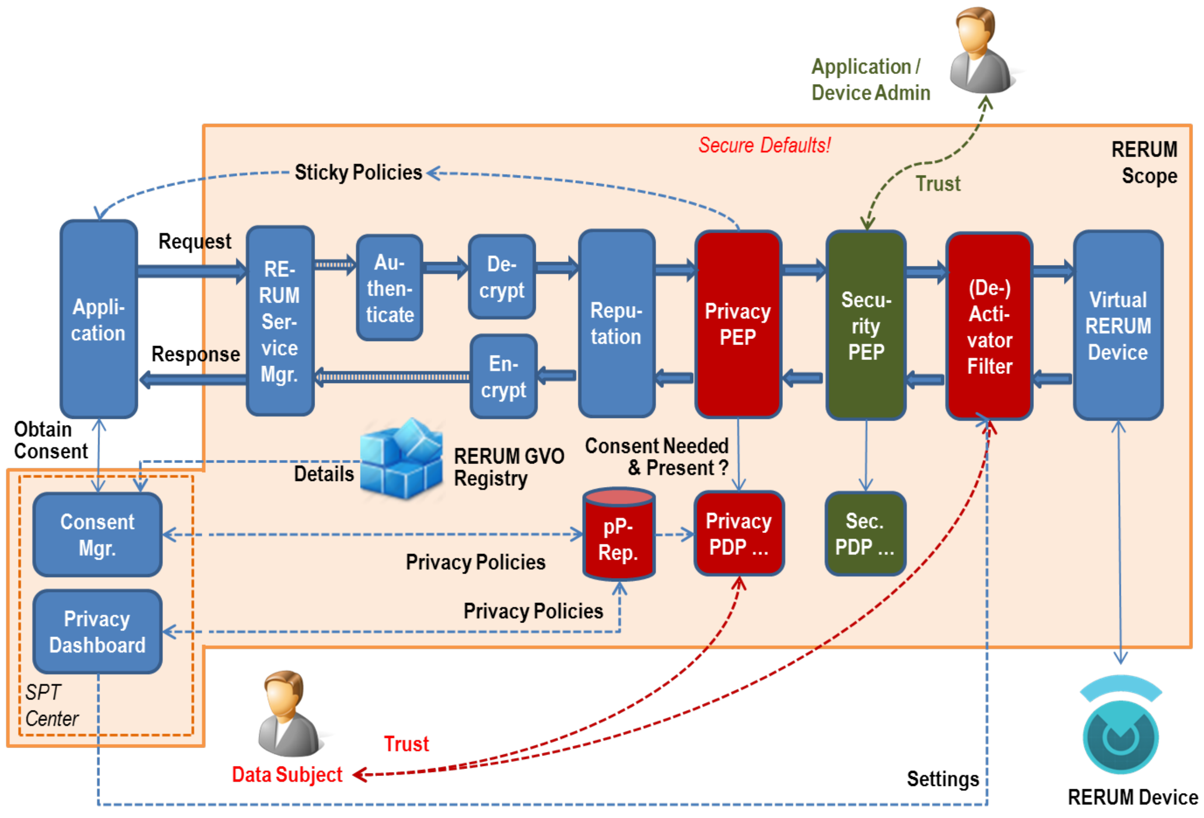


Figure 18: Separation of Security Policy and Privacy Policy Enforcement

subjects using trusted software and maybe also hardware selected by the data subjects.

The position of the security and privacy policy enforcement components in the flow of personal data in Figure 18 is just exemplary. For instance it may make sense to enforce privacy policies after security policies for incoming requests, letting the security policy enforcement point filter out obviously undesirable requests first. For outgoing responses it usually will be preferable from a data subject's point of view to enforce privacy policies again as last level of data subject control after the security policy enforcement has taken place.

Note, that besides privacy policies, there is also the RERUM Deactivator and Activator of Data Collection, whose GUI is provided by the Privacy Dashboard (see Section 3.3). That component allows the data subjects to block data transmission of individual VRD, VE and associated services temporarily without the need to alter privacy settings. These settings are taking preference over any policy settings.

Sticky Policies (derived from a consent, see Section 4.1) may be attached to data to ensure the data controller knows which compliant behaviour the data subject is expecting. Sticky Policies can't be enforced. Rather they enable good-natured data controllers to behave in a compliant manner.

3.1.13 Mapping to existing standards

In XACML [197] [150] nomenclature the consent manager is a kind of *policy administration point* (PAP). It allows for the creation, modification and revocation (deletion) of privacy policies. Usual sub-components of a PAP are a *policy repository* (PRep) for storing the policies and policy sets as well as a *policy authoring function*. It needs access to the policies, policy sets and policy metadata, if any. The consent manager uses a *privacy policy repository* (pPRep) which is also part of the privacy *policy enforcement point* (pPEP) infrastructure described in Section 3.2. It generates privacy policies from requests for consent and stores them in the pPRep.

An access control policy specifies that "consent is needed". The GVO registry (see D2.3, Section 6.6.2) records which data subject(s) are associated with a particular data source. Some component must provide a functionality to retrieve a list of data subjects associated with a given data source. The consent manager must be able to figure out, who is concerned by a given request for consent and ask all those data subjects for their consent. The pPDP also needs such a functionality to check whether all data subjects concerned have given their consent. In XACML nomenclature, a privacy *policy information point* (pPIP) in cooperation with the GVO registry and the pPRep provides this functionality.

The RERUM GVO registry represents an IoT device registry. The device (sensor, actor, ...) specifications contained in it are referenced by the policies stored in the privacy policy repository (pPRep). The pPRep data base in XACML nomenclature is part of the privacy policy enforcement infrastructure (see Section 3.2). A privacy *policy retrieval point* (pPRP) gets the policies from it, on request of the privacy policy decision point (pPDP). The pPRep contains the privacy policies based on the privacy preferences of as well as the consents granted by that the data subject. The policies are primarily generated by the consent manager and the privacy dashboard. The consent manager translates an (adjusted) request for consent into an appropriate XACML privacy policy and submits it to the pPRep, where the pPRP can access it. So if, after a callback or maybe in some cases a redirect, the data controller returns to the pPEP, the corresponding pPDP can make use of the newly generated privacy policy settings.

There could be e.g. consents missing or deny access, or data collection is temporarily blocked, or privacy preferences obscure privacy policies based on consents. Then one maybe could permit XACML *opaque rewrite* [197] in the pPEP to allow the data controller to get at least some data, even if not as exact as they would have liked. With opaque rewrite the data controller may not be getting the quality information they are expecting, even without being aware of this. With *transparent rewrite*, the data controller would be notified of the different quality, but privacy preferences and settings may get leaked this way. Use of transparent rewrite needs careful analysis of the individual deployment scenario.

The local IoT infrastructure could provide access to sensors, actors and services in form of web services. *Web service discovery* is the process of finding a suitable Web service for a given task. Web service providers augment a web service endpoint with an interface description using the *Web Services Description Language* (WSDL) [55]. So a data controller knows what service is available and how to use the service, when seeking access to private data. The RERUM GVO repository may cooperate with a component that provides such a WSDL interface description. This description the data controller could query to find out the specification of sensors, actors and services. That information would enable it to compose suitable requests for consent.

If need be, the consent manager may provide the data controller authorisation credentials in form of OAUTH tokens [111], together with service end point information. Then at the pPEP the data controller

Table 3: Technical implementation summary for Consent Manager

Consent Manager		
Technical Level (description given in 3)	Level 3	Component was not designed as such for the IoT-domain.
Suggested Method(s) for Implementation	See this subsection, with standards in Section 3.1.13.	
	See this subsection, and Sections 4.1 and 4.7.	
Technical Readiness of Implementation within RERUM	Design	yes
	Experiments, Simulations	no
	Trial	no

can present these OAUTH tokens to be permitted to gather sensed data and to process them as described in the consent content.

To reduce consent complexity and visualise consents in a human-user-friendly manner, techniques as described in Section 3.1.3 may be helpful, even if these aren't standards. To offer support for consent automation as described in [99] (Semi-Automated Consent - SAC) may also be a promising approach, which however requires further research out of scope of RERUM.

3.1.14 Consent for user attributes

Privacy policies may govern not only to the access to IoT data, but also to the access to traditional address-style and role-information user attributes. The RERUM consent manager also accepts such traditional requests for consent. If the data source is not specified in the RERUM GVO registry, this should be indicated clearly in the request for consent as an additional input parameter. This parameter, or rather indicator, is propagated to the privacy policy repository to allow different deployment of non-IoT privacy policies, if so desired by the privacy policy enforcement infrastructure administrator.

3.1.15 Summary

Data subjects often must consent when personal data are generated, collected, stored, and processed. A valid consent in the European mind set should be given prior to data collection, informed, specific for a purpose, voluntary, explicit, documented and revokable. Data subjects must comprehend the disclosed information and be competent to give consent, with their attention minimally distracted from the consent procedure.

Data subjects need information to grant or withhold consent in an IoT situation, for instance purpose of data collection, identity and trustworthiness of data controller, details of data collected and recording sensors, data processing, fusing, storing, and sharing practices, privacy impacts, user rights and controls, and technical and security related details.

Consent complexity may be reduced for instance by using simple language and standardised terms, icons, templates, and complete standardised requests for consent. Trust labelling and scoring, tracking of changes, and comparison of requests for consent allow for quick overview. Preferences of an negotiation with aware and educated data subjects can lower consent complexity and shape consent practices.

In IoT environments, the installed sensors, actors, data services, and other services (like aggregation or local pre-processing), their meta data and capabilities can be specified precisely, supplemented by ranges and wildcards. The RERUM "GVO Registry" provides an IoT device and service registry. Machine processable parts of requests for consent can be converted to privacy policies and be utilised by a privacy policy enforcement point.

A shared RERUM "Consent Manager", located conceptually at the RERUM "Security Centre", is provided per IoT infrastructure and supports privacy compliant behaviour. It needs to be trusted by data subjects and data controllers. The Consent Manager interacts with the data subject, who must be a conscious registered RERUM user, via a graphical interface. The Consent Manager displays the data controller's (application's) request for consent to the user, assists with negotiations and option selection, and obtains the user's consent or refusal. Privacy policies derived from granted or refused requests for consent allow to permit or deny the data controller access to data sources. Sticky policies may be attached to retrieved data to enable the data controller behaving compliant to the data subject's expectations. In XACML nomenclature the Consent Manager represents a Policy Administration Point.

The core Consent Manager functionality is to support the data controller in requesting consent and the data subject in granting and revoking of consent, and to derive privacy policies. The Consent Manager functionality also allows for (partial) consent automation based on user preferences in cooperation with the RERUM "Privacy Dashboard", maintaining consent history, updating of requests-for-consent, and protection from request-for-consent flooding. The Consent Manager may provide request-for-consent reading guidance, and visualise sensor and actor specifications of an IoT situation in form of a floor plan, provide data controllers with suitable request-for-consent templates, and supply data subjects with interaction and feedback facilities and consent review recommendations.

Privacy policies derived from user preferences specified in the Privacy Dashboard take preferences over those derived from granted requests-for-consents. In case of conflicts users are asked whether they wish to grant exceptions to their privacy preferences for certain requests-for-consent.

As summarised in Table 3, the Consent Manager has not been a component up to now designed as such for the IoT domain. In this section we have outlined the IoT specifics a consent manager component. These specifics we can take advantage of to ease the data subject's burden of consent. We pointed out a several implementation options, especially in Section 3.1.13. More details on sticky policies are given in Section 4.1. Consent for authorisation is addressed in more detail in Section 4.7. The description of the Consent Manager in this section provides a high-level design for the IoT domain. Experiments, simulations or trials of the consent manager component are not scheduled within the context of RERUM.

3.2 Privacy Policy Enforcement Point

Deliverable D3.1 [201] already introduced the concept of a Policy Enforcement Point (PEP) for access policies. In very short, a PEP is a component that intercepts the communications to the service and decides whether to grant access to them or not. D3.1 already explained how XACML policies could be used for defining security criteria for accessing a service. Section 3.1 shows how the privacy criteria can be defined in a similar way with XACML v3.0 with the privacy extension. In a similar way, RERUM reuses the same principle for the privacy policies: Using a component able to interpret XACML policies, RERUM uses that component to interpret privacy policies corresponding to the resource to be invoked.

Hence, used jointly with a consent manager, a PEP can be used to check that the access to data is made taking not only security criteria into account, but privacy criteria as well. In concrete, this feature corresponds to Contribution 3: Lightweight and Efficient Pseudonym System of D2.1 [167] and will be shown in trial scenario T-UC-I2C in D5.1 [168]. This PEP specialized in privacy criteria is named Privacy Policy Enforcement Point (pPEP), which is the object of this section. To distinguish the pPEP with the PEP used for defining security rules, we will name the latter as Security Policy Enforcement Point (sPEP).

The Privacy Policy Enforcement Point will work in a similar way as the PEP already presented in D3.1 [201] for the access control. That is, it will act as a filter in the way introduced in D2.3 [219] and will:

- intercept incoming http requests,
- check the integrity of a security token included in the request for obtaining user attributes,
- execute a crossed check of the user attributes and the rest of the information contained in the request against the privacy policies provided by the consent manager, and
- if the privacy policy grants access to the information requested, it will let the request pass to the next element in the chain of filters, which will normally be the sPEP.

In any other case, it will reject the request by returning a HTTP reject code instead.

Figure 18 shows how the pPEP fits in the overall process of serving requests and how is related with the sPEP. As Figure 18 shows, it is foreseen to have two different PEPs (pPEP and sPEP) for evaluating Privacy and Security policies independently. The privacy policies are generated by the Consent Manager and privacy dashboard, and both the sPEP and pPEP are integrated in RERUM following the chain of filters already presented in D2.3 [219].

The main advantage of this conceptual view is that it allows to have different providers for both the sPEP and pPEP. This will allow, for instance, that a Data Controller, which is the entity legally responsible for enforcing the proper access to these data could delegate this control on an external pPEP provided by a trusted third party. That would allow the Data Controlled to focus on its own business logic while the Data Processor providing the RERUM service focuses on providing and checking the security policies.

But this conceptual view has also a very big drawback, especially in terms of performance. The concept of chain of filters is very powerful because it allows setting up a potentially infinite number of filters before the request, either for enriching it or to ban it. But it requires that in each step of the chain the request is forwarded to the next step, which will increase the processing times and network load for the requested service. For this reason, the implementation of the pPEP for the RERUM prototype follows an intermediate but more pragmatic approach that Section 3.2 describes in detail.

This split between a pPEP possibly provided by a Data Processor and a sPEP provided by the Data Controller, in this case RERUM, brings another question: Who trusts whom? The pPEP on the sPEP or the reverse? By definition, it is the data controller who obliges the Data Processor to sign a PLA (Privacy level agreement) that legally entitles the Data Processor to enforce the security constraints. Hence, the Data Controller would trust the Data Processor to include this pPEP as a filter to the process. RERUM adds additional controls to this that are explained in Section 3.10.

3.2.1 Privacy enforcing feasibility on delegated scenarios

Till now, we have been talking about RERUM applications acting on behalf of a human user that gets authorized on a RERUM user basis, that is, each RERUM application trying to access any RERUM service gets authorized (or rejected) based on the user that they are using to access RERUM and, as such, this used needs to be a valid RERUM registered user.

But RERUM is supposed to support IoT. In IoT, it is legitimate that an application mines external platforms, such as RERUM, to gather data that will be used for their own purposes. In fact, in RERUM, the Tarragona trials are built on this concept. The RERUM Applications built for these trials do not access the RERUM services on behalf of the human beings that will be accessing to them (mainly for a matter of performance). Instead, these applications basically consists in two parts: One part is a batch program that is executed retrieving data from the RERUM services, and the other part is a graphical application that allows human users to have access to the functionalities provided by these applications based on the data retrieved. That is, the human beings of these RERUM applications are authenticated for those applications, but not for RERUM, because these applications, which are not part of RERUM, use their own authentication and authorization mechanisms and do not register their users in RERUM. The user that will be utilized to authorize the access to the RERUM services will indeed be a valid RERUM registered user, but a single one specifically created for the batch program accessing the data.

Though it is possible to declare a different purpose for each batch process and even different registered users for each of them, this is a strong limitation for the privacy enforcement, because it will only be possible to define policies that declare:

- Access granted to a whole set of data corresponding to the set retrieved by the corresponding batch program for a given purpose or
- Access granted to a given application instead of individual users for a given purpose.

The impact of the first item can be reduced by grouping the access to closely related pieces of data, such as 'First Name' and 'Last Name', according to EU recommendations. An additional possibility is to define multiple security policies for each of the group of data accessed for ensuring that the application only accesses the data that it is allowed to.

The second limitation, however, is much tougher to enforce and at least cannot be enforced directly from the RERUM framework. Once the data has gone out to any system, there is currently no technical way to enforce that the data provided will be used with the purpose that it was declared to be used. Of course, PLAs can tie legally Data Controllers accessing these data, but this only entitles them to legal responsibility in case of non-compliance, instead of ensuring they will comply.

In this concrete case, any system providing data needs relying on the Data Processor on fulfilling the obligations it is legally bound to. But there is still a way that the RERUM framework could help on this:

Should the applications accessing RERUM services installed their own PEP such as the RERUM one, they would be able to check the access of their users to the services provided by them to ensure compliance with the Privacy Policies. Of course, this brings the problem of how to properly distribute the privacy policies to these PPEPs installed outside RERUM from the consent manager, but this is discussed on the Section 4.5.2.

3.2.2 Combination of multiple policies

The inclusion of Privacy Polices implies the need for being able to combine several security policies for a given resource. Till now, security criteria could be defined in a single security policy, but the inclusion of privacy policies brings the need for combining more than one single policy because a single RERUM service may access several piece of data, each of them protected by a single privacy policy. This implies that for checking the privacy of accessing a given RERUM service, IT will be necessary to check several privacy policies in a single operation. This can be achieved by defining Policy Sets with a special XACML policy, to be generated when creating the privacy policy and more exactly when associating it to a given resource. More details on how the Policy Sets for privacy are created can be found on Section 4.4.1.

Moreover, as Privacy Policies are likely to be associated to specific pieces of information rather than individual resources, it is needed to support different levels of Policy applicability, so some policies can be applicable at Global level (for any services) while others are applicable only locally for the service being called. This will be achieved with the XACML Policy sets as well. Figure 19 shows how this is achieved with a proper assignment of Policy Sets.

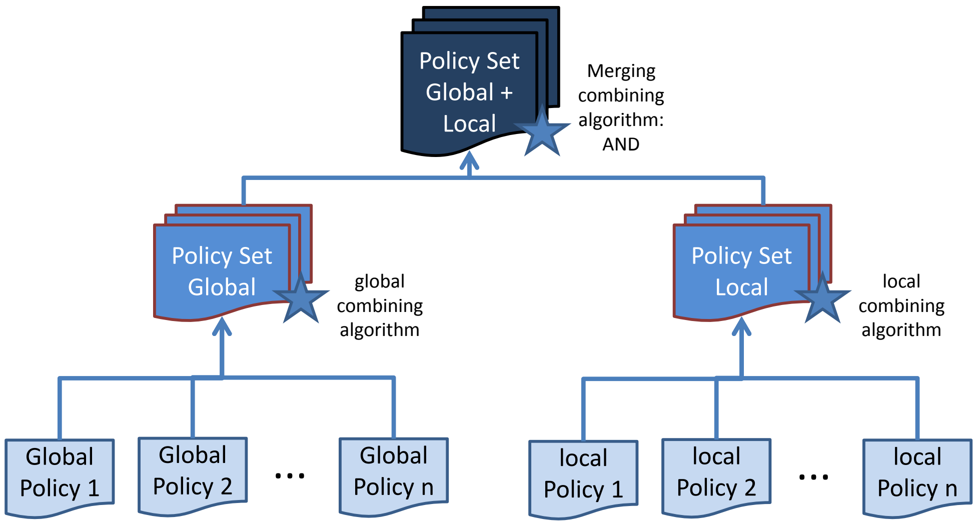


Figure 19: Combining local and global policies

In short words, all policies are independent from each other and are applied together using a policy set. This policy set includes both the identifiers of the policy files it is referring to and the logic criteria to join them all, that is, the combining algorithm. In our concrete case, local and global policies are combined together with their respective local and global combining algorithms and finally and AND operation is carried out between them, so both local and global policies are fulfilled.

The drawback of this approach is that it is necessary to refresh all the applicable policies in the policy sets each time there is a new policy or an existing one is removed, but this is carried out during the deployment process.

In the concrete case of the policy files, however, there is an additional policy set due to the way they are generated. In concrete, both the Consent Manager and the Privacy Dashboard can produce Privacy Policies to be evaluated by the pPEP. But Privacy Policies from the dashboard have priority on the ones from the Consent Manager. For this reason, they include an additional Policy Set to set that. The Figure 20 shows how this is achieved for the privacy policies.

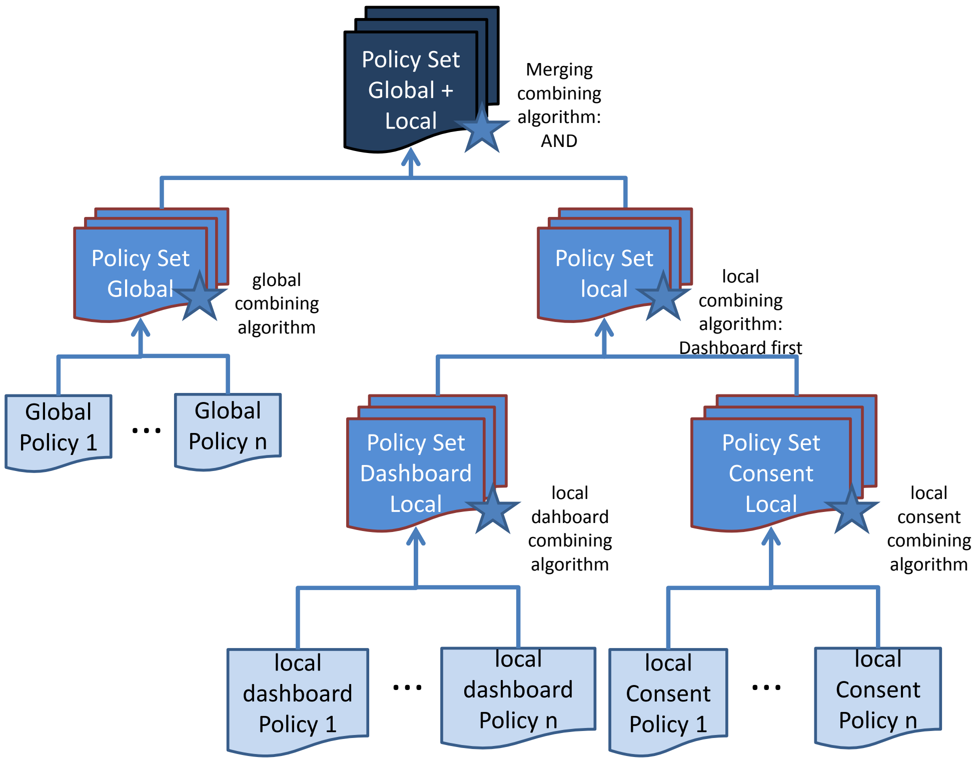

Policy Set Global + Local
Merging combining algorithm: AND
Policy Set Global
global combining algorithm
Policy Set local
local combining algorithm: Dashboard first
Global Policy 1
...
Global Policy n
Policy Set Dashboard Local
local dahboard combining algorithm
Policy Set Consent Local
local consent combining algorithm
local dashboard Policy 1
...
local dashboard Policy n
local Consent Policy 1
...
local Consent Policy n

Figure 20: Combining local and global privacy policies

The procedure is basically the same with an additional layer for giving precedence to the dashboard privacy files on the ones generated by the consent manager.

3.2.3 Summary

RERUM reuses the authorization components already defined in D3.1 [201] by upgrading them so they can work with different policy stores and combine multiple set of policies. This provides a mechanism to evaluate and enforce the privacy policies generated by the Consent Manager.

However, in delegated scenarios, such as the ones run in Tarragona or in traffic monitoring applications, it is possible that the application decides that their users do not register in RERUM, but use a single RERUM registered user that is specifically created for that application. In that case, RERUM can only check the access for this specific RERUM registered user. The same is applicable for privacy policies. As a result, privacy policies for such applications need to be defined based on the application that is trying to access the data instead of the people that will later access to it.

Figure 4 summarises how this section contributes to the state of the art.

Table 4: Technical Implementation Summary for Privacy Policy Enforcement Point

Privacy Policy Enforcement Point		
Technical Level (description given in 3)	Level 3	Though there are already authorization components suitable for evaluating privacy components, the ability of RERUM authorization engine to evaluate security / privacy components based on information included even in the body of the request and in a generic way has never implemented before in an IoT environment.
Suggested Method(s) for Implementation	reuse of D3.1 authorization components, see Section 3.2	
	Upgrade D3.1 authorization components for supporting different policy stores and combining local and global policies, see section 3.2	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	Yes

3.3 Deactivator/Activator of Data Collection

Data minimization is one of the core principles of privacy-by-design. RERUM ensures on a scenario-basis that the collection of personal data is minimized as far as possible, at best dispensed with at all. Where personal data collection is unavoidable in some scenarios, RERUM strictly follows an opt-in approach in compliance with the European mindset on privacy protection. That means that data are collected only if the user actively allows a RERUM Device to do so. An activator / deactivator of data collection is provided specific for RERUM scenarios, such as in the smart transportation use case. The user starts transmitting data, when he/she actively installs a smartphone application and then (privacy by default) explicitly switches on data collection.

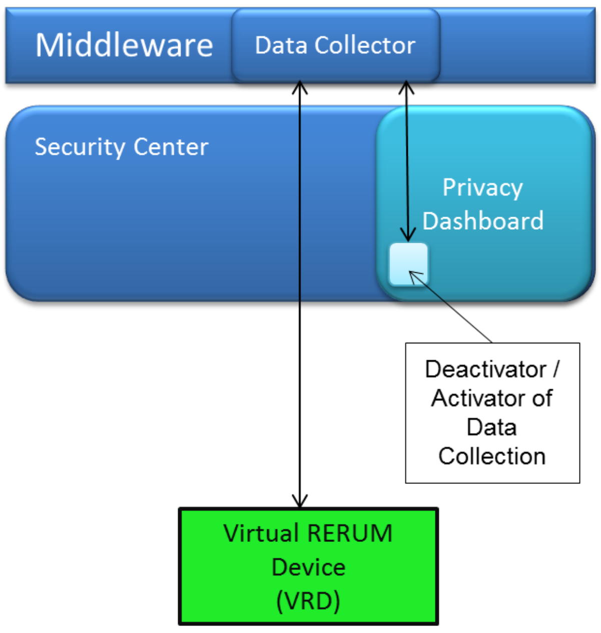

Middleware
Data Collector
Security Center
Privacy Dashboard
Deactivator / Activator of Data Collection
Virtual RERUM Device (VRD)

Figure 21: Location of Activator / Deactivator of Data Collection (as described in D2.3)

In D2.3 [219] the location and the functionality of the Activator / Deactivator of Data Collection and its relation with other Middleware components was described, see Figure 21.

3.3.1 Functionality

The sequence of actions is as shown in Figure 22:

In Figure 22, we assume that there is a device which continuously collects data from a user. The device can be a simple sensor platform and does not provide an interface for opting in or out of data collection.

1. The device collects data and sends it to some application in the cloud. This communication was approved by the user initially.

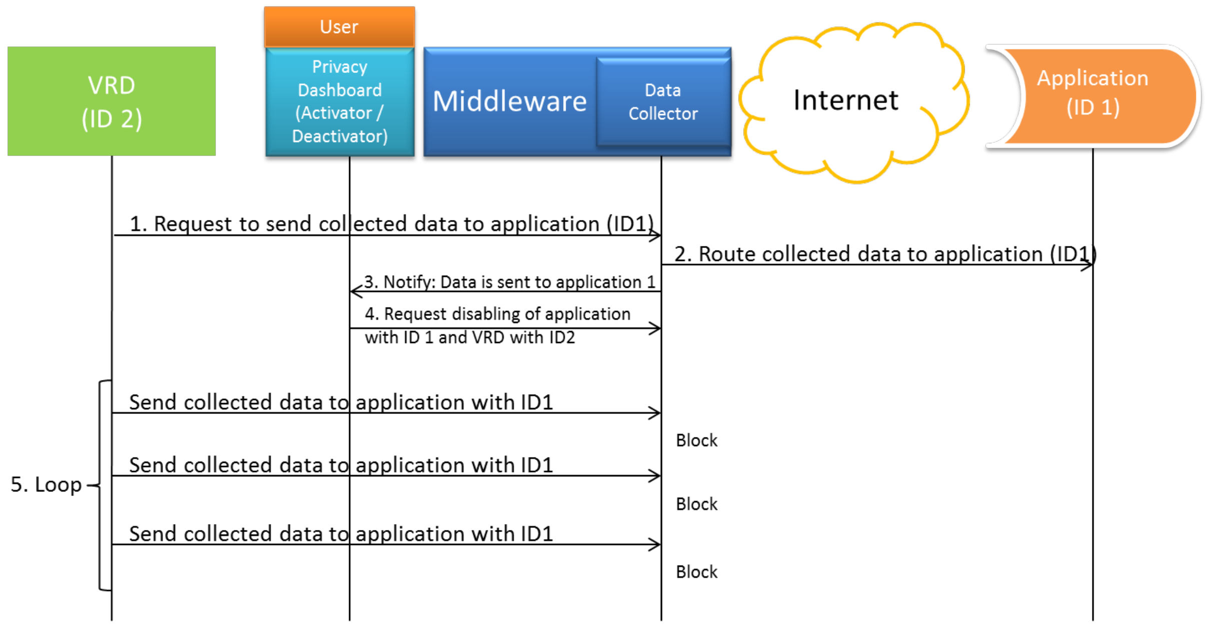

User
Privacy Dashboard (Activator / Deactivator)
VRD (ID 2)
Middleware
Data Collector
Internet
Application (ID 1)
1. Request to send collected data to application (ID1)
2. Route collected data to application (ID1)
3. Notify: Data is sent to application 1
4. Request disabling of application with ID 1 and VRD with ID2
Send collected data to application with ID1
Block
Send collected data to application with ID1
Block
Send collected data to application with ID1
Block
5. Loop

Figure 22: Sequence of actions in case of a data collection opt-out by means of the Deactivator / Activator of Data Collection (from D2.3)

2. The Data Collector collects the data from RDs and routes them to the desired destination.
3. At the same time, it notifies the Activator / Deactivator of Data Collection, that data are being sent.
4. The user logs in to his Privacy Dashboard (see below) and decides to opt-out from the data collection of device ID2. He/She clicks on a button and hereby opts-out of the data collection. The Privacy Dashboard notifies the Activator / Deactivator of Data Collection, which notifies the Data collector in the Middleware to block any requests from the application ID1 to the device ID2.
5. Whenever the device ID2 tries to send data or the application ID1 tries to request data, the intermediary Data Collector will block any message or request from either the device or the application.

Finally, the user may opt-in and again allow device ID2 to send data or he/she may physically shutdown the device.

3.3.2 Conflicts with privacy policies and preference policies

A user can determine who may access his data by defining privacy policies. If the user has agreed on a purpose by recording his consent and providing it to the service provider, the service provider will rely upon that consent to ensure a proper service provision.

In the privacy dashboard, see Section 3.3, a user can define which services he prefers. Thus a user devices may automatically consume a service and reveal personal information, whenever a preference policy applies.

The Deactivator / Activator of Data Collection interrupts every data transition from a defined device. It is hereby irrelevant if the service provider relies on the agreement of concept or if preference policies dictate an automatic agreement of service provision by a provider. The Deactivator / Activator of Data Collection is an explicit opt-in tool, which overrides every previously defined policy if the user decides to opt-out of a service, supporting the user's rights to *collection limitation* and *individual participation and transparency* (see privacy requirements defined in D2.2 section 2.6.3 [62]).

An exemplary behaviour of the Deactivator / Activator of Data Collection after the definition of privacy and preference policies is showed in Figure 23.

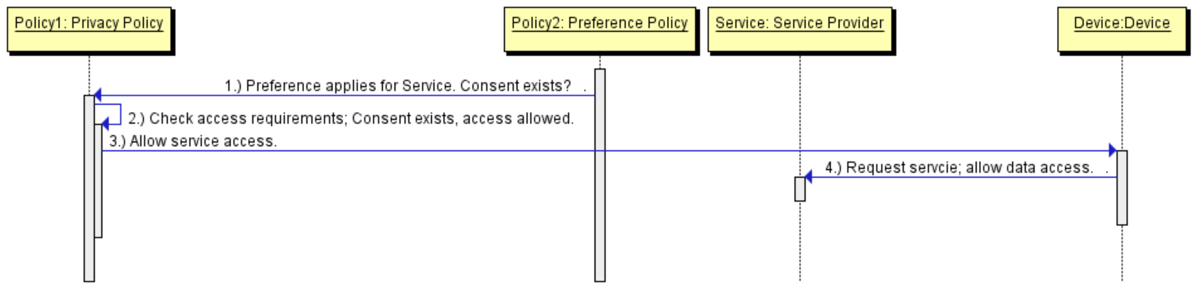

Policy1: Privacy Policy
Policy2: Preference Policy
Service: Service Provider
Device:Device
1.) Preference applies for Service. Consent exists?
2.) Check access requirements; Consent exists, access allowed.
3.) Allow service access.
4.) Request servcie; allow data access.

Figure 23: Interplay of privacy and preference policies

The steps are as follows:

1. A preference policy states that a device should consume a service when a given context applies. When the context applies, a consume control is triggered for a device. Before the device consumes the service, consent and privacy policies have to be checked.
2. Privacy policy are checked to see if consent was given and access is allowed for that service. While one could assume that access is granted due to the user's preference for that service, the service could have changed its purpose for data processing and therefore need a renewed consent.
3. Assuming that the service has not changed, consent was given and policy requirements are fulfilled. The device gets a notice to grant access to the service.
4. The device requests the service and allows access.

Up to this point, Figure 23 explains how privacy and preference policies play together. Figure 24 shows how the Deactivator / Activator of Data Collection allows explicit interaction of the user.

This sequence is extended by the Data Collector of RERUM's middleware. The Activator / Deactivator is a component that interplay with the Data Collector to achieve the user opt-out. The steps are the following:

1. The service requests data from the device. The requests is sent to the Data Collector.
2. The Data Collector redirects the request to the Device.
3. The Device responds with the requested Data.
4. The Data Collector redirects the data stream of the Device to the Service.

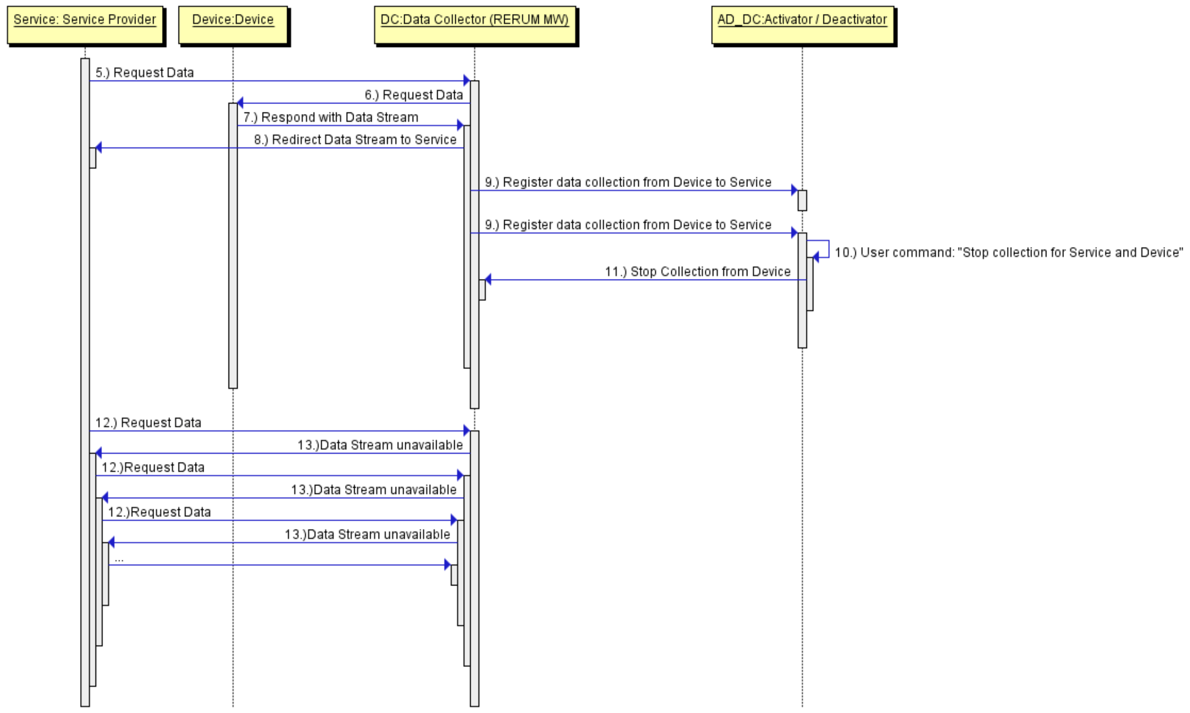

Service: Service Provider
Device:Device
DC:Data Collector (RERUM MW)
AD_DC:Activator / Deactivator
5.) Request Data
6.) Request Data
7.) Respond with Data Stream
8.) Redirect Data Stream to Service
9.) Register data collection from Device to Service
9.) Register data collection from Device to Service
10.) User command: "Stop collection for Service and Device"
11.) Stop Collection from Device
12.) Request Data
13.)Data Stream unavailable
12.)Request Data
13.)Data Stream unavailable
12.)Request Data
13.)Data Stream unavailable
...

Figure 24: Behaviour of data collection in case of privacy and preference policies and deactivation

5. The Data Collector registers a data stream from the Device to the Service at the Activator / Deactivator. From this point, the user is informed at the Privacy Dashboard of an existing data transmission and he can activate / deactivate the data collection.
6. The users decides to stop the collection. He does not want to refuse the existing consent or the agreed terms with the service, he just wants t stop the service for a short period.
7. The Activator / Deactivator sends a command to the Data Collector to stop redirecting data for the Device and the Service. If another Service is consuming data from the Device, it will not be deactivated until the user explicitly deactivates this communication as well.
8. The service provider is unaware of the deactivation and keeps sending data requests to the data collector.
9. The data collector responds with an error message, as if the Device was powered off.

The last message is formed as an error message to avoid revealing that the user deactivated the collection. It might be privacy sensitive to inform the service provider in which moment the user decides to stop the service. In general, a ranking can be determined: *Preference Policies* determine when a service request is triggered, but it does not allow to consume a service by itself. *Privacy policies* state if a service is allowed to access personal data, checking existing consent, access requirements, and so on (see Section 3.1 for details). *The Activator / Deactivator* interrupts the collection of data independent of given consent or preferences. It is an explicit interaction of the user, which supersedes every policy.

The activation / deactivation is done by the user in the Privacy Dashboard. The Dashboard is a graphical mashup of privacy functionality, further described in the following section.

Table 5: Technical Implementation Summary for Deactivator / Activator of Data Collection

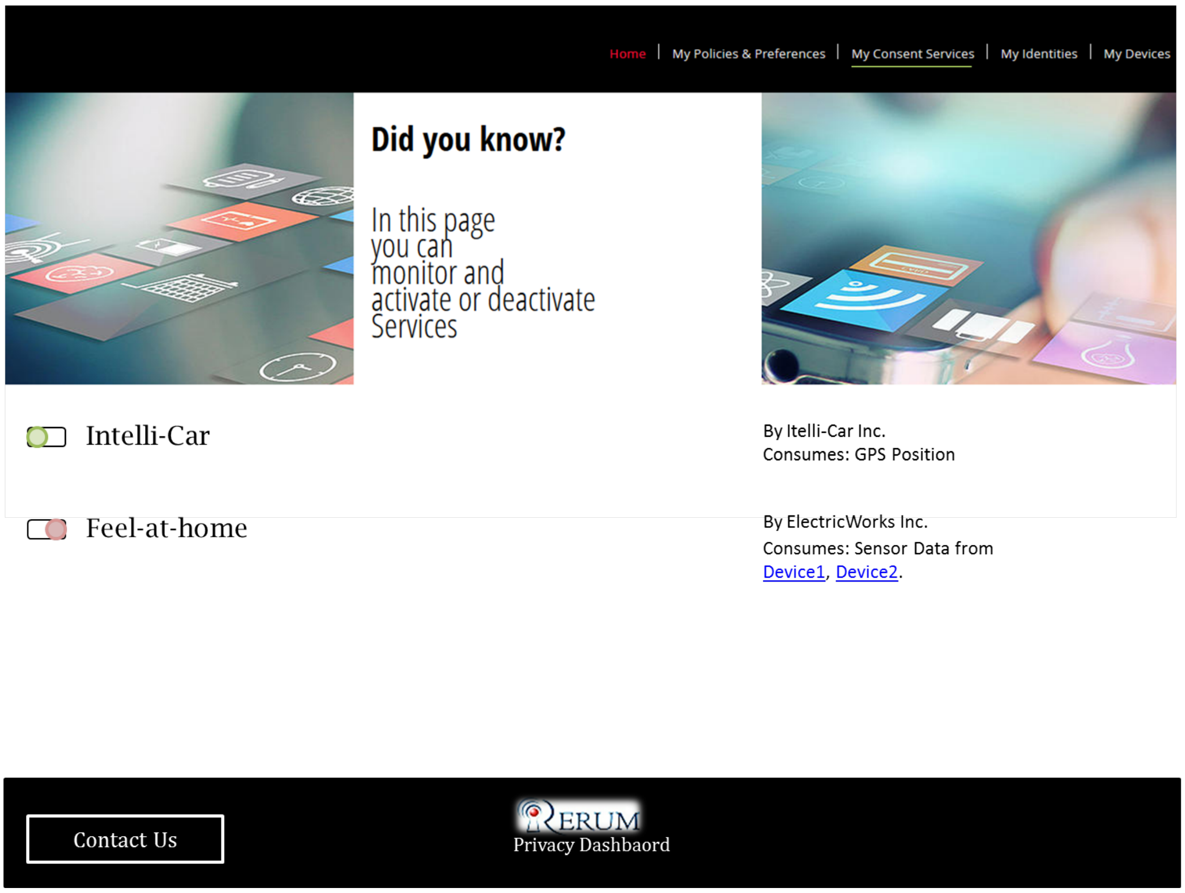
Activator / Deactivator of Data Collection		
Technical Level (description given in 3)	Level 1	The component creates an interface to the data collector in the RERUM Middleware. Similar collector components can be found in many architectures, see for example [9].
Suggested Method(s) for Implementation	Web interface for the privacy dashboard, see Section 3.4. 23	
	Exemplary, the activator / deactivator could be deployed in the Atos User Interface Portal, see D5.2 [145], Section 6.	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	No

3.3.3 Summary

The Activator / Deactivator of data collection is RERUM's component for individual opt-in and opt-out of users from all applications in all of RERUM's use cases. For the component, it is irrelevant if the sensing elements are attached to a RERUM device or are provided by a third party, it simply cuts off the data stream by interacting with the RERUM Middleware. The Activator / Deactivator fulfils therefore the requirement for user interaction and control identified in D2.1 [167]. The Activator / Deactivator can be realized with tools already available, for details see Table 5.

3.4 Privacy Dashboard

Due to the fact that not all people that utilize IoT applications have technical background, it is not viable to elicit a detailed policy language editor for users, such as a XACML editor, to define privacy policies. This is done in the Privacy Dashboard instead. The Privacy Dashboard is a graphical user interface, which visualizes a RERUM Device's behaviour and allows setting a specific behaviour according to users' preferences (Figure 25). The user preferences are then translated to detailed XACML policies / policy database entries without the user's assistance. Additionally, the RERUM Privacy Dashboard allows tracking how many Physical Entities are connected to the RERUM Middleware and which kind of data they are disclosing.



Did you know?

In this page you can monitor and activate or deactivate Services

Intelli-Car

By Itelli-Car Inc.
Consumes: GPS Position

Feel-at-home

By ElectricWorks Inc.
Consumes: Sensor Data from Device1, Device2.

Figure 25: RERUM Privacy Dashboard Sketch

3.4.1 Privacy Dashboard - a privacy pattern

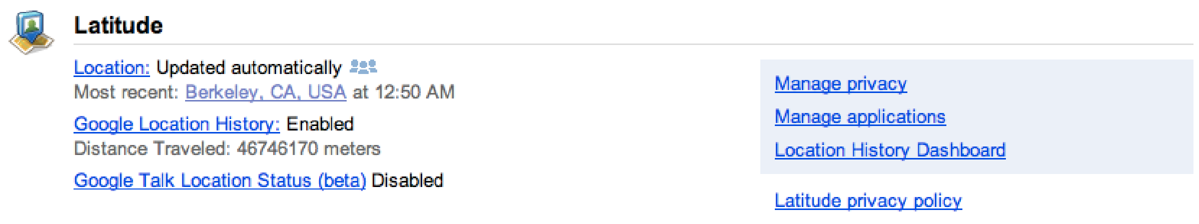
The intent of the privacy dashboard is to help users gain an overview of the personal information collected about them, particularly when the data sources, personal data and related services in question are as numerous and unobtrusive, as in IoT. The privacy dashboard supports the privacy principles of access, transparency and feedback. A privacy dashboard answers the common data subject's (users) question ”What do you know about me?”. It does so in a way that the user can understand and take appropriate action if necessary. It has been described as a privacy pattern in [73].

Data controller collect, aggregate, and process personal information from data subjects (users). Particularly information in the IOT, collected by sensors, and changes over time. It is collected, or aggregated in ways that might be unexpected, invisible or easily forgotten. Still data subjects (users) need to have options for access, correction and deletion.

How can a service communicate the kind and extent of potentially disparate data that has been collected or aggregated by an IoT service or IoT infrastructure? Data subjects (users) may not remember or realise what data a particular IoT data controller (service) has collected, and thus can’t be sure that a service isn’t collecting too much data. Users who aren’t regularly informed of what data a service has collected may be surprised when learning about the data controller’s data collection practices in some other context. Without visibility of the actual data collected, data subjects may not fully understand the abstract description of what types of data are collected; simultaneously, data subjects may be overwhelmed by access to raw data without knowing what that data means.

An informational privacy dashboard can provide collected summaries of the collected or processed personal data for a particular user. While access to raw data may be useful for some purposes, a dashboard provides a summary or highlight of important personal data. It aims to make the data meaningful to the user with examples, visualisations and statistics.

However a privacy dashboard is not only a purely informational instrument. Where data subjects have choices for deletion or correction of stored data, or are permitted to declare their privacy preferences, a dashboard view of collected data is an appropriate place for these controls. Data subjects may be motivated to make use of them on realising the extent of their collected data.


Latitude

Location: Updated automatically
Most recent: Berkeley, CA, USA at 12:50 AM
Google Location History: Enabled
Distance Traveled: 46746170 meters
Google Talk Location Status (beta) Disabled

Manage privacy
Manage applications
Location History Dashboard

Latitude privacy policy

Figure 26: Google Dashboard for Latitude Screenshot

A well known example is the “Google Privacy Dashboard”. The Google Dashboard shows a summary of the content stored and/or shared by many (but not all) of Google’s services (Latitude, Google’s location sharing service, is shown in Figure 26). For each service, a summary (with counts) of each type of data is listed, and in some cases an example of the most recently collected data is described. An icon signifies which pieces of data are public. Links are also provided in two categories: to actions that can be taken to change or delete data, and to privacy policy / help pages.

However, as in other access mechanisms, showing a user’s data back to them can create new privacy problems. Implementers should be careful not to provide access to sensitive data on the dashboard to people other than the data subject. For example, showing the search history associated with a particular cookie to any user browsing with that cookie can reveal the browsing history of one family member to another that uses the same computer. Also, associating all usage information with a particular account or identity (in order to show a complete dashboard) may encourage designers to associate data that would otherwise not be attached to the user account at all. Designers must take care to balance the access value against the potential advantages of De-personalization.

3.4.2 RERUM Privacy Dashboard functionality

The Dashboard is used to track connected physical entities, devices and disclosed data. It must allow to register devices and entities connected with a user / data subject (or automatic discovery like network nodes?). It triggers activation and deactivation of data collection via a GUI. The GUI also allows to declare user privacy preferences, make privacy settings, and control the disclosure of personal data via activator/deactivator, ….

For instance the privacy dashboard displays to the data subject which sensors are gathering which data and who may currently read them, as well as what are the available configuration options (see for instance Figure 12). The user can set the various options and the privacy dashboard translates the current selection into a privacy policy in the pPRep. There the policies are used by the privacy policy enforcement point (pPEP).

The RERUM Privacy Dashboard should not be implemented as a central component per data subject. This would be too privacy infringing, especially if such a component gets compromised. Rather there should be one Privacy Dashboard per IoT infrastructure. There could even be a “well known” access point for user data subjects, just like “Impressum / Contact” in current web sites.

Table 6: Technical Implementation Summary for Privacy Dashboard

Privacy Dashboard		
Technical Level (description given in 3)	Level 1	Implementations exist, see for example [73].
Suggested Method(s) for Implementation	Web server with a web-portal implementation as shown in [19] or as an extension as shown in [227]] .	
	http://code.w3.org/privacy-dashboard/	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	No

3.4.3 Summary

The privacy dashboard provides transparency for users: It informs the user of privacy relevant events, the service providers he is involved with, and the interactions his RERUM Devices are currently carrying out. The privacy dashboard fulfils the requirement for notice and access defined in D2.1 [167]. The dashboard can be implemented with an adaptation of already existing tools as presented in Table 6.

3.5 Anonymising and Pseudonymising Managment

In RERUM Deliverable D3.1 [201] and also in the introduction to privacy-by-design the principle of data minimization was described as key concept for true privacy-by-design. Anonymisation and pseudonymisation help to mitigate privacy breaches by tracking and identification [151], that means, that an attacker tracks the behaviour of a system participant and tries to link this information to the identity of the participant. With anonymisation and pseudonymisation, the attacker will not be able to link tracked behaviour, thus minimising the subject related data in the RERUM's architecture.

Anonymisation mechanisms are implemented per scenario. For this data is directly anonymised after it is sensed. RERUM utilizes state-of-the art mechanisms to achieve anonymisation. For example, traffic data is anonymised whenever reasonable with existing anonymising networks such as [154] or [86]. Anonymous authorization is supported by group signatures [53] and privacy enhancing authorization is explored in [63].

Pseudonymisation requires extensive management, depending on the type of pseudonymisation that is considered. Generally, there are four types of pseudonymisation techniques, based on

- asymmetric encryption,
- identity-based attributes,
- group signatures, and
- symmetric encryption.

RERUM regards identity-based attributes as best suited for pseudonym issuing and management. Albeit group signatures have been recognised as the best state-of-the-art pseudonym mechanism, RERUM has developed a new identity-based pseudonym technology based on cartographic one-way functions, such as SHA2 [212] and SHA3 [191].

One of the main issues in developing pseudonym system has been obtaining new pseudonyms. In [151] the authors have studied how to "refill" pseudonyms and ask: *"is it better to load a large amount of pseudonyms at one time or to load a small amount of pseudonyms at several times?"*. RERUM takes a different approach by allowing the dynamical generation of virtually unlimited pseudonyms with efficient cryptographic methods that are adequate for constrained IoT devices. As opposed to group signatures, hash-algorithms are widely implemented in standard cartographic libraries, are less computational demanding and far superior in low energy consumption (see [190]).

Pseudonym management is handled with an intuitive hash-tree mechanism further described in Section 4.6. The architectural integration remains the same, the pseudonymising management resides on the Security Center and is closely coupled to the authentication authority and the stream processing component of the RERUM's middleware. The relationship between middleware, anonymising and pseudonymising management is depicted in Figure 27.

3.5.1 Summary

The anonymising and pseudonymising component provides identity protection for users and devices alike. The component is reachable over the RERUM's security privacy centre thus independent from the party requiring new pseudonyms. Pseudonym management and agreement can be handled individually by devices as the pseudonym mechanism has been designed with computational and battery efficiency

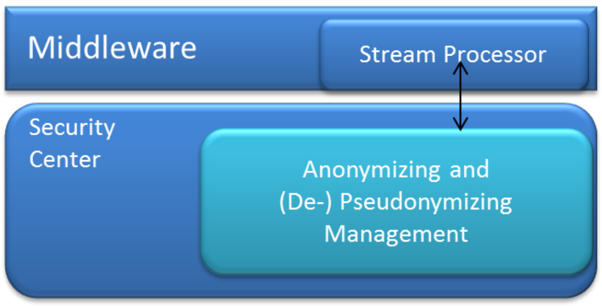


Figure 27: Location of the anonymising and pseudonymising management (as described in D2.3)

Table 7: Technical Implementation Summary for Anonymizing and Pseudonymizing Management

Anonymizing and Pseudonymizing Management		
Technical Level (description given in 3)	Level 2	Cryptographic hash functions exist, the protocol for pseudonym agreement and weak depseudonymization has been defined in RERUM.
Suggested Method(s) for Implementation	Hash libraries such as [225] or [5] for cryptographic operations combined with appropriate data structures such as arrays or dictionaries.	
	None available.	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	No

in mind, but in case of highly constrained devices, pseudonym generation, agreement and management can be delegated to the anonymising and pseudonymising component itself. The novelty of the approach is again underlined in Tables 7 and 8.

3.6 De-Pseudonymizer

There are two general situations where a pseudonym has to be “reverted” in some way:

Strong De-Pseudonymizer Given a pseudonym, find the user or entity to whom this pseudonym belongs (or belonged). In particular, this may be necessary or convenient for billing or legal requirements. (For billing purposes, other mechanisms like encryption could be used. Note that the billing information should contain a minimal set of personal information required for the purpose of billing). This type of de-pseudonymising routines or procedures in pseudonym systems may be a weakness, if it is exploited by attackers to recover personal data.

Weak De-Pseudonymiser or Pseudonym Agreement Given a user or entity and a moment in time (usually the current time or a time not far in the past), find the pseudonym that this entity has or had at that moment. This weak type of de-anonymisation can be also called “pseudonym agreement”. A restricted number of well defined entities has the capability of finding the pseudonym for an entity or user (or the capability of “agreeing on a pseudonym” for an entity).

The main use of the weak de-anonymisation routine is as follows: assume an entity (say a sensor) provides a service associated to a pseudonym. (So far, the “real name” or application name of the sensor has been pseudonymised). In this way, the data of the sensor is kept linked to a pseudonym (not to a sensor name in cleartext) in the “cloud” or in the databases. In general, an attacker that may be able to read the data in the database is not able (or at least has trouble) to revert the link to the real identities. Assume that an authorised user want to access to the service provided by this sensor. In order to do so, he must know under which pseudonym the data is indexed, for the purpose of retrieving it.

Notice that knowing the list of entities in the system, a weak de-pseudonymiser can be used to implement a strong de-pseudonymiser: list all entities, calculate all valid pseudonyms for those entities in the given time, and find the pseudonym in this list. But his procedure is costly and thus it is difficult to use in general (which may be an advantage, because a strong de-pseudonymiser should only be used in very special cases, say where a judge requires it).

In many scenarios requirements will make it necessary to de-pseudonymise certain entities and to track the identities that were behind certain actions. RERUM does not explicitly support this type of routine. However, if absolutely necessary the weak de-pseudonymiser can be used instead (with a cost in performance). We strongly note, that to protect privacy de-pseudonymising routines must be secured against misuse. RERUM’s weak de-anonymisation does exactly this; RERUM —in a similar way as in key agreement protocols and encryption— requires entities to know a shared secret (key) in order to gain the capability of finding the pseudonym of an entity. Thus, (weak) de-pseudonymising in RERUM is a functional component that supports the scenarios mentioned above, but at the same time is unavailable to unauthorised entities, i.e. attackers, due to the use of secrets in combination with one-way functions to create pseudonyms.

We further reason on several state-of-the-art techniques in Section 4.6.6.

3.6.1 Summary

The de-pseudonymiser is a part of the anonymising / pseudonymising component. It allows re-linking of pseudonyms for use cases that need to identify the action of a user. The de-pseudonymisation is weak:

Table 8: Technical Implementation Summary for Weak De-Pseudonymizer

Weak De-Pseudonymiser		
Technical Level (description given in 3)	Level 2	Cryptographic hash functions exist, the protocol for pseudonym agreement and weak de-pseudonymisation has been defined in RERUM.
Suggested Method(s) for Implementation	Hash libraries such as [225] or [5] for cryptographic operations combined with appropriate data structures such as arrays or dictionaries.	
	None available.	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	No

This means that not every pseudonym can be re-linked at will, as this is undesired in many use cases. The pseudonym management has to be agreed on in such a way that re-linking is possible. This is further described in Section 3.6.

3.7 Geo-Location PET

In RERUM Deliverable D2.3 [219] Section 6.11.2.7 we firstly introduced the need for a privacy enhancing technology for geo-location privacy. RERUM will support traffic analysis by floating car observation. In this context, location privacy has been a topic of interest as ubiquitous systems will, on the one hand, be able to track, record and analyse every user movement, revealing a user's habits, routines and tendencies. On the other hand, traffic analysis and vehicular networks are envisioned to improve safety and traffic efficiency. Real traffic data can be used for simulations to improve e.g. road construction before it is carried out in real life.

Many proposals to avoid tracking exist, such as [23], [115] and [41]. To understand why RERUM needs a different geo-location privacy approach than those proposed in current research, the existing approaches have to be categorised. Most geo-location privacy approaches hide traffic participants in vehicular networks, where messages from vehicles are routed through traffic participants, using other vehicles as routing nodes. In a simple vehicular network (abbreviated "VANET"), a traffic participant requests nearby vehicles which could route his message over the network. A central party would identify the nearby vehicles through the vehicle's GPS positions and would broadcasts the position to the requester. In more advanced scenarios, for example note the description in [23], the vehicles broadcast sets of their positions, speeds, motion vectors and acceleration as so called Beacons every 100 to 300 milliseconds. Mechanisms in VANETs protect these Beacons and other VANET messages by hiding the vehicle's identity with pseudonyms and obfuscating the sending routes. Similar to mix-cascades and onion routing for network traffic, VANET privacy mechanisms use mixing of message routes and identities, creating so called mix-zones, as seen in [23].

In RERUM's floating car observation use case, the situation is different. A traffic participant does not need other participants to broadcast its message. The traffic data measurement is transmitted directly to a service provider, possibly using a cellular mobile network (e.g. 3G or 4G). As the network transmission can continuously identify the participant, anonymous routing techniques have to be applied. This will not be a research focus of RERUM, as many applicable anonymous network solutions exist such as the TOR [154] and the AN.ON [86] networks.

In addition, most VANET privacy mechanism protect message routing, but not the message content itself. The message content, i.e., the measured GPS positions and driving speeds, is the privacy sensitive data in RERUM's use case. We therefore need to identify suitable techniques for transmitting detailed traffic information, but at the same time protect the traffic participant. This is done by enlarging the set of indistinguishable measurements. Every traffic participant simulates not one, but several participants sending measurements. As the number of simulated participants is generated randomly, the anonymity set varies in such a way that the change of distinguishing single participants from simulated or other real participants becomes insignificant. At the same time, the measurement data is left unaltered. There is no aggregation or perturbation of measurements for the service provider.

We provide a detailed description of the mechanism in Section 4.8. We also give privacy considerations Sections 4.8.5 and 4.8.6.

3.7.1 Summary

The geo-location PET is a part of the on device S&P&T mechanisms and resides in the RERUM Device. It receives data from the GPS sensing element in the device to create privacy enhanced data sets. RERUM's geo-location PET is at the time of this writing unique for floating car observation privacy (see Table 9).

Table 9: Technical Implementation Summary for Privacy Enhanced Geo Location

GEO-Location Privacy		
Technical Level (description given in 3)	Level 3-	This components introduces a novel privacy enhancing technology for traffic observation. Related work does target VANETs which is not directly usable in RERUM's use case.
Suggested Method(s) for Implementation	Vector generation is implemented on the measuring device, e.g., as an android application.	
	None available. For related work, see section 3.7.	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	No

The geo-location privacy component maybe switched off by policies (Sections 4.8.2 and 4.8.3) to fulfil the privacy requirement for individual user participation and control.

3.8 Security functional components as privacy basis

In this section we summarise several security components described in RERUM Deliverable D3.1 [201] together with a short description of how we think they may be used to enhance privacy. This covers 5 of the 8 security components sketched in D2.3, Section 6.11.1 and detailed in D3.1. The other 3 security components are needed and summarised in the description of the Privacy Policy Enforcement Point in Sections 3.2 and 4.4 respectively.

3.8.1 Data encrypter/decrypter for privacy

As written in RERUM Deliverable D2.3, the data encrypter/decrypter is a basic mechanism of the RERUM architecture, and it is part of the Secure Communication component. CS-based encryption/decryption has been implemented and integrated within this component. Data are encrypted in the sensors, and decrypted in the RERUM Gateway or in the client that receives these data; hence, privacy is feasible as CS encryption can provide strong computational secrecy that is a core building block in order to achieve privacy. We have successfully demonstrated the data encrypter/decrypter (Figure 28) in various events (IoT Week 2015, etc.).

In Table 10 we summarised why cryptographically strong encryption is a building block to achieve IoT privacy.

3.8.2 D2D authenticator for privacy

With device-to-device (D2D) authentication we allow to authenticate the RERUM device towards another RERUM device (RD). This security mechanisms, as well as technical solutions to achieve it, have been explained in detail in RERUM Deliverable D3.1 [201]. One of the candidates described in more detail is Datagram Transport Layer Security (DTLS). RERUM also seeks to implement DTLS on the Re-MOTE (RERUM's hardware platform). DTLS includes origin authentication on the network layer. Secondly, RERUM described how to best enable authentication between devices on a higher layer using digital signatures. Digital signatures on devices give data origin authentication by means of public keys on application level data (h-data). Additionally, other authentication mechanisms like MACs based on symmetric keys can be used. All of these mechanisms have been described to work on RDs.

Device-to-device (D2D) authentication is a cornerstone for achieving privacy: First, in order to respond to data requests we then need to authenticate the requesting device in order to decide whether or not to allow access to the data, this again relates to the principle of data minimisation. Secondly, in order to address the data minimisation principle we need to know the target of a communication channel in order to send data only to authorised partners. Furthermore, privacy can use Device-to-device (D2D) authentication to allow for accountability: Keeping track which devices, e.g. the device's IDs, have accessed/requested data. This only makes sense if the request that was logged was indeed coming from the device with that ID. In Table 10 we summarised why cryptographically strong D2D authentication is a cornerstone to achieve IoT privacy.

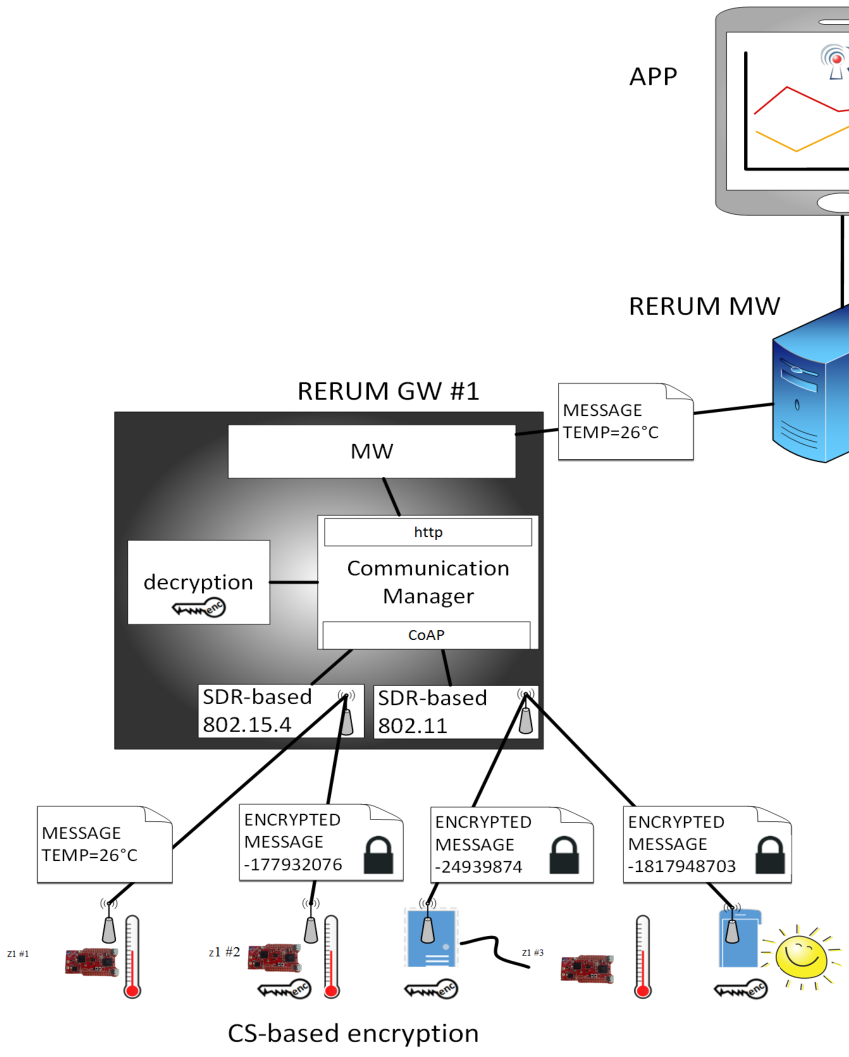


Figure 28: RERUM Encrypter/Decrypter demonstration

RERUM IoT privacy requirements		D2D Authenticator	Encryption
1. Consent and Choice			
2. Purpose legitimacy & specification			
3. Collection limitation			
4. Data Minimisation	YES	strongly identify the RD that requests and gets the data	data is only accessible to authorised parties
5. Accuracy and Quality			
6. Notice and Access	YES	accessing RD can be authenticated and logged	
7. Individual participation & transparency			
8. Accountability	YES	accessing RD can be authenticated and logged	

Table 10: Support from D2D Authenticator and Encryption for the RERUM IoT privacy requirements

3.8.3 Credential bootstrapping client/authority

RERUM deploys cryptographic security mechanisms that are vital to enhance privacy. All technical solutions need key material. We have described the key material in great detail in RERUM Deliverable D3.1 [201]. Hence, all of the security mechanisms require to have access to this material which means also that it has been distributed to all parties that need it.

For this deliverable, we assume that the minimally needed key-material has been distributed to the devices. Either by using some establishment protocol as the novel RSSI-based key-derivation for compressive sensing [88]. Or by using the secure credential bootstrapping process that is described in Section5.2 of D3.1 [201].

In short, we assume for this deliverable and for privacy that all underlying security mechanisms have the right credentials. For example, the previously described D2D Authenticator, can use correct credentials from a trusted authority stored in the Trusted Credential Store to identify another RD as being his trusted gateway device.

3.9 Privacy Enhanced Integrity Generator / Verifier

The Integrity Generator / Verifier (from RERUM Deliverable D2.3 Section 6.11.1.1 [219]) is a crucial security mechanism to protect data and commands from unauthorised modifications and allow authentication of the origin. Integrity is the "property that data has not been altered or destroyed in an unauthorised manner" [118]. It can be achieved on the transport-layer and on the message level. Transport-layer integrity protects the channel between two communicating entities, such that inside the channel integrity cannot be violated with out being detected by the communication partner. Message-level integrity creates an integrity check value, e.g., using digital signature, over the message and then send message and signature over an unsecured communication channel. Figure 29 shows the components

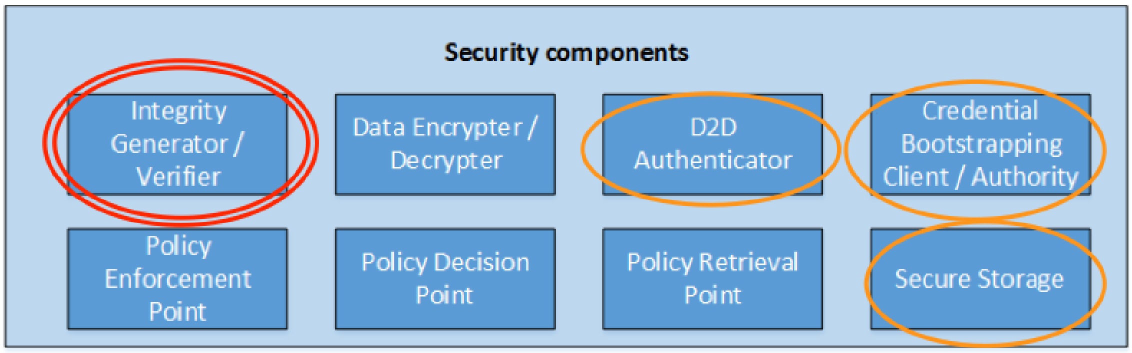


Figure 29: Overview of the location of the Integrity Generator / Verifier function inside RERUM's architecture (taken from: RERUM Deliverable D2.3 [219])

being part of the communication security as devised in the RERUM architecture specified in its first version in D2.3 Section 6.11.1.1 [219]) Thus this mechanism is present throughout the communication channels within RERUM, but also towards the outside of the RERUM architecture.

In this deliverable we describe in more detail the added functionality that RERUM has devised to allow editing of integrity protected data while preserving as much of the integrity and origin authentication. It is desirable to achieve message level integrity as we can achieve the goal of end-to-end security (see Figure 30a). Digital signatures are the usual cryptographic building block that allows to achieve this property. However, standard signature schemes suffer from a problem termed the **digital document sanitizing problem** by [163]. The original work describes this problem as follows:

> A digital signature does not allow any alteration of the document to which it is attached. *Appropriate alteration* of some signed documents, however, should be allowed because there are security requirements other than that for the integrity of the document. In the disclosure of official information, for example, sensitive information such as personal information or national secrets is masked when an official document is sanitized so that its nonsensitive information can be disclosed when it is demanded by a citizen. If this disclosure is done digitally by **using the current digital signature schemes, the citizen cannot verify the disclosed information correctly because the information has been altered to prevent the leakage of sensitive information**. That is, with current digital signature schemes, the confidentiality of official information is incompatible with the integrity of that information. This is called the *digital document sanitizing problem* [...] [163]

In RERUM we researched how data generated on RERUM Devices (RD) can be cryptographically signed on the RD such that it allows appropriate alteration with malleable signature schemes. Those appropriate alterations can result in data minimisation to increase privacy. In Section 3.9 we first give the results from our published case-study on the positive effects of perturbation of energy consumption data. This has been published in [189]. With this motivation in Section 3.9.2 we discuss malleable signature schemes that allow such *appropriate alteration*. We have disseminated the results academically in [66, 67, 182, 184–186, 189]. We conclude in Section 3.9.2 giving the interaction with the technical mechanisms to achieve the component's function in a privacy tolerant manner using malleable signatures.

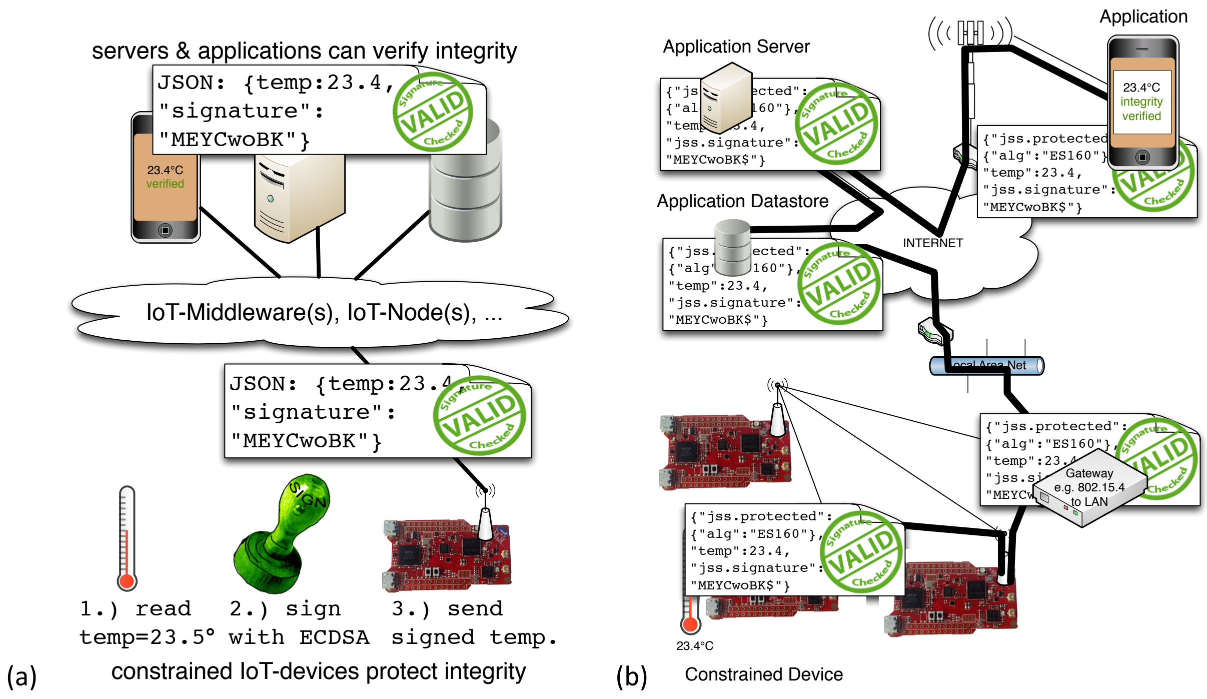

servers & applications can verify integrity
JSON: {temp:23.4, "signature": "MEYCwoBK"}
23.4°C verified
IoT-Middleware(s), IoT-Node(s), ...
JSON: {temp:23.4, "signature": "MEYCwoBK"}
SIGN
1.) read temp=23.5°
2.) sign with ECDSA
3.) send signed temp.
(a)
constrained IoT-devices protect integrity
Application Server
Application
23.4°C integrity verified
Application Datastore
INTERNET
{"jss.protected": {"alg":"ES160"}, "temp":23.4, "jss.signature": "MEYCwoBK$"}
Local Area Net
Gateway e.g. 802.15.4 to LAN
23.4°C
(b)
Constrained Device
Signature VALID Checked

Figure 30: (a) Seamless integrity protection end-to-end during the complete data-lifecycle [182]; (b) proposed solution: signing JSON-formatted data on the constrained device

3.9.1 Case study on some data blurring techniques (published in [189])

As stated previously in Section 2.3.4 information blurring is a tool to increase privacy if data needs to be transmitted. This follows the principle of data minimisation. Many information blurring techniques require to change data only within predefined limits. However if that data was integrity protected information blurring will interfere with integrity, i.e. it changes data that was protected against undetectable subsequent changes. In RERUM we do not wanted to forgo integrity protection completely, but only lower the integrity protection.+ to allow certain data blurring or data minimisation.

For example take the case of temperature data. Only a controlled change of signed data is required, e.g. temperature information needs to be sanitized to reduce their resolution, before being made available to the city. Namely, imagine sensed temperature values of a very precise resolution, e.g. 23.542°C. A redaction in resolution can mean $23.$■°C, or 2■.■■■°C which can be noted also as ≥ 20°C. However, which resolution is required to protect the privacy or which is consented to be released to a certain requesting entity might not be decidable by the initially data gathering sensor itself, but only by the gateway or the RERUM middleware. These *allowed* modifications shall not result in an invalid signature and must not involve the sensor's signature on the data. Also the required overhead rules out re-sending the changed data to the respective sensor to re-sign the changed data. This problem, that integrity protected data must be changed in order to protect some data's confidentiality, but such that the integrity is not tampered, has been termed the "digital document sanitization problem" [162]. This requires allowing a verifier to identify that the unmodified data is original and —if modified— the modification was done with the consent of the original signer. The initial signer shall remain still identifiable by means of the signer's public key.

To highlight and motivate the need for subsequent changes to increase privacy without fully invalidating previously applied integrity protection RERUM conducted a small case study.

The results of the following have been part of this study conducted for RERUM by Henrich C. Pöhls, Max Mössinger, Benedikt Petschkuhn, Johannes Rückert on the privacy invasiveness of energy consumption monitoring traces. The study shows that energy consumption traces retain information extractable by basic behavioural detection algorithms even if they are not very fine grained. Thus, energy consumption traces of individual homes are (a) private data and (b) they needs to pass by a PET that anonymizes and perturbates the data. Note, the idea was not to remove the usefulness, e.g. allow to get averages for forecasts or detect inhabitants presence.

The results of the complete study have also been successfully disseminated to the academic community and published at IEEE CAMAD in 2014 [189].

3.9.1.1 Overview of case study

We analyse accuracy, privacy, compression-ratio and computational overhead of selected aggregation and perturbation methods in the Internet of Things (IoT). We measure over a real-life data set of detailed energy consumption logs of a single family household. This studies setting was within the Use Case of Energy Consumption (UC-I1). The main privacy concern was the possibility to deduct behavioural patterns from the energy readings gathered for in-house circuits. Current market ready IoT deployments (e.g. for the smarthome, domotics, smartgrid) gather data at a few central places, e.g., energy consumption at smart meters, needing only the deployment of few devices. Still, new applications shall be able to evolve based on top of that data, e.g., provide an intelligence and self-adapting home environment learning from the energy patterns. Aggregation aside, especially perturbation (adding noise) is meant to achieve privacy gains for the indirectly monitored inhabitants (see UNI PASSAU's work on achieving differential privacy in combination with a malleable signature scheme described in Section 5.3.2). We modelled privacy by simple, threshold-driven machine-learning algorithms that extract features of behaviour. The accuracy of those extraction is used as privacy metric. We state for different parameters of the aggregation, reduction and perturbation if the output still allows detections, as this follows the EU's data protection principle of "minimisation": increased privacy due to less detailed data, but still good enough accuracy for the purpose. As we have detailed logs about timing of actions, e.g. using the microwave to heat milk for the morning coffee, correlating to circuit measurements and we know exactly what devices each circuit contains, additionally we know from user diaries what actions (sleep, wake, watching TV, vacation) he performed, which allows us to identify behavioural patterns in traces and make assumptions on the privacy gained. Accuracy is measured by comparison with the original data in terms of total sums over great time periods of several months. The result is that many detections for sensible predictions and intelligent reactions are still possible with lower quality data.

3.9.1.2 Research question and methodology of case study

This case study is mainly motivated by the fact that under EU privacy laws the data gathered must be "necessary for the performance of a contract to which the data subject is party" [82]. We wanted to know if we really need the high precision in which the IoT could gather data. We applied and evaluated different parameters for aggregation and perturbation on a real-life data set in order to find what level of reduced data quality and hence additional privacy we could achieve. Alongside, aggregation yields compression. Privacy, in this context means not disguising the identity of the data's subject. Rather we want to lower data quality to the bare "necessity" [82] to suit a given purpose of an application. This is following the requirement of data minimisation (No. 4 of RERUM's privacy requirements). The terms purpose and consent are used according to RERUM's privacy requirements identified in Section2.7. Purpose is based on European legislation, e.g., [84], meaning that getting data such that an application can learn and forecast behavioural patterns, like detecting and then deducting that you are usually at home between 12-16 on saturdays and sundays, but away on weekdays, can be a legitimate purpose, e.g. to adjust your heating system and schedule your parcel delivery. Hence, a data subject could give their informed consent to just that purpose.

However, the question 'How low can the granularity and data quality become such that the application still works?' was posed to the research community in our previous publication [184].

The study was on on electrical energy consumption data. According to M. Jawurek [123, p. 80], aggregation can be applied on three different dimensions: spatial, temporal or arbitrary. We therefore gathered several detailed energy consumption profiles of several in-house circuits of one family household and hence we will focus on temporal aggregation. As we have detailed self observation logs from the family about timing of actions, e.g. using the microwave to heat milk for the morning coffee, additionally we know what devices each circuit contains. From this we devised threshold machine learning algorithms (see Section 3.9.1.5) that correlate energy measurements with actions (e.g., sleep, wake, watching TV, vacation) performed. These algorithms allow identifying behavioural patterns in traces and later make assumptions on the privacy gained by aggregation and perturbation methods. Future work might use more sophisticated algorithms from the domain of machine learning.

3.9.1.3 Data set

Data was gathered in one household of a family, using in-circuit 'smart meters' measuring the energy consumption of devices connected to each electrical circuit. Each in-circuit-smart-meter sends a 'tick' on every consumed Watt hour (1 Wh) that is recorded together with a timestamp[4]. The data set contains separately the energy consumption of several circuits: (a) *living room* with a TV (approx. 100W) and several independent lights (150W in total), (b) *study room* with computers and a TV (approx. 40-70W). The data was collected over a period of seven months with around 926,000 entries. We automatically obtained the uptime of certain IP-enabled appliances, e.g., SmartTV, and the inhabitants kept diaries and

[4]based on `volkszaehler.org`

we conducted interviews. Thus, we had ground truth to identify which actions correlate to consumption data traces to check the accuracy of the feature extraction algorithms.

3.9.1.4 Modifications for information blurring: Aggregation over time, perturbation and reduction of resolution on time

We differentiate the following three modifications for information blurring:

(a) Aggregation over time,
(b) Perturbation of the data with noise,
(c) Reduction of resolution on the scale of time.

Aggregation: Aggregation is a mechanism to increase privacy by merging different single data points. It is not geared towards disguising the identity of the data's subject, but attempts to enhance privacy by lowering the accuracy of data, hereby limiting the possibility to deduce private information. According to M. Jawurek ([123],p.80), aggregation can be applied on three different dimensions: spatial, temporal or arbitrary. For this first instance of the case study we calculated the harmonic and the arithmetic mean over different time intervals.

Definition 1 (Harmonic Mean). $A = \frac{n}{\sum_{i=0}^{n} \frac{1}{x_i}}$

The harmonic mean showed to be tolerant towards energy peaks and offers a good accuracy. Hence, we choose the harmonic mean for aggregation.

Definition 2 (Arithmetic Mean). $A = \frac{1}{n} \sum_{i=1}^{n} x_i$

Already few peaks negatively affected the accuracy of the aggregated result using an arithmetic mean in many of our cases. Hence, we did not choose an arithmetic mean.

For the aggregation we can use the different arithmetic functions mentioned. The time interval can be adjusted to suit the application. We ran with different intervals, i.e., 10 minutes, 1, 4, 8 and 24 hours.

Perturbation: Perturbation and the reduction of resolution both aim to abstract data to a level, on which the deduction of private information can hardly be performed. The basic method of perturbation relies on the introduction of random noise (i.e. data fragments) to the data items respectively the final aggregate, causing a distortion in the original values. Adding sufficient noise to prevent an attacker from deriving data items or patterns from the result while preserving the utility of the data is challenging [123, p.74-75]. In some cases, this challenge is difficult if not impossible to overcome. For example consider perturbation on an energy profile to avoid burglary when you are away. Perturbation needs to add enough noise to prevent an attacker from differentiating whether the inhabitants are present or absent. At the same time, exact data might be needed to perform certain computations, e.g. for the purpose of billing [123]. Consequently, perturbation is only applicable if the calculations don't need to be perfectly accurate. Furthermore, random perturbation carries the risk of revealing some kind of structure within the randomness, which could be used to compromise the original data set [149].

In this case study we tuned perturbation by adding different noise. First and foremost, the parameters to identify are a suitable maximum and minimum noise to be added. Secondly, the noise can be random, or pseudo-random, or following some specific distribution.

Reduction of resolution: Reduction of resolution operates as the name implies by reducing the accuracy of the collected data, for example extending a time attribute from minutes to hours or even days. There is however a key difference in comparison to aggregation: In case of an aggregation over time, the mean of the values within a time interval is calculated and generalised over all entries within this interval. Reduction of resolution on the other hand doesn't change the values, but instead determines one timestamp within the observed time interval with which the timestamp of every entry is overwritten. So in contrast to aggregation, the measured values will be left untouched, yielding perfect accuracy. A conceivable use case would be reducing the resolution of consumption traces of a smart home before storing them externally, effectively limiting the amount of sensitive personal information that may be derived [123]. The interval is again the property that can be adjusted to suit the application.

3.9.1.5 Comparison regarding extractability of features, compression, data quality (accuracy) and computational overhead

We compare using four metrics:

- Feature Extraction
    - detecting if inhabitants present (example for Behavioural Detection)
    - detecting use of a certain device
- Compression
- Data quality in terms of accuracy of averages
- Computational Overhead.

Feature Extraction: For feature extraction we used simple feature extraction algorithm to detect (1) if inhabitants are present and (2) if a certain device is used.

The algorithm detecting presence on our energy consumption data set is based on comparing the average consumed energy over defined time intervals. It starts with a one week training phase over data for which the inhabitants indicated their presence. Then, it iterates over the whole data set using the defined interval as step-size. In each step the algorithm checks if a part of the interval features an average which is greater or equal to the average determined in the training phase. In case of a hit, we assume having detected presence and mark the interval accordingly. For example, we executed it with a target of a resolution of four hours, as Figure 31 this would allow to forecast consumption at different time intervals a day, e.g. morning, lunchtime. Figure 32 only targets to detect the presence on a daily basis.

As the presence within the household is not reasonably detectable by utilising the data of one circuit only, we applied the algorithm on an accumulated data set including the consumption of the living and the study room. We performed the detection over intervals of 4 hours and one day, on both the original data set, as well as on an accordingly aggregated data set. As the algorithm identified presence on the original data set almost flawless, we used these results as reference. Further comparison with the algorithm's results on the aggregated data set was based on the

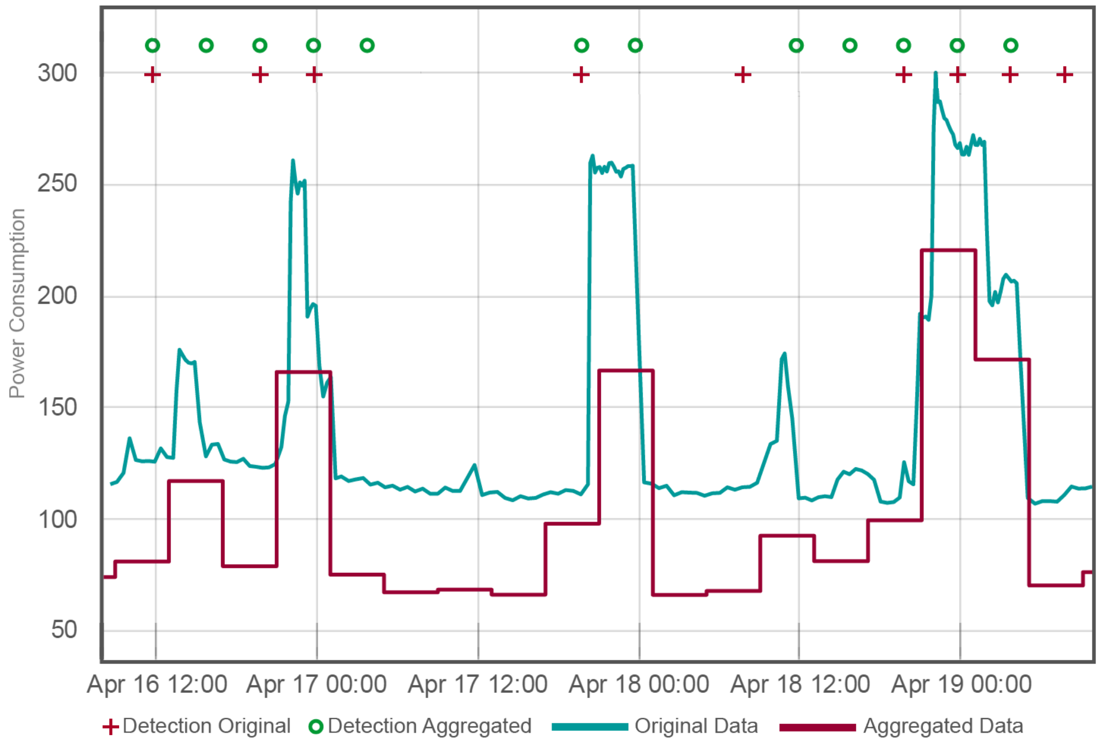


Figure 31: Presence detection over an interval of 4 hours.

receiver operation characteristic notation: if both mark an interval, this is a true positive (TP), the opposite is a true negative (TN). If an interval is marked only by the algorithm using the original dataset, this is a false negative (FN). In case of an interval being marked only by the algorithm utilising the aggregated data, a false positive (FP) is issued. The accuracy is then computed as $\frac{TP+TN}{TP+TN+FP+FN}$.

Behavioural Detection: We implemented behaviour detection based on detecting devices being turned on. We utilised device specific power consumption signatures for the purpose of identification, for instance the TV requires between 40W and 70W while being powered on. We then matched the data with the signatures to detect occasions where this device is known to be on. From a privacy point of view, when observed over longer times this allows the derivation of behaviour patterns, e.g. reveals your favourite TV show. Figure 33 illustrates the algorithm, the marked areas allow to easily identify the points in time when the TV has been switched on. After aggregating the data set, this method is no longer applicable, since there is no way to differentiate between distinctive power input anymore. In case of reduction of resolution this is different however, since the power consumption as well as the sequence of events is sustained. The activation of devices can not be mapped to an absolute point in time though. To measure the privacy gain we compare the number of detected devices before and after the application of aggregation respectively perturbation. The accuracy is determined by calculating $\frac{number\_of\_dev\_detected_{after}}{number\_of\_dev\_detected_{before}}$. The resulting figure describes the percentage of devices which can still be detected in relation to the previously detectable devices.

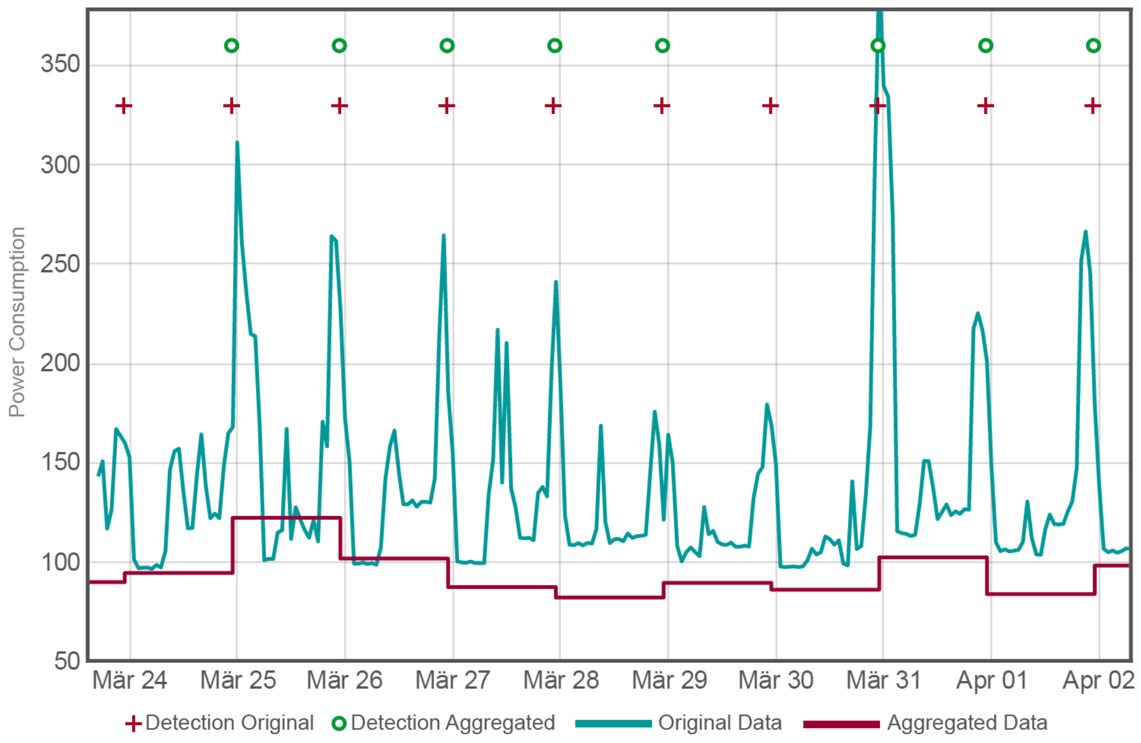


Figure 32: Presence detection over an interval of 24 hours.

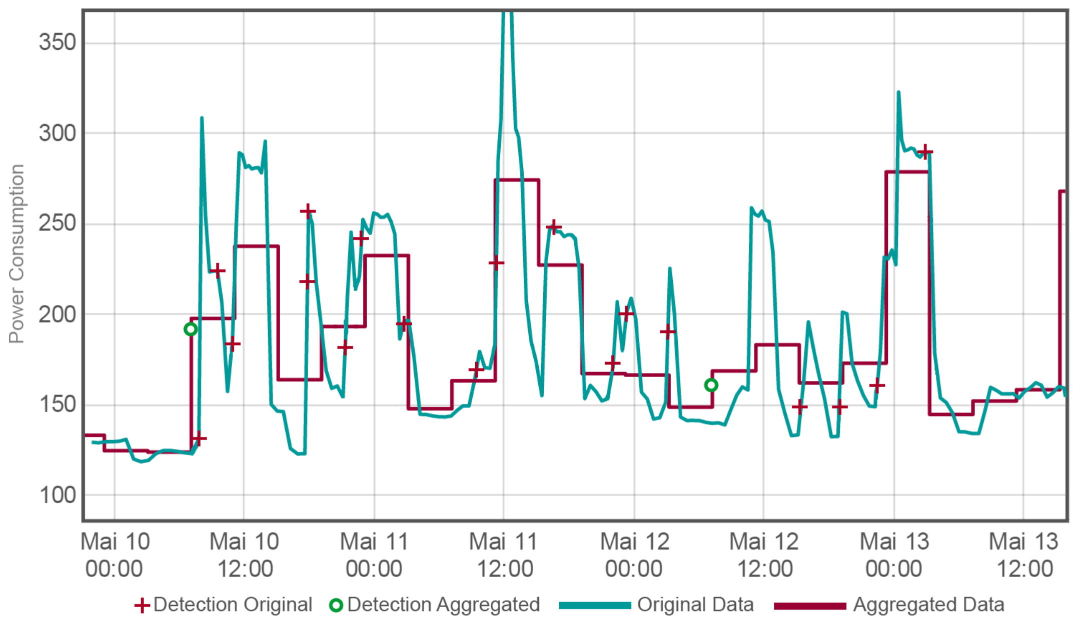


Figure 33: Identification of SmartTV based on peak of certain height

To ensure the objectivity of our results, we also utilised an external peak detection algorithm based on Matlab, providing a well-established mathematical foundation. Thereby, a peak corresponds to a local maxima and has to be greater than its direct neighbours [74]. Semantically, a peak can be interpreted as some kind of activity. We applied the algorithm to the raw data set. Afterwards the same algorithm was executed on the data aggregated over 10 minute intervals with the harmonic mean. The results are illustrated in Figure 34. Since every peak corresponds to activity, the reduction of 27 peaks to merely 2 indicates a clear privacy improvement. To estimate the privacy advantage, the formula $\frac{number\_of\_peaks_{after}}{number\_of\_peaks_{before}}$ gives the percentage of peaks in relation to the original number of peaks. Since perturbation introduces random noise, the number of peaks is increased instead of reduced. Thus $peaks_{before}$ are equal to correct peaks, while $peaks_{after}$ include numerous deceptive peaks. Consequently the quotient has to be turned around in case of perturbation, yielding the ratio of correct to incorrect peaks.

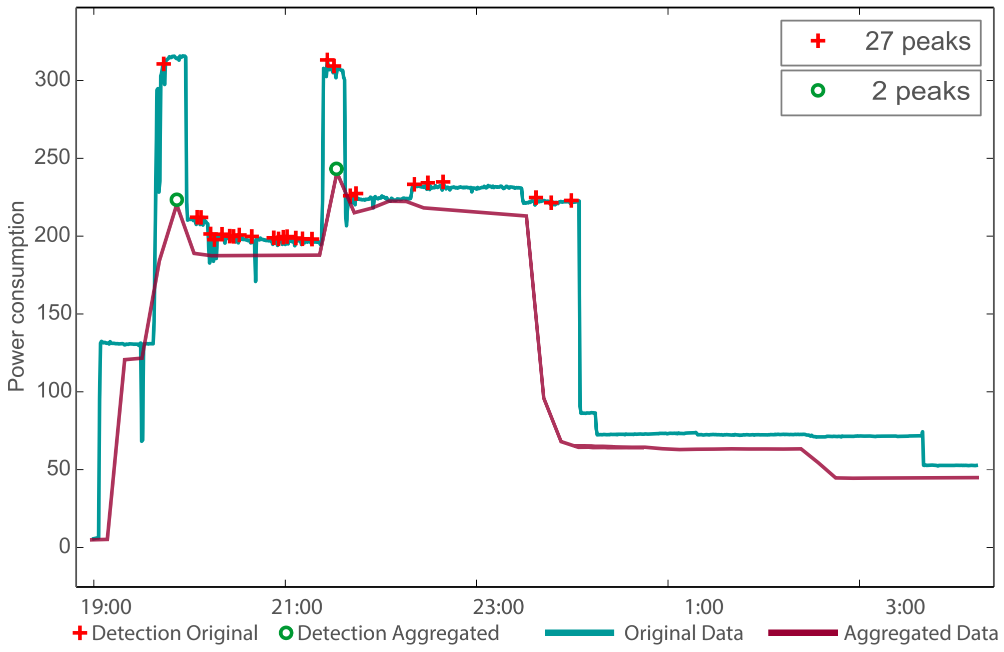


Figure 34: Peak detection on original and aggregated data

Compression Ratio: The amount of transmitted data is an important factor in the IoT. Our compression metric indicates the percentage by which the aggregation or perturbation is reducing the original data set and is calculated as $1 - \frac{number\_of\_entries_{after}}{number\_of\_entries_{before}}$.

Energy Consumption Accuracy: The data set contained timestamped ticks. So we transformed them into a different representation, by calculating: $total_{trans}(kWh) = \frac{W\_t_{now}}{1000} \cdot \frac{(t_{now}(ms) - t_{prior}(ms))}{3.600.000}$. Given the total consumption in kWh, we set the accuracy function to the difference between the original and the processed data: $accuracy = 1 - \frac{|total_{after} - total_{before}|}{total_{after}}$.

Computational Overhead: Computation time is the average over ten runs on an Intel(R) Xeon(R) CPU 5110 @ 1.60GHz single core system.

Comparing different parameters for aggregation and perturbation we check if the resulting data still allows deductions. In other words, we check if “data minimisation” [81] can take place.

Aggregation: We aggregated using the harmonic mean over different time intervals ranging from 10 minutes to 1 hours. As Figure 11 shows, long time intervals results in far less data.

Aggregation over Time - Feature Extraction Accuracy (%)	
Presence Detection - 4 hours	97,3 %
Presence Detection - 24 hours	92,5 %
Turning-On of devices - 4 hours	10,4 %
Turning-On of devices - 8 hours	5,4 %
Peak detection - 10 minutes	7,4 %
Peak detection - 1 hour	6,2 %
Interval of 8 hours	
Accuracy (%)	99,2 %
Compression (%)	99,7 %
Comp. Overhead (s)	0.8 sec

Table 11: Results for aggregation over different time intervals

Obviously, it reduces the amount of private information, but still as our analysis of 4h and 24h presence detection over interval shows, it remains usable data, e.g., for statistical predictions in the smart grid.

Reduction: As Figure 12 shows that the dataset with a reduced temporal resolution, i.e. 1 minute and 8 hours had no impact on the empirical accuracy. Although 8 hours are double the interval of presence detection, only a marginal impact on the presence detection is observed. It remains to be seen if this due to peculiarities of this household.

Reduction of Resolution: Feature Extraction Accuracy (%)		
Presence Detection - 4 hours		74,2 %
Presence Detection - 24 hours		78,5 %
Turning-On of devices - 4 hours		15,9 %
Turning-On of devices - 8 hours		10,8 %
Peak detection - 10 minutes		100 %
Peak detection - 1 hour		100 %
Interval	**1 minute**	**8 hours**
Accuracy (%)	100 %	99,9 %
Compression (%)	0,0 %	0,0 %
Comp. Overhead (s)	14.3 sec	8.1 sec

Table 12: Results for reduction of resolution of time

Perturbation: From the standpoint of privacy protection the notion of differential privacy seems to be promising [75]. We just kept it much simpler, knowing that we loose on privacy [130]: First, we take the average determined by the `AVG` function of MySQL, standard deviation determined by the `STD` function of MySQl. Second, we calculate the new value by adding the noise to the previous value. We utilise a uniform or a gauss distribution, calculated in python as follows:

$new\_val = old\_val + rand.uniform(\frac{avg}{2}, (avg + \frac{avg}{2}))$

$new\_val = old\_val + rand.gauss(avg, std\_dev)$

Perturbation: Feature Extraction Accuracy (%)		
Presence Detection - 4 hours		88,1 %
Presence Detection - 24 hours		99,5 %
Turning-On of devices - Gauss		2,5 %
Turning-On of devices - Uniform		1,6 %
Peak detection - Gauss		13,8 %
Peak detection - Uniform		23 %
Distribution:	**Gaussian**	**Uniform**
Accuracy (%)	22,1 %	21,2 %
Compression (%)	0,0 %	0,0 %
Comp. Overhead (s)	8.7 sec	8.6 sec

Table 13: Results for perturbation

Figure 13 again shows that large and generic detections, even if simplistic, can hardly be disturbed by noisy data. Which again means, that simple noise is to be tolerated for some applications and hence "the data collected [..] should be strictly necessary for the specific purpose previously determined by the data controller (the ″data minimisation″ principle)" [81]. However, simple noise does not add to a statistically provable consumer privacy [130].

3.9.2 Technical mechanisms to achieve component's function

As a result of the above case-study we note, that for privacy reasons data often is in the need to be modified. RERUM wants to achieve integrity for end-to-end communication. If data is protected by a classical digital signature scheme, e.g. RSASSA-PKCS-v1.5-SIGN [128] the moment the verifier only has access to the original signature value and a somehow modified, e.g. reduced resolution, of the signed message the signature will no longer be valid. The case study showed that sensible predictions and thus intelligent reactions are still possible with lower quality data. However, the origin of the data, and maybe also the amount of data redaction, must be kept at verifiable level. Hence, RERUM wants to allow subsequent modification to increase privacy. Malleable Signatures ($\mathcal{MSS}$) are RERUM's chosen tool to maintain a lower bound of integrity and allow to

- verify that only authorised modification have been applied, and
- authenticate the origin of the authorised modified data, and

- identify the origin of the unmodified data and the consent for modification.

Malleable signature schemes ($\mathcal{MSS}$) enable a third party to alter signed data in a controlled way, maintaining a valid signature after an authorised change.

3.9.2.1 Functionality of malleable signature schemes ($\mathcal{MSS}$)

In RERUM Deliverable D3.1 [201] we already presented the harmonised notation for Malleable signature schemes ($\mathcal{MSS}$). To make this deliverable self contained we briefly introduce them here. Malleable signature schemes (MSS) allow generating a signature over data that allows a specified third-party to modify signed data and re-compute a potentially different signature, which is again valid for the modified data; the re-computation of the signature can be done without the signer's signature generation key. The signature on the modified data is valid under the signer's public verification key if and only if the signer-specified rules for subsequent modifications are adhered to. As such, malleable signatures schemes (MSS) shall offer:

1. integrity protection for the message, protecting against subsequent malicious or random, but unauthorised modifications, and
2. authentication of origin and consent to authorised modifications of the message, as the party that applied the reduced integrity protection on a message, by signing it, can be identified by the corresponding verification key, with
3. accountability for the message's current state, potentially not requiring an interaction with the signer, and
4. cryptographically strong[5] privacy guarantees for the original version of data if it was modified (sanitized or redacted).

Note, the latter —cryptographically strong privacy— means that "[n]obody should be able to restore sanitized parts of a message. For example, if we have pseudonyms in medical documents then, of course, the original names should not be recoverable." [33]

3.9.2.2 Applied cryptographic research conducted for RERUM in the area of malleable signature schemes

RERUM thoroughly analysed the current state of MSS to understand what algorithms to choose for RERUM's idea to apply malleable signatures already on devices. We identified two different currently studied cryptographic constructions: redactable signature schemes (RSS) and sanitizable signature schemes (SSS). These results led to a harmonised view on both forms of constructions, which was presented and disseminated as early as possible in RERUM's deliverable D3.1 (Sect. 2.3.9 of [201]) and published in [66]. Moreover, we found certain gaps that needed to be addressed in order for those schemes to become useful. Several research results have been obtained in the course of this deliverable. They are filling gaps in functionality while still keeping RSS and SSS cryptographically strongly private. They have let to the following published papers:

[5]at least as strong as formally defined by Brzuska et al. [33]

[66] at ESSOS'11 This work harmonises and describes the subtle but important differences between redactable signature schemes and sanitizable signature schemes (see Sect. 2.3.9 of RERUM Deliverable D3.1 [201]).

[186] at ARES'15 This work adds accountability to redactable signature schemes (see Sect.4.2.5).

[67] in Journal of E-Business and Telecommunications 445-2014 This work adds the flexibility to redact arbitrary content from tree-structured data (e.g. JSON) (see Sect.4.2.6).

[185] at ACNS'14 This work explicitly captures how to allow merging two redacted versions from the same source message into one message, allowing to save space as only one signature is needed and giving privacy hiding that the message was previously split (see Sect.4.2.7).

3.9.2.3 Message level application for end-to-end integrity

When sensory information is gathered by constrained devices (see [29] for classification) and the data is then forwarded to other constrained devices or to servers. It might be immediately processed, but often it is stored in message queues to be picked up later by applications to achieve the desired functionality. For example assume the sensor with the thermistor to continuously push his readings into a message queue on some server. Asynchronously this message queue is read by several different applications. Protecting the integrity for those type of loosely connected processing can be achieved by message-level protection mechanisms. Using a cryptographically secure signature scheme allows verifying that data has not been modified in unauthorised ways. Additionally, you gain origin-authentication, i.e., verifying which entity signed the data. Note, all the methods for RERUM must be capable of being executed on the constrained device, e.g. the ReMOTE. This is inline with the goal to provide integrity end-to-end or on the transport level, but starting at devices, to protect against Loss of U-DATA Integrity (Threat#05), Loss of C&C-DATA Integrity (Threat#06) and Loss of S/W Integrity (Threat#07) from RERUM Deliverable D2.1 [167]. For the Integrity Generator / Verifier we had differentiated between two types:

- **integrity protection applied on the transport-layer**:
  Transport-layer integrity protects the channel between two communicating entities, such that inside the channel integrity cannot be violated with out being detected by the communication partner.
- **integrity protection applied on the message-level**:
  Message-level integrity creates an integrity check value, e.g., using digital signature, over the message and then send message and signature to the communication partner. The latter can be done even over an unsecured (regarding integrity) communication channel.

On the transport layer, this can be achieved by DTLS (for channels between constrained devices) and TLS (for channels between gateways and servers) or by DTLS all the way. End-to-end integrity protection can be achieved with transport layer technology, i.e. by the use of an DTLS channel between the application and the device, see Figure 35a. While this truly protects data from the constrained device to the application the drawback is that there is a need to establish a direct link, which is needed for confidentiality protection (which DTLS offers) but unnecessary for integrity alone.

Transport-layer channels can also be established hop-by-hop see Figure 35b. However, this does not offer end-to-end integrity protection.

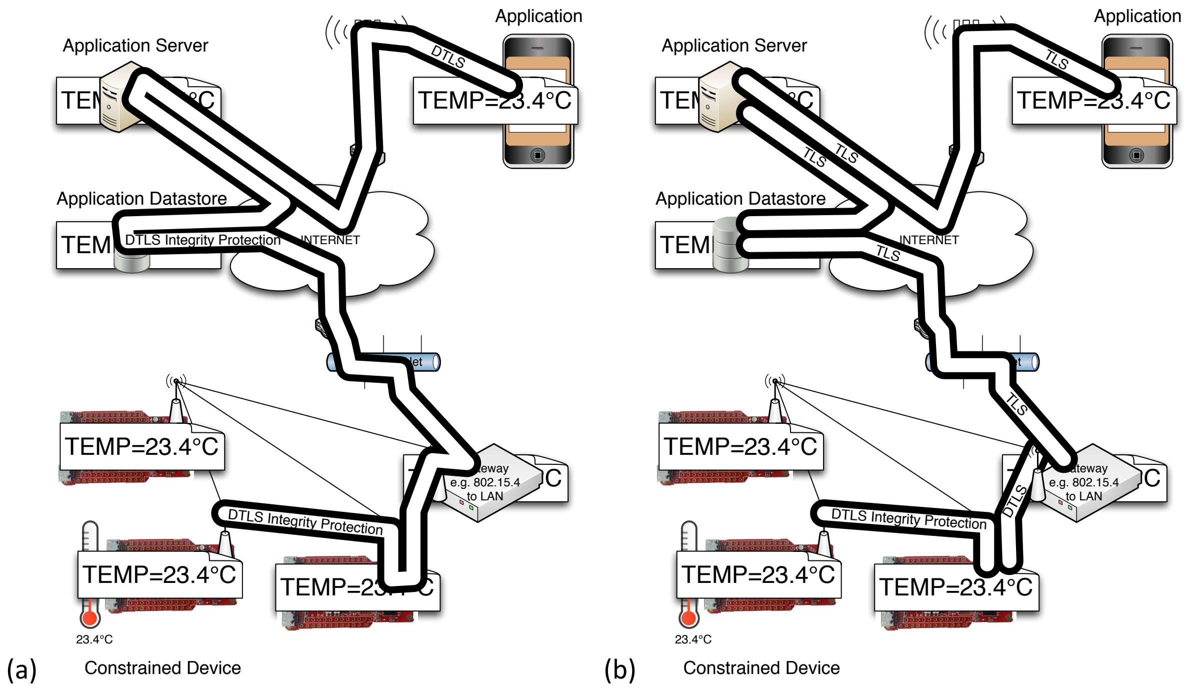


Figure 35: Integrity protection on transport layer: (a) DTLS channel protects data from the constrained device to the application; drawback is that there is a need to establish a direct link, which is needed for confidentiality protection but unnecessary for integrity alone (b) DTLS channel protects data on each communication hop between two constrained devices, but every hop can modify the data on its way to the application; integrity is not protected end-to-end

The cryptographic primitives, as well as the data types to transport signatures on the message-level have been described in RERUM Deliverable D3.1.[201] the resulting data type is currently implemented for testing in prototypes and the results have been published and presented at the Workshop on Extending Seamlessly to the Internet of Things (esIoT), collocated at the Ninth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2015) in July, 2012 [182].

3.9.2.4 Functionality and interaction with enhanced Integrity Generator / Verifier

The usual functions are *Sign* and *Verify*. *Sign* would allow to generate the integrity check value. *Verify* respectively verifies the integrity of the supplied message with respect to the supplied integrity check value. In a nutshell, malleable signature schemes have one additional algorithm to do the authorised modification and re-compute the integrity check value. We denote this functionality by *Sanit*. Each of the cryptographic schemes that were devised in the course of the research and given in this deliverable will give more details.

The Integrity Generator / Verifier runs in the RERUM Device (RD). If for example sensory data is gathered, the resources manager senses the environment and produces data. If this data is to be signed —for example before sending it wirelessly— the Integrity Generator / Verifier is called to produce an Integrity protecting cryptographic integrity check value. In the following interaction diagrams we assume that a malleable signature might be the algorithm chosen. The Integrity Generator / Verifier is designed to handle other signature schemes as well, e.g. AES-based MAC or standard digital signatures like ed25519. Figure 36 shows how to generate a signed JSON object.

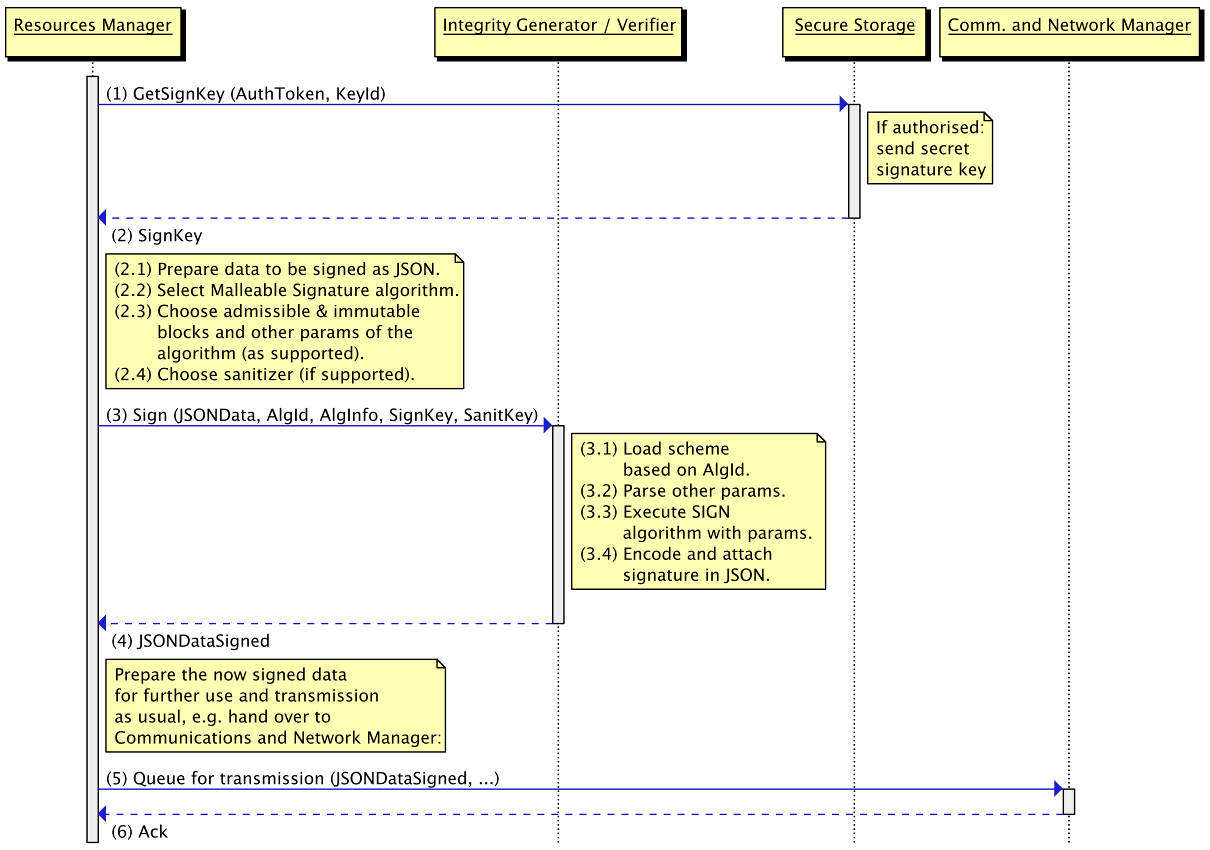

Resources Manager
Integrity Generator / Verifier
Secure Storage
Comm. and Network Manager
(1) GetSignKey (AuthToken, KeyId)
If authorised: send secret signature key
(2) SignKey
(2.1) Prepare data to be signed as JSON.
(2.2) Select Malleable Signature algorithm.
(2.3) Choose admissible & immutable blocks and other params of the algorithm (as supported).
(2.4) Choose sanitizer (if supported).
(3) Sign (JSONData, AlgId, AlgInfo, SignKey, SanitKey)
(3.1) Load scheme based on AlgId.
(3.2) Parse other params.
(3.3) Execute SIGN algorithm with params.
(3.4) Encode and attach signature in JSON.
(4) JSONDataSigned
Prepare the now signed data for further use and transmission as usual, e.g. hand over to Communications and Network Manager:
(5) Queue for transmission (JSONDataSigned, ...)
(6) Ack

Figure 36: Interaction when signing data, e.g. with a malleable signature scheme

As mentioned, Integrity shall be verified as well. Figure 37 shows how to verify received or stored data. This can be used to check the integrity and origin of a received command, data file or of received sensor reading from other RDs. It can also be used to check the validity of an over the air (OAP) update file. Validity for an OAP file here means that the integrity check says unchanged and the origin can be verified to be the trusted security center. In order to identify the trusted origin, Integrity verification with digital signatures defines against which public key(s) a signature must verify. Hence, also in Figure 37 the first interaction is not with the Integrity Generator / Verifier, but with the Trusted Credential Store that resides in Secure Storage. Even public verification keys must be kept in Secure Storage to withstand attacks of them being overloaded. If an attacker could convince an RD to believe that the attacker supplied key is the one of his security center, then the attacker could masquerade. After retrieving the key(s) —we need also the sanitizer's key for $\mathcal{SSS}$— the Integrity Generator / Verifier is called. Please see Figure 47 for different interactions in order to get detailed accountability information in the case of different malleable signature schemes, i.e. interactive vs. non-interactive accountability.

In the case of the signature offering some form of authorised subsequent modifications the Integrity

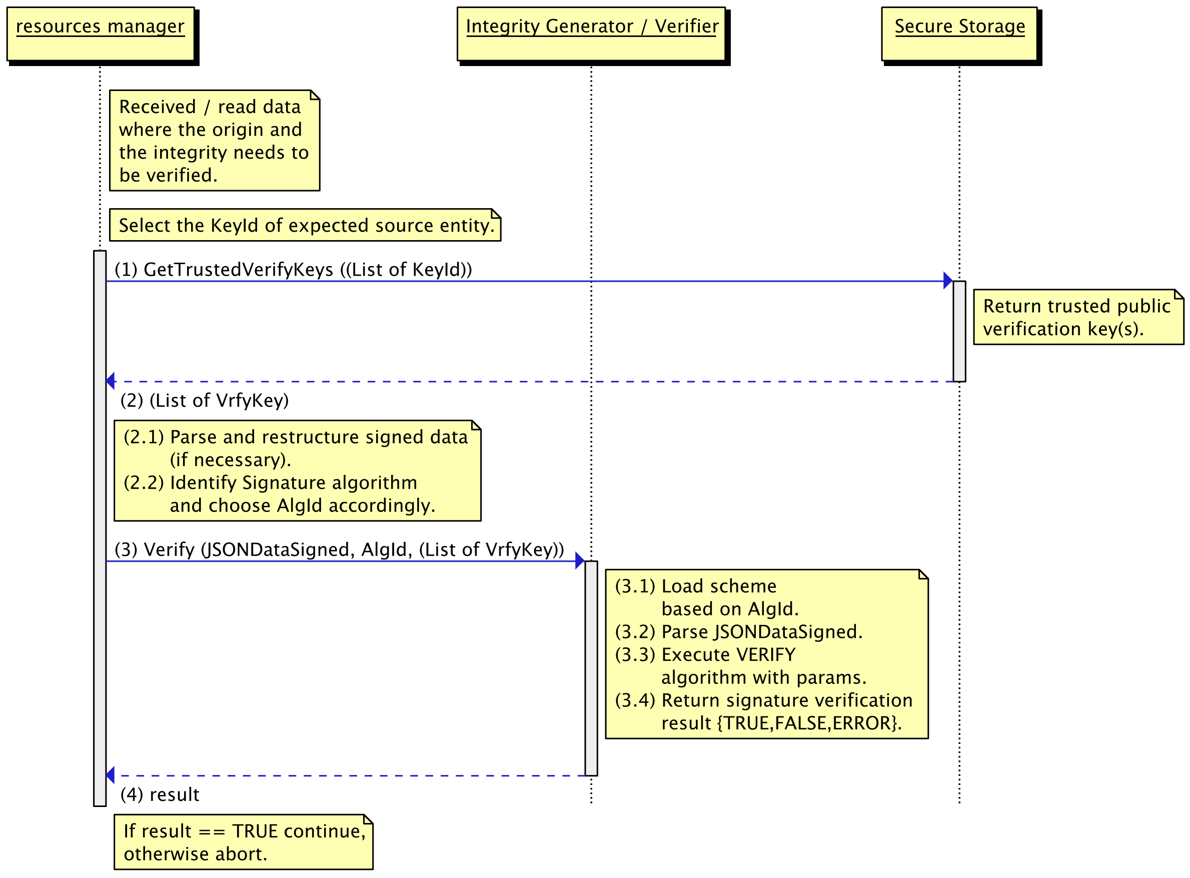

resources manager
Integrity Generator / Verifier
Secure Storage
Received / read data where the origin and the integrity needs to be verified.
Select the KeyId of expected source entity.
(1) GetTrustedVerifyKeys ((List of KeyId))
Return trusted public verification key(s).
(2) (List of VrfyKey)
(2.1) Parse and restructure signed data (if necessary).
(2.2) Identify Signature algorithm and choose AlgId accordingly.
(3) Verify (JSONDataSigned, AlgId, (List of VrfyKey))
(3.1) Load scheme based on AlgId.
(3.2) Parse JSONDataSigned.
(3.3) Execute VERIFY algorithm with params.
(3.4) Return signature verification result {TRUE,FALSE,ERROR}.
(4) result
If result == TRUE continue, otherwise abort.

Figure 37: Interaction when verifying signed data, assuming we want no additional information about who is accountable in the case of malleable signature schemes.

Generator / Verifier component is called to re-compute the signature after an authorised modification has happened. This algorithm is called *Sanit*. Depending on the schemata the modification requires key material, i.e. sanitizer's secret key. Hence, also in Figure 38 the first interaction is not with the Integrity Generator / Verifier, but with the Trusted Credential Store that resides in Secure Storage to retrieve necessary secrets.

In Figure 38 we assume that the signed data is in JSON Sensor Signature format. We expect to implement JSS handling in our prototype of the Integrity Generator / Verifier during Task 5.3.

3.9.3 Summary

The Integrity Generator / Verifier Enhancement for Authorised Malleability reaches level 3. We have devised new cryptographic malleable signature schemes and published them. You will find three new schemes: No.1 published in [186] at ARES'15 in Section 4.2.5, No.2 [67] in Section 4.2.6, and No.3 published at ACNS'14 [185] in Section 4.2.7. Those schemes are currently starting to be subjected to laboratory trials, in pending Task 5.3. Table 14 shows the summary table for this components technical readiness and novelty.

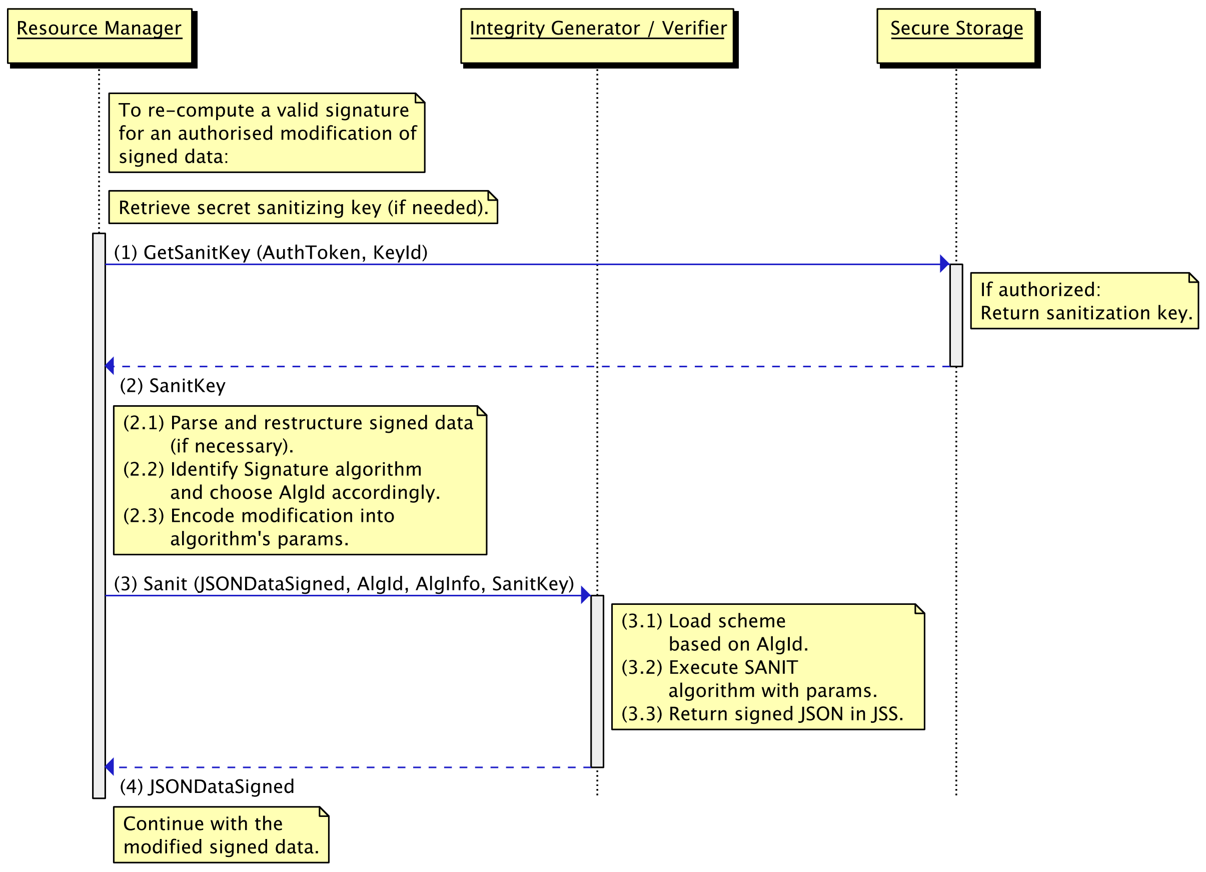

Resource Manager
Integrity Generator / Verifier
Secure Storage
To re-compute a valid signature for an authorised modification of signed data:
Retrieve secret sanitizing key (if needed).
(1) GetSanitKey (AuthToken, KeyId)
If authorized: Return sanitization key.
(2) SanitKey
(2.1) Parse and restructure signed data (if necessary).
(2.2) Identify Signature algorithm and choose AlgId accordingly.
(2.3) Encode modification into algorithm's params.
(3) Sanit (JSONDataSigned, AlgId, AlgInfo, SanitKey)
(3.1) Load scheme based on AlgId.
(3.2) Execute SANIT algorithm with params.
(3.3) Return signed JSON in JSS.
(4) JSONDataSigned
Continue with the modified signed data.

Figure 38: Interaction when doing an authorised modification and re-computing a verifiable signature, assuming a malleable signature scheme was used for signing the data in the first place

Table 14: Technical Implementation Summary for Privacy Enhancement for Authorised Malleability of the Integrity Generator / Verifier Component

Integrity Generator / Verifier Enhancement for Authorised Malleability		
Technical Level (description given in 3)	Level 3	New MSS schemes have been designed to fit RERUM requirements but not too overshoot in cryptographic strength as this induces unnecessary overhead. Currently selected optimal schemes are implemented for devices. Initial implementations show them to be fast enough to run on RERUM gateways. Laboratory prototype for testing is underway. Depending on overhead to be determined it might be ready for trials in Y3.
Suggested Method(s) for Implementation	Private Malleable Signature Schemes	
	Section 4.2	
Technical Readiness of Implementation within RERUM	Design	**yes**, several new cryptographic schemes
	Experiments, Simulations	**yes**, underway as scheduled for T5.3
	Trial	maybe

3.10 Privacy Policy Checker and Attribute Need Reporter

This document has already dealt with the privacy of the data requested from the services. More specifically, there is a consent manager that produces privacy policies that are used to specify privacy criteria in the form of XACML policies that are enforced later in their respective PEPs. However, both security and privacy policies are normally dependent on the identity of the RERUM registered user accessing the service and their attributes, and the access to these attributes should be authorized as well. As explained in D3.1 [201], the provision of these attributes is delegated in an identity provider, which should be responsible for authorizing any requests sent to it, and an Identity Agent component is responsible for retrieving this information.

But then comes the problem of what information will be retrieved from the Identity Agent and how to guarantee that its access has been granted by the security policies. Regarding the information to be retrieved by the Identity Agent, it would be possible and easier to have the Identity Agent retrieving all the information of the RERUM registered user at the start of the session, but this might result in the Identity Agent both trying to access user information that it is not allowed to or information that it is allowed to be accessed but it really does not need to access. The latter might happen due to the attribute being not referenced in any privacy or security policy of the system.

Ideally, the identity provider should have their own authorisation layer that would check the access of the RERUM registered user utilised by the Identity agent to query the user information. In that case it would not be necessary for RERUM components to check for privacy of the requested attributes, because it would have been already done by the security layer of the identity provider. But in real world, many identity providers, especially legacy ones, lack a fine-grain authorisation layer that checks for access of each individual attribute, and many more do not have any privacy check.

Besides this problem, even if RERUM was trying to access only those attributes that it is authorised, it would be against the data minimisation principle to ask for all of them only because it has been granted so. Instead, the Identity Agent should ask only for those attributes that are needed for the authorisation process because they are referenced in the policies of the System.

In short, this section deals with how to deal with privacy for the authorisation process when the identity provider does not provide that functionality.

It can still be argued that actually it would be possible to ask for each attribute only when the policy is going to be evaluated, but this have a very big problem. Retrieving attributes from a identity provider usually consumes much time, which has much more to do with the number of times this information is asked for than with the amount of information retrieved. If the attributes were retrieved each time a policy is being evaluated, this operation that consumes much time would be repeated once and again, especially taking into account that accessing a given service may require evaluating multiple policies. Hence, retrieving all needed attributes at once is much more efficient in terms of number of messages exchanged with the identity provider and hence response time. This is why it is preferable to have some way to know in advance what information will be needed and retrieve it each time the session is renewed. The Indentity Agent already introduced in section 4.4.4 of D3.1 already dealt with the problem of holding a cache of attributes, but did nothing to try to limit the set of attributes retrieved to the needed in the authorisation process. This section also deal with the problem of identifying a complete set of attributes needed in the authorisation process to be able to ask them in a single operation.

3.10.1 RERUM approach to Privacy in Authorisation: PPC and ANR

To deal with these problems, RERUM provides two new components named 'Privacy Policy Checker' (PPC) and 'Attribute Need Reporter' (ANR) that enrich the Identity Agent. The PPC checks the Privacy Policies against the access policies each time the Identity Agent tries to access the user attributes of the RERUM registered user that is being used for the request (requests with no valid RERUM registered users are rejected), and the ANR renews the set of user attributes to be checked by the PPC each time Privacy or Access Policies change in the system, due to any operation of creation, removal or modification on each of them. This way RERUM is:

- ensuring that no access to any banned user attribute is attempted even if the security policies ask for them and
- asking only for those user attributes that are subject to be used in the authorisation process.

Finally, it could still be argued that even if the privacy policies guaranteed that only granted attributes are retrieved, the Identity Agent might still retrieve some unnecessary user attributes. That could happen if the user does not access all the services in the system during a given session. That user information could be referenced in some privacy policy corresponding to any of those services not invoked during the session. That is, it is theoretically possible that some user attribute are retrieved for legitimate proposes but unnecessarily. That is true, but as explained before, the performance of asking once for each session will be much higher than asking for each request. Asking for information to an external identity provider can consume much time. The main factors for that are the time that the identity provider needs to receive the request through the network and analyze it, which is very similar for one single attribute as for many. For this reason, asking for a single block of many attributes normally takes much less time than asking many times for a single attribute.

Additionally, it could happen that some users have access to many more services than others, but requires more user attributes to be checked for these purpose than the ones that access only some services. For instance, if two different applications requiring different user attributes were accessing the same RERUM installation, they could ask for different sets of user attributes. If a RERUM registered user utilised only one of these applications, the user attributes needed for the other application would still be in the list produced by the ANR. But here the PPC would check if the access to these user attributes has been really granted. The attributes needed for the second application would only be retrieved if the human being previously agreed on that.

This innovation is in line with the privacy principles of RERUM and is a privacy refinement of the authorisation process. As such, it is meant to be transparent for the system unless the request is rejected. The benefit for the smart cities are an enhanced treatment of the privacy of the owner of the data, let it be the municipalities or the citizens using the system. In concrete, citizens that registered as RERUM registered users in RERUM and later decided to withdraw completely or partially the consent on accessing their user attributes will benefit from this, but at the cost of their access to the application being affected accordingly to the lack of these attributes (see Section 4.5.3).

In summary, these components provide:

1. an additional privacy check that will be especially useful when dealing with legacy identity providers that do not properly check for privacy when providing user information,

2. a cache of user attributes used for authorisation that will boost authorisation performance avoiding the need for asking for these attributes for each request, and
3. only those attributes that are necessary for the authorisation policies will be required each time.

3.10.2 Example of user attribute retrieval

Let us suppose a system with the following policies:

1. **Global policy** 'only_active_users_work' states that only users whose attribute 'active' equals to 'true' are granted access. As any global policy, it applies to all RERUM services.
2. **Local** 'policy temperature_set_only_by_administrators' applies only to RERUM service set_temperature and states that only users whose attribute 'role' is set to 'administrator' are allowed to set the temperature.
3. **Local policy** 'turn_on_tv_after_midnight' applies only to the RERUM service 'turn on tv' and states that the tv can be set after midnight only if the user attribute 'age' is bigger than 6.

In this case, the starting list of needed attributes will be: ('active', 'role', and 'age'). The hour of the operation being requested is a system attribute but not an attribute of the user, and for this reason, it is not included on the list.

For this example let us additionally suppose that the user has rejected access to his attribute 'age' even for authorisation purposes. For this reason, the system will need to contain the following privacy policies regarding these attributes:

1. **Policy** 'privacy_active' granting access for attribute 'active' for purpose 'Authorisation'.
2. **Policy** 'privacy_role' granting access for attribute 'role' for purpose 'Authorisation'.
3. **Policy** 'privacy_age' granting access for attribute 'age' for purpose 'Authorisation'.

With this list of needed attributes, the IdA iterates for each of this attributes, and obtains the following results, after executing the PPC for checking their corresponding privacy policies:

- 'active', 'true'
- 'role', 'administrator'
- 'age', 'access rejected'

These will be the values that will be passed to the authorisation process. As a result, the user will be able to access the service 'set temperature', but will not be able to access the service 'turn on tv' regardless of his role or age, because one of the values were needed for evaluating the access policy.

3.10.3 Summary

ANR provides a way to make an initial filter on the set of user attributes to be retrieved by the IdA to limit it to only those attributes that are referenced in the policies present in the system.

PPC provides a mean for checking that the access to the user attributes utilized in the authorisation process have actually been granted for the corresponding RERUM registered user for authorisation purposes.Though, strictly speacking, this check should normally be provided by the Identity Provider, the use of legacy systems make necessary to provide such complementary measure.

The joint use of ANR with PPC and IdA provide a mechanism for the authorisation process to ask in advance for those user attributes that will be needed and have been granted by the RERUM registered user.

Besides, as the IdA asks for this user attributes at the start of the session of the user, the result is a significant decrease in the number of messages sent to the Identity Provider to retrieve this information.

Table 15 summarises how this section contributes to the state of the art.

Table 15: Technical Implementation Summary for Privacy Policy Checker (PPC) and Attribute Need Reporter

Technical Mechanisms to achieve Component's Function		
Technical Level (description given in 3)	Level 3	Providing privacy and minimisation components for user data is something already widely implemented. But Privacy projects tend to obviate the differences between accessing user attributes by the services and by the authorisation process. Privacy policies regarding the access to user attributes for authorisation must necessarily be more limited than the ones used for the services because the only user attribute they should be allowed to refer is the user-id to avoid entering circular loops. This section enters not only the specifics of the privacy of user attributes used for authorisation but additionally its design and implementation
Suggested Method(s) for Implementation	Analyze available policies in the system to get an initial list of user attributes needed for authorisation and privacy purposes 4.5	
	Evaluate privacy policies for each user attribute contained in the list of user attributes needed for the authorisation process, see Section 4.5	
Technical Readiness of Implementation within RERUM	Design	Yes
	Experiments, Simulations	No
	Trial	Yes

3.11 Summary

In this chapter we covered the RERUM Privacy Functional Components specified in D2.3 [219] and shown in Figure 6.

RERUM's Security and Privacy Centre is formed by five RERUM components: the "**User Consent Manager**", the "**Privacy Dashboard**", the "**Deactivator / Activator of Data Collection**", and the "**Anonymising and Pseudonymising Management**" including the "**De-Pseudonymiser**". Components residing in the RERUM Device (which are part of the on-device "S&P&T components") are the RERUM "**Privacy Policy Enforcement Point (PEP)**" and the RERUM "**Geo-Location PET**".

The User Consent Manager supports the data controller to request consent and the data subject in granting / revoking consent. It derives privacy policies from consents and allows for semi-automation of consent granting based on consent handling preferences. In the Privacy Dashboard the user can specify privacy preferences, which that component translates into privacy policies. The Privacy Dashboard also keeps the user updated about relevant events and maintains a history of present and past interactions. Data collection is controlled by the RERUM Activator / Deactivator of Data Collection, it enables the user to opt-in and opt-out individually from all applications. This is done by preventing the data stream from passing RERUM Middleware (see Figure 6). The Anonymising and Pseudonymising Management component protects the identity of users and devices. Pseudonym management and agreement is handled in the Security and Privacy Centre. However this can also be done by a separate component residing in the RERUM Device. The related De-Pseudonymiser allows re-linking of pseudonyms for special use cases (see Section 3.6 for details). Privacy policies generated by the Consent Manager and the Privacy Dashboard are evaluated and enforced by the Privacy PEP. This latter component grants access to services and intercepts communications if needed. The Geo-Location PET provides location privacy. It receives GPS data from users and processes them to privacy enhanced data sets for service providers.

We summarised the security components mentioned in D3.1 [201], the "**Data Encrypter / Decrypter**", the "**D2D Authenticator**", the "**Credential Bootstrapping Client / Authority**", and the "**Trusted Credential Storage**", and detailed their relevance for enhancing privacy. All these security core components are related to essential communication security mechanisms shown in Figure 29 as specified in D2.3 [219].

We presented two new privacy components not sketched in D2.3, but outlined in D3.1 and D2.5 [157].

One, a crucial security component shown in Figure 29, is the RERUM "**Privacy Enhanced Integrity Generator / Verifier**". It protects data and commands from unauthorised modifications and allows authentication of the origin. For it we devised new cryptographic malleable signature schemes published in [186], [67], and [185].

The other new privacy component consists of two parts, the "**Privacy Policy Checker**" and the "**Attribute Need Reporter**". The first computes the user attributes needed. The latter ensures access control to these attributes. An Identity Agent security component is retrieving this information. The joint use of Attribute Need Reporter with Privacy Policy Checker and Identity Agent provides a mechanism for the authorisation process to ask for required attributes granted by the RERUM registered user.

4 RERUM privacy enhancing protocols and mechanisms

This chapter provides an in-depth description of the RERUM privacy enhancing protocols and mechanisms specifically developed for or adapted to and improved for RERUM needs. We also elaborate on relevant aspects of certain RERUM privacy enhancing components.

(1) **Sticky policies:** Section 4.1; a privacy policy containing the data subject's expectations and wishes regarding their personal data may be attached ("stuck") to the data in transmission and at rest. This allows data processors to learn about and comply with the data subject's requirements.

(2) **Malleable signatures on devices:** Section 4.2; the malleable signature schemes we have newly designed for RERUM. They are currently being implemented for RERUM devices.

(3) **Data Perturbation with integrity preservation on the gateway:** Section 4.3; we balance the conflicting interests of privacy and integrity including accuracy by specifying a privacy gateway that uses data perturbation on redactably signed meter values providing a privacy guarantee of differential privacy with only a small computational overhead.

(4) **Privacy Policy Enforcement Point:** Section 4.4; we explain how the authorisation components already defined in D3.1 are upgraded so they can additionally support privacy policies and combine them at both local and global levels.

(5) **Enhanced privacy for user information retrieval:** Section 4.5; we detail how the new component PPC and ANR work jointly with the IdA to enrich it to support privacy in the authorisation process.

(6) **Pseudonyms:** Section 4.6; The presented pseudonym generation and management mechanism is based on Hash-Trees, using an innovative top-down approach. It is computational and battery efficient and supports efficient de-pseudonymization as well.

(7) **Consent for authorisation:** Section 4.7; specifically for RERUM we developed a concept for privacy-enhanced tokens for authorisation in constrained environments, which is actively developed within the IETF. Here the same mechanism used for generating pseudonyms can also be used for generating privacy-enhanced tokens.

(8) **GeoLocation position hiding:** Section 4.8; we explain the technical details of our RERUM position hiding mechanism where a traffic participant sends a random number of vectors, which are again determined by random timers. The approach allows the adaption of user preferences, and temporary opt-out of the data collection even initiated automatically by default in privacy-critical situations.

(9) **Compressive sensing encryption:** Section 4.9; we propose a method that makes compressive sensing more immune to CPA attacks involving a chaos sequence and generation of a secret sparsifying basis.

(10) **Leakage resilient MAC:** Section 4.10; we present an innovative leakage resilient MAC which can actually be used in practical applications.

4.1 Sticky policies

In Section 3.1.5 we pointed out that a given consent has to hold at all times, including data that is in transit through multiple parties. Machine-readable policies resulting from the consent can be attached (or "sticked") to a data set helping to define allowed actions and consent obligations for that data set.

4.1.1 Sticky Policy mechanism

We refer to the sticky policy mechanism suggested in [176] which allows access to personal data only upon satisfaction of the attached policies. These is achieved by encrypting the data set and disclosing decryption information to parties fulfilling the the policies. The sticky policy mechanism can be described by three basic steps, as shown in Figure 39.

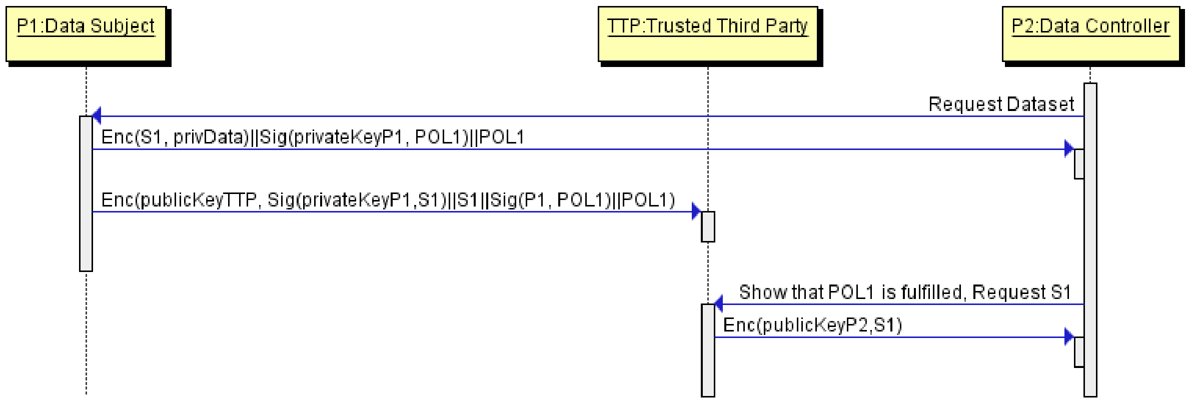

P1:Data Subject
TTP:Trusted Third Party
P2:Data Controller
Request Dataset
Enc(S1, privData)||Sig(privateKeyP1, POL1)||POL1
Enc(publicKeyTTP, Sig(privateKeyP1,S1)||S1||Sig(P1, POL1)||POL1)
Show that POL1 is fulfilled, Request S1
Enc(publicKeyP2,S1)

Figure 39: A simple Sticky Policy Mechanism

The three parties are assumed, person one ("P1") is the data subject creating data sets, the second person ("P2") is the data controller processing the data, and the third person is a trusted third party ("TTP"), which is able to verify that the data controller fulfils policy obligations.

Step 1 P2 requests personal data from P1. P1 generates a data set *privData* and according policies *POL1*. The data set is encrypted with a secret *S1* and the policies are attached as metadata to to the encrypted data. Alternatively, the policies could be stored in a public registry with only a policy pointer sticked to the data set's as metadata. Person one signs the policy with his private key *privateKeyP1* and sends the data, the policy and the signature to P2.

Step 2 P1 sends an an encrypted message to the trusted third party with S1, his signature over S1, POL1 and its signature.

Step 3 P2 wants to access the data set, which is encrypted with S1. P2 understands the attached policies POL1, he requests S1 from TTP, showing that he can fulfil the requirements from POL1. P2 receives S1, if TTP is convinced that P2 can fulfil the policies satisfyingly.

It should be noted, that in this small example, there is no need for a trusted third party, P2 could ask P1 himself for S1. In case of data in transit through multiple parties, P1 might not be available, thus TTP is assumed a party with much higher availability and connectivity than the data subject himself.

4.1.2 Sticky Policies in the RERUM architecture

The integration of sticky policies in the RERUM architecture relies on the policy generation as described in D2.3. Figure 40 illustrates where data is protected and policies attached.

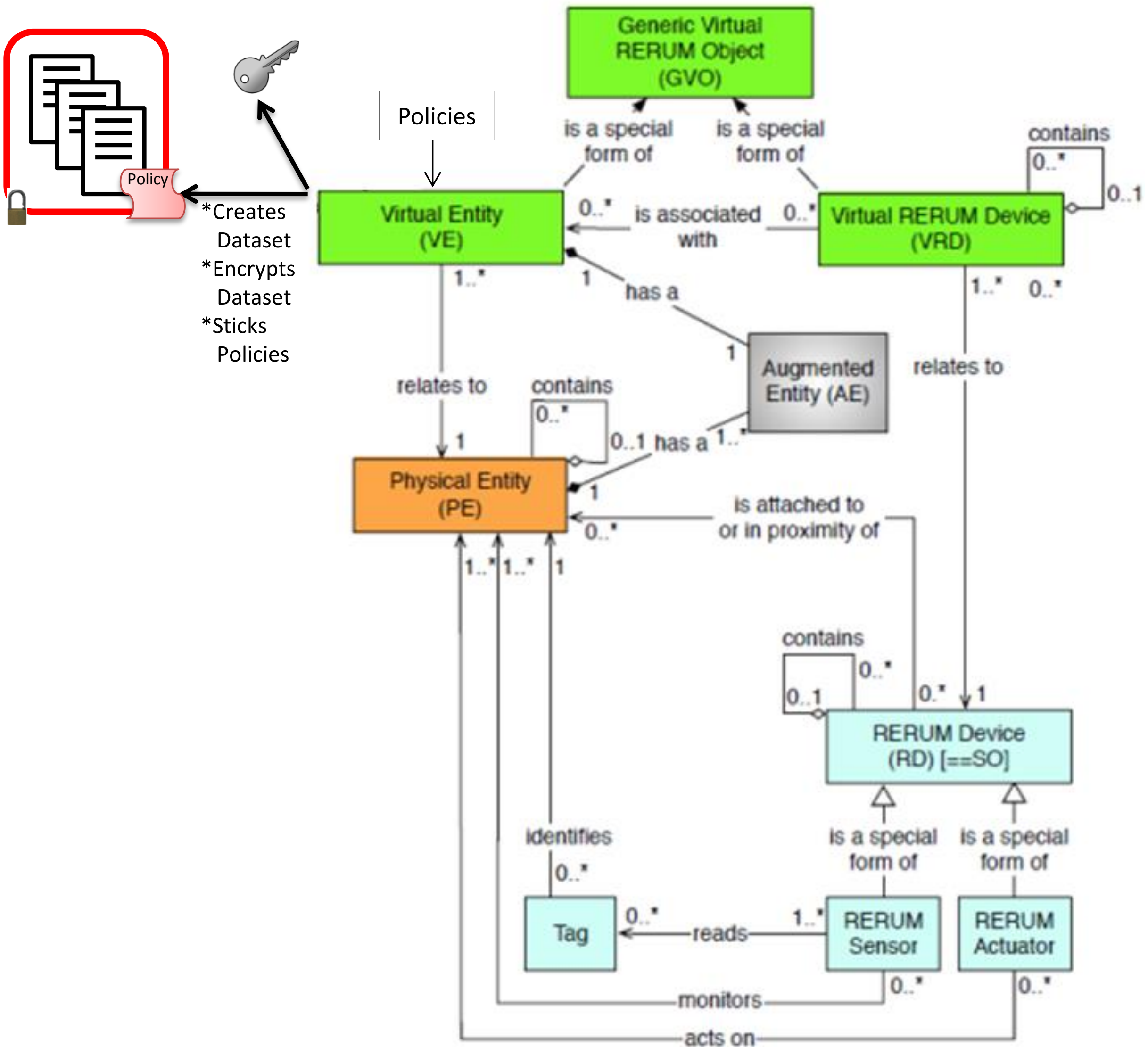


Figure 40: Sticky Policies in the RERUM ARM

For the application of sticky policies, policy generation and the provision of data is needed. Datastreams, which are not protected by sticky policies, are provided by RERUM devices, while pre-processed datasets are provided at the virtual entity. Policies are stored per physical entity at the corresponding virtual entity (see RERUM Deliverable D2.3 [219], Section 6.11.2.2). Therefore, a protected dataset can be generated at the virtual entity. That means a dataset is encrypted and attached policies to, and then sent to a requesting party. The corresponding secret is sent to either a trusted third party, which could be another, more powerful device of the data subject, or to a global privacy enforcement point at the

RERUM Security Center. Exercising the generation of a sticky policy protected dataset, the virtual entity would follow these steps:

Step 1 The virtual entity was requested a multiparty dataset. Multiparty datasets are always protected by sticky policies. The virtual entity generates the dataset and a corresponding secret S1. The policies to be attached are taken from the policy database of the virtual entity.

Step 2 The virtual entity encrypts the dataset with S1 and attaches the policies to the dataset. The virtual entity signs the policies with its private key (or with another secret which is verifiable by a public counterpart).

Step 3 The encrypted dataset, the policies and the signature are sent to the requesting party.

Step 4 The secret S1 and the policies are again signed by the virtual entity and sent in a confidential way to the trusted third party. The trusted third party in RERUM could be a device of the data subject which has a higher availability and connectivity or a trusted service found in RERUM's security and privacy center.

Step 5 The requesting party shows to the trusted third party that it can fulfil the requirements of the sticky policy. The TTP provides the secret in a confidential way to the requester.

Depending on the policies, there might be many requirements to be fulfilled before acquiring the set's secret. Pearson et al. [176] describe following possible policy requirements:

- proposed use of the data — for example: for research, transaction processing, ...
- use of the data only within a given set of platforms with certain security characteristics, a given network, or a subset of the enterprise
- specific obligations and prohibitions such as allowed third parties, people, or processes
- blacklists, notification of disclosure and deletion, or minimization of data after a certain time
- a list of trusted authorities (TAs) that will provide assurance and accountability in the process of granting access to the protected data, potentially the result of a negotiation process.

It should be noted that sticky policies first and foremost describe the obligations needed to process the data, but it cannot prevent misbehaviour after the data has been decrypted.

4.1.3 Summary

Sticky policies are a soft mechanism for privacy protection that allows service providers to be compliant with and to respect a user's wish for privacy. Sticky policies are used to

- attach policies to data,
- protect a data set until a service provider proves that he fulfils privacy requirement (this works up to a certain point), and
- allow a service provider to respect a user's wish, even with data sets from an unknown user.

4.2 Malleable signatures on devices

As described in Section 3.9, we present in this section the malleable signature schemes that we have newly designed for RERUM to address gaps of current schemes and to fit the needs of RERUMs use. We will list the gaps, for which we devised new schemes and we describe the newly needed security properties in Section 4.2. In Sections 4.2.5, 4.2.6, and 4.2.7 we give the details of three new schemes and offer rigorous proofs of their cryptographic security which includes cryptographic privacy. We do not offer the full background on the harmonised notation here again; the reader is referred to RERUM Deliverable D3.1 Section 2.3.9 [201] or the published papers, e.g. [66] for the notation and overview of security properties.

This section then offers our list of candidate functions which we currently try to implement in Contiki OS to run on the RERUM device, i.e. Zolertia's Re-MOTE in Section 4.2.8. We already have prototypes of many algorithms written in JAVA; details are given for the schemes below in the respective section on performance. Those prototypes show speeds that makes RERUM positively assume that they can be run in reasonable time in JAVA on a normal workstation. We plan to have some of the schemes implemented also as node-red[6] components, such that they can be easily integrated into IoT workflows in the non-constrained environment of the IoT processing chain.

Once the implementations are done, we can offer the first results from our laboratory experiments; see RERUM Deliverable D5.1 [168] for details on the laboratory tests planned. As part of these RERUM foresees to measure the runtime-overhead again for constrained (Zolertia Re-MOTE) and semi-constrained devices such as the RERUM gateway (e.g. RaspberryPi). We conclude in Section 4.2.9.

In the following four sections we highlight four important gaps that RERUM work has managed to close.

4.2.1 Gap 1: missing block-level-scope of properties

Sanitizable signatures bear an inherent risk. As introduced by *Ateniese* et al. [6] they explicitly allow for controlled modifications of a signed message. In particular, a $\mathcal{SSS}$ allows that a signed message $m = (m[1], m[2], \ldots, m[\ell])$ can be changed to a different message m'. For each, so called *block*, denoted as $m[i] \in \{0,1\}^*$, the signer has to decide whether a sanitization by a semi-trusted third party, called the *sanitizer*, is admissible during signature generation. The sanitization neither requires the signer's private key nor requires any protocol interaction with the signer. Hence, the sanitizer is able to derive a new verifying message-signature pair (m', σ') on its own behalf.

In an $\mathcal{SSS}$ a semi-trusted party is allowed to change signed data and thus the signing RERUM device gives up control over the message contents, while they are attributed to be originating from the signing device. The security model for accountable sanitizable signatures, introduced in [6], formalised and extended in [33], only allows to decide which party is accountable for the *complete* message-signature pair (m, σ).

Let us give an example, a message with some blocks is depicted in Figure 41. Assume that a message is split into blocks as depicted and that we want to remove some precision from the temperature to preserve privacy. Now without a block-level property, a sanitizer changing m into m' by adapting the precise centigrades of the temperature would become accountable for the whole message. This is what

[6] `http://nodered.org/`

Table 16: JSON example

```
{ ”temperature”:
   {”value”: 23.4,”type”: ”celsius”},
      ”rollingHourlyAverage”: {
            {”value”: 20.4,”type”: ”celsius”}
      }
}
```

is meant by message level properties. But, for a use in the IoT, RERUM assumed that it would be interesting to allow applications to detect that the blocks m_2 and m_3, that carry the temperature at the precision of one grade celsius, is actually original. With block-level properties this can be achieved.

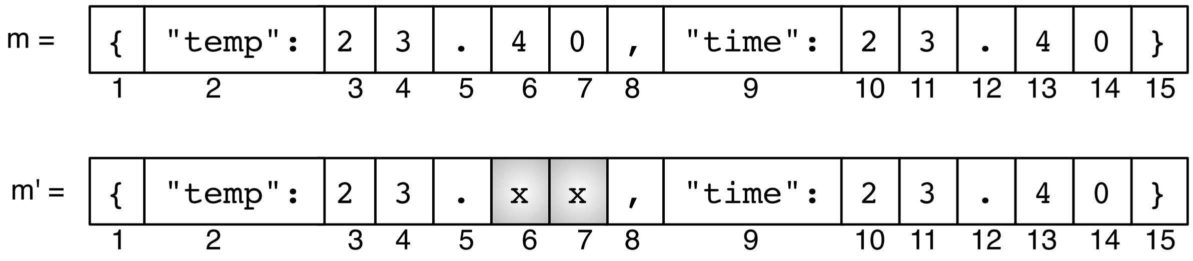


Figure 41: An example of groups-of-blocks that might require a per-block treatment

.

Moreover, this requires the verifier to obtain additional information that must be generated by the signing device as it involves the secret signing key.

4.2.1.1 Solution: Scheme No.1 (described in Section 4.2.5)

Scheme No.1 was devised to addresses this issue and allows accountability to be checked on each individual block. The details are described in Section 4.2.5.

4.2.2 Gap 2: Non-leaf node redaction in tree-data-structures must offer contextual integrity

Let us consider an example for a tree-structure. In the Javascript Object Notation (JSON) we represent some sensed data.[7] Table. 16 has a JSON formatted temperature reading of currently 23.4 degree celsius.

For the sake of this example assume that alongside the sensor also sends an rolling average and that it is encoded like in Table. 16 that is giving the tree depicted in Figure 42. We can think that a tree is encoded in the represented JSON. The tree encoded is depicted in Figure 42.

[7]JSON is very popular in the IoT domain see `http://postscapes.com/internet-of-things-protocols`

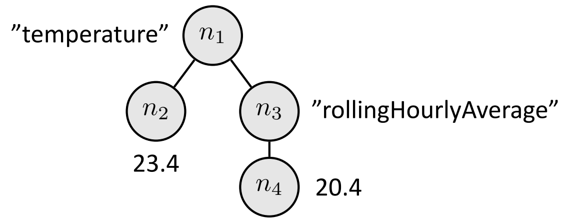


Figure 42: Tree encoded in the JSON from Table. 16

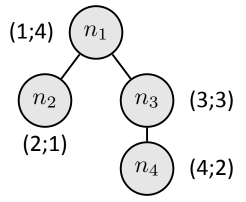


Figure 43: Original Tree with Traversal Numbers

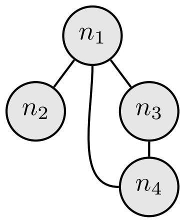


Figure 44: Transitive closure of the child-of relation

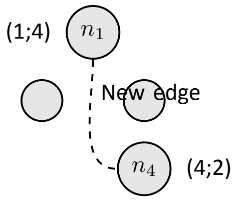


Figure 45: After removal of n_2 and n_3; with orig. traversal numbers

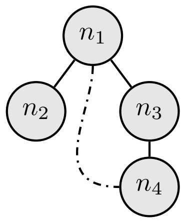


Figure 46: Added explicitly authorized potential edge

Consider that we still talk about the JSON encoded as the tree, depicted in Figure 42. We now want to show what happens if you could be able to redact, non-leaf nodes from this tree. That same tree data structure is depicted in Figure 43, ignoring the numbers in brackets for now. To remove the leaf n_4, the node n_4 itself and the edge $e_{3,4}$ is removed. By consecutive removal of leaves, complete sub-trees can be redacted [32]. However, existing schemes only allowing redaction of leaves fail to solely redact the data stored in, e.g., n_3. In RERUM we assume that the redacting entity and the signing entity might not be able to agree on a data structure a-priori and that the signing entity does not know what will later be in need of removal. As such, RERUM requires to leave the flexibility to redact non-leaf content, e.g. n_3. Assume we would remove the leave representing the actual temperature of 23.4 (n_2) and the intermediate node that marks n_4 as an hourly average (n_3). This would result in removing the current temperature and just sent the hourly average instead. This is depicted in Figure 45 Assuming that this is also a valid structure of a sensor reading, this still is signed, but will transport different data. Of course this would be interesting for privacy reasons. However, the wanted or unwanted tree depicted in Figure 45 needs a new edge after the removal of intermediate nodes. To connect n_4 to the remaining tree, the third party requires to add a *new* edge $e_{1,4}$, which was not present before. However, $e_{1,4}$ is in the *transitive closure* of the original tree, as shown in Figure 44. For example, the existing scheme introduced in [138] allows redaction of non-leaves, stating that this flexibility is useful in many scenarios. Note, in their scheme non-leaf redaction is modelled as a two step process: first, all children of the to-be-redacted node are re-located to its parent. The to-be-redacted node is now a leaf and can be redacted as such. Allowing non-leaf removal has its merits, but generally allowing this behaviour —without control— can lead to a reduced structural integrity protection, as we describe next.

Hence, protecting structural integrity is equal to protecting that information encoded in the tree hierarchy, e.g. the hourly average marker, can not be removed if not wanted. If one only signs the ancestor relationship of the nodes, all edges that are part of the transitive closure are part of the signature. This is depicted in Figure 44. This allows a third party to add edges to the tree. This possibility was named "Level Promotion" in [204]. This may not *always* be wanted. Thus, RERUM requires it to become controllable.

The scheme introduced in [138] behaves like this: it builds upon the idea that having all pre- and post-order traversal numbers of the nodes in a tree, one can uniquely reconstruct it. To make their scheme hiding occurred redactions, the traversal numbers are randomised in an order-preserving manner, which does not have an impact on the reconstruction algorithm, as the relation between nodes does not change. For our discussion, this step can be left out.[8] Assume we redact n_3, as depicted in Figure 45: the traversal-numbers are still in the correct relation. Hence, the edge $e_{1,4}$, which has not explicitly been present before, passes verification. One might argue that nesting of elements must adhere to a specific codified structure. However, JSON has unlike XML no schemata, and if elements containing the same elements, like hierarchically structured composed data, e.g. Table 16. Hence, redaction of non-leaves is not acceptable in the generic case and may lead to several new attack vectors, similar to the ones of XPath [103]. We conclude that the signing entity must *explicitly* sign only the authorised transitive edges, if the aforementioned behaviour is not wanted, or use an $\mathcal{RSS}$ which only permits leaf-redactions.

4.2.2.1 Solution: Scheme No.2 (described in Section 4.2.6)

Scheme No.2 was devised as a solution to this gap. It addresses this issue and allows controllable relocations and thus secure non-leaf redactions in tree based data structures like nested JSON. Details of this scheme are in Section 4.2.6.

4.2.3 Gap 3: Schemes can be silently updated by the entity with the secret singing key

State-of-the-art security models do not capture the possibility that the signer can "update" signatures, i.e., add new elements. Neglecting this, third parties can generate forgeries. Moreover, there are constructions which permit creating a signature by merging two redacted messages, if they stem from the same original.

4.2.3.1 Solution: Scheme No.3 (described in Section 4.2.7)

Scheme No.3 offers an explicit formal description of the merge process (and the update). This allows splitting and combining data at different steps in the IoT data processing chain by redacting it. Scheme No.3 is described in detail in Section 4.2.7

[8] Indeed, the randomisation step does not hide anything [32, 203].

4.2.4 Gap 4: Accountability for subsequent changes must not require an interaction with the original signer

State-of-the-art security models do allow for a security property called transparency. This is a stronger privacy property and captures the impossibility for a verifier to identify just from a valid signature over a message wether it is original or if it has subsequently been modified in an authorised manner. To still guarantee some form of accountability, e.g. allow the original signer to cryptographically repute a subsequently changed, but still verifying signed data, schemes offer an interactive protocol. As the interaction in step 3 of the sequence diagram in Figure 47 shows, this means invoking a service at the entity (possibly an RD) that initially generated the signature.

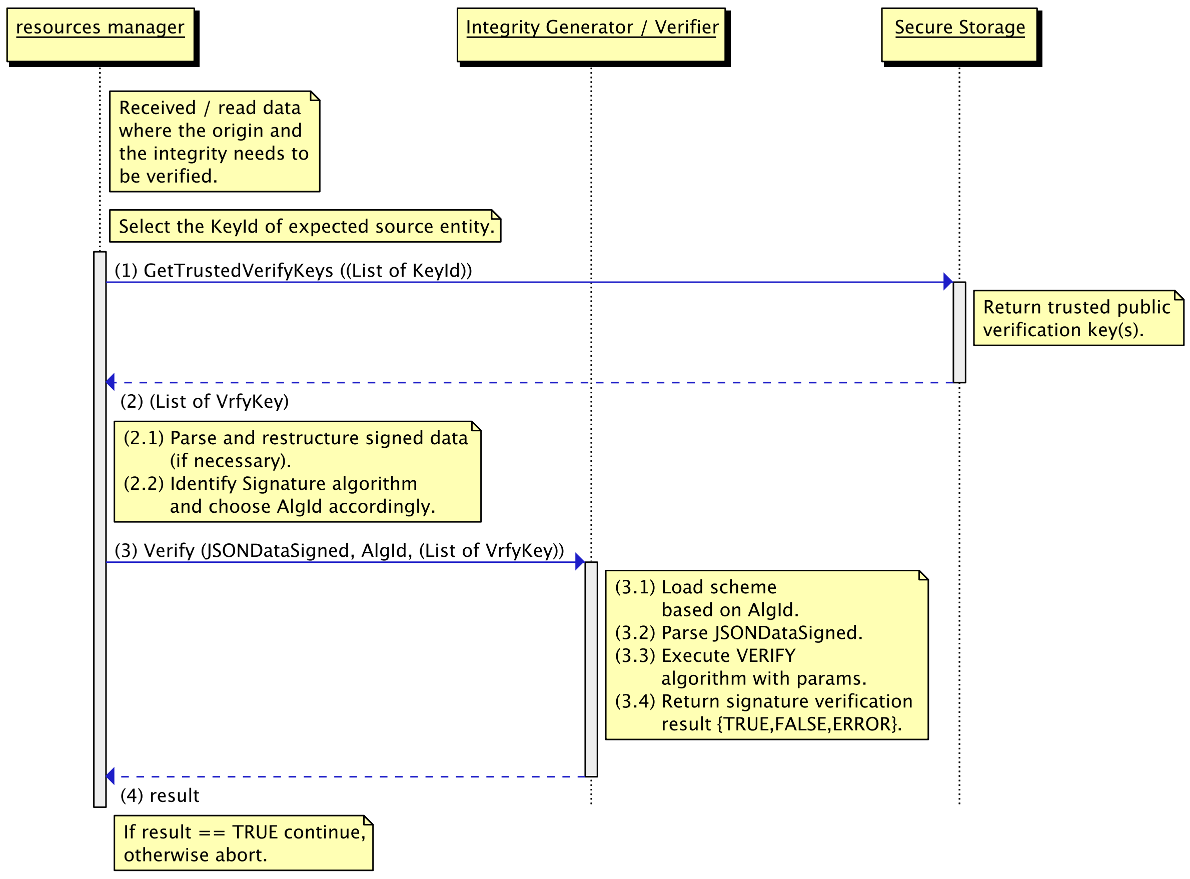

resources manager
Integrity Generator / Verifier
Secure Storage
Received / read data where the origin and the integrity needs to be verified.
Select the KeyId of expected source entity.
(1) GetTrustedVerifyKeys ((List of KeyId))
Return trusted public verification key(s).
(2) (List of VrfyKey)
(2.1) Parse and restructure signed data (if necessary).
(2.2) Identify Signature algorithm and choose AlgId accordingly.
(3) Verify (JSONDataSigned, AlgId, (List of VrfyKey))
(3.1) Load scheme based on AlgId.
(3.2) Parse JSONDataSigned.
(3.3) Execute VERIFY algorithm with params.
(3.4) Return signature verification result {TRUE,FALSE,ERROR}.
(4) result
If result == TRUE continue, otherwise abort.

Figure 47: Interaction when verifying signed data, assuming we want additional information about who is accountable; in the case of a transparent malleable signature schemes this requires additional interaction (step 3).

This is additional overhead, it is cryptographically nice to have transparent schemes, however non-interactivity showed more usefulness in RERUM's use cases.

4.2.4.1 Solution: Scheme No.1 and No.3 (described in 4.2.5, 4.2.7)

Both scheme solves this by offering non-interactive accountability. Scheme No.3 offers an explicit formal description of the merge process (and the update). This allows splitting and combining data at different steps in the IoT data processing chain by redacting it. Scheme No. 1 offers group-level public accountability.

4.2.5 New Scheme No.1 (published in [186])

These results have been published as a paper titled 'Scope of Security Properties of Sanitizable Signatures Revisited' authored by Hermann de Meer, Henrich C. Pöhls, Joachim Posegga and Kai Samelin [186]. We restate all the paper's results and highlight how they are motivated by RERUM and can be facilitated for privacy inline.

Due to transparency, a strong privacy notion, outsiders cannot see if the signature for a message was created by the signer or by the semi-trusted party. Accountability allows the signer to prove to outsiders if a message was original or touched by the semi-trusted party. Currently, block-level accountability requires to drop transparency. We devised a new scheme such that it can allow for accountability for sanitizable signatures with transparency on the block-level. Additionally, we generalise the concept of block-level properties to groups. This offers a even more fine-grained control and leads to more efficient schemes. We prove that group-level definitions imply both the block-level and message-level notions. We derive a provably secure construction, achieving our enhanced notions. A further modification of our construction achieves efficient group-level non-interactive public accountability. This construction only requires a constant amount of signature generations to achieve this property. Finally, we have implemented our constructions and the scheme introduced by *Brzuska* et al. at PKC '09 and provide a detailed performance analysis of our reference implementation in JAVA.

In turn, the notion of *non-interactive public accountability* was introduced in 2012 [35]. A non-interactive publicly accountable $\mathcal{SSS}$ allows that *every* third party is able to decide which party is accountable for a given message-signature pair (m, σ), without requiring any additional information besides what is given from the signature. If the accountable party cannot be derived without the auxiliary information, the scheme is said to be *transparent* [6]. In the same paper, *Brzuska* et al. introduced the paradigm of treating properties on the block-level [35]. In particular, they derive the notion of *block-level* non-interactive public accountability, i.e., a third party can decide which party is accountable for *each block* $m[i]$. They require to sacrifice transparency; our construction keeps this stronger privacy notion. Hence, we achieve block-level interactive accountability and transparency.

For determine the trustworthiness based on the presence or absence of modifications of data it is of paramount importance to know which parts of a given message have been sanitized; even if one part of a message is altered, other parts technically proven to be original may still provide highly trustworthy data. See the example depicted in Figure 41. This shall be done non-interactively, when the existence of the sanitizer has no major impact on the privacy concerns of the involved parties. However, sometimes, the knowledge whether a sanitizer has sanitized a message may lead to problems, e.g., if the existence of the sanitizer must be hidden. Then the scheme must be transparent. However, transparency and non-interactive public accountability are mutually exclusive [35]. Moreover, the current notion of block-level non-interactive public accountability requires the use of linearly many signatures based on the amount of blocks in a message [35]. We therefore generalise the concept of block-level properties to group-level properties, which leads to a reduced complexity in many scenarios. Consider the case of ordering office supplies once more: it may be sufficient to derive the accountability office-wise instead of item-wise. In most scenarios, it still allows for meaningful accountability. Our new generalised definitions contain already existing notions as a border-case. In other words, our work offers generalisation and consolidation of the state-of-the-art and allows to use the ideas of [35] but offers the stronger privacy guarantee of transparency. Hence, we unite both approaches.

4.2.5.1 No.1: Goal is a fine-grained scope of security properties (especially accountability)

The standard security properties of $\mathcal{SSS}$s have first been introduced by *Ateniese* et al. [6]. They have later been formalised and extended by *Brzuska* et al. [33]. Limiting sanitizers to certain values has also been discussed [44, 109, 134, 188]. Later, *Brzuska* et al. introduced the concept of unlinkability, a privacy notion which prohibits a third party from linking two messages [34]. Currently, the notion of unlinkability combined with transparency requires the more costly utilisation of group signatures [34]. We thus focus on the security properties presented in [33]. In particular, unlike *Canard* et al. [45, 46], the signer needs to define which blocks are admissible during the signature generation, while we focus on a setting of a single signer and a single sanitizer, as transparent $\mathcal{SSS}$s for more than one sanitizer currently also require the use of more expensive group signatures [34, 45]. We do note that our ideas remain applicable in unlinkable or multi-sanitizer environments without any adjustments.

Proxy signatures allow for delegating the signing rights entirely, while sanitizable signatures allow to *alter* a specific message. Due to their different goals we do not discuss proxy or redactable signatures in any more depth.

Current block-level accountability notions require at least linearly many signatures, in terms of the number of blocks. We therefore generalise the idea of block-level properties to group-level notions. This allows blocks to be grouped together, which results in a significant theoretical and, as shown in the performance evaluation for JAVA, also practical performance increase: we only require linearly many operations for the number of groups, not blocks. Hence, we close existing gaps and generalise and merge existing ideas. We formalise the notion of group-level accountability for transparent sanitizable signatures and give a provably secure construction based on standard signature schemes and tag-based chameleon hashes [33, 136]. An alteration of our construction allows to achieve group-level non-interactive public accountability equal to [35], with only a constant amount of signature generations. This is required as a main draw-back is that the accountability for many existing RSS required an interaction with the signing device. This would induce a huge overhead and would also not allow to decouple the sanitization/redaction process from the on-device signing process.

4.2.5.2 No.1: Cryptographic preliminaries

We shortly revisit the utilised algorithms, nomenclature and notations for scheme no.1 here, to make this self contained. They are derived from [33], but have been extended to allow for group-level notions. For a message $m = (m[1], \ldots, m[\ell])$, we call $m[i] \in \{0,1\}^*$ a *block*. "," denotes a uniquely reversible concatenation, while $\perp \notin \{0,1\}^*$ denotes a special symbol not being a string, e.g., to indicate an error or an exception.

ADM describes the sanitizable blocks. W.l.o.g. we assume that ADM contains the total number of blocks in m, denoted by ℓ, and a list of the indices of the modifiable blocks. By including ℓ in ADM, we inhibit all attacks that maliciously try to append or remove blocks at the beginning or end.

GRP contains a set of sets which expresses which admissible blocks $m[i]$ are grouped together. In particular, we have GRP $\subseteq 2^{\mathbb{N}}$. We require that the elements of GRP are pairwise disjunct, i.e., $\forall i, j, i \neq j$: $\text{GRP}_i \cap \text{GRP}_j = \emptyset$. Moreover, $|\bigcup_{s_i \in \text{GRP}} s_i| = |\text{ADM}|$ must yield. In other words, every *admissible* block belongs to exactly one group. To clarify this, let GRP $= \{\{1, 5\}, \{3, 4, 6\}\}$. This means, that GRP$[1] = (m[1], m[5])$ and GRP$[2] = (m[3], m[4], m[6])$. For simplicity we also use GRP$[i]$ to denote the uniquely reversible concatenation of each block in GRP$[i]$. We order the set by order of appearance of the ordered blocks. The cardinality of GRP, i.e., the number of groups, is denoted as γ. Hence, in our example, $\gamma = 2$. To simplify the algorithmic description every non-admissible block belongs to the special group GRP$[0]$. Hence, in our prior example we have GRP$[0] = (m[2])$, if $\ell = 6$. Further, we assume that ADM and GRP can *always* be correctly reconstructed from σ, which accounts for the work done in [101].[9]

A secure $\mathcal{SSS}$ consists of the following algorithms:

Definition 3 (Sanitizable Signature Scheme)**.** *A $\mathcal{SSS}$ consists of at least seven PPT algorithms ($KGen_{sig}$, $KGen_{san}$,* Sign, Sanit, Verify, Proof, Judge*):*

KGen$_{sig}$, KGen$_{san}$. *There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys consisting of a private key and the corresponding public key, based on the security parameter λ:*

$$(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) \leftarrow KGen_{sig}(1^\lambda)$$

$$(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow KGen_{san}(1^\lambda)$$

Sign. *: The* Sign *algorithm takes as input the security parameter λ, a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0, 1\}^*$, the secret key* sk$_{sig}$ *of the signer, the public key* pk$_{san}$ *of the sanitizer, as well as ADM and GRP. It outputs the message m and a signature σ (or $\perp$, indicating an error):*

$$(m, \sigma) \leftarrow \mathsf{Sign}(1^\lambda, m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \textit{ADM}, \textit{GRP})$$

Sanit. *Algorithm* Sanit *takes the security parameter λ, a message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0, 1\}^*$, a modification instruction MOD, a signature σ, the public key* pk$_{sig}$ *of the signer and the secret key* sk$_{san}$ *of the sanitizer. It modifies the message m according to the modification instruction MOD. We model MOD to contain a list of pairs $(i, m[i]')$, indicating that block i shall be modified into the string $m[i]'$. Note, MOD can be empty or the string $m[i]'$ can be equal to $m[i]$. This allows the sanitizer to take accountability for a given group without modifying it. For simplicity, we write GRP$[j] \in$ MOD, if at least one block $j \in$ GRP$[i]$ is to be modified.* Sanit *generates a new signature σ' for the modified message $m' =$ MOD(m). Then* Sanit *outputs m' and σ' (or $\perp$ in case of an error):*

$$(m', \sigma') \leftarrow \mathsf{Sanit}(1^\lambda, m, \textit{MOD}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$$

[9]The notation of GRP can be integrated into ADM. However, for historical reasons, we keep them separate and preserve ADM's original meaning.

Verify. *The* Verify *algorithm outputs a decision* $d \in \{\mathtt{true}, \mathtt{false}\}$*, indicating the correctness of a signature* σ *for a message* m *with respect to the public keys* pk_{sig} *and* pk_{san}*.*

$$d \leftarrow \mathsf{Verify}(1^\lambda, m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san})$$

Proof. *The* Proof *algorithm takes as input the security parameter* λ*, the secret signing key* sk_{sig}*, a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a signature* σ *as well a set of (polynomially many) additional message-signature pairs* $\{(m_i, \sigma_i) | i \in \mathbb{N}\}$ *and the public key* pk_{san}*. It outputs a string* $\pi \in \{0,1\}^*$ *(or* $\perp$ *in case of an error):*

$$\pi \leftarrow \mathsf{Proof}(1^\lambda, \mathsf{sk}_{sig}, m, \sigma, \{(m_i, \sigma_i) | i \in \mathbb{N}\}, \mathsf{pk}_{san})$$

Judge. *Algorithm* Judge *takes as input a message* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$ *and a valid signature* σ*, the public keys of the parties and a proof* π*. It outputs a decision* $d \in \{\mathtt{Sig}, \mathtt{San}, \perp\}$*, indicating whether the message-signature pair has been created by the signer or the sanitizer (or* $\perp$ *in case of an error):*

$$d \leftarrow \mathsf{Judge}(1^\lambda, m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi)$$

We require the usual correctness properties to hold. In particular, every genuinely signed or sanitized message verifies as valid. Moreover, every genuinely created proof makes the judge decide in favour of the signer. See [33] for a formal definition.

For the following definitions, we require that a public key must be efficiently derivable from its corresponding secret key.

Ateniese et al. introduced a set of desirable properties [6], later formalised by *Brzuska* et al. [33–35]. We list the informal enumerate of all of them for the paper to be self-contained:

- *Unforgeability* assures that third parties cannot produce a signature for a "fresh" message. Fresh means the message has not been signed by the signer, nor issued by the sanitizer. This is similar to the unforgeability requirements of standard signature schemes [33].
- *Immutability* prevents the sanitizer from modifying blocks not admissible [33].
- *Privacy*, prevents third parties from recovering any original information from sanitized message parts. Its extension *unlinkability* [34] describes the "impossibility to use the signatures to identify sanitized message-signature pairs originating from the same source" [34].
- *Transparency* prevents third parties to decide which party is accountable for a given message-signature pair (m, σ). This is important, if the existence of a sanitizer must be hidden, e.g., if sanitization leads to disadvantages of any party involved [33].
- *Accountability* makes the origin (signer or sanitizer) of a signature undeniable. Hence, it allows a judge to settle disputes over the origin of a signature [33]. The judge may request additional information from the signer. *Brzuska* et al. distinguish between Signer- and Sanitizer-Accountability [33].
- *Non-Interactive Public Accountability* allows that a third party can always decide which party is accountable for a given *message*-signature pair (m, σ) [35].
- *Block-level Non-Interactive Public Accountability* allows that a third party can always decide which party is accountable for a *block*-signature pair $(m[i], \sigma)$ [35].

```
Experiment $\mathsf{Immutability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathsf{sig}}, sk_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}(1^\lambda)$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathsf{sig}}, \cdot, \cdot, \cdot), \mathsf{Proof}(sk_{\mathsf{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\mathsf{san}})$
    let $(m'_i, \sigma'_i)$ for $i = 1, \ldots, q$
    denote the answers from Sign
  return $1$, if:
    $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{\mathsf{sig}}, pk^*) = \mathtt{true}$, and
    $\forall i : pk^* \neq pk_{\mathsf{san},i}$ or
    $m^*[j_i] \neq m_i[j_i]$, where $j_i \notin \mathsf{ADM}_i$
    //shorter messages are padded with $\perp$
```

Figure 48: Immutability

We now give the formal definitions of immutability, privacy, (signer- and sanitizer-) accountability, transparency, and block-level public accountability to increase readability of the upcoming text which introduces new properties and implications. Note, we have already altered the definitions to account for the possibility of grouping blocks.

Definition 4 (Immutability). *A sanitizable signature scheme $\mathcal{SSS}$ is immutable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Immutability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 48 returns 1 is negligible (as a function of λ). To break immutability, the adversary must be able to alter blocks not designated to be sanitized, or to make the signature verify under a new public key [33].*

Definition 5 (Privacy). *A sanitizable signature scheme $\mathcal{SSS}$ is private, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Privacy}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 49 returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ). Here, the adversary has to decide which message was used to produce the desired outcome [33].*

Definition 6 (Signer Accountability). *A sanitizable signature scheme $\mathcal{SSS}$ is* signer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Sig} - \mathsf{Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 50 returns 1 is negligible (as a function of λ). In this game, the adversary has to generate a proof π^* which makes* Judge *to decide that the sanitizer is accountable, if it is not [33].*

Definition 7 (Sanitizer Accountability). *A sanitizable signature scheme $\mathcal{SSS}$ is* sanitizer accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{San} - \mathsf{Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 51 returns 1 is negligible (as a function of λ). In this game, the adversary has to generate a message-signature (m^*, σ^*) which makes* Proof *generate a proof π, leading the* Judge *to decide that the signer is accountable, if it is not [33].*

Definition 8 (Transparency). *A sanitizable signature scheme $\mathcal{SSS}$ is* proof-restricted transparent*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Transparency}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 52 returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ). The basic idea is that the adversary is not able to decide whether it sees a freshly signed signature or a signature created through* Sanitize*. Note, we have already altered the definitions of [33, 35] to account for our new group-level definitions.*

```
**Experiment** $\mathsf{Privacy}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
    $(pk_{\mathsf{sig}}, sk_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}(1^\lambda)$
    $(pk_{\mathsf{san}}, pk_{\mathsf{san}}) \leftarrow \mathsf{KGen}_{\mathsf{san}}(1^\lambda)$
    $b \leftarrow \{0,1\}$
    $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathsf{sig}}, \cdot,\cdot,\cdot), \mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot, sk_{\mathsf{san}})}_{\mathsf{Proof}(sk_{\mathsf{sig}},\cdot,\cdot,\cdot,\cdot), \mathsf{LoRSanit}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot, sk_{\mathsf{sig}}, sk_{\mathsf{san}}, b)}(pk_{\mathsf{sig}}, pk_{\mathsf{san}})$
        where oracle LoRSanit on input of:
        $m_{0,i}, \mathsf{MOD}_{0,i}, m_{1,i}, \mathsf{MOD}_{1,i}, \mathsf{ADM}_i, \mathsf{GRP}_i$
        if $\mathsf{MOD}_{0,i} \not\subseteq \mathsf{ADM}_i$, return $\perp$
        if $\mathsf{MOD}_{1,i} \not\subseteq \mathsf{ADM}_i$, return $\perp$
        if $\mathsf{MOD}_{0,i}(m_{0,i}) \neq \mathsf{MOD}_{1,i}(m_{1,i})$, return $\perp$
        let $(m_i, \sigma_i) \leftarrow \mathsf{Sign}(1^\lambda, m_{b,i}, sk_{\mathsf{sig}}, pk_{\mathsf{san}}, \mathsf{ADM}_i, \mathsf{GRP}_i)$
        return $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(1^\lambda, m_i, \mathsf{MOD}_{b,i}, \sigma, pk_{\mathsf{sig}}, sk_{\mathsf{san}})$
    return $1$, if $a = b$
```

Figure 49: Privacy

```
**Experiment** $\mathsf{Sig} - \mathsf{Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
    $(pk_{\mathsf{san}}, sk_{\mathsf{san}}) \leftarrow \mathsf{KGen}_{\mathsf{san}}(1^\lambda)$
    $b \leftarrow \{0,1\}$
    $(pk^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot, sk_{\mathsf{san}})}(pk_{\mathsf{san}})$
        let $(m'_i, \sigma'_i)$ for $i = 1, \ldots, q$
        denote the answers from the oracle Sanit
    return $1$, if:
        $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}) = \mathtt{true}$, and
        $(pk^*, m^*) \neq (pk_{\mathsf{sig},i}, m'_i)$ for all $i = 1, \ldots, q$, and
        $\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}, \pi^*) = \mathtt{San}$
```

Figure 50: Signer Accountability

4.2.5.3 No.1: New scope for properties is groups of blocks

In this section, we introduce the notion of group-level accountability. We show how a signer can use our new definition to simulate existing notions. Hence, we do not restate existing block-level definitions here, as they are border-cases of our new definitions. We first give the definition of group-level non-interactive public accountability which does not offer transparency and then group-level accountability with transparency[10]. We are the first to give a construction which allows for block-by-block (or group-by-group resp.) accountability, while fully achieving transparency.

[10]Note, as transparency prohibits a third party from deciding who issued the message-signature pair (m, σ), it directly inhibits the instant non-interactive and public form of accountability [35].

```
**Experiment** $\mathsf{San-Accountability}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{sig}, \cdot, \cdot, \cdot), \mathsf{Proof}(sk_{sig}, \cdot, \cdot, \cdot, \cdot)}(pk_{sig})$
    let $(m_i, \mathsf{ADM}_i, pk_{san,i}, \mathsf{GRP}_i)$ and $\sigma_i$ for $i = 1, \ldots, q$
    denote the queries and answers of oracle Sign
  $\pi \leftarrow \mathsf{Proof}(1^\lambda, sk_{sig}, m^*, \sigma^*, \{(m_i, \sigma_i) | 0 < i \leq q\}, pk^*)$
  return $1$, if:
    $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk_{sig}, pk^*) = \mathtt{true}$, and
    $(pk^*, m^*) \neq (pk_{san,i}, m_i)$ for all $i = 1, \ldots, q$, and
    $\mathsf{Judge}(1^\lambda, m^*, \sigma^*, pk_{sig}, pk^*, \pi) = \mathtt{Sig}$
```

Figure 51: Sanitizer Accountability

```
**Experiment** $\mathsf{Transparency}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{sig}, sk_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\lambda)$
  $(pk_{san}, sk_{san}) \leftarrow \mathsf{KGen}_{san}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{sig}, \cdot, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{san}), \mathsf{Proof}(sk_{sig}, \cdot, \cdot, \cdot, \cdot)}_{\mathsf{Sanit/Sign}(\cdot, \cdot, \cdot, \cdot, sk_{sig}, sk_{san}, b)}(pk_{sig}, pk_{san})$
    where Sanit/Sign for input $m_i$, $\mathsf{MOD}_i$, $\mathsf{ADM}_i$, $\mathsf{GRP}_i$
    $\sigma_i \leftarrow \mathsf{Sign}(1^\lambda, m_i, sk_{sig}, pk_{san}, \mathsf{ADM}_i, \mathsf{GRP}_i)$,
    $(m'_i, \sigma'_i) \leftarrow \mathsf{Sanit}(1^\lambda, m_i, \mathsf{MOD}_i, \sigma_i, pk_{sig}, sk_{san})$
    if $b = 1$:
      $\sigma'_i \leftarrow \mathsf{Sign}(1^\lambda, m'_i, sk_{sig}, pk_{san}, \mathsf{ADM}_i, \mathsf{GRP}_i)$,
    finally return $(m'_i, \sigma'_i)$.
  return $1$, if $a = b$ and $\mathcal{A}$ has not
  queried any $m_i$ output by Sanit/Sign to Proof.
```

Figure 52: Transparency

4.2.5.4 No.1: Group-level non-interactive public accountability

To simplify the notion of (block-level) public accountability, *Brzuska* et al. define that the algorithm Judge decides upon reception of an empty proof, i.e., $\pi = \perp$ [35]. In this paper, we keep their approach for consistency. Formally, we require the algorithm Detect. It takes as input the security parameter λ, a message m and a valid signature σ together with the sanitizer's public key pk_{san} and the signer's public key pk_{sig}. Most notably, it also takes as an input a group index i and then returns $\mathtt{San}$ or $\mathtt{Sig}$, indicating which party is accountable for the i^{th} group. This is compareable to *Brzuska* et al.'s definition [35].

Detect is defined as follows: on input of the security parameter λ, a valid message-signature pair (m, σ), the corresponding public keys pk_{sig} and pk_{san}, and the group index i, Detect outputs the accountable party for group i (or $\perp$ in case of an error).

$$d \leftarrow \mathsf{Detect}(1^\lambda, m, \sigma, pk_{sig}, pk_{san}, i), d \in \{\mathtt{San}, \mathtt{Sig}, \perp\}$$

```
**Experiment** Group − Pub − $\mathsf{Acc}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathsf{sig}}, sk_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}(1^\lambda)$
  $(pk_{\mathsf{san}}, sk_{\mathsf{san}}) \leftarrow \mathsf{KGen}_{\mathsf{san}}(1^\lambda)$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathsf{sig}}, \cdot, \cdot, \cdot), \mathsf{Sanit}(\cdot, \cdot, \cdot, \cdot, sk_{\mathsf{san}})}(pk_{\mathsf{san}}, pk_{\mathsf{sig}})$
    Let $(m_i, \mathsf{ADM}_i, pk_{\mathsf{san},i}, \mathsf{GRP}_i)$ and $(m_i, \sigma_i)$ for $i = 1, 2, \ldots, k$
    be the queries and answers to and from oracle Sign.
    Let $(m_j, \mathsf{MOD}_j, \sigma_j, pk_{\mathsf{sig},j})$ and $(m'_j, \sigma'_j)$ for $j = 1, 2, \ldots, k'$
    be the queries and answers to and from oracle Sanit.
    return $1$ if
      Verify$(1^\lambda, m^*, \sigma^*, pk_{\mathsf{sig}}, pk^*) = \mathtt{true}$, and
      $\exists q$ : Detect$(1^\lambda, m^*, \sigma^*, pk_{\mathsf{sig}}, pk^*, q) = \mathtt{Sig}$
      $(\mathsf{GRP}[q]^*, pk^*)$ was never queried to Sign
      as a group of any $m_i$ queried
    return $1$, if
      Verify$(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}) = \mathtt{true}$, and
      $\exists q$ : Detect$(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}, q) = \mathtt{San}$
      $(\mathsf{GRP}[q]^*, pk^*)$ was never queried to Sanit
      as a group of any $\mathsf{MOD}_i$
  return $0$
```

Figure 53: Group-level Non-Interactive Public Accountability

Definition 9 (Non-Interactive Public Accountability). *A sanitizable signature scheme $\mathcal{SSS}$ together with an algorithm Detect is* group-level non-interactive publicly accountable*, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* Group − Pub − $\mathsf{Acc}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 53 returns 1 is negligible (as a function of λ).*

4.2.5.5 No.1: Group-level accountability with transparency

Next, we define accountability with a detail of group-level, while fully preserving transparency. Let us give an informal definition of group-level accountability first:

> A $\mathcal{SSS}$ offers *group-level accountability*, if for all valid message-signature pairs (m, σ) the algorithm Proof outputs a proof π which allows the algorithm GJudge to decide, if the given *group*-signature pair $(\mathsf{GRP}[i], \sigma)$ originates from the signer or from the sanitizer, even in the presence of malicious signers or sanitizers.

As the algorithm Judge only decides the accountability for the complete message/signature pair, we require an additional algorithm able to derive it for each group. The additional algorithm GJudge is defined as follows:

$$d_i \leftarrow \mathsf{GJudge}(1^\lambda, m, \sigma, pk_{\mathsf{sig}}, pk_{\mathsf{san}}, \pi, i)$$

To incorporate the standard accountability notion for the message-level, we define that a sanitizer is accountable for a complete message-signature pair (m, σ), if there exists at least one group i, for which the sanitizer has taken accountability. Vice versa, if there exists no group for which the sanitizer has taken

```
**Experiment** $\mathsf{Group} - \mathsf{Signer} - \mathsf{Acc}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$
  $(pk_{\mathsf{san}}, sk_{\mathsf{san}}) \leftarrow \mathsf{KGen}_{\mathsf{san}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $(pk^*, \pi^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,sk_{\mathsf{san}})}(pk_{\mathsf{san}})$
    Let $(m_j, \mathsf{MOD}_j, \sigma_j, pk_{\mathsf{sig},j})$ and $(m'_j, \sigma'_j)$ for $j = 1, 2, \ldots, k$
    be the queries and answers to and from oracle Sanit.
  return $1$, if:
    $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}) = \mathtt{true}$, and
    $\exists q : \mathsf{GJudge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}, \pi^*, q) = \mathtt{San}$
      $(\mathsf{GRP}[q]^*, pk^*)$ was never queried to Sanit
      as a group of any $\mathsf{MOD}_i$
```

Figure 54: Group-level Signer Accountability

accountability, the signer's accountability follows. This is the expected behavior, as originally defined in [33]. For group-level accountability, we now give new definitions, that include the existing definitions as a border case:

Definition 10 (Group-level Signer Accountability)**.** *A sanitizable signature scheme $\mathcal{SSS}$ is group-level signer accountable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* Group– Signer– $\mathsf{Acc}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 54 returns 1 is negligible (as a function of λ). Basically, to win the game the adversary has to generate a tuple $(\mathsf{pk}^*, m^*, \sigma^*, \pi^*)$, which leads* GJudge *to decide that the sanitizer is accountable for a group $\mathsf{GRP}[q] \in m^*$, while it is not.*

Definition 11 (Group-level Sanitizer Accountability)**.** *A sanitizable signature scheme $\mathcal{SSS}$ is group-level sanitizer accountable, if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* Group-Sanitizer-$\mathsf{Acc}_{\mathcal{A}}^{\mathcal{SSS}}(\lambda)$ *given in Figure 55 returns 1 is negligible (as a function of λ). Basically, to win the game the adversary has to generate a tuple $(\mathsf{pk}^*, m^*, \sigma^*)$ for which* Proof *generates a proof π which leads* Judge *to decide that the signer is accountable for a group $\mathsf{GRP}[q] \in m^*$, while it is not.*

Following the reasoning given in [35], we can express the aforementioned constellation by the following theorems:

Theorem 1. *Every $\mathcal{SSS}$ which is group-level signer accountable, is also signer accountable.*

Theorem 2. *Every $\mathcal{SSS}$ which is group-level sanitizer accountable, is also sanitizer accountable.*

Moreover, we can easily emulate block-level properties, if we generate a new group for each block. This proves, that our notions are a generalisation of existing notions and are stronger. Moreover, message-level properties, as considered by *Brzuska* et al. [33], can easily be achieved by putting all admissible blocks into one single group. Hence, all existing definitions are contained as a border-case of our new ones.

Definition 12 (group-level Accountability)**.** *A sanitizable signature scheme $\mathcal{SSS}$ is group-level accountable, if it is group-level signer accountable and group-level sanitizer accountable.*

```
**Experiment** Group − Sanitizer − $\mathsf{Acc}^{\mathcal{SSS}}_{\mathcal{A}}(\lambda)$
  $(pk_{\mathsf{sig}}, sk_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}(1^\lambda)$
  $b \leftarrow \{0,1\}$
  $(pk^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\mathsf{sig}}, \cdot, \cdot), \mathsf{Proof}(sk_{\mathsf{sig}}, \cdot, \cdot, \cdot, \cdot)}(pk_{\mathsf{sig}})$
    Let $(m_i, \mathsf{MOD}_i, \sigma_i, pk_{\mathsf{san},i}, \mathsf{GRP}_i)$ and $(m_i, \sigma_i)$ for $i = 1, 2, \ldots, k$
    be the queries and answers to and from the oracle Sign.
  $\pi \leftarrow \mathsf{Proof}(1^\lambda, sk_{\mathsf{sig}}, m^*, \sigma^*, \{(m_i, \sigma_i) | 0 < i \leq q\}, pk^*)$
  return $1$, if:
    $\mathsf{Verify}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}) = \mathtt{true}$, and
    $\exists q : \mathsf{GJudge}(1^\lambda, m^*, \sigma^*, pk^*, pk_{\mathsf{san}}, \pi^*, q) = \mathtt{Sig}$ and
      $(\mathsf{GRP}[q]^*, pk^*)$ was never queried to Sign
      as a group of any $m_i$
```

Figure 55: group-level Sanitizer Accountability

4.2.5.6 No.1: Constructions to achieve the new properties

In this section, we derive two new constructions, denoted No.1.1 and No.1.2. The first construction (No.1.1) achieves group-level accountability with transparency. The second construction (No.1.2) allows a more efficient group-level non-interactive public accountability requiring only a constant number of signatures.

4.2.5.7 No.1: Cryptographic prerequisites

All constructions make use of the tag-based chameleon hash by *Brzuska* et al. [33]. In particular, the chameleon hash must be collision-resistant under random tagging-attacks as assumed and shown in [33].

Definition 13 (Chameleon Hash with Tags)**.** *A chameleon hash* $\mathcal{CH} := (\mathsf{CHKeyGen}, \mathsf{CHash}, \mathsf{CHAdapt})$ *with tags consists of three efficient algorithms:*

CHKeyGen*. The algorithm* CHKeyGen *takes as input the security parameter* 1^λ *and outputs the key pair required for the chameleon hash:*

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$$

CHash*. The algorithm* CHash *takes as input the public key* pk*, a string* m *to hash, a tag* TAG *and a randomness* $r \in \{0,1\}^\lambda$*. It outputs the digest* h*:*

$$h \leftarrow \mathsf{CHash}(1^\lambda, \mathsf{pk}, \mathit{TAG}, m, r)$$

CHAdapt*. The algorithm* CHAdapt *takes as input the private key* sk*,* m*,* m'*,* TAG*,* TAG$'$*,* r*. It outputs the new randomness* r'*:*

$$r' \leftarrow \mathsf{CHAdapt}(1^\lambda, \mathsf{sk}, \mathit{TAG}, m, r, \mathit{TAG}', m')$$

```
**Experiment** $\mathsf{Rand} - \mathsf{Tag}_{\mathcal{A}}^{\mathcal{CH}}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$
    $(\mathsf{TAG}, m, r, \mathsf{TAG}', m', r') \leftarrow \mathcal{A}^{\mathsf{OAdapt}(sk,\cdot,\cdot,\cdot,\cdot)}(pk)$
        where oracle OAdapt for the $i^{\text{th}}$ query
        $(\mathsf{TAG}_i, m_i, r_i, m'_i)$ with $\mathsf{TAG}_i \in \{0,1\}^\lambda$
        let $\mathsf{TAG}'_i \leftarrow \{0,1\}^\lambda$ and compute
        $r'_i \leftarrow \mathsf{CHAdapt}(sk, \mathsf{TAG}_i, m_i, r_i, \mathsf{TAG}'_i, m'_i)$
        return $(\mathsf{TAG}'_i, r'_i)$
    return $1$, if
        $(\mathsf{TAG}, m) \neq (\mathsf{TAG}', m')$ and
        let $i = 1, \ldots, q$ denote the $i^{\text{th}}$ oracle query
        $\mathsf{CHash}(pk, \mathsf{TAG}, m, r) = \mathsf{CHash}(pk, \mathsf{TAG}', m', r')$ and
        $\forall i, j : \{(\mathsf{TAG}, m), (\mathsf{TAG}', m')\} \neq \{(\mathsf{TAG}_i, m_i), (\mathsf{TAG}'_i, m'_i)\}$
        $\wedge \{(\mathsf{TAG}, m), (\mathsf{TAG}', m')\} \neq \{(\mathsf{TAG}'_i, m'_i), (\mathsf{TAG}'_j, m'_j)\}$
```

Figure 56: Collision-Resistance against Random Tagging Attacks [33]

As usual, we require all correctness properties to hold. In particular, we require that

$$\mathsf{CHash}(pk, \mathsf{TAG}, m, r) = \mathsf{CHash}(pk, \mathsf{TAG}', m', r')$$

must yield, if r' has been generated genuinely using CHAdapt.

Definition 14 (Collision-Resistance vs. Random-Tag Attacks). *A tag-based chameleon hash $\mathcal{CH}$ is said to be collision-resistant under random-tagging attacks, if the probability that the experiment depicted in Figure 56 returns 1 is negligible (as a function of λ) [33].*

A concrete secure instantiation is found in [33]. Note, the distribution of r' is computationally indistinguishable from uniform [33].

4.2.5.8 No.1.1: Group-level accountable and transparency in Scheme No.1

Next, we introduce a provably secure construction, denoted $\mathcal{SSS}_{No.1.1}$, which is transparent, private, immutable, group-level accountable and unforgeable.

Our construction uses the ideas by [33, 101]. In particular, each group is hashed using a tag-based chameleon hash. However, instead of using one tag for the complete message m, we use different tags for each group $\mathsf{GRP}[i]$. We utilise a standard UNF-CMA signature scheme $\mathcal{SS} = (\mathsf{SKeyGen}, \mathsf{SSign}, \mathsf{SVerify})$ to generate the final signature. We also require a pseudorandom function $\mathcal{PRF}$ mapping n-bit input on a n-bit output for n-bit keys and a pseudorandom generator $\mathcal{PRG}$ mapping n-bit inputs to $2n$-bit outputs.

Definition 15 (Group-level Accountable and Transparent $\mathcal{SSS}_{No.1.1}$). *A $\mathcal{SSS}_{No.1.1}$ consists of the following PPT algorithms $(KGen_{sig}, KGen_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{GJudge}, \mathsf{Judge})$:*

$\mathsf{KGen}_{\mathsf{sig}}$. *Generate a key pair of the underlying signature algorithm* SKeyGen*, i.e.,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$*. Pick a key* $\kappa \leftarrow \{0,1\}^\lambda$ *for the* $\mathcal{PRF}$*. Output* $(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) = (\mathsf{pk}, (\mathsf{sk}, \kappa))$*.*

$\mathsf{KGen}_{\mathsf{san}}$. *Generate a key pair of the underlying chameleon hash. Output* $(\mathsf{pk}_{san}, \mathsf{sk}_{san}) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$. Sign *On input of* $m = (m[1], \ldots, m[\ell])$*,* $m[i] \in \{0,1\}^*$*,* pk_{san}*,* sk_{sig}*, ADM, and GRP,* Sign *draw* $\gamma+1$ *nonces* $n_i \leftarrow \{0,1\}^\lambda$ *and compute:* $x_i \leftarrow \mathcal{PRF}(\kappa, n_i)$ *and* $\mathit{TAG}_i \leftarrow \mathcal{PRG}(x_i)$ *for all* $i = 0, \ldots, \gamma$*. Draw* $\gamma+1$ *additional nonces* $r_i \leftarrow \{0,1\}^\lambda$*. Let:*

$$h[i] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}, \mathit{TAG}_i, (i, \mathit{GRP}[i]), r_i)$$

for all $i = 1, \ldots, \gamma$*. Now, let:*

$$h[0] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}, \mathit{TAG}_0, (\mathit{TAG}_1, \ldots, \mathit{TAG}_\gamma, m), r_0)$$

Set

$$\sigma_c \leftarrow \mathsf{SSign}(\mathsf{sk}, (h[0], \ldots, h[\gamma], \mathit{GRP}[0], \mathsf{pk}_{san}, \mathit{ADM}, \mathit{GRP}))$$

Output (m, σ)*, where*

$$\sigma = (\sigma_c, (\mathit{TAG}_i)_{0 \le i \le \gamma}, (n_i)_{0 \le i \le \gamma}, \mathit{ADM}, \mathit{GRP}, (r_i)_{0 \le i \le \gamma})$$

Verify. *On input of* pk_{sig}*,* pk_{san}*,* m *and*

$$\sigma = (\sigma_c, (\mathit{TAG}_i)_{0 \le i \le \gamma}, (n_i)_{0 \le i \le \gamma}, \mathit{ADM}, \mathit{GRP}, (r_i)_{0 \le i \le \gamma})$$

for each $i \in \mathit{GRP}$ *compute:*

$$h[i] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}, \mathit{TAG}_i, (i, \mathit{GRP}[i]), r_i)$$

and

$$h[0] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}, \mathit{TAG}_0, (\mathit{TAG}_1, \ldots, \mathit{TAG}_\gamma, m), r_0)$$

Output:

$$\mathsf{SVerify}(\mathsf{pk}, (h[0], \ldots, h[\gamma], \mathit{GRP}[0], \mathsf{pk}_{san}, \mathit{ADM}, \mathit{GRP}), \sigma_c)$$

Sanit. *On input of* pk_{sig}*,* sk_{san}*,* m*, MOD and* σ*, first check, if the received message-signature pair is valid using* Verify*. Check, if* $\mathit{MOD} \subseteq \mathit{ADM}$*. If not, stop outputting* $\perp$*. For each* ***group*** $\mathit{GRP}[i] \in \mathit{MOD}$*, draw a nonce* $n'_i \leftarrow \{0,1\}^\lambda$ *and a new tag* $\mathit{TAG}'_i \leftarrow \{0,1\}^{2\lambda}$*. If* $\mathit{GRP}[i] \notin \mathit{MOD}$*, the tags, randoms and nonces are copied from the original signature, i.e.,* $n'_i = n_i$ *and* $\mathit{TAG}'_i = \mathit{TAG}_i$*. If* $\mathit{MOD} \neq \emptyset$*, draw an additional nonce* $n'_0 \leftarrow \{0,1\}^\lambda$ *and an additional tag:* $\mathit{TAG}'_0 \leftarrow \{0,1\}^{2\lambda}$*. Compute:*

$$r'_i \leftarrow \mathsf{CHAdapt}(\mathsf{sk}_{san}, \mathit{TAG}_i, (i, \mathit{GRP}[i]), r_i, \mathit{TAG}'_i, \mathit{GRP}[i]')$$

for each $\mathit{GRP}[i] \in \mathit{MOD}$ *and*

$$\begin{aligned} r'_0 \leftarrow \mathsf{CHAdapt}(&\mathsf{sk}_{san}, \mathit{TAG}_0, (\mathit{TAG}_1, \ldots, \mathit{TAG}_\gamma, m), \\ &r_0, \mathit{TAG}'_0, (\mathit{TAG}'_1, \ldots, \mathit{TAG}'_\gamma, m')) \end{aligned}$$

Output (m', σ')*, where* $m' \leftarrow \mathit{MOD}(m)$ *and*

$$\sigma' = (\sigma_c, (\mathit{TAG}')_{0 \le i \le \gamma}, (n'_i)_{0 \le i \le \gamma}, \mathit{ADM}, \mathit{GRP}, (r'_i)_{0 \le i \le \gamma})$$

Proof. *On input of* sk_{sig}*,* m*,* $\sigma = (\sigma_c, (\textit{TAG}_i)_{0\leq i\leq\gamma}, (n_i)_{0\leq i\leq\gamma}, \textit{ADM}, \textit{GRP}, (r_i)_{0\leq i\leq\gamma})$*,* pk_{san} *and a sequence of message-signature pairs* $\{(m_i, \sigma_i) \mid i \in \mathbb{N}\}$*, search for all groups the matching signatures, s.t.:*

$$\mathsf{CHash}(\mathsf{pk}_{san}, \textit{TAG}_i, (i, \textit{GRP}[i]), r_i) =$$
$$\mathsf{CHash}(\mathsf{pk}_{san}, \textit{TAG}'_i, (i, \textit{GRP}'[i]), r'_i)$$

Do the same for the outer chameleon hash:

$$\mathsf{CHash}(\mathsf{pk}_{san}, \textit{TAG}_0, (\textit{TAG}_1, \ldots, \textit{TAG}_\gamma, m), r_i) =$$
$$\mathsf{CHash}(\mathsf{pk}_{san}, \textit{TAG}'_0, (\textit{TAG}'_1, \ldots, \textit{TAG}'_\gamma, m'), r'_i)$$

Set $\textit{TAG}_i \leftarrow \mathcal{PRG}(x_i)$*, where* $x_i \leftarrow \mathcal{PRF}(\kappa, n_i)$*. Output* π*, where*

$$\pi = ((\textit{TAG}_i)_{0\leq i\leq\gamma}, m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, (r_i)_{0\leq i\leq\gamma}, (x_i)_{0\leq i\leq\gamma})$$

If any errors occur, output $\perp$*. In other words,* Proof *outputs the original blocks as the proof for the complete message.*

GJudge. *On input of* m*,* σ*,* pk_{sig}*,* pk_{san}*, an index* i*, and the proof* π*:*

$$\pi = ((\textit{TAG}^\pi_i)_{0\leq i\leq\gamma}, m^\pi, \mathsf{pk}^\pi_{sig}, \mathsf{pk}^\pi_{san}, (r^\pi_i)_{0\leq i\leq\gamma}, (x^\pi_i)_{0\leq i\leq\gamma})$$

Then check, if σ *verifies. Afterwards, check, if* $\mathsf{pk}^\pi_{san} = \mathsf{pk}_{san}$*. Else, return* $\perp$*. Let*

$$d_i \leftarrow \begin{cases} \mathtt{San} & \text{if the collision is non-trivial and} \\ & \textit{TAG}^\pi_i = \mathcal{PRG}(x^\pi_i) \\ \mathtt{Sig} & \text{else} \end{cases}$$

If $\textit{TAG}^\pi_0 \neq \mathcal{PRG}(x^\pi_0)$ *and there exists no non-trivial collision for the outer chameleon-hash, set* $d_i = \mathtt{Sig}$*. Output* d_i*, or* $\perp$ *on error.*

Judge. *On input of* m*,* σ*,* pk_{sig}*,* pk_{san} *and the proof* $\pi = ((\textit{TAG}^\pi_i)_{0\leq i\leq\gamma}, m^\pi, \mathsf{pk}^\pi_{sig}, \mathsf{pk}^\pi_{san}, (r^\pi_i)_{0\leq i\leq\gamma}, (x^\pi_i)_{0\leq i\leq\gamma})$ *let* $d_i \leftarrow \mathsf{GJudge}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi, i)$ *for each group* $\textit{GRP}[i] \in \textit{GRP}$*. If* $\textit{TAG}^\pi_0 \neq \mathcal{PRG}(x^\pi_0)$ *and there exists no non-trivial collision for the outer chameleon-hash, output* $\mathtt{Sig}$*. On error, output* $\perp$*. If* $\exists i : d_i \neq \mathtt{Sig}$*, then output* $\mathtt{San}$ *and* $\mathtt{Sig}$ *otherwise.*

Theorem 3 (The Construction $\mathcal{SSS}_{No.1.1}$ is Secure and Transparent.)**.** *If the underlying signature scheme* $\mathcal{SS}$ *is unforgeable, the used chameleon hash is collision resistant under random tagging attacks, while* $\mathcal{PRF}$ *and* $\mathcal{PRG}$ *are pseudorandom, our construction is transparent, private, immutable, group-level accountable and unforgeable.*

The proofs are found in 4.2.5.10.

4.2.5.9 No.1.2: Group-level publicly accountable in Scheme No.1

Next, we present a provably secure construction which is private, immutable, group-level non-interactive publicly accountable and unforgeable based on our first construction. This construction alters our first construction such that it removes transparency, but efficiently gives group-level non-interactive public accountability. We achieve this with a constant number of signatures compared to *Brzuska* et al.'s construction [35] where the number of signatures increases linearly with the number of blocks:

Definition 16 (Group-level Publicly Accountable $\mathcal{SSS}_{No.1.2}$). *A $\mathcal{SSS}_{No.1}$ consists of the following PPT algorithms (KGen$_{sig}$, KGen$_{san}$,* Sign, Sanit, Verify, *Detect,* Proof, Judge). *Note,* Proof *just returns $\perp$, so it is here only to fit the $\mathcal{SSS}$ standard list of algorithms.*

KGen$_{\text{sig}}$. *Generate a key pair of the underlying signature algorithm* SKeyGen*, i.e.,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(1^\lambda)$. *Output* $(\mathsf{pk}_{sig}, \mathsf{sk}_{sig}) = (\mathsf{pk}, \mathsf{sk})$.

KGen$_{\text{san}}$. *Generate two key pairs, one for the underlying chameleon hash and one for an unforgeable signature scheme. In particular, let* $(\mathsf{pk}_{san}.\mathsf{pk}_c, \mathsf{sk}_{san}.\mathsf{sk}_c) \leftarrow \mathsf{CHKeyGen}(1^\lambda)$ *and* $(\mathsf{pk}_{san}.\mathsf{pk}_s, \mathsf{sk}_{san}.\mathsf{sk}_s) \leftarrow \mathsf{SKeyGen}(1^\lambda)$.
Output $(\mathsf{pk}_{san}, \mathsf{sk}_{san}) = ((\mathsf{pk}_{san}.\mathsf{pk}_c, \mathsf{pk}_{san}.\mathsf{pk}_s), (\mathsf{sk}_{san}.\mathsf{sk}_c, \mathsf{sk}_{san}.\mathsf{sk}_s))$

Sign. *On input of the message $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$,* pk_{san}, sk_{sig}, *ADM, and GRP, draw γ nonces: $s_i = r_i \leftarrow \{0,1\}^\lambda$ and γ additional tags, i.e., TAG$_i \leftarrow \{0,1\}^{2\lambda}$ Let:*

$$h[i] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}.\mathsf{pk}_c, \mathit{TAG}_i, (i, \mathit{GRP}[i]), r_i)$$

for all $i = 1, \ldots, \gamma$. Generate:

$$\sigma_c \leftarrow \mathsf{SSign}(\mathsf{sk}_s, (h[1], \ldots, h[\gamma], \mathit{GRP}[0], \mathsf{pk}_{san}, \mathit{ADM}, \mathit{GRP}, (r_i)_{0<i\leq\gamma})$$

and

$$\sigma_d \leftarrow \mathsf{SSign}(\mathsf{sk}_s, (h[1], \ldots, h[\gamma], s_1, \ldots, s_\gamma, m))$$

Output:

$$\sigma = (\sigma_c, \sigma_d, (\mathit{TAG}_i)_{0<i\leq\gamma}, (r_i)_{(0<i\leq\gamma)}, (s_i)_{0<i\leq\gamma}, \mathit{ADM}, \mathit{GRP})$$

Verify. *On input of* pk_{sig}, pk_{san}, *m, $\sigma = (\sigma_c, \sigma_d, (\mathit{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \mathit{ADM})$ compute:* $h[i] \leftarrow \mathsf{CHash}(\mathsf{pk}_{san}.\mathsf{pk}_c, \mathit{TAG}_i, (i, \mathit{GRP}[i]), s_i)$ *Check, if σ_d either verifies under* $\mathsf{pk}_{san}.\mathsf{pk}_s$ *or* pk_{sig}. *If σ_d verifies under* pk_{sig}, *also check, if the r_i protected by σ_c and σ_d are equal, i.e., if $r_i = s_i$. If so, output:*

$$\mathsf{SVerify}(\mathsf{pk}, (h[i]_{0<i\leq\gamma}, \mathit{GRP}[0], \mathsf{pk}_{san}, \mathit{ADM}, \mathit{GRP}, (s_i)_{0<i\leq\gamma}), \sigma_c)$$

Sanit. *On input of* pk_{sig}, sk_{san}, *m, MOD and $\sigma = (\sigma_c, \sigma_d, (\mathit{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \mathit{ADM})$ check, if the received message-signature pair is valid using* Verify. *If not, stop and output $\perp$. For each* ***group*** *GRP$[i] \in$ MOD, draw new tags TAG$'_i \leftarrow \{0,1\}^{2\lambda}$. If GRP$[i] \notin$ MOD, set TAG$'_i$ = TAG$_i$ and $s'_i = r_i$. Afterwards, compute:*

$$s'_i \leftarrow \mathsf{CHAdapt}(\mathsf{sk}_{san}.\mathsf{sk}_c, \mathit{TAG}_i, \mathit{GRP}[i], r_i, \mathit{TAG}'_i, \mathit{GRP}[i]')$$

Output (m', σ')*, where* $m' \leftarrow$ MOD(m) *and*

$$\sigma' = (\sigma_c, \sigma'_d, (\textit{TAG}')_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s'_i)_{0<i\leq\gamma}, \textit{ADM}, \textit{GRP})$$

where $\sigma'_d \leftarrow \mathsf{SSign}(\mathsf{sk}_{san}.\mathsf{sk}_s, (h[1], \ldots, h[\gamma], s'_1, \ldots, s'_\gamma, m'))$ *Again, we want to emphasize, that* $r'_i = r_i$*, where* GRP$[i] \in$ MOD*, is only possible with negligible probability, if the* TAG *is changed.*

Proof. *Always return* $\perp$.

Detect. *On input of* $m = (m[1], \ldots, m[\ell])$, $m[i] \in \{0,1\}^*$*, and*

$$\sigma = (\sigma_c, \sigma_d, (\textit{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \textit{ADM}, \textit{GRP}),$$

pk_{sig}, pk_{san}, $\perp$ *and an index* i*, first check, if* σ *verifies. For group* GRP$[i] \in$ GRP *let:*

$$d_i \leftarrow \begin{cases} \mathtt{San} & \text{if } r_i \neq s_i \\ \mathtt{Sig} & \text{else} \end{cases}$$

Output d_i*, or* $\perp$ *on error resp.*

Judge *On input of* m*,*

$$\sigma = (\sigma_c, \sigma_d, (\textit{TAG}_i)_{0<i\leq\gamma}, (r_i)_{0<i\leq\gamma}, (s_i)_{0<i\leq\gamma}, \textit{ADM}),$$

pk_{sig}, pk_{san} *and* $\perp$*, first check, if* σ *verifies.*
For each GRP$[i] \in$ GRP*, call* $d_i \leftarrow \mathsf{GJudge}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \perp, i)$.
On error, output $\perp$*. If* $\exists i : d_i \neq \mathtt{Sig}$*, then output* $\mathtt{San}$ *and* $\mathtt{Sig}$ *otherwise.*

Theorem 4 (The Construction No.1.2 is Secure.). *If the underlying signature scheme* $\mathcal{SS}$ *is unforgeable, the used chameleon hash is collision resistant under random tagging attacks, while* $\mathcal{PRF}$ *and* $\mathcal{PRG}$ *are pseudorandom, our construction is private, immutable, group-level non-interactive publicly accountable and unforgeable.*

The proofs are in Section 4.2.5.10.

4.2.5.10 No.1: Performance measurements for prototypes of No.1.1 and No.1.2 in JAVA

Kai Samelin has implemented scheme No.1 and the construction by *Brzuska* et al. [33] for better comparison in JAVA. The source code used for this evaluation can be made available on request. The tests were performed on a *Fujitsu Celsius* with an *Intel* Q9550 Quad Core @2.83 GHz and 3 GiB of RAM. We only used one core and utilised RSA as the signature algorithm. The moduli have been fixed to 512, $1,024$, $2,048$ and $4,096$-Bit. We evaluated every algorithm with 100, 500 and $1,000$ blocks. We fixed the amount of admissible blocks to 50% and always sanitized all admissible blocks. Moreover, to maintain comparability, each group is exactly one block of the message signed, i.e., $\gamma = |\text{ADM}|$. We omit the key pair generation, as we assume that the key pairs are pre-generated. Proof and Judge are very fast, as they contain only a database lookup and are therefore omitted. The results can be seen in Table 17, and Table 18, Table 19.

As seen, the performance is nearly the same for all three schemes. Hence, our constructions are as useable as the one by *Brzuska* et. al [33].

Table 17: Performance of Scheme No.1.1 with Transparency; Median Runtime in ms

	Signing			Verifying			Sanitizing		
λ \ ℓ	100	500	$1,000$	100	500	$1,000$	100	500	$1,000$
512 Bit	16	63	125	15	46	78	157	766	1,641
$1,024$ Bit	28	112	14,132	20	96	22	1,007	4,948	9,720
$2,048$ Bit	110	391	750	62	328	657	7,109	35,328	70,997
$4,096$ Bit	563	1,546	2,798	250	1,235	2,469	54,719	272,672	545,062

Table 18: Performance of Scheme No.1.2 with Group-level Public Accountability; Median Runtime in ms

	Signing			Verifying			Sanitizing			Detecting		
λ \ ℓ	100	500	$1,000$	100	500	$1,000$	100	500	$1,000$	100	500	$1,000$
512 Bit	16	78	140	15	47	94	172	797	1,578	16	46	94
$1,024$ Bit	47	172	313	31	141	265	1,047	5,062	10,438	32	125	266
$2,048$ Bit	172	516	969	94	437	875	7,547	36,079	72,735	93	421	859
$4,096$ Bit	922	2,157	4,141	328	1,546	3,546	55,453	271,329	562,683	360	1,546	3,109

4.2.5.11 No.1.1: Security of proofs for Scheme No.1.1 with transparency

It is enough to show that the scheme is group-level accountable, transparent and immutable due to the implications given in this work and by *Brzuska* et al. [33, 35]. We prove each property on its own. Most of the proofs are kept short, as they are comparable to the ones given in [33, 35].

Theorem 5 (Construction No.1.1 is Secure (with Transparency).)**.** *If the underlying signature scheme $\mathcal{SS}$ is UNF-CMA, while the used chameleon hash is collision-resistant under random-tagging attacks, our construction is transparent, private, immutable, unforgeable and group-level accountable. We prove each property on its own.*

Construction No.1.1 is immutable. Let $\mathcal{A}$ denote an efficient adversary breaking the immutability of our scheme. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the unforgeability of the underlying signature scheme as follows. We simulate $\mathcal{A}$'s environment by simulating the signing oracle; the signature of the underlying signature scheme (σ_c) is generated by $\mathcal{B}$'s own oracle. Eventually, $\mathcal{A}$ will output a forgery attempt, i.e., a tuple (pk^*, m^*, σ^*). This finishes the simulation. We have to distinguish between three cases: (1) We have $pk_{\mathsf{san}} \neq pk_{\mathsf{san},i}$ for all queries. As pk_{san} has been signed, the underlying signature scheme has been broken. (2) For some j and $i_j \notin \mathsf{ADM}_j$, $m^*_j[j_i] \neq m_j[j_i]$ yields. As m^* has therefore not been queried, the unforgeability of the underlying signature scheme

Table 19: Performance of the Scheme by *Brzuska* et al. [33]; Median Runtime in ms

	Signing			Verifying			Sanitizing		
λ \ ℓ	100	500	$1,000$	100	500	$1,000$	100	500	$1,000$
512 Bit	15	46	93	16	31	78	156	781	1,532
$1,024$ Bit	31	125	219	32	109	203	984	4,875	9,703
$2,048$ Bit	110	391	765	62	328	672	7,109	34,747	70,782
$4,096$ Bit	594	1,547	2,750	250	1,250	2,453	57,390	273,625	537,110

has been broken as well. (3) For some group $\textsf{GRP}_j[i]$, the message has been replaced by a hash or vice versa resp. As this implies $\textsf{GRP}_j[i] \neq \textsf{GRP}^*[i]$, the signature must have been forged, as $\textsf{GRP}$ is signed. If neither case happens, the simulation aborts. The signature forgeries can be extracted in all cases and are then returned by $\mathcal{B}$ as a valid forgery of the underlying signature scheme. Hence, $\mathcal{B}$'s success probability *equals* the one of $\mathcal{A}$. □

Construction No.1.1 is transparent. Transparency follows from the definitions of CHash and CHAdapt, as the distribution of r' and h are computationally indistinguishable from uniform [33]. Moreover, the pseudorandom generators output numbers which are computationally indistinguishable from uniform as well. We do not consider any tag-collisions here, as they only appear with negligible probability. Transparency follows. □

Construction No.1.1 is group-level sanitizer accountable. Please note, in the case where $h[i] \neq h^*[i]$, where h denotes the digest of a group, a direct forgery of the underlying signature scheme is implied. This is also true for $pk_{\mathsf{san},i} \neq pk^*$ and $\textsf{ADM}_i \neq \textsf{ADM}^*$ and $\textsf{GRP}_i \neq \textsf{GRP}^*$. Also note, that in this case the Proof-oracle can trivially be simulated by picking κ itself. Hence, we can focus on the chameleon hash. To be successful, the adversary against group-level signer accountability needs to make sure that the proof algorithm Proof cannot find at least one non-trivial colliding pair of chameleon hash digests. Hence, we have:

$$\mathsf{CHash}(pk_{\mathsf{san}}, \textsf{TAG}_{j,0}, ((\textsf{TAG}_{j,i})_{0\le i\le\gamma}, m), r_{j,0}) =$$
$$\mathsf{CHash}(pk^*, \textsf{TAG}^*_{j,0}, ((\textsf{TAG}_{j,i})^*_{0\le i\le\gamma}, m^*), r^*_0)$$

for some query j. However, this collision is non-trivial and Proof can find it, which prohibits the attack. This also applies to the outer chameleon hash, protecting against match-and-mix attacks. Building an extractor is straight forward and therefore omitted. Sanitizer accountability for groups follows. □

Construction No.1.1 is group-level signer accountable. Let $\mathcal{A}$ denote an efficient adversary breaking the group-level signer accountability of our scheme. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the collision-resistance against random-tagging attacks of the underlying chameleon hash in the follow way. As before, $\mathcal{B}$ simulates $\mathcal{A}$'s environment. However, calls to the sanitization oracle are simulated using $\mathcal{B}$'s $OAdapt$-oracle and signed by its own generated signature key pair. Eventually, $\mathcal{A}$ returns $(pk^*, \pi^*, m^*, \sigma^*)$.

By definition π^* must contain two (non-trivial) colliding tuples:

$$\mathsf{CHash}(pk_{\mathsf{san}}, \mathsf{TAG}_{j,0}, (\mathsf{TAG}_{j,i})_{0\leq i\leq\gamma}, r_{j,i}) = \\ \mathsf{CHash}(pk^*, \mathsf{TAG}^*_i, (\mathsf{TAG}_{j,i})_{0\leq i\leq\gamma})^*, r^*_i)$$

This finishes the simulation. Afterwards, $\mathcal{B}$ outputs the colliding tuples. These tuples break the collision-resistance of the chameleon hash as the tags are drawn at random. Any tag-collision is therefore only possible with negligible probability. Hence, $\mathcal{B}$'s success probability equals the one of $\mathcal{A}$. Please note that this also applies for the outer chameleon hash, protecting against match-and-mix attacks. Building an extractor is straight forward and therefore omitted. Hence, the attack discovered by *Gong* et al. does not apply here, as we add an additional chameleon hash, protecting the whole message, similar to [101]. Signer accountability for groups follows. □

4.2.5.12 No.1.2: Security of proofs for Scheme No.1.2 with public accountability

Theorem 6 (Construction No.1.2 is Secure (with Public Accountability).)**.** *If the underlying signature scheme $\mathcal{SS}$ is UNF-CMA, while the used chameleon hash is collision-resistant under random-tagging attacks, our construction is private, immutable, unforgeable and group-level non-interactive publicly accountable. Following our definitions and [33, 35], it is enough to show that privacy, immutability and group-level non-interactive public accountability hold to prove the security of our scheme.*

Constr. No.1.2 is immutable, private, unforgeability, group-level non-interactive publicly accountable.
The proofs for privacy, immutability and unforgeability are exactly the same as for Scheme No.1.1's construction, with two notable exceptions: We do not achieve transparency, as we sign the original $r[i]$. However, the randomness does not leak any information about the original message, as the tags are drawn at random. Moreover, the "outer" signature protects against mix-and-match attacks. In other words, the sanitizer is only able to draw a new tag, which changes the random coin, but not the message, while the random coins for the chameleon hash are always distributed uniformly, which implies privacy.

Therefore, we only need to show that our scheme is group-level non-interactive publicly accountable. Assume that there is an efficient adversary $\mathcal{A}$ against group-level non-interactive public accountability. We can then construct an adversary $\mathcal{B}$ using $\mathcal{A}$ as a black box to break the unforgeability of the underlying signature scheme as follows: $\mathcal{B}$ forwards any queries to its own oracles and returns the answers to $\mathcal{A}$. $\mathcal{B}$ also flips a coin $b \leftarrow \{0, 1\}$. Eventually, $\mathcal{A}$ returns a tuple (pk^*, m^*, σ^*). If $b = 1$, $\mathcal{B}$ sets $pk_{\mathsf{san}} \leftarrow pk^*$ and $(pk_{\mathsf{sig}}, sk_{\mathsf{sig}}) \leftarrow \mathsf{KGen}_{\mathsf{sig}}$ else, $\mathcal{A}$ sets $pk_{\mathsf{sig}} \leftarrow pk^*$ and $(pk_{\mathsf{san}}, sk_{\mathsf{san}}) \leftarrow \mathsf{KGen}_{\mathsf{san}}$.

Consequently, we have to distinguish between two cases, i.e., a malicious sanitizer and a malicious signer. The probability that the simulation is done for the correct case is exactly $\frac{1}{2}$. We will omit cases where the random coins are equal, as this only occurs with negligible probability.

4.2.5.13 Malicious signer

As $r'_i \neq r_i$, the underlying signature scheme must been forged, as σ_d protects all r_i, as $r'_i = r_i$ occurs only with negligible probability.

4.2.5.14 Malicious sanitizer

We know that $r'_i = r_i$ only occurs with negligible probability. Therefore, σ_d must be a valid forgery.

In both cases, an extractor can trivially be build. □

4.2.5.15 No.1: Main achievement: Group-level non-interactive or interactive accountability

Scheme No.1 brings the new notion of group-level properties for sanitizable signatures. We have formalised the notions of group-level accountability, both in an offline and an online variant. The offline variant allows to achieve transparency which positively answers an open research question posed by *Brzuska* et al. [35]. This broader scope of groups of blocks in the definitions includes all existing notions of accountability on block- [35] or message-level [33]. Hence, this is a real generalisation, which closes current gaps. We have derived two novel yet provably secure constructions, achieving our new notions. Both constructions show how the group-level definitions allow to choose between performance and accountability: the signer can control the granularity of accountability. The performance analysis for JAVA based implementations highlights that the scheme by *Brzuska* et al. [33] and our two new schemes are reasonable performant, even the construction that also achieve the stronger security notion of transparency. The sanitization algorithm is are expected to be run at the

4.2.6 New scheme No.2 (published in [67])

These results have been published as a paper titled 'Redactable Signature Schemes for Trees With Signer-Controlled Non-Leaf-Redactions' authored by Hermann de Meer, Henrich C. Pöhls, Joachim Posegga and Kai Samelin [67]. We restate all the paper's results and highlight how they are motivated by RERUM and can be facilitated for privacy inline.

Exiting redactable signature schemes ($\mathcal{RSS}$) allow to remove parts from signed documents, while the resulting signature is valid. Some existing $\mathcal{RSS}$s for trees allow to redact non-leaves. Then, new edges have to be added to the tree to preserve its structure. This alters the position of the nodes' children, and may alter the semantic meaning encoded into the structure.

As JSON data, if nested is a tree and because trees are a commonly used to structure data. The integrity protection with intended malleability needs to protect these data structures against the above described and other unauthorized modifications.

4.2.6.1 No.2: Goal is to allow non-leaf redactions in tree-based data structures (e.g. JSON)

The concept of $\mathcal{RSS}$s has been introduced in [127, 218]. Following these ideas, $\mathcal{RSS}$s have been proposed to work for lists [51, 204], and have extended for trees [32, 138] and graphs [138]. *Brzuska* et al. derived a set of desired properties for redactable tree-structured documents including a formal model for security notions [32]. Following their definitions, most of the schemes proposed are not secure, e.g., the work done in [109, 127, 138, 163, 218, 235]. In particular, a third party can derive that something has been redacted, which impacts on the intention of an $\mathcal{RSS}$. However, Brzuska et al.'s model is limited to leaf-redaction only.

Recently, schemes with *context-hiding*, a very strong privacy notion, and variations thereof, e.g., [2, 7, 8] appeared. In those schemes, a derived signature does not leak whether it corresponds to an already existing signature in a statistical sense. Most recent advances generalize similar ideas, e.g., [2, 7, 8, 26, 30].

Secure, i.e. completely controllable in terms of intermediate node relocation, non-leaf redactions have not been further studied.

Scheme No.2 must be secure in a security model where the signer has the flexibility to allow redaction of any node. The new model allows granting explicitly level promotions —via granting re-locations of specified sub-trees— which resembles the *implicit* possibility of previous redactable schemes. The signer is *explicitly* prohibiting the redaction of nodes individually, as the signer must explicitly sign an edge for re-locations. Re-locations of sub-trees can be used to emulate non-leaf redactions, but allow even more flexibility: we can relocate sub-trees without redactions. We also allow that a sanitizer can prohibit such re-locations by redacting the authorized potential edge.

While [203] either allows or disallows non-leaf redactions completely, scheme No.2 allows the signer to decide which non-leaves can be redacted: the signer defines to which "upper-level" node the children can be connected to.

We derive a provably secure construction, based on cryptographic accumulators [14, 22], in combination with *Merkle*'s Hash-Tree-Technique. Thus, our construction requires only standard cryptographic primitives. However, we need to strengthen existing definitions of accumulators. In particular, we introduce the notions of indistinguishability and strong one-wayness of accumulators.

In our construction, the signer controls the protection of the order of siblings. Hence, our scheme is capable of signing both ordered and unordered trees.

4.2.6.2 No.2: Cryptographic preliminaries

Nodes are denoted as n_i. The root is denoted as n_1. With c_i, we refer to all the content of node n_i, which is additional information that might be associated with a node, i.e., data, element name and so forth. We use the work done in [32] as our starting point. Their model only allows removing *a single leaf* at a time and does not support non-leaf redactions.

Definition 17 (Redactable Signature Scheme). *An $\mathcal{RSS}$ consists of four efficient (PPT) algorithms: $\mathcal{RSS} := (\textit{KeyGen}, \textit{Sign}, \textit{Verify}, \textit{Modify})$. All algorithms output $\perp$ in case of an error. Also, they take an implicit security parameter λ (in unary).*

KeyGen. *The algorithm KeyGen outputs the key pair of the signer, i.e., $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textit{KeyGen}(1^\lambda)$, λ being the security parameter.*

Sign. *On input of* sk*, T, and* ADM*, Sign outputs a signature σ.* ADM *controls what changes by Modify are admissible. In detail,* ADM *is the set containing all signed edges, including the ones where a sub-tree can be re-located to. In particular, $(n_i, n_j) \in$* ADM*, if the edge (n_i, n_j) must verify. These edges cannot be derived from T alone. Let $\sigma \leftarrow \textit{Sign}(\mathsf{sk}, T, \textsc{adm})$.*

Verify. *On input of* pk*, the tree T and a signature σ, Verify outputs a bit $d \in \{0, 1\}$, indicating the validity of σ, w.r.t.* pk *and T: $d \leftarrow \textit{Verify}(\mathsf{pk}, T, \sigma)$. Note,* ADM *is not required.*

```
**Experiment** Unforgeability$_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $(T^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot,\cdot)}(pk)$
        let $i = 1, 2, \ldots, q$ index the queries/answers to/from Sign
    return $1$, if
        $\mathsf{Verify}(pk, T^*, \sigma^*) = 1$ and
        for all $1 \leq i \leq q$, $T^* \notin \mathsf{span}_{\vdash}(T_i, \sigma_i, \mathsf{ADM}_i)$
```

Figure 57: Unforgeability

Modify*. The algorithm Modify takes* pk*, the tree* T*, a signature* σ*,* ADM*, and an instruction* MOD*.* MOD *contains the actual change to be made: redact a sub-tree, relocate a sub-tree, or prohibit relocating a sub-tree. On modification,* ADM *is adjusted. If a node* n_i *is redacted, the edge to its father needs to be removed. Moreover, if there exists a sub-tree which can be re-located under the redacted node, the corresponding edges need to be removed from* ADM *as well. The alteration of* ADM *is crucial to maintain privacy and transparency. Hence, we have:* $(T', \sigma', \mathsf{ADM}') \leftarrow$ *Modify*$(\mathsf{pk}, T, \sigma, \mathsf{ADM}, \mathsf{MOD})$.

We require the usual correctness requirements to hold [32]. A word of clarification: we assume that ADM *is always correctly derivable from* σ*. However, we always explicitly denote* ADM *to increase readability of our security definitions.*

4.2.6.3 No.2: Extend the security model to fine-grained scope

We build around the framework given in [32], extending it to cater for the flexibility of non-leaf redactions and re-locations.

Definition 18 (Unforgeability). *No one should be able to compute a valid signature on a tree* T^* *verifying for* pk *outside span*$_{\vdash}(T, \sigma, \mathsf{ADM})$*, without access to the corresponding secret key* sk*. Here, span*$_{\vdash}(T, \sigma, \mathsf{ADM})$ *expresses the set of trees derivable by use of Modify on* T*,* σ *and* ADM*. This is analogous to the standard unforgeability requirement for signature schemes [98]. A scheme* $\mathcal{RSS}$ *is unforgeable, if for any PPT adversary* $\mathcal{A}$*, the probability that the game depicted in Figure 57 returns* 1*, is negligible.*

Definition 19 (Privacy:). *No one should be able to gain any knowledge about parts redacted. This is similar to the standard indistinguishability notation for encryption schemes [97]. An* $\mathcal{RSS}$ *is private, if for any PPT adversary* $\mathcal{A}$*, the probability that the game shown in Figure 58 returns* 1*, is negligibly close to* $\frac{1}{2}$*. In a nutshell, privacy says that everything which has been redacted remains hidden. However, if in real documents redactions are obvious, e.g., due to missing structure, one may trivially be able to decide that not the complete tree was given to the verifier. However, this cannot be avoided: our definitions assume that no other sources of knowledge apart from (several)* σ_i'*,* T_i' *and* ADM_i' *are available to the attacker.*

```
**Experiment** $\mathsf{Privacy}_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $d \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot,\cdot),\mathsf{LoRModify}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,sk,b)}(pk)$
    where oracle $\mathsf{LoRModify}(T_{j,0}, \mathsf{ADM}_{j,0}, \mathsf{MOD}_{j,0}, T_{j,1}, \mathsf{ADM}_{j,1}, \mathsf{MOD}_{j,1}, sk, b)$
        if $\mathsf{MOD}_{j,0}(T_{j,0}) \neq \mathsf{MOD}_{j,1}(T_{j,1})$ return $\perp$
        $(T_{j,0}, \sigma_0, \mathsf{ADM}_{j,0}) \leftarrow \mathsf{Sign}(sk, T_{j,0}, \mathsf{ADM}_{j,0})$
        $(T_{j,1}, \sigma_1, \mathsf{ADM}_{j,1}) \leftarrow \mathsf{Sign}(sk, T_{j,1}, \mathsf{ADM}_{j,1})$
        $(T'_{j,0}, \sigma'_0, \mathsf{ADM}'_{j,0}) \leftarrow \mathsf{Modify}(pk, T_{j,0}, \sigma_0, \mathsf{ADM}_{j,0}, \mathsf{MOD}_{j,0})$
        $(T'_{j,1}, \sigma'_1, \mathsf{ADM}'_{j,1}) \leftarrow \mathsf{Modify}(pk, T_{j,1}, \sigma_1, \mathsf{ADM}_{j,1}, \mathsf{MOD}_{j,1})$
        if $\mathsf{ADM}'_{j,0} \neq \mathsf{ADM}'_{j,1}$, abort returning $\perp$
        return $(T'_{j,b}, \sigma'_b, \mathsf{ADM}'_{j,b})$
    return $1$, if $b = d$
```

Figure 58: Privacy

```
**Experiment** $\mathsf{Transparency}_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $d \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot,\cdot),\mathsf{ModifyOrSign}(\cdot,\cdot,\cdot,sk,b)}(pk)$
    where oracle $\mathsf{ModifyOrSign}(T, \mathsf{ADM}, \mathsf{MOD}, sk, b)$
        if $\mathsf{MOD} \notin \mathsf{ADM}$, return $\perp$
        $(T, \sigma, \mathsf{ADM}) \leftarrow \mathsf{Sign}(sk, T, \mathsf{ADM})$
        $(T', \sigma', \mathsf{ADM}') \leftarrow \mathsf{Modify}(pk, T, \sigma, \mathsf{ADM}, \mathsf{MOD})$
        if $b = 1$:
            $(T', \sigma', \mathsf{ADM}') \leftarrow \mathsf{Sign}(sk, T', \mathsf{ADM}')$
        return $(T', \sigma', \mathsf{ADM}')$
    return $1$, if $b = d$
```

Figure 59: Transparency

Definition 20 (Transparency:). *A party who receives a signed tree T should not be able to tell whether it received a freshly signed tree (case $b = 1$ in Figure 59) or a tree derived by Modify [32]. We say that an $\mathcal{RSS}$ is transparent, if for any PPT adversary $\mathcal{A}$, the probability that the game shown in Figure 59 returns 1, is negligibly close to $\frac{1}{2}$.*

4.2.6.4 No.2: Relations between security properties

The implications and separations between the security properties given in [32] do not change — the proofs are very similar and therefore omitted in this work. In particular, transparency implies privacy, while transparency and unforgeability are independent.

4.2.6.5 No.2: Background on cryptographic accumulators

For our construction, we deploy accumulators. They have been introduced in [22]. The basic idea is to hash a set $\mathcal{S}$ into a short value a, normally referred to as the accumulator. For each element $y_i \in \mathcal{S}$ a short witness w_i is generated, which allows to verify that y_i has actually been accumulated into a. We only need the basic operations of an accumulator, e.g., neither trapdoor-freeness [147, 205] nor dynamic updates [43], or revocation techniques [37] are required. A basic accumulator consists of four efficient algorithms, i.e., $\mathcal{AH} := \{\mathsf{KeyGen}, \mathsf{Hash}, \mathsf{Proof}, \mathsf{Check}\}$:

KeyGen. Outputs the public key *pk* on input of a security parameter λ:
$pk \leftarrow \mathsf{KeyGen}(1^\lambda)$

Hash. Outputs the accumulator a, and an auxiliary value aux, given a set $\mathcal{S}$, and *pk*:
$(a, \mathsf{aux}) \leftarrow \mathsf{Hash}(pk, \mathcal{S})$

Proof. On input of an auxiliary value aux, the accumulator a, a set $\mathcal{S}$, and an element $y \in \mathcal{S}$, Proof outputs a witness w, if y was actually accumulated:
$w \leftarrow \mathsf{Proof}(pk, \mathsf{aux}, a, y, \mathcal{S})$

Check. Outputs a bit $d \in \{0, 1\}$, indicating if a given value y was accumulated into the accumulator a with respect to *pk* and a witness w:
$d \leftarrow \mathsf{Check}(pk, y, w, a)$

All correctness properties must hold [14]. Next, we define the required security properties of accumulators.

Definition 21 (Strong One-Wayness of Accumulators.). *It must be hard to find an element not accumulated, even if the adversary can chose the set to be accumulated. The needed property is strong one-wayness of the accumulator [14]. We say that an accumulator is strongly one-way, if the probability that the game depicted in Figure 60 returns 1, is negligibly close to 0. Note, in comparison to [14, 171], we consider probabilistic accumulation and allow to query adaptively.*

```
Experiment $\mathsf{Strong-One-Wayness}_{\mathcal{A}}^{\mathcal{AH}}(\lambda)$
    $pk \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $(a^*, y^*, p^*) \leftarrow \mathcal{A}^{\mathsf{Hash}(pk,\cdot)}(1^\lambda, pk)$
        where oracle Hash for input $\mathcal{S}_i$:
            $(a_i, \mathsf{aux}_i) \leftarrow \mathsf{Hash}(pk, \mathcal{S}_i)$
            return $(a_i, \{(y_j, p_j) \mid y_j \in \mathcal{S}_i, p_j \leftarrow \mathsf{Proof}(pk, \mathsf{aux}_i, a_i, y_j, \mathcal{S}_i)\})$
    look for $k$ s.t. $a_k = a^*$. If such $k$ does not exist, return $0$.
    return $1$, if $\mathsf{Check}(1^\lambda, pk, y^*, p^*, a^*)$ and $y^* \notin \mathcal{S}_k$
```

Figure 60: Accumulator Strong One-Wayness

Definition 22 (Indistinguishability of Accumulators.). *We require that an adversary cannot decide how many additional members have been digested. We say that an accumulator is indistinguishable, if the probability that the game depicted in Figure 61 returns 1, is negligibly close to $\frac{1}{2}$. Here, the adversary can* *choose three sets, and has to decide which sets have been accumulated (either the first and the second, or the first and the third). Note, only the witnesses for the first set are returned.*

```
**Experiment** Indistinguishability$_{\mathcal{A}}^{\mathcal{AH}}(\lambda)$
    $pk \leftarrow \mathsf{KeyGen}(1^\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    $d \leftarrow \mathcal{A}^{\mathsf{LoRHash}(\cdot,\cdot,\cdot,b,pk)}(1^\lambda, pk)$
        where oracle LoRHash for input $\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1$:
            $(a, \mathsf{aux}) \leftarrow \mathsf{Hash}(pk, \mathcal{S} \cup \mathcal{R}_b)$
            return $(a, \{(y_i, p_i) \mid y_i \in \mathcal{S}, p_i \leftarrow \mathsf{Proof}(pk, y_i, \mathsf{aux})\})$
    return $1$, if $d = b$
```

Figure 61: Accumulator Privacy

An accumulator not fulfilling these requirements has been proposed by *Nyberg* in [171]; the underlying *Bloom*-Filter can be attacked by probabilistic methods and therefore leaks the amount of members [65]. This is not acceptable for our construction, as it impacts on privacy. A concrete instantiation of such an accumulator achieving our requirements is the probabilistic version of [14]. In a nutshell, instead of fixing the base for the RSA-function, it is drawn at random. A more detailed discussion is given in [65]. We do note that our definition of indistinguishability already assumes a probabilistic accumulator; [65] also accounts for deterministic ones. Additional information about accumulators can be found in [14, 22, 43].

4.2.6.6 No.2.: Background hash-trees and privacy

Removing sub-trees requires to give a hash of the removed node to the verifier, in order to calculate the same $\mathcal{MH}(n_1)$. This directly impacts on privacy and transparency, because the hash depends on removed information that shall remain private. One example for an $\mathcal{RSS}$ which suffers from this problem is given in [114]. It can be attacked in the following way: the attacker asks its left-or-right oracle to sign a root with one child only, but without redacting anything. The other input is a tree with the root and two children, while the *left* child is to redacted. This results in the same tree: the root with one child. However, in the case the first input is used, their "fake-digest" is the right node, while in the other case the fake-digest is the left node. This can clearly be distinguished and privacy is broken.

A more detailed analysis of the *Merkle*-Hash-Tree is given in [138], which also gives an introduction on the possible attacks on non-private schemes. To overcome the limitation of *Merkle*-Hash-Trees, we use accumulators instead of standard collision-resistant hash-functions. We do note that the idea to use accumulators has already been proposed in [138]. However, they state that accumulators are not able to achieve the desired functionality. We show that they are sufficient by giving a concrete construction. The number of signatures to be generated is one.

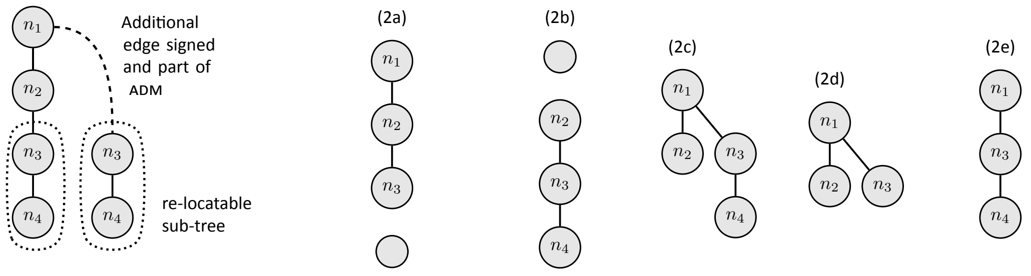


Figure 62: Expanded tree with duplicates and examples of valid trees after redactions or re-locations (2a-e).

4.2.6.7 No.2: Construction of scheme No.2

Scheme No.2's construction makes use of *Merkle*-Hash-Trees. The *Merkle*-Hash $\mathcal{MH}$ of a node x is calculated as: $\mathcal{MH}(x) := \mathsf{Hash}(\mathsf{Hash}(c_x)||\mathcal{MH}(x_1)||\ldots||\mathcal{MH}(x_n))$, where Hash is a collision-resistant hash-function, c_x the content of the node x, x_i a child of x, n the number of children of the node x, while $||$ denotes a uniquely reversible concatenation of strings. $\mathcal{MH}(n_1)$'s output depends on all nodes' content and on the *right order* of the siblings. Hence, signing $\mathcal{MH}(n_1)$ protects the integrity of the nodes in an ordered tree and the tree's structural integrity. Obviously, this technique does not allow to hash unordered trees: an altered order most likely causes a different digest value.

We allow explicit re-location of sub-trees. If a non-leaf is subject to redaction, all sub-trees of the node need to be re-located. If this is possible and what their new ancestor will be must be under the sole control of the signer. We limit re-locations directing towards the root to avoid forming loops, which was possible in the original publication [180]. We now sketch our solution, and give the concrete algorithms afterward. Our re-location definition does not require to delete the ancestor node. This behaviour of re-locating only is discussed later on.

4.2.6.8 No.2: Sketch of the construction

In our solution, the signer replicates all re-locatable nodes and the underlying sub-trees to all locations where a sanitizer is allowed to relocate the sub-tree to. The replicas of the nodes are implicitly used to produce the re-locatable edges. Each additional edge is contained in ADM. To prohibit simple copy attacks, i.e., leaving a re-located sub-tree in two locations, each node n_i gets an associated unique nonce r_i. The whole tree gets signed using a *Merkle*-Hash-Tree, but using an accumulator instead of a standard hash. To redact parts, the sanitizer removes the nodes in question, and no longer provides the corresponding witnesses. As accumulators work on sets, it does not matter in what order the members are checked: if ordered trees are present, the ordering between siblings has to be explicitly signed. To do so, we sign the "left-of" relation, as already used and proposed in [32, 51, 204]. Note, this implies a quadratic complexity in the number n of siblings, i.e, $n + \frac{n(n-1)}{2}$. To relocate a sub-tree, one only applies the necessary changes to T, without any further changes. Moreover, a sanitizer can prohibit consecutive re-locations by altering ADM. This control is similar to consecutive sanitization control [163]. Verification is straight forward: for each node x inside the tree check, if x's content, x's children and x's order to other siblings is contained in x's *Merkle*-Hash. This is done recursively. Further, all node's nonces must be unique for this tree. Finally, the root's signature is checked.

4.2.6.9 No.2.: Algorithmic description of Scheme No.2

$\Pi :=$ (KeyGen, Sign, Verify) denotes a standard unforgeable signature scheme [98]. Note, to shorten the *algorithmic* description, we abuse notation and define that Hash directly works on a set and returns all witness/element pairs (w_i, y_i). We denote the accumulation as $(a, \mathcal{W} = \{(w_i, y_i)\}) \leftarrow \mathcal{AH}(\mathit{pk}, \{y_1, \ldots, y_n\})$. We use *//comment* to indicate comments.

```
KeyGen($\lambda$).
    $\mathit{pk}_{\mathcal{AH}} \leftarrow \mathcal{AH}.\mathsf{KeyGen}(1^\lambda)$
    $(\mathit{pk}_S, \mathit{sk}_S) \leftarrow \Pi.\mathsf{KeyGen}(1^\lambda)$
    return $((\mathit{pk}_S, \mathit{pk}_{\mathcal{AH}}), \mathit{sk}_S)$


Expand($T$, ADM).
    For all edges $e_i \in \mathsf{ADM} \setminus T$ (must be done bottom-up)
        Replicate the sub-tree underneath the node addressed by $e_i$
        to the designated position. //Note: this is recursive!
    Return this expanded tree


Sign($\mathit{sk}, T, \mathsf{ADM}$).
    //We implicitly assume a parameter $s \in \{\text{ordered}, \text{unordered}\}$,
    //denoting if the order must be protected
    For each node $n_i \in T$:
        $r_i \xleftarrow{\$} \{0,1\}^\lambda$
        Append $r_i$ to each node $n_i \in T$
    Expand tree: $\Omega \leftarrow \mathsf{Expand}(T, \mathsf{ADM})$ //Note: $r_i$ is copied as well
    Do the next step with the expended tree $\Omega$:
        If $s =$ unordered: //$\mathcal{MH}(\cdot)$ denotes the digest calculated by $\mathcal{AH}$
            $(d_1, \{(y_k, w_k)\}) \leftarrow \mathcal{AH}(\mathit{pk}, \{c_1||r_1, \mathcal{MH}(x_1), \ldots, \mathcal{MH}(x_n)\})$
        Else ($s =$ ordered): //ordered tree
            $(d_1, \{(y_k, w_k)\}) \leftarrow \mathcal{AH}(\mathit{pk}, \{c_1||r_1, \mathcal{MH}(x_1), \ldots, \mathcal{MH}(x_n), \Xi_x\})$,
            where $\Xi_x = \{r_i||r_j \mid 0 < i < j \leq n\}$
    Sign the root-hash: $\sigma_s \leftarrow \Pi.\mathsf{Sign}(\mathit{sk}_S, d_1||s)$
    $\mathcal{W} = \{(y_k, w_k)\}$ denotes the set of all witness/element pairs returned
    return $\sigma = (\sigma_s, \mathcal{W}, \mathsf{ADM})$


Modify($\mathit{pk}, T, \sigma, \mathsf{ADM}, \mathsf{MOD}$).
    use Verify to verify the tree $T$
    Expanded tree $\Omega \leftarrow \mathsf{Expand}(T, \mathsf{ADM})$
    Case 1: MOD instruction to redact sub-tree $T_s$ (only via leaf-redaction):
        //1. remove all $n_l \in T_s$ (incl. replicas) from $\Omega$:
        Set $\Omega' \leftarrow \Omega \setminus n_l$
        //2. remove all $n_l \in T_s$ from $T$:
        Set $T' \leftarrow T \setminus T_s$
        Create ADM$'$ by removing all ingoing edges all nodes in $T_s$ from ADM
        return $\sigma' = (T', \sigma_s, \mathcal{W} \setminus \{(y_k, w_k) \mid y_k \in \Omega'\}, \textsf{ADM}')$
    Case 2: MOD instruction to re-locate $T_s$:
        Set $T' \leftarrow \textsf{MOD}(T)$
        return $\sigma$
    Case 3: MOD instruction to remove re-location edges $e$:
        Set $\textsf{ADM}' \leftarrow \textsf{ADM} \setminus e$
        //Note: This expansion is done with the modified ADM'.
        Let $\Omega' \leftarrow \text{Expand}(T, \textsf{ADM}')$
        return $\sigma' = (T, \sigma_s, \mathcal{W} \cap \{(y_k, w_k) \mid y_k \in \Omega'\}, \textsf{ADM}')$


Verify$(pk, T, \sigma)$.
    Check if each $r_i \in T$ is unique.
    Check $\sigma$ using $\Pi$.Verify
    Let the value protected by $\sigma_s$ be $d_1' = d_1||s$
    For each node $x \in T$:
        For all children $x_i$ of $x$ do:
            //Note: checks if children are signed
            Let $d \leftarrow$ Check$(pk, d_i, w_i, d_x)$ //$d_x$ denotes the node's digest
            If $d = 0$, return $0$
            If $s =$ ordered:
                //Is every "left-of"-relation signed?
                //Note: only linearly many checks
                For all $0 < i < n$:
                    $d \leftarrow$ Check$(pk, r_i||r_{i+1}, w_{x,x+1}, d_x)$
                    If $d = 0$, return 0
    return 1
```

Arguably, allowing re-location without redaction may also be too much freedom. However, it allows the signer to allow a flattening of hierarchies, i.e., to remove the hierarchical ordering of treatments in a patient's record. We want to stress that copying complete sub-trees may lead to an exponential blow-up in the number of nodes to the signed. This happens, in particular, if re-locations are nested. However, if only used sparely, our construction remains useable, as a performance analysis shows next.

4.2.6.10 No.2: Performance of prototypes in JAVA of Scheme No.2

The implemented Scheme No.2 demonstrates its usability using the old algorithm given in [180], i.e., where every accumulator is signed, not only the root. As the accumulator, we chose the original construction [22] in its randomized form. Tests were performed on a *Lenovo Thinkpad T61* with an *Intel* T8300 Dual Core @`2.40` GHz and 4 GiB of RAM. The OS was *Ubuntu* Version 10.04 LTS (64 Bit) with Java-Framework `1.6.0_26-b03` (OpenJDK). We took the median of 10 runs: we only want to demonstrate that our construction is practical as a proof-of-concept. We measured trees with unordered siblings and one with ordered siblings. Trees were randomly generated in an iterative fashion. Re-locations were not considered: only leaf-removal has been implemented. Time for generation of keys for the hash is included. We excluded the time for creating the required signature key pair. However, both becomes

	Generation of σ			Verification of σ		
Nodes	10	100	$1,000$	10	100	$1,000$
Ordered	276	6,715	57,691	26	251	2,572
Unordered	103	599	5,527	21	188	1,820
SHA-512	4	13	40	4	13	40

Table 20: Median Runtime in ms

negligible in terms of the performance for large trees. On digest calculation, we store all intermediate results in RAM to avoid any disk access impact.

As shown, our construction runtime remains within useable limits. The advanced features come at a price; our scheme is considerably slower than a standard hash like SHA-512. Signatures are more often verified than generated, so the overhead for verification has a greater impact. All other provable secure and transparent schemes, i.e., [32] and [51], have the same complexity and therefore just differ by a constant factor. [32] and [51] do not provide a performance analysis on real data. Compared to [203], where a performance analysis of a prototype is provided, this construction offers equal speed or is faster.

4.2.6.11 No.2: Security of the construction of No.2

Our scheme is unforgeable, private and transparent. Assuming $\mathcal{AH}$ is strongly one-way, and the signature scheme Π is UNF-CMA, our scheme is unforgeable, while the indistinguishability of $\mathcal{AH}$ implies privacy and transparency. The formal proofs are in Section 4.2.6.11. We now show that our construction fulfils the given definitions. Namely, these are unforgeability, privacy, and transparency. We prove each property on its own. Note, we can ignore collisions of randoms, as they only appear with negligible probability.

Theorem 7. *Construction No.2 is Unforgeable. If $\mathcal{AH}$ is strongly one-way, while the signature scheme Π is unforgeable, our scheme is unforgeable.*

Proof. Let $\mathcal{A}$ be an algorithm winning the unforgeability game. We can then use $\mathcal{A}$ in an algorithm $\mathcal{B}$ to either to forge the underlying signature scheme Π or to break the strong one-wayness of $\mathcal{AH}$. Given the game in Figure 57 we can derive that a forgery must fall in at least one of the two following cases, for at least one node d in the tree:

- Type 1 Forgery: The value d protected by σ_s has never been signed by the signing oracle.
- Type 2 Forgery: The value d protected by σ_s has been signed, but $T^* \notin \text{span}_{\vdash}(T, \sigma, \text{ADM})$ for any tree T signed by the signing oracle.

Type 1 Forgery: In the first case, we can use the forgery generated by $\mathcal{A}$ to create $\mathcal{B}$ which forges a signature. We construct $\mathcal{B}$ using $\mathcal{A}$ as follows:

1. $\mathcal{B}$ generates the key pair of $\mathcal{AH}$, i.e., *pk* $\leftarrow$ $\mathsf{KeyGen}(1^\lambda)$. It passes *pk* to $\mathcal{A}$. *pk*$_S$ is provided by $\mathcal{B}$'s challenger.
2. All queries to the signing oracle from $\mathcal{A}$ are genuinely answered with one exception: instead of signing digests itself, $\mathcal{B}$ asks it own signing oracle to generate the signature. Afterward, $\mathcal{B}$ returns the signature generated to $\mathcal{A}$.
3. Eventually, $\mathcal{A}$ outputs a pair (T^*, σ^*). $\mathcal{B}$ looks for the message/signature pair (m^*, σ_s^*) inside the transcript not queried to its own signing oracle, i.e., the accumulator value with the signature σ_s^* of the root of (T^*, σ^*). Hence, there exists a value not signed by $\mathcal{B}$'s signing oracle. This pair is then returned as $\mathcal{B}$'s own forgery attempt.

As every tree/signature pair was accepted as valid, but not signed by the signing oracle, $\mathcal{B}$ breaks the unforgeability of the signature algorithm. Here, we have a tight reduction for the first case.

Type 2 Forgery: In the case of a type 2 forgery, we can use $\mathcal{A}$ to construct $\mathcal{B}$, which breaks the strong one-wayness of the underlying accumulator. We construct $\mathcal{B}$ using $\mathcal{A}$ as follows:

1. $\mathcal{B}$ generates a key pair of a signature scheme Π.
2. It receives *pk* of $\mathcal{AH}$. Both public keys are forwarded to $\mathcal{A}$.
3. For every request to the signing oracle, $\mathcal{B}$ uses its hashing oracle to generate the witnesses and the accumulators. All other steps are genuinely performed. The signature is returned to $\mathcal{A}$.
4. Eventually, $\mathcal{A}$ outputs (T^*, σ^*). Given the transcript of the simulation, $\mathcal{A}$ searches for a pair (w^*, y^*) matching an accumulator a, while y^* has not been queried to hashing oracle under a. Note, the root accumulator has been returned: otherwise, we have a type 1 forgery. $\mathcal{B}$ outputs (a, w^*, y^*).

As every new element accepted as being part of the accumulator, while not been hashed by the hashing oracle, breaks the strong one-wayness of the accumulator, we have a tight reduction again.

□

Theorem 8. *Construction No.2 is Private. If $\mathcal{AH}$ is indistinguishable our scheme is private. Note: the random numbers do not leak any information, as they are distributed uniformly and are not ordered. Hence, we do not need to take them into account.*

Proof. Let $\mathcal{A}$ be an algorithm winning the privacy game. We can then use $\mathcal{A}$ in an algorithm $\mathcal{B}$ to break the indistinguishability of the accumulator $\mathcal{AH}$. We construct $\mathcal{B}$ using $\mathcal{A}$ as follows:

1. $\mathcal{B}$ generates a key pair of a signature scheme Π.
2. It receives *pk* of $\mathcal{AH}$. Both public keys are forwarded to $\mathcal{A}$.
3. For every request to the signing oracle, $\mathcal{B}$ produces the expanded trees given ADM. Then, it uses its hashing-oracle to generate the accumulators, and then proceeds honestly as the original algorithm would do. Finally, it returns the generated signature σ to $\mathcal{A}$.

4. For queries to the Left-or-Right oracle, $\mathcal{B}$ extracts the common elements to be accumulated for both trees — this set is denoted $\mathcal{S}$. Note, $\mathcal{S}$ may be empty. The additional elements for the first hash are denoted $\mathcal{R}_0$, and $\mathcal{R}_1$ for the second one. $\mathcal{B}$ now queries its own Left-or-Right oracle with $(\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1)$ for each hash. The result is used as the accumulator and the witnesses required: $\mathcal{B}$ genuinely performs the rest of the signing algorithm and hands over the result to $\mathcal{A}$.
5. Eventually, $\mathcal{A}$ outputs its own guess d.
6. $\mathcal{B}$ outputs d as its own guess.

As we only pass queries, $\mathcal{B}$ succeeds, whenever $\mathcal{A}$ succeeds. □

Theorem 9. *Construction No.2 is Transparent. We already know that the given construction for No.2 scheme is private. As neither the underlying signature, the witness' values, nor the accumulator itself change during a redaction, no building block leaks additional information. Transparency follows.*

4.2.6.12 No.2: Main achievement: Controlled redaction of arbitrary nodes of a tree

Construction No.2 offers a way for redacting arbitrary nodes of a tree without leading to severe problems by degrading structural integrity. The revised security model captures that the signer has to explicitly mark redactable nodes. The derived new construction is based on accumulators. The construction for Scheme No.2 can handle ordered and unordered trees. We have implemented the scheme in JAVA, and as our performance measurements show, it is reasonably fast on non-constrained devices.

4.2.7 New Scheme No.3 (published in [185])

The results we obtained during the research on malleable signatures shows that merging blocks from two versions derived by authorised modifications from the same original was not formally founded. This and other important results were published in the paper 'On updatable redactable signatures' authored by Henrich C. Pöhls and Kai Samelin [185]. We restate all the paper's results and highlight how they are motivated by RERUM and can be facilitated for privacy inline.

Assume we sign a set $\mathcal{S} = \{v_1, v_2, \ldots, v_\ell\}$, generating a signature σ protecting $\mathcal{S}$.[11] The use of a redactable signature scheme ($\mathcal{RSS}$) now allows removing elements from $\mathcal{S}$: a verifying signature σ' for a subset $\mathcal{S}' \subseteq \mathcal{S}$ can be derived by anyone. This action is called a *redaction*. For this, no secret key is not required, i.e., redacting is a public operation. This possibility is contrary to standard digital signatures, which do not permit any alterations. Public redactions are especially useful, if the original signer is not reachable anymore, e.g., in case of death, or if it produces too much overhead to resign a message every time an alteration is necessary, e.g., if communication is too costly. Hence, $\mathcal{RSS}$s partially address the "digital document sanitization problem" [162]. Formally, $\mathcal{RSS}$s are a proper subset of ($\mathcal{P}$-)homomorphic signatures [2]. The obvious applications for $\mathcal{RSS}$s are privacy-preserving handling of medical records, the removal of the date-of-birth from certificates from job applications, and the removal of identifying information for age-restricted locations from XML-files or the cloud [18, 113, 138, 188, 196, 203, 204, 218]. Real implementations are given in [188, 213, 235]. However, existing provably secure constructions offer the possibility of "dynamic updates". In a nutshell, dynamic updates allow the signer to add new elements to existing signatures. This captures the ideas given in [21, 133]. Hence,

[11][32, 51, 204] show how to treat more complex data-structures with an $\mathcal{RSS}$ for sets.

a signer can add new elements without the need to re-sign everything, and also without the need to retransmit or store already signed and transmitted values. This aids applications in the IoT domain.

4.2.7.1 No.3: Goal is to formally capture the actions of update and merge

In the field of $\mathcal{RSS}$s, all existing provably *private* constructions only consider how to redact elements. The opposite — reinstating previously redacted elements, i.e., merging signatures — in a controlled way has neither been formalized nor have security models been properly discussed. Notions of mergeability are initially given by *Merkle* for hash-trees [158], but these are not private in the context of $\mathcal{RSS}$s. The closest existing works mentioning merging in our context are [127, 146, 181, 187]. However, neither of the mentioned schemes is fully private in our model, while [127] is even forgeable — merging from any signed set is possible.

As aforementioned, current security models do not correctly capture the possibility that some signatures can be updated, i.e., that the signer can freely add new elements. Additionally, they also do not discuss that signatures can, under certain circumstances, be merged. We propose a countermeasure: we augment the state-of-the-art security model with explicit access to an "update-oracle", which an adversary can query adaptively. We also rigorously define the notions of "update privacy" and "update transparency". Jumping ahead, both properties describe which information can be derived from an updated signature. We introduce a formal definition of "mergeability", i.e., under which circumstances signatures can be merged into a single one. With private and transparent mergeability, we give the first security model of the inverse operation of redaction, extending the work done in [146]. Again, both properties aim to formalize which information an adversary can obtain from a merged signature. We prove that merging signatures has no negative impact on existing security properties. We show how the new and old notions are related to each other, extending the work by *Brzuska* et al. [32]. We derive a provably secure construction, meeting our enhanced definitions. For our construction, we deploy trapdoor-accumulators. This construction is of independent interest. Moreover, it turns out that we do not require any kind of standard signature scheme, which is a very surprising result on its own. Also, our construction proves that the statement given in [138] that accumulators are not sufficient for $\mathcal{RSS}$s is not true.

4.2.7.2 No. 3: Cryptographic preliminaries

We heavily modify the security model introduced by *Brzuska* et al. [32], as we explicitly allow merging and updating signatures. We do so by introducing the algorithms Merge[12] and Update.

Definition 23 (Mergeable and Updatable $\mathcal{RSS}$)**.** *A mergeable and updatable $\mathcal{RSS}$ consists of six efficient algorithms. Let $\mathcal{RSS} := (\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify}, \mathit{Redact}, \mathit{Update}, \mathit{Merge})$, such that:*

KeyGen**.** *The algorithm KeyGen outputs the public and private key of the signer, i.e.,*
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathit{KeyGen}(1^\lambda)$*, where λ is the security parameter*

Sign**.** *The algorithm Sign gets as input the secret key* sk *and the set $\mathcal{S}$.*
It outputs $(\mathcal{S}, \sigma, \tau) \leftarrow \mathit{Sign}(1^\lambda, \mathsf{sk}, \mathcal{S})$. Here, τ is a tag

[12]Merge was named "combine" in [146]

Verify. *The algorithm Verify outputs a bit $d \in \{0, 1\}$ indicating the correctness of the signature σ, w.r.t.* pk *and τ, protecting $\mathcal{S}$. 1 stands for a valid signature, while 0 indicates the opposite. In particular: $d \leftarrow$ Verify$(1^\lambda, \mathsf{pk}, \mathcal{S}, \sigma, \tau)$*

Redact. *The algorithm Redact takes as input a set $\mathcal{S}$, the public key* pk *of the signer, a tag τ, and a valid signature σ and a set $\mathcal{R} \subset \mathcal{S}$ of elements to be redacted. The algorithm outputs $(\mathcal{S}', \sigma', \tau) \leftarrow$ Redact$(1^\lambda, \mathsf{pk}, \mathcal{S}, \sigma, \mathcal{R}, \tau)$, where $\mathcal{S}' = \mathcal{S} \setminus \mathcal{R}$. $\mathcal{R}$ is allowed to be $\emptyset$. On error, the algorithm outputs $\perp$*

Update. *The algorithm Update takes as input a verifying set/signature/tag tuple $(\mathcal{S}, \sigma, \tau)$, the secret key* sk *and a second set $\mathcal{U}$. It outputs $(\mathcal{S}', \sigma', \tau) \leftarrow$ Update$(1^\lambda, \mathsf{sk}, \mathcal{S}, \sigma, \mathcal{U}, \tau)$, where $\mathcal{S}' = \mathcal{S} \cup \mathcal{U}$, and σ' is a verifying signature on $\mathcal{S}'$. On error, the algorithm outputs $\perp$*

Merge. *The algorithm Merge takes as input the public key* pk *of the signer, two sets $\mathcal{S}$ and $\mathcal{V}$, a tag τ, and the corresponding signatures $\sigma_\mathcal{S}$ and $\sigma_\mathcal{V}$. We require that $\sigma_\mathcal{S}$ and $\sigma_\mathcal{V}$ are valid on $\mathcal{S}$ and $\mathcal{V}$. It outputs the merged set/signature/tag tuple $(\mathcal{U}, \sigma_\mathcal{U}, \tau) \leftarrow$ Merge$(1^\lambda, \mathsf{pk}, \mathcal{S}, \sigma_\mathcal{S}, \mathcal{V}, \sigma_\mathcal{V}, \tau)$, where $\mathcal{U} = \mathcal{S} \cup \mathcal{V}$ and $\sigma_\mathcal{U}$ is valid on $\mathcal{U}$. On error, the algorithm outputs $\perp$*

We assume that one can efficiently, and uniquely, identify all the elements $v_i \in \mathcal{S}$ from a given set $\mathcal{S}$. All algorithms, except Sign and Update, are public operations, as common in $\mathcal{RSS}$s. In other words, all parties can redact and merge sets, which includes the signer, as well as any intermediate recipient. The correctness properties must also hold, i.e., every genuinely signed, redacted, merged, or updated set must verify. The same is true for updates and merging signatures. This must even hold transitively, i.e., the history of the signature must not matter. τ does not change on any operation. As we allow merging signatures, unlinkability cannot be achieved: τ makes signatures linkable.

4.2.7.3 No.3: Extended security model

Next, we introduce the extended security model and define the notions of transparency, privacy, unforgeability, merge privacy, merge transparency, update privacy, and update transparency. We then show how these properties are related to each other. As before, we use the definitions given in [32, 164, 203, 204] as our starting point.

As common in $\mathcal{RSS}$s, all of the following definitions specifically address the additional knowledge a third party can gain from the signature σ alone: if in real documents the redactions or updates are obvious due to additional context information or from the message contents itself, e.g., missing parts of a well known document structure, it may be trivial for attackers to detect them. This observation is general and also applies to schemes which offer context-hiding and cannot be avoided.

Definition 24 (Unforgeability). *No one must be able to produce a valid signature on a set $\mathcal{S}^*$, verifying under* pk *with elements not endorsed by the holder of* sk*, i.e., the signer. That is, even if an attacker can adaptively request signatures on different documents, and also can adaptively update them, it remains impossible to forge a signature for a new set or new elements not queried. In Figure 63 we use $\mathcal{S}_{\tau^*}$ to remember all elements signed by the oracle under tag τ^* and $\mathcal{T}$ to collect all tags. This unforgeability definition is analogous to the standard unforgeability requirement of standard digital signature schemes [98]. We say that an $\mathcal{RSS}$ is unforgeable, if for every probabilistic polynomial time (PPT) adversary $\mathcal{A}$ the probability that the game depicted in Figure 63 returns 1, is negligible.*

```
Experiment Unforgeability$_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  Set $\mathsf{T} \leftarrow \emptyset$
  $(\mathcal{S}^*, \sigma^*, \tau^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(1^\lambda, sk, \cdot)}_{\mathsf{Update}(1^\lambda, sk, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, pk)$
    For each query to oracle Sign:
      let $(\mathcal{S}, \sigma, \tau)$ denote the answer from Sign
      Set $\mathcal{S}_\tau \leftarrow \mathcal{S}$
      Set $\mathsf{T} \leftarrow \mathsf{T} \cup \{\tau\}$
    For each call to oracle Update:
      let $(\mathcal{S}, \sigma, \tau)$ denote the answer from Update
      Set $\mathcal{S}_\tau \leftarrow \mathcal{S}_\tau \cup \mathcal{S}$
  return $1$, if
    $\mathsf{Verify}(1^\lambda, pk, \mathcal{S}^*, \sigma^*, \tau^*) = 1$ and
    $\tau^* \notin \mathsf{T}$ or $\mathcal{S}^*, \nsubseteq \mathcal{S}_{\tau^*}$
```

Figure 63: Unforgeability

```
Experiment Privacy$_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $b \xleftarrow{\$} \{0, 1\}$
  $d \leftarrow \mathcal{A}^{\mathsf{Sign}(1^\lambda, sk, \cdot), \mathsf{LoRRedact}(1^\lambda, \cdot, \cdot, \cdot, \cdot, sk, b)}_{\mathsf{Update}(1^\lambda, sk, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, pk)$
    where oracle LoRRedact
      for input $\mathcal{S}_0, \mathcal{S}_1, \mathcal{R}_0, \mathcal{R}_1$:
      If $\mathcal{R}_0 \nsubseteq \mathcal{S}_0 \vee \mathcal{R}_1 \nsubseteq \mathcal{S}_1$, return $\perp$
      if $\mathcal{S}_0 \setminus \mathcal{R}_0 \neq \mathcal{S}_1 \setminus \mathcal{R}_1$, return $\perp$
      $(\mathcal{S}, \sigma, \tau) \leftarrow \mathsf{Sign}(1^\lambda, sk, \mathcal{S}_b, \tau)$
      return $(\mathcal{S}', \sigma', \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma, \mathcal{R}_b, \tau)$.
  return $1$, if $b = d$
```

Figure 64: Privacy

```
Experiment Transparency$_{\mathcal{A}}^{\mathcal{RSS}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $b \xleftarrow{\$} \{0, 1\}$
  $d \leftarrow \mathcal{A}^{\mathsf{Sign}(1^\lambda, sk, \cdot), \mathsf{Sign/Redact}(1^\lambda, \cdot, \cdot, sk, b), \mathsf{Update}(1^\lambda, sk, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, pk)$
    where oracle Sign/Redact for input $\mathcal{S}, \mathcal{R}$:
      if $\mathcal{R} \nsubseteq \mathcal{S}$, return $\perp$
      $(\mathcal{S}, \sigma, \tau) \leftarrow \mathsf{Sign}(1^\lambda, sk, \mathcal{S})$,
      $(\mathcal{S}', \sigma', \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma, \mathcal{R}, \tau)$
      if $b = 1$:
        $(\mathcal{S}', \sigma', \tau) \leftarrow \mathsf{Sign}(1^\lambda, sk, \mathcal{S}')$
      return $(\mathcal{S}', \sigma', \tau)$
  return $1$, if $b = d$
```

Figure 65: Transparency

Definition 25 (Privacy). *The verifier should not be able to gain any knowledge about redacted elements without having access to them. In this definition, the adversary chooses two tuples $(\mathcal{S}_0, \mathcal{R}_0)$ and $(\mathcal{S}_1, \mathcal{R}_1)$, where $\mathcal{R}_i \subseteq \mathcal{S}_i$ describes what shall be removed from $\mathcal{S}_i$. A redaction of $\mathcal{R}_0$ from $\mathcal{S}_0$ is required to result in the same set as redacting $\mathcal{R}_1$ from $\mathcal{S}_1$. The two sets are input to a "Left-or-Right"-oracle which signs $\mathcal{S}_b$ and then redacts $\mathcal{R}_b$. The adversary wins, if it can decide which pair was used by the oracle as the input to create its corresponding output. This is similar to the standard indistinguishability notion for encryption schemes [97]. We say that an $\mathcal{RSS}$ is private, if for every PPT adversary $\mathcal{A}$ the probability that the game depicted in Figure 64 returns 1, is negligibly close to $\frac{1}{2}$. Note, this definition does not capture unlinkability.*

Definition 26 (Transparency). *The verifier should not be able to decide whether a signature has been created by the signer directly, or through the redaction algorithm Redact. The adversary can choose one*

```
**Experiment** Merge Privacy$^{\mathcal{RSS}}_{\mathcal{A}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $b \xleftarrow{\$} \{0,1\}$
  $d \leftarrow \mathcal{A}^{\mathsf{Sign}(1^\lambda,sk,\cdot),\mathsf{LoRMerge}(1^\lambda,\cdot,\cdot,\cdot,sk,b)}_{\mathsf{Update}(1^\lambda,sk,\cdot,\cdot,\cdot)}(1^\lambda, pk)$
    where oracle LoRMerge
    for input $\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1$:
      if $\mathcal{R}_0 \nsubseteq \mathcal{S} \vee \mathcal{R}_1 \nsubseteq \mathcal{S}$, return $\perp$
      $(\mathcal{S}, \sigma_\mathcal{S}, \tau) \leftarrow \mathsf{Sign}(1^\lambda, sk, \mathcal{S})$
      $(\mathcal{S}', \sigma'_\mathcal{S}, \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma_\mathcal{S}, \mathcal{R}_b, \tau)$
      $(\mathcal{S}'', \sigma''_\mathcal{S}, \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma_\mathcal{S}, \mathcal{S} \setminus \mathcal{R}_b, \tau)$
      return $\mathsf{Merge}(1^\lambda, pk, \mathcal{S}', \sigma'_\mathcal{S}, \mathcal{S}'', \sigma''_\mathcal{S}, \tau)$
  return $1$, if $b = d$
```

Figure 66: Merge Privacy

```
**Experiment** Merge Transparency$^{\mathcal{RSS}}_{\mathcal{A}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $b \xleftarrow{\$} \{0,1\}$
  $d \leftarrow \mathcal{A}^{\mathsf{Sign}(1^\lambda,sk,\cdot),\mathsf{Sign/Merge}(1^\lambda,\cdot,\cdot,sk,b)}_{\mathsf{Update}(1^\lambda,sk,\cdot,\cdot,\cdot)}(1^\lambda, pk)$
    where oracle Sign/Merge for input $\mathcal{S}, \mathcal{R}$:
      if $\mathcal{R} \nsubseteq \mathcal{S}$, return $\perp$
      $(\mathcal{S}, \sigma, \tau) \leftarrow \mathsf{Sign}(1^\lambda, sk, \mathcal{S})$
      if $b = 0$:
        $(\mathcal{T}', \sigma'_\mathcal{T}, \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma_\mathcal{S}, \mathcal{R}, \tau)$
        $(\mathcal{R}', \sigma'_\mathcal{R}, \tau) \leftarrow \mathsf{Redact}(1^\lambda, pk, \mathcal{S}, \sigma_\mathcal{S}, \mathcal{S} \setminus \mathcal{R}, \tau)$
        $(\mathcal{S}', \sigma', \tau) \leftarrow \mathsf{Merge}(1^\lambda, pk, \mathcal{T}', \sigma'_\mathcal{T}, \mathcal{R}', \sigma'_\mathcal{R}, \tau)$
      if $b = 1$:  $(\mathcal{S}', \sigma', \tau) \leftarrow (\mathcal{S}, \sigma_\mathcal{S}, \tau)$
      return $(\mathcal{S}', \sigma', \tau)$
  return $1$, if $b = d$
```

Figure 67: Merge Transparency

tuple $(\mathcal{S}, \mathcal{R})$, where $\mathcal{R} \subseteq \mathcal{S}$ describes what shall be removed from $\mathcal{S}$. The pair is input for a "Sign/Redact" oracle that either signs and redacts elements (using Redact) or remove elements as a redaction would do ($\mathcal{S} \setminus \mathcal{R}$) before signing it. The adversary wins, if it can decide which way was taken. We say that an $\mathcal{RSS}$ is transparent, if for every PPT adversary $\mathcal{A}$, the probability that the game depicted in Figure 65 returns 1, is negligibly close to $\frac{1}{2}$.

Definition 27 (Merge Privacy). *If a merged set is given to another third party, the party should not be able to derive any information besides what is contained in the merged set, i.e., a verifier should not be able to decide which elements have been merged from what set. In this definition, the adversary can choose three sets $\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1$. The oracle LoRMerge signs $\mathcal{S}$ and then generates two signed redacted versions $\mathcal{S}' = \mathcal{S} \setminus \mathcal{R}_b$ and $\mathcal{S}'' = \mathcal{R}_b$. Then, it merges the signatures again. The adversary wins, if it can decide if $\mathcal{R}_0$ or $\mathcal{R}_1$ was first redacted from $\mathcal{S}$ and then merged back. We say that an $\mathcal{RSS}$ is merge private, if for every PPT adversary $\mathcal{A}$, the probability that the game depicted in Figure 66 returns 1, is negligibly close to $\frac{1}{2}$.*

Definition 28 (Merge Transparency). *If a set is given to a third party, the party should not be able to decide whether the set has been created only by Sign or through Sign and Merge. The adversary can choose one tuple $(\mathcal{S}, \mathcal{R})$ with $\mathcal{R} \subseteq \mathcal{S}$. This pair is input to a Sign/Merge oracle that signs the set $\mathcal{S}$ and either returns this set/signature pair directly $(b = 1)$ or redacts the $\mathcal{S}$ into two signed "halves" $\mathcal{R}$ and $\mathcal{T}$ only to merge them together again and return the set/signature pair derived using Merge $(b = 0)$. The adversary wins, if it can decide which way was taken. We say that an $\mathcal{RSS}$ is merge transparent, if for every PPT adversary $\mathcal{A}$, the probability that the game depicted in Figure 67 returns 1, is negligibly close to $\frac{1}{2}$.*

Note, the notions of merge transparency and merge privacy are very similar to the notions of privacy and transparency, as they achieve comparable goals.

Definition 29 (Update Privacy). *If an updated set is given to another third party, the party should not be able to derive which elements have been added. In the game, the adversary wins, if it can decide which*

```
**Experiment** Update Privacy$^{\mathcal{RSS}}_{\mathcal{A}}(\lambda)$
 $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \xleftarrow{\$} \{0,1\}$
 $d \leftarrow \mathcal{A}^{\text{Sign}(1^\lambda, sk, \cdot), \text{LoRUpdate}(1^\lambda, \cdot, \cdot, \cdot, sk, b)}_{\text{Update}(1^\lambda, sk, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, pk)$
    where oracle LoRUpdate for input $\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1$:
       $(\mathcal{S}', \sigma'_{\mathcal{S}}, \tau) \leftarrow \text{Sign}(1^\lambda, sk, \mathcal{S} \cup \mathcal{R}_b)$
       return $\text{Update}(1^\lambda, sk, \mathcal{S}', \sigma'_{\mathcal{S}}, \mathcal{R}_{1-b}, \tau)$
 return $1$, if $b = d$
```

Figure 68: Update Privacy

```
**Experiment** Update Transparency$^{\mathcal{RSS}}_{\mathcal{A}}(\lambda)$
 $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \xleftarrow{\$} \{0,1\}$
 $d \leftarrow \mathcal{A}^{\text{Sign}(1^\lambda, sk, \cdot), \text{Sign/Update}(1^\lambda, \cdot, \cdot, sk, b)}_{\text{Update}(1^\lambda, sk, \cdot, \cdot, \cdot, \cdot)}(1^\lambda, pk)$
    where oracle Sign/Update for input $\mathcal{S}, \mathcal{R}$:
       if $b = 1$:  $(\mathcal{S}', \sigma', \tau) \leftarrow \text{Sign}(1^\lambda, sk, \mathcal{S} \cup \mathcal{R})$,
       if $b = 0$:  $(\mathcal{T}', \sigma'_{\mathcal{T}}, \tau) \leftarrow \text{Sign}(1^\lambda, sk, \mathcal{S})$
                    $(\mathcal{S}', \sigma', \tau) \leftarrow \text{Update}(1^\lambda, sk, \mathcal{T}', \sigma'_{\mathcal{T}}, \mathcal{R}, \tau)$
       return $(\mathcal{S}', \sigma', \tau)$
 return $1$, if $b = d$
```

Figure 69: Update Transparency

elements were added after signature generation. In this definition, the adversary can choose three sets $\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1$. The oracle LoRUpdate signs $\mathcal{S} \cup \mathcal{R}_b$ and then adds $\mathcal{R}_{b-1}$ to the signature. The adversary wins, if it can decide which set was used for the update. A scheme $\mathcal{RSS}$ is update private, if for every PPT adversary $\mathcal{A}$, the probability that the game depicted in Figure 68 returns 1, is negligibly close to $\frac{1}{2}$.

Definition 30 (Update Transparency)**.** *A verifying party should not be able to decide whether the received set has been created by Sign or through Update. The adversary can choose one pair $(\mathcal{S}, \mathcal{R})$. This pair is input to a Sign/Update oracle that either signs the set $\mathcal{S} \cup \mathcal{R}$ ($b = 1$) or signs $\mathcal{S}$ and then adds $\mathcal{R}$ using Update ($b = 0$). The adversary wins, if it can decide which way was taken. We say that a scheme $\mathcal{RSS}$ is update transparent, if for every PPT adversary $\mathcal{A}$, the probability that the game depicted in Figure 69 returns 1, is negligibly close to $\frac{1}{2}$.*

Note, that the notions of update transparency and update privacy are, on purpose, kept very similar to the notions of privacy and transparency due to their similar goals.

Definition 31 (Secure $\mathcal{RSS}$)**.** *We call an $\mathcal{RSS}$ secure, if it is unforgeable, transparent, private, merge transparent, merge private, update private, and update transparent.*

4.2.7.4 No.3: Relations between security properties

We now give some relations between the security properties. This section can be kept brief, as we tailored the definitions to be similar (in terms of relation) to the ones given in [32]. This is intentional, to keep consistent with existing wording and to blend into the large body of existing work. We have to explicitly consider the update-oracle, as it may leak information about the secret key *sk*.

Theorem 10 (Merge Transparency $\implies$ Merge Privacy)**.** *Every scheme which is merge transparent, is also merge private.*

Proof. Intuitively, the proof formalizes the following idea: if an adversary can decide which elements have been merged, then it can decide that the signature cannot be created by Sign, but by Merge.

Assume an (efficient) adversary $\mathcal{A}$ that wins our merge privacy with probability $\frac{1}{2} + \epsilon$. We can then construct an (efficient) adversary $\mathcal{B}$ which wins the merge transparency game with probability $\frac{1}{2} + \frac{\epsilon}{2}$. According to the merge transparency game, $\mathcal{B}$ receives a public key *pk* and oracle access to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Sign/Merge}}$, and $\mathcal{O}^{\mathsf{Update}}$. Let $\mathcal{B}$ randomly pick a bit $b' \in \{0,1\}$. $\mathcal{B}$ forwards *pk* to $\mathcal{A}$. Whenever $\mathcal{A}$ requests access to the signing oracle $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{B}$ honestly forwards the query to its oracle and returns the unmodified answer to $\mathcal{A}$. The same is true for $\mathcal{O}^{\mathsf{Update}}$. When $\mathcal{A}$ requests access to $\mathcal{O}^{\mathsf{LoRMerge}}$, i.e., when it sends a query $(\mathcal{S}, \mathcal{R}_0, \mathcal{R}_1)$, then $\mathcal{B}$ checks that $\mathcal{R}_0 \subset \mathcal{S} \wedge \mathcal{R}_1 \subset \mathcal{S}$ and forwards $(\mathcal{S}, \mathcal{R}_{b'})$ to $\mathcal{O}^{\mathsf{Sign/Merge}}$ and returns the answer to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs its guess d. Our adversary $\mathcal{B}$ outputs 0, if $d = b'$ and 1 otherwise. What is the probability that $\mathcal{B}$ is correct? We have to consider two cases:

1. If $b = 0$, then $\mathcal{O}^{\mathsf{Sign/Merge}}$ signs, redacts, and merges the set. This gives exactly the same answer as $\mathcal{O}^{\mathsf{LoRRedact}}$ would do, if using the bit b'. Hence, $\mathcal{A}$ can correctly guess the bit b' with probability at least $\frac{1}{2} + \epsilon$, if $b = 0$.
2. If $b = 1$, then $\mathcal{O}^{\mathsf{Sign/Merge}}$ always signs the set as is. Hence, the answer is independent of b'. $\Pr[\mathcal{B} = 1 \mid b = 1] = \frac{1}{2}$ follows.

Hence, due to the probability of $\frac{1}{2}$ that $b = 1$, it follows that $\Pr[\mathcal{B} = b] = \frac{1}{2} + \frac{\epsilon}{2}$. Hence, $\mathcal{B}$ has non-negligible advantage, if ϵ is non-negligible. □

Theorem 11 (Merge Privacy $\not\Rightarrow$ Merge Transparency)**.** *There is a scheme which is merge private, but not merge transparent.*

Proof. At sign, we append a bit $d = 0$. For all other algorithms d is cut off, and appended after the algorithm finished. However, we set $d = 1$ once signatures are merged. Obviously, we leave all other properties intact. □

Theorem 12 (Update Transparency $\implies$ Update Privacy)**.** *Every scheme which is update transparent, is also update private.*

Proof. The proof is essentially the same as for Th. 10. □

Theorem 13 (Update Privacy $\not\Rightarrow$ Update Transparency)**.** *There is a scheme which is update private, but not update transparent.*

Proof. The proof is essentially the same as for Th. 11. □

Theorem 14 (Merge Transparency is independent)**.** *There is a scheme which fulfills all mentioned security goals but merge transparency.*

Proof. The proof is essentially the same as for Th. 11. □

Theorem 15 (Update Transparency is independent)**.** *There is a scheme which fulfills all mentioned security goals but update transparency.*

Proof. The proof is essentially the same as for Th. 11. □

Theorem 16 (Unforgeability is independent)**.** *There is a scheme which fulfills all mentioned security goals but unforgeability.*

Proof. We simply use a verify algorithm which always accepts all inputs. □

Theorem 17 (Transparency $\implies$ Privacy)**.** *Every scheme which is transparent, is also private. Similar to [32].*

Theorem 18 (Privacy $\not\Rightarrow$ Transparency)**.** *There is a scheme which is private, but not transparent. Similar to [32].*

Theorem 19 (Transparency is independent)**.** *There is a scheme which fulfills all mentioned security goals but transparency. Similar to [32].*

Even though the transparency properties give stronger security guarantees, legislation requires that altered signatures must be distinguishable from new ones [35]. However, privacy is the absolute minimum to be useful [35]. We therefore need to split the definitions: depending on the use-case, one can then decide which properties are required.

4.2.7.5 No.3: Construction based on trapdoor-accumulators

Cryptographic accumulators have been introduced by *Benaloh* and *de Mare* [22]. They hash a potentially very large set $\mathcal{S}$ into a short single value a, called the accumulator. For each element accumulated, a witness is generated, which vouches for the accumulation. A trapdoor-accumulator allows generating proofs for new elements not contained by use of a trapdoor. Our construction is based upon such an accumulator. Using an accumulator allows us to achieve mergeability "for free", as we can add and remove witnesses and the corresponding elements freely. We do not require non-membership witnesses [143], or non-deniability [147] for our scheme to work. We do note that there exists the possibility of dynamically updating an accumulator [43]. However, they also allow removing accumulated elements, while they need to adjust every single witness. This is not necessary for our goals. However, accumulators are very versatile. We leave it as open work to discuss the impact of accumulators with different properties plugged into our construction.

4.2.7.6 No.3: Algorithmic description and security model of trapdoor accumulators

We now introduce trapdoor accumulators. The definition is derived from [14].

Definition 32 (Trapdoor Cryptographic Accumulators)**.** *A cryptographic trapdoor accumulator $\mathcal{ACC}$ consists of four efficient (PPT) algorithms. In particular, $\mathcal{ACC} := (Gen, Dig, Proof, Verf)$ such that:*

Gen*. The algorithm* Gen *is the key generator. On input of the security parameter λ, it outputs the key pair $(\mathsf{sk}_{\mathcal{ACC}}, \mathsf{pk}_{\mathcal{ACC}}) \leftarrow Gen(1^\lambda)$*

Dig*. The algorithm Dig takes as input the set $\mathcal{S}$ to accumulate, the public parameters $\mathsf{pk}_{\mathcal{ACC}}$. It outputs an accumulator value $a \leftarrow Dig(1^\lambda, \mathsf{pk}_{\mathcal{ACC}}, \mathcal{S})$*

Proof*. The deterministic algorithm* Proof *takes as input the secret key $\mathsf{sk}_{\mathcal{ACC}}$, the accumulator a, and a value v and returns a witness p for v. Hence, it outputs $p \leftarrow Proof(1^\lambda, \mathsf{sk}_{\mathcal{ACC}}, a, v)$*

```
Experiment Strong − Coll. − Res.$_{\mathcal{A}}^{\mathcal{ACC}}(\lambda)$
  $(sk_{\mathcal{ACC}}, pk_{\mathcal{ACC}}) \leftarrow \mathsf{Gen}(1^\lambda)$
  $(S^*, st) \leftarrow \mathcal{A}_1(1^\lambda, pk_{\mathcal{ACC}})$ //$st$ denotes $\mathcal{A}$'s state
  $a \leftarrow \mathsf{Dig}(1^\lambda, pk_{\mathcal{ACC}}, S^*)$
  $(v^*, p^*) \leftarrow \mathcal{A}_2^{\mathsf{Proof}(1^\lambda, sk_{\mathcal{ACC}}, a, \cdot)}(st, a)$
  return 1, if
    $\mathsf{Verf}(1^\lambda, pk_{\mathcal{ACC}}, a, v^*, p^*) = 1$,
    and $v^*$ has not been queried to Proof
```

Figure 70: Strong Collision-Resistance

Verf. *The verification algorithm* Verf *takes as input the public key* $\mathsf{pk}_{\mathcal{ACC}}$*, an accumulator* a*, a witness* p*, and a value* v *and outputs a bit* $d \in \{0, 1\}$*, indicating whether* p *is a valid witness for* v *w.r.t.* a *and* $\mathsf{pk}_{\mathcal{ACC}}$*. Hence, it outputs* $d \leftarrow \mathit{Verf}(1^\lambda, \mathsf{pk}_{\mathcal{ACC}}, a, v, p)$

We require the usual correctness properties to hold. Refer to [14] for a formal definition of the correctness properties for accumulators.

Definition 33 (Strong Collision-Resistance)**.** *An adversary should not be able find a valid witness/element pair* (p^*, v^*) *for a given accumulator* a*, even if it is allowed to adaptively query for elements not contained in the original set accumulated and to choose the original set to be accumulated. We call a family of trapdoor accumulators strongly collision-resistant, if the probability that the experiment depicted in Figure 70 returns* 1*, is negligible. We do note that this definition is very similar to the standard unforgeability of signature schemes. The naming is due to historical reasons [14].*

4.2.7.7 Trapdoor-accumulators

Next, we show how a trapdoor-accumulator can be build. We use the ideas given in [14], but make use of the trapdoor $\varphi(n)$.

Construction 1 (Trapdoor-Accumulator $\mathcal{ACC}$)**.** *We require a division-intractable hash-function* $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ *mapping to odd numbers. A formal definition is given in [94]. Let* $\mathcal{ACC} := (\mathit{Gen}, \mathit{Dig}, \mathit{Proof}, \mathit{Verf})$ *such that:*

Gen. *Generate* $n = pq$*, where* p *and* q *are distinct safe primes of length* λ*.*[13] *Return* $(\varphi(n), (n, \mathcal{H}))$*, where* $\varphi(pq) := (p - 1) \cdot (q - 1)$*.*

Dig. *To improve efficiency, we use the build-in trapdoor. A new digest can therefore be drawn at random. Return* $a \in_R \mathbb{Z}_n^\times$*.*

Proof. *To generate a witness* p_i *for an element* v_i*, set* $v_i' \leftarrow \mathcal{H}(v_i)$*. Output* $p_i \leftarrow a^{v_i'^{-1} \pmod{\varphi(n)}} \bmod n$

Verf. *To check the correctness of a proof* p *w.r.t. an accumulator* a*, the public key* $\mathsf{pk}_{\mathcal{ACC}}$*, and a value* v*, output* 1*, if* $a \overset{?}{=} p^{\mathcal{H}(v)} \pmod{n}$*, and* 0 *otherwise*

[13]A prime p is safe, if $p = 2p' + 1$, where p' is also prime.

We do note that this construction is related to GHR-signatures [94]. Due to the build-in trapdoor, we do not require any auxiliary information as proposed in [14]. The use of safe primes allows us to almost always find a root for odd numbers. If we are not able to do so, we can trivially factor n. The proofs that our trapdoor-accumulator is strongly collision-resistant can be found in the appendix.

We want to explicitly stress that an adversary can simulate the Proof-oracle itself for the elements used for Dig. It calculates $a = x^{\prod_{v_i \in S} \mathcal{H}(v_i)} \mod n$ for a random $x \in_R \mathbb{Z}_n^\times$ and for each proof p_i, it lets $p_i = x^{\prod_{v_j \in S, i \neq j} \mathcal{H}(v_j)} \mod n$. For new elements, this technique does not work. Note, a is drawn at random for efficiency. We can also use the slower method aforementioned: a will be distributed exactly in the same way.

4.2.7.8 No.3: Construction of an updateable and mergeable $\mathcal{RSS}$

The basic ideas are: (1) Our trick is to fix the accumulator a for *all* signatures. Additionally, each element is tagged with a unique string τ to tackle mix-and-match attacks. Hence, all derived subset/signature pairs are linkable by the tag τ. τ is also accumulated to avoid trivial "empty-set"-attacks. (2) Redactions remove v_i and its corresponding witness p_i. The redactions are private, as without knowledge of the proof p_i nobody can verify if v_i is "in" the accumulator a. (3) Mergeability is achieved, as supplying an element/witness pair allows a third party to add it back into the signature. (4) Unforgeability comes from the strong collision-resistance of $\mathcal{ACC}$. (5) Dynamic updates are possible due to a trapdoor in $\mathcal{ACC}$, only known to the signer. (6) Privacy directly follows from definitions, i.e., the number of proofs is fixed, while the proofs itself are deterministically generated, without taking already generated proofs into account. We do note that we can also use aggregate-signatures to reduce the signature size [27]. However, we want to show that an accumulator is enough to build $\mathcal{RSS}$s. Having a suitable security model, we can now derive an efficient, stateless, yet simple construction. Our construction is inspired by [127]. However, their construction is forgeable and non-private in our model, as they allow for arbitrary merging, and do not hide redacted elements completely. One may argue that a very straight-forward construction exists: one signs each element $v_i \in \mathcal{S}$ and gives out the signatures. However, our approach has some advantages: we can exchange the accumulator to derive new properties, e.g., prohibiting updates using a trapdoor-free accumulator [147]. Moreover, we prove that using accumulators are sufficient, opposing the results of [138].

Construction 2 (Updatable and Mergeable $\mathcal{RSS}$)**.** *We use $||$ to denote a uniquely reversible concatenation of strings. Let $\mathcal{RSS} := (KeyGen, Sign, Verify, Redact, Update, Merge)$ such that:*

KeyGen*. The algorithm KeyGen generates the key pair in the following way:*

1. *Generate key pair required for $\mathcal{ACC}$, i.e., run $(\mathsf{sk}_{\mathcal{ACC}}, \mathsf{pk}_{\mathcal{ACC}}) \leftarrow Gen(1^\lambda)$*
2. *Call $a \leftarrow Dig(\mathsf{pk}_{\mathcal{ACC}}, \emptyset)$*
3. *Output $(\mathsf{sk}_{\mathcal{ACC}}, (\mathsf{pk}_{\mathcal{ACC}}, a))$*

Sign*. To sign a set $\mathcal{S}$, perform the following steps:*

1. *Draw a tag $\tau \in_R \{0,1\}^\lambda$*
2. *Let $p_\tau \leftarrow Proof(\mathsf{sk}_{\mathcal{ACC}}, a, \tau)$*

3. *Output* $(\mathcal{S}, \sigma, \tau)$*, where* $\sigma = (p_\tau, \{(v_i, p_i) \mid v_i \in \mathcal{S} \wedge p_i \leftarrow \mathit{Proof}(\mathsf{sk}_{\mathcal{ACC}}, a, v_i || \tau)\})$

Verify*. To verify signature* $\sigma = (p_\tau, \{(v_1, p_1), \ldots, (v_k, p_k)\})$ *with tag* τ*, perform:*

1. *For all* $v_i \in \mathcal{S}$ *check that* $\mathit{Verf}(\mathsf{pk}_{\mathcal{ACC}}, a, v_i || \tau, p_i) = 1$
2. *Check that* $\mathit{Verf}(\mathsf{pk}_{\mathcal{ACC}}, a, \tau, p_\tau) = 1$
3. *If Verf succeeded for all elements, output* 1*, otherwise* 0

Redact*. To redact a subset* $\mathcal{R}$ *from a valid signed set* $(\mathcal{S}, \sigma)$ *with tag* τ*, with* $\mathcal{R} \subseteq \mathcal{S}$*, the algorithm performs the following steps:*

1. *Check the validity of* σ *using Verify. If* σ *is not valid, return* $\perp$
2. *Output* $(\mathcal{S}', \sigma', \tau)$*, where* $\sigma' = (p_\tau, \{(v_i, p_i) \mid v_i \in \mathcal{S} \setminus \mathcal{R}\})$

Update*. To update a valid signed set* $(\mathcal{S}, \sigma)$ *with tag* τ *by adding* $\mathcal{U}$ *and knowing* $\mathsf{sk}_{\mathcal{ACC}}$*, the algorithm performs the following steps:*

1. *Verify* σ *w.r.t.* τ *using Verify. If* σ *is not valid, return* $\perp$
2. *Output* $(\mathcal{S} \cup \mathcal{U}, \sigma', \tau)$*, where* $\sigma' = (p_\tau, \{(v_i, p_i) \mid v_i \in \mathcal{S}\} \cup \{(v_k, p_k) \mid v_k \in \mathcal{U}, p_k \leftarrow \mathit{Proof}(\mathsf{sk}_{\mathcal{ACC}}, a, v_k || \tau)\})$

Merge*. To merge two valid set/signature pairs* $(\mathcal{S}, \sigma_{\mathcal{S}})$ *and* $(\mathcal{T}, \sigma_{\mathcal{T}})$ *with an equal tag* τ*, the algorithm performs the following steps:*

1. *Verify* $\sigma_{\mathcal{S}}$ *and* $\sigma_{\mathcal{T}}$ *w.r.t.* τ *using Verify. If they do not verify, return* $\perp$
2. *Check, that both have the same tag* τ
3. *Output* $(\mathcal{S} \cup \mathcal{T}, \sigma_{\mathcal{U}}, \tau)$*, where* $\sigma_{\mathcal{U}} = (p_\tau, \{(v_i, p_i) \mid v_i \in \mathcal{S} \cup \mathcal{T}\})$*, where* p_i *is taken from the corresponding signature*

Construction for Scheme No.3 fulfils all security goals (all but unforgeability even perfectly), and is therefore useable in practice. The proofs of security are in the appendix. All reductions are tight, i.e., we have no reduction losses. We want to explicitly clarify that we do not see the transitive closure of the updates as forgeries. If we want to disallow the "transitive update merging", we can deploy accumulators which also update the witnesses, e.g., [43]. This requires a new security model, which renders existing constructions insecure, which we wanted to avoid. We leave this as future work.

4.2.7.9 No.3: Security proofs for Scheme No.3

Theorem 20 (Our Construction is Unforgeable)**.** *Our construction is unforgeable, if the underlying accumulator is strongly collision-resistant.*

Proof. We do not consider tag collisions, as they only appear with negligible probability. $S^* \subseteq S_\tau$ for some signed τ is not a forgery, but a redaction. We denote the adversary winning the unforgeability game as $\mathcal{A}$. We can now derive that the forgery must fall into exactly one of the following categories:

Case 1: $\mathcal{S}^*, \nsubseteq \mathcal{S}_{\tau^*}$, and τ^* was used as a tag by Sign

Case 2: $\mathcal{S}^*$, verifies, and τ^* was never used as a tag by Sign

Each case leads to a contradiction about the security of our accumulator.

4.2.7.10 Case 1

In this case, an element v^* not been returned by the Proof-oracle for the accumulator a, but is contained in $\mathcal{S}^*$,. We break the strong collision-resistance of the underlying accumulator by letting $\mathcal{B}$ use $\mathcal{A}$ as a black-box:

1. $\mathcal{B}$ receives $pk_{\mathcal{ACC}}$ from the challenger
2. $\mathcal{B}$ requests an accumulator a for $\emptyset$
3. $\mathcal{B}$ receives a from its own challenger
4. $\mathcal{B}$ forwards $pk = (pk_{\mathcal{ACC}}, a)$ to $\mathcal{A}$
5. For each query to the signing oracle, $\mathcal{B}$ answers it honestly: it draws τ honestly and uses the Proof-oracle provided to get a witness for each $v_j \in \mathcal{S}_i$ queried, with τ concatenated as the label. Also, $\mathcal{B}$ gets a proof for τ
6. For each call to the Update-oracle, $\mathcal{B}$ uses its Proof-oracle provided to get a witness for each $v_j \in \mathcal{S}_i$ queried, with τ concatenated as the label
7. Eventually, $\mathcal{A}$ outputs a pair (S^*, σ^*)
8. $\mathcal{B}$ looks for (v^*, p^*), v^* not queried to Proof, in (S^*, σ^*) and returns them

In other words, there exists an element $v^* \in \mathcal{S}^*$, with a corresponding witness p^*. If v^* has not been asked to the Proof-oracle, $\mathcal{B}$ breaks the collision-resistance of the underlying accumulator by outputting (v^*, p^*). This happens with the same probability as $\mathcal{A}$ breaks unforgeability in case 1. Hence, the reduction is tight.

4.2.7.11 Case 2

In case 2, the tag τ^* has not been accumulated. We break the strong collision-resistance of the underlying accumulator by letting $\mathcal{B}$ use $\mathcal{A}$:

1. $\mathcal{B}$ receives $pk_{\mathcal{ACC}}$ from the challenger
2. $\mathcal{B}$ requests an accumulator a for $\emptyset$
3. $\mathcal{B}$ forwards $pk = (pk_{\mathcal{ACC}}, a)$ to $\mathcal{A}$

4. For each query to the signing oracle, $\mathcal{B}$ answers it honestly: it draws τ honestly and uses the Proof-oracle provided to get a witness for each $v_j \in \mathcal{S}_i$ queried, with τ concatenated as the label. Also, $\mathcal{B}$ gets a proof for τ
5. For calls to the Update-oracle, $\mathcal{B}$ uses its Proof-oracle provided to get a witness for each $v_j \in \mathcal{S}_i$ queried, with τ concatenated as the label
6. Eventually, $\mathcal{A}$ outputs a pair (S^*, σ^*, τ^*)
7. $\mathcal{B}$ returns (p^*_τ, τ^*). Both is contained in σ^*

In other words, there exists an element $\tau^* \in \sigma^*$ with a corresponding witness p^*_τ, as otherwise σ^* would not verify. We know that τ^* was not queried to Proof, because otherwise we have case 1. This happens with the same probability as $\mathcal{A}$ breaks the unforgeability in case 2. Note, we can ignore additional elements here. Again, the simulation is perfect. □

Theorem 21 (Construction No.3 is Merge Private and Transparent)**.** *Our construction is merge private and merge transparent.*

Proof. The distributions of merged and freshly signed signatures are *equal*. In other words, the distributions are the same. This implies, that our construction is *perfectly* merge private and *perfectly* merge transparent. □

Theorem 22 (Construction No.3 is Transparent and Private)**.**

Proof. As the number of proofs only depends on n, which are also deterministically generated, without taking existing proofs into account, an adversary has zero advantage on deciding how many additional proofs have been generated. Moreover, redacting only removes elements and proofs from the signatures. Hence, fresh and redacted signatures are distributed *identically*. *Perfect* transparency, and therefore also *perfect* privacy, is implied. □

Theorem 23 (Construction No.3 is Update Private and Transparent)**.** *Our construction is update private and update transparent.*

Proof. The distributions of updated and freshly signed signatures are *equal*. In other words, the distributions are the same. This implies, that our construction is *perfectly* update private and *perfectly* update transparent. □

Theorem 24 (The Accumulator is Strongly Collision-Resistant)**.**

Proof. Let $\mathcal{A}$ be an adversary breaking the strong-collision-resistance of our accumulator. We can then turn $\mathcal{A}$ into an adversary $\mathcal{B}$ which breaks the unforgeability of the GHR-signature [94] in the following way:

1. $\mathcal{B}$ receives the modulus n, the hash-function $\mathcal{H}$, and the value s. All is provided by the GHR-challenger
2. $\mathcal{B}$ sends *pk* $= (n, \mathcal{H})$ to $\mathcal{A}$. Then, $\mathcal{B}$ waits for $\mathcal{S}$ from $\mathcal{A}$

3. $\mathcal{B}$ sends s to $\mathcal{A}$. Note, we have a *perfect* simulation here, even as we ignore $\mathcal{S}$, as the GHR-signature scheme draws s in the *exact* same way as we do for our accumulator
4. For each Proof-oracle query v_i, $\mathcal{B}$ asks its signing oracle provided, which returns a signature σ_i. Send σ_i as the witness p_i back to $\mathcal{A}$
5. Eventually, $\mathcal{A}$ comes up with an attempted forgery (v^*, p^*)
6. $\mathcal{B}$ returns (v^*, p^*) as its own forgery attempt

$\square$

Now let $y = v^*$, and $p = \sigma^*$. As $s = p^{\mathcal{H}(y)} \pmod{n}$, and we have embedded our challenges accordingly, $\mathcal{B}$ breaks the GHR-signature with the same probability as $\mathcal{A}$ breaks the strong collision-resistance of our trapdoor-accumulator. [94] shows how to break the strong-RSA-assumption with the given forgery.

4.2.7.12 No.3: Main contribution: Formal notion of mergeability as an inverse of redaction

We have revised existing notions of redactable signature schemes. We derived a security model, addressing the shortcomings of existing ones. Moreover, in have formalized the notion of mergeability, the inverse of redactions. These properties allow using this $\mathcal{RSS}$ for IoT data that is subject to distributed workflows. E.g. application scenarios where first some IoT data that was signed gets redacted and thus fragmented into different versions, e.g. for different applications that forward the redacted version to a database in their cloud-storage. If this data is then brought together with other fragments of the same originally signed data, it can be recombined. As noted, this $\mathcal{RSS}$ can not offer unlinkability.

4.2.8 Candidate malleable signature schemes for on-device usage

The list of malleable signature schemes is long. RERUM has done research on the state of the art and decided to go for malleable signature schemes with two different malleabilities. Due to their distinct properties or due to their ease in their construction RERUM has chosen the following schemes as candidates for implementation. In order to be confident that this list contains suitable candidates RERUM implemented several algorithms as prototypes. Currently these developments are ongoing, the implementations on Zolertia's ReMOTE will be subjected to testing as part of Task 5.3. Table 21 gives a quick overview of the schemes that RERUM thinks are interesting. It also indicates if we have started implementing prototypes to run on Zolertia's ReMOTE.

As Table 21 shows we have implemented well known simple schemes but also tried to limit the amount of new cryptographic functionalities we need to program in order to get the building blocks on real RERUM Devices. They all require an digital signature scheme existentially unforgeable under chosen message attacks (UNF-CMA). Fortunately, ECC signatures like those described in Deliverable 3.1 of RERUM, like Ed25519 and NIST, achieve this. RERUM is currently evaluating the overhead of different signature implementations with respect to speed (runtime), code size (programmable flash usage) and energy consumption. This work is carried out as part of this WP and WP5, so results are expected at the end of the laboratory experiments task. Detailed results are expected to be published latest in first quarter of 2016.

Scheme	**Properties**	**Prototype Started**	**Required Building Block**
EuroPKI'12 [35]	$\mathcal{SSS}$; non-interactive publicly accountable; two signature invocations	yes	standard hash (i.e. SHA) + UNF-CMA Digital Signature (i.e. Ed25519 or equivalent)
No.1 (ARES'13 [186])	$\mathcal{SSS}$; non-interactive publicly accountable on the level of blocks; constant amount of signature invocations	yes	tag-based chameleon hash + UNF-CMA Digital Signature (i.e. Ed25519 or equivalent)
No.2 (ICEITE'14 [67])	$\mathcal{RSS}$; tree-structured data (e.g. JSON); non-leaf redaction	yes	accumulating hash + UNF-CMA Digital Signature (i.e. Ed25519 or equivalent)
No.3 (ACNS'14 [185])	$\mathcal{RSS}$; sets (e.g. simple JSON); update and merge	yes	accumulating hash + UNF-CMA Digital Signature (i.e. Ed25519 or equivalent

Table 21: Overview of the schemes that RERUM plans to bring onto Zolertia Re-MOTE for lab experiments.

4.2.9 Summary

Malleable signatures enable the co-existence of privacy protecting changes and offer a reduced but lower-bounded integrity protection. As such, RERUM's advancement of the state of the art in malleable signatures allows applications to specify all of the following in a cryptographically secure fashion:

Who can modify in terms of distributing sanitizer secret keys.

What can be modified in terms of splitting messages into blocks and specifying which are admissible.

Have flexibility for redactions in tree-based data structures allowing to flexibly redact also non-leafs in the tree-representation i.e. in JSON.

Detect what changed in terms of being non-interactive and being fine-grained on the level of single blocks.

4.3 Data Perturbation with integrity preservation on the gateway

Cornerstones of the Smart Grid (SG) are the Smart Meter (SM) and the Smart Meter Gateway (SMGW) as depicted in Figure 73. Both devices are trusted and installed by a SG stakeholder, i.e., the power grid provider. A SM sends energy consumption values via the SMGW to a collecting SG stakeholder. We assume that the SM produces accurate and timely readings. This allows the stakeholder to get a fine resolution picture of the energy consumption at customer's premises, which can be used for purposes like demand forecasting or creating energy profiles [239]. To counter act malicious tampering, both SM and SMGW protect the integrity and authenticity of the transmitted data. All communication between the SM within a household and the SMGW is secured for wired as well as for wireless connections. Classical digital signatures offer such a protection: they allow detecting any change that occurred after the signature's generation. Cryptographically, a digital signature scheme is said to be *unforgeable*, e.g., RSA-PSS [20]. Hence, data requested by SG stakeholders is encrypted and signed by the SMGW before being sent [38].

Having tampering solved by digital signatures, one problem remains: The fine grained values impose a privacy threat to the residential customer. Several works show that too fine-grained energy values allow detecting appliances within the household [165], detecting the use mode of the appliances [80] as well as deducting the residential customers' behaviour [148]. To mitigate those threats current research and governmental organisations suggest using Privacy Enhancing Technologies (PET). For example, the German "Bundesamt für Sicherheit in der Informationstechnik (BSI)" is using pseudonymization as a privacy protecting mechanism [38]. In [124] it has been shown that de-pseudonymization is feasible in the Smart Grid and pseudonymization is vulnerable to linkage attacks. However, pseudonymization is only one tool from the PET toolbox. PET is rather a holistic concept than just one technical solution. One main principle of PET is to reduce the amount of information to a minimum required for a specific application, i.e., data minimisation. Another PET tool is the reduction of the data's accuracy or timeliness. However, the application of such a PET as this one would require that in one way or another the data needs to be modified for privacy preserving reasons by a party other than the SM or the SMGW.

4.3.1 Problem #1: Balancing Data Utility (incl. Integrity and Authenticity) and Privacy

We see one problem in the opposing goals: On the one side the SG stakeholder needs access to integrity protected values gathered by a trusted untampered SM. On the other side consumer requires some trusted privacy component to perform data perturbation to protect the consumer's privacy. The main point we would like to raise is that the entity trusted to generate data is controlled and trusted by the SG stakeholder. With its goals and incentives to gather fine-grained data, this entity is untrusted to maintain the consumer's privacy. Vice versa, the SG stakeholder will not be able to rely on data gathered by an untrusted consumer-controlled device. Figure 71 depicts this situation.

4.3.2 Problem #2: Judging and Comparing Privacy Invasiveness

There is no debate that certain applications of the smart grid will need more data than others. At the moment exact nature of such future smart grid applications is unsure, so is the required data utility. This section remains open towards future SG applications' need for data utility and future individual

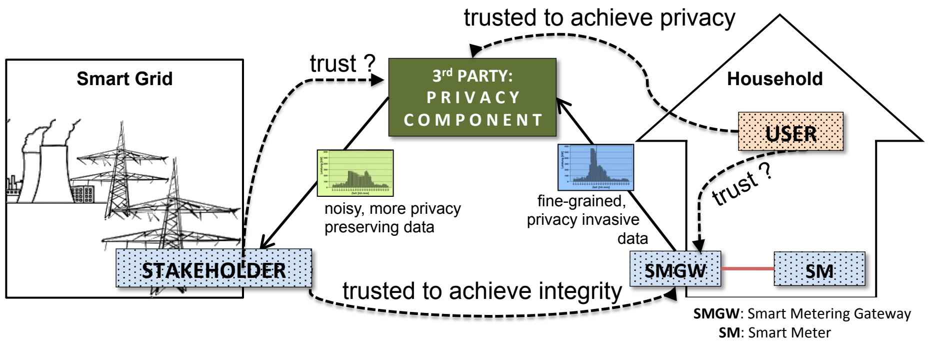

trusted to achieve privacy
trust ?
3rd PARTY: PRIVACY COMPONENT
Smart Grid
Household
USER
trust ?
noisy, more privacy preserving data
fine-grained, privacy invasive data
STAKEHOLDER
SMGW
SM
trusted to achieve integrity
SMGW: Smart Metering Gateway
SM: Smart Meter

Figure 71: Trust towards components by SG stakeholders and privacy-aware household

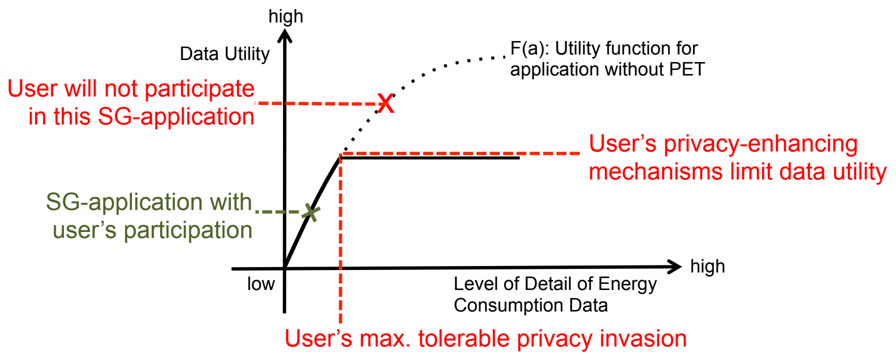


Figure 72: Data-utility might be hindered by PET, but consumers will participate in applications within consumer's privacy preferences

consumers' privacy-tolerances. We believe that with an informed choice the user's willingness to participate in SG-applications will increase and that SG-applications will hence respect consumer's privacy preferences. Figure 72 shows that participation in applications are possible, if they require a data quality that is below the consumer's privacy preference. Privacy preserving mechanisms or unwillingness to participate limit the maximum data utility.

4.3.3 Contribution

This section describes a technology that allows balancing the conflicting interests of privacy and integrity[14]. We follow an approach called data perturbation, which is widely used in the field of privacy preserving data mining and differential privacy [75]. Data perturbation based mechanisms preserve privacy of distinct customers by letting an entity tamper with the data. We will call this entity the *privacy gateway* (PGW). The drawbacks of data perturbation are twofold: First it obviously must result in a reduced data utility and second the data tampering entity must be trusted. The first is an inherent problem of PET whereas the impact on utility needs to be limited to a level where the application is still executable. We counter the latter by applying a redactable signature instead of a classical digital signature at the SMGW.

[14]which here includes accuracy

The contribution of this section is to provide a differential privacy guarantee in the BSI Smart Metering Setting (see Figure 74) and to control the amount of integrity violations needed to achieve the privacy: We achieve control, integrity protection and origin authentication for the SG stakeholder by letting the SMGW sign a *range of values* around actual energy consumption using a redactable signature scheme ($\mathcal{RSS}$). The residential customer's *privacy gateway* (PGW) still has the possibility 'tamper' with the data to increase privacy by choosing *one value* out of the signed range.

We gain all the advantages of data perturbation combined with those of redactable signatures:
(1) data perturbation still allowing the stakeholders to address customers individually allowing for applications like providing energy efficiency recommendations;
(2) data perturbation gives an ad omnia privacy guarantee of differential privacy with a small computational overhead;
(3) redactable signatures allow the verifier to gain reassurance that the SMGW actually signed this value. Hence, the signer limits allowed values according to maximum tolerable reduction of data utility;
(4) redactable signatures allow third parties to do the choosing without any interaction with the signer, hence the customer does not need to trust a third party like a Smart Metering Operator (SMO) or the Smart Metering Gateway Administrator to protect her privacy.

4.3.4 System Description and Integrity Requirements

The BSI proposed a technical guideline [38] for intelligent metering systems. While this technical guideline is controversial discussed in literature due to its broad as well as expensive security and its slim privacy concept [226], it allows for a controlled data communication between a household and SG stakeholders. The concept is depicted in Figure 73.

SMGW checks whether a requesting stakeholder like a Distribution System Operator (DSO) or a Demand Side Manager (DSM) are allowed to access values like energy consumption or to send commands to the Controllable Local Systems (CLS). SMGW communicates via the residential Home Area Network (HAN) with CLS. In Addition the SMGW provides over the HAN data for the end consumer as well as the service technician. Within the Local Metrological Network (LMN) SMs for electricity, heat, gas and water are installed. SMs communicate consumption values to SMGW via the LMN.

Stakeholders like the DSO can ask the SMGW to get consumption data. The time interval between the gathering may vary but in the UK a collection rate once every 15 minutes is discussed and considered to be sufficient to guarantee net stability. Even finer grained consumption values are advantageous for forecasting.

4.3.5 Privacy Threats

Service providers in the SG like DSO or DSM need to collect data from individual households for their services. This data allows to infer information about households. The general research focus for privacy incursion has been about energy consumption values which are considered the household's output channel. Note that research barely considers the other direction, the input channel to the household. Inferred information of energy consumption values can be structured in the following three categories: First, appliance detection, second, use mode detection, and third, behaviour detection. Note that all these attacks are possible for any party that has access to the plain data. Hence, encryption will help

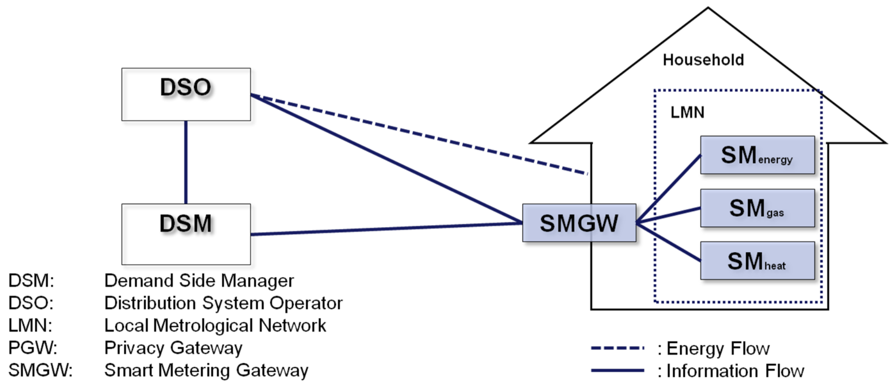


Figure 73: BSI System Structure

to protect the confidentiality during transmission of data, i.e., achieve privacy against third-parties, but will not mitigate privacy attacks by the party finally receiving and decrypting the plain data.

In the first category an analyser tries to find out which appliances run in a household site. This information can be used for advertising purposes. In the second category an analyser tries to find out how those devices are used. Experiments with high frequency data shows that even the TV channel can be deduced with a high percentage rate [105]. In the third category data is used to investigate how many people live in a household and what those people do. In [148] wake and sleep cycles as well as presence and absence have been deduced.

The information transmitted over the channel from SG service providers to the household bears privacy risks which depend on the application. Demand Response (DR) application allow to infer incentive sensitivity as well as a customer's preferences. In a simple version of DR the DSM ask the customer to reduce the amount of consumed energy in a certain time frame. In return the customer gets a financial compensation. To measure the compensation amount the DSM needs to know the energy consumption of this time as well as data to compare in order to determine the real reduction. This data can be the consumption from former periods. With this data and to know when the customer accepts and executes DR requests, the DSM can infer incentive sensitivity information of the customer.

To mitigate privacy threats appliance and use mode detection as well as behaviour deduction, several privacy enhancing technologies have been introduced. PET are based upon the principle of data minimisation and concealing. The main drawback of those techniques are that either customers can not be addressed individually or that fine granular data is not available.

4.3.6 Differential Privacy: Perturbation to protect Privacy

A different approach than data minimisation and concealing is the addition of noise to consumption data. While the outlook from the standpoint of privacy protection is very promising, the effect of the introduced error to data utility in SG is still in research. Data perturbation done in a right way, allows to reach the differential privacy ad omnia guarantee.

Let the function $k()$ be a "randomisation" or "perturbation" or "sanitisation" algorithm that takes a Database D and constructs a "sanitised" version $k(DB)$. The algorithm provides differential privacy if the nothing can be learned about an individual X that couldn't be learned from looking at the rest of the data-set, excepting X. More precisely, assuming that D_1 and D_2 are two Databases which differ in at most one individual (say, "X", as in the sentence above). Then the algorithm $k()$ provides differential-privacy if the two sanitised databases are so close that it is practically unfeasible to detect differences between them that could be attributed to the individual X. This can be formalised as follows (the definition is from [75]):

Definition 34. *Let k be a (randomising) sanitising algorithm and ϵ a positive number (that can be chosen arbitrarily, but a-priori fixed). Then we say that $k()$ provides ϵ-differential privacy iff for all databases D_1 and D_2 which differ in at most one element (that is: on the fact of one individual being present or not)), and for all $S \subseteq Range(k)$.*

$$\frac{Pr(k(D_1) \in S)}{Pr(k(D_2) \in S)} \leq e^{\epsilon}$$

where the probability space in each case is over the "coin flips" (or randomisation) of the mechanism $k()$.

As an instantiation of using k to achieve privacy consider a DSO asking SMGW for current consumption data. The SMGW is retrieving this information and uses a function k, that adds noise taken from a Laplace distribution.

Definition 35 (Sanitising Mechanism k)**.** *The Sanitising Mechanism k is : $k(D) = f(D) + \mathfrak{L}(\frac{\Delta(f)}{\epsilon})$. The mechanism is ϵ-differential private for all functions $f : D \rightarrow R^x$, where $\mathfrak{L}(\frac{\Delta(f)}{\epsilon})$ denotes the noise which is taken from the Laplace distribution, $\Delta f = max||f(D_1) - f(D_2)||$ and where D_1, D_2 differ in exactly one single dataset.*

Addition of noise as well as function f performed over the data base are done by a trusted entity, known as curator. In the SM case, the database needs to hold stored consumption values for specific points in time.

4.3.7 Redactable Signatures ($\mathcal{RSS}$): Fine control of Integrity

Assume the message to be signed is a set which contains ℓ values as elements: $\mathcal{M} = \{m_1, \ldots, m_\ell\}$. This section uses a set-like notation without loss of generality.[15] The fundamental difference to classic signatures is that a $\mathcal{RSS}$ allows anyone to *redact* an element from the signed list, such that the signature still verifies. Basically, a redacted list no longer contains all elements from $\mathcal{M}$. Assume $\mathcal{R} \subseteq \mathcal{M}$, than removing elements in $\mathcal{R}$ from $\mathcal{M}$ leaves a subset $\mathcal{M}' = \mathcal{M} \setminus \mathcal{R}$. The most important differentiator between a classical signature is that a redactable signature scheme allows deriving an adapted signature σ', which still verifies. This action is called *redaction* and can be performed by anyone; the secret signing key is not required. Hence the original signer is not involved. However, a secure $\mathcal{RSS}$ is unforgivable comparable to classic digital signature schemes; this ensures that each element $m_i \in \mathcal{M}$ is protected against modifications other than complete removal. To continue the example, assume you redact all the other $\ell - 1$ elements, leaving only one value m_i in the signed set: $\mathcal{M}' = \{m_i\}$. Due to the $\mathcal{RSS}$ you can adjust the signature to σ'. A positive consecutive verification of the signature σ' over $\mathcal{M}'$ means that

[15]Set-like notation eases understanding of the decomposition of a message as mathematical notions like intersection and union become applicable.

all elements in $\mathcal{M}'$ are authentic. In other words without use of the secret signing key you can produce a valid signature for remaining unchanged elements. Hence m_i that remained in $\mathcal{M}'$ can be verified to having not been altered and originating from the original signer, which remains identifiable via its public key.

4.3.8 Algortihmic Description of $\mathcal{RSS}$

The following notation is derived from [204], which is based of Brzuska et al. [32].

Definition 36 (Redactable Signature Schemes). *An $\mathcal{RSS}$ consists of four efficient algorithms $\mathcal{RSS} := (\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify}, \mathit{Redact})$:*

KeyGen. *The algorithm KeyGen outputs the public key* pk *and private key* sk *of the signer, where λ denotes the security parameter:*
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathit{KeyGen}(1^\lambda)$

Sign. *The algorithm Sign gets as input the secret key* sk *and the message $\mathcal{M} = \{m_1, \ldots, m_\ell\}$, $m_i \in \{0,1\}^*$: $(\mathcal{M}, \sigma) \leftarrow \mathit{Sign}(1^\lambda, \mathsf{sk}, \mathcal{M})$*

Verify. *The algorithm Verify outputs a decision $d \in \{\mathtt{true}, \mathtt{false}\}$, indicating the validity of the signature σ, w.r.t.* pk*, protecting $\mathcal{M} = \{m_1, \ldots, m_\ell\}$, $m_i \in \{0,1\}^*$: $d \leftarrow \mathit{Verify}(1^\lambda, \mathsf{pk}, \mathcal{M}, \sigma)$*

Redact. *The algorithm Redact takes as input the message $\mathcal{M} = \{m_1, \ldots, m_\ell\}$, $m_i \in \{0,1\}^*$, the public key* pk *of the signer, a valid signature σ and a set of elements $\mathcal{R}$ to be redacted. It returns a modified message $\mathcal{M}' \leftarrow \mathcal{M} \backslash \mathcal{R}$ (or $\perp$, indicating an error): $(\mathcal{M}', \sigma') \leftarrow \mathit{Redact}(1^\lambda, \mathsf{pk}, \mathcal{M}, \sigma, \mathcal{R})$*

We require the correctness properties for $\mathcal{RSS}$s to hold: Hence, every genuinely signed or redacted message will verify. A formal definition is given in [32].

4.3.9 Security of $\mathcal{RSS}$

This section describes the required security properties and models on an informal level, the formal properties are described and proven in [32, 33, 101, 204]. A secure $\mathcal{RSS}$ must be unforgeable and private to be meaningful [32]. Unforgeability allows detecting Integrity violations, e.g., only the genuine signed message or a valid redaction thereof can bear a valid signature created by the owner of the secret signing key. A public verification key linked to a a known entity and an unforgeable signature allows authentication of origin.

4.3.9.1 Unforgeability.

No one should be able to compute a valid signature on a message not previously issued without having access to any private keys [32].
This is analogous to the unforgeability requirement for standard signatures [98], except excluding all valid redactions from the set of forgeries.The attacker can generate genuinely signed messages using an oracle, but has no access to the secret key. He has breached unforgeability if and only if he is able to compute a signature on a 'fresh' message, which is valid under the corresponding public verification key

fixed at the beginning. A message is considered 'fresh' if it either has not previously queried from the oracle and if it can not have been created by one or more redaction(s) from a message queried from the oracle.

4.3.9.2 Privacy (weakly and a strongly)

A *private* $\mathcal{RSS}$ prevents everyone except the signer from recovering any information (esp. the original value) about elements redacted, given the redacted $\mathcal{M}'$ and a valid signature σ' over $\mathcal{M}'$.
Note that information leakage through the modified message itself is out of scope. A weakly private $\mathcal{RSS}$ allows a third party to derive that elements have been redacted without gathering more information about their contents. Assume that each redacted element's value being replaced with ■ remains a visible element of $\mathcal{M}'$ [109]. The definition of a strongly private $\mathcal{RSS}$ is very similar, but redacted elements are considered *not* being visible as elements of $\mathcal{M}'$.

4.3.10 Solution: Signing a range of values with an $\mathcal{RSS}$

4.3.10.1 Solution towards problem #1.

We allow the SMGW to provide the Smart Grid stakeholders like DSO and DSM with signed and henceforth trustable SM values, e.g., energy consumption values. At the same time, we allow the customer to achieve a desired level of privacy, by allowing the energy consumption value to be tampered with, e.g., adding noise. The party running PETs to achieve the consumer's privacy is termed Privacy Gateway (PGW). Our solution is depicted in Figure 74. We assume that all information between the SMGW and the DSO and the DSM are running over the curator termed 'Privacy Gateway' (PGW).

Note that it is the SG stakeholder who knows and requests a desired level of data utility. This means in case of perturbation by noise to limit the maximum allowed noise. Of course, the SMGW could run privacy preserving algorithms directly and add noise to keep the customer's differential privacy. However this solution would require that the residential customer trusts the SM operator (SMO) to protect her privacy. The same problems occurs if the PGW is placed before the SMGW and would directly tamper with the readings from the SM. However, our solution allows the party doing the addition of noise to be trusted to preserve the customer's privacy, as the customer remains in full control. The task of the PGW is to tamper energy consumption values in order to protect the privacy of residential customers. The task of the SMGW is to sign the energy consumption values and the maximum tolerable perturbation in order to protect the integrity and trustworthiness of the SM readings. Both parties act on behalf of different stakeholders and hence are in different trust zone. Our solution uses redactable signatures to solves this conflict.

4.3.10.2 Solution towards problem #2.

For brevity, we will now focus only on the transmission of a consumption value, other information that the SMGW sends alongside, like timestamps, are not considered.

The SMGW must make sure that values are not tampered in an unauthorised malicious way. Depending on the application DSO and DSM can tolerate a certain level of inaccuracy, e.g., allow that a certain amount of noise degrades their data utility. We denote the maximum amount of noise that can be added to an accurate reading by δ_{max}. Assuming SM measures the actual consumption value v DSO/DSM will

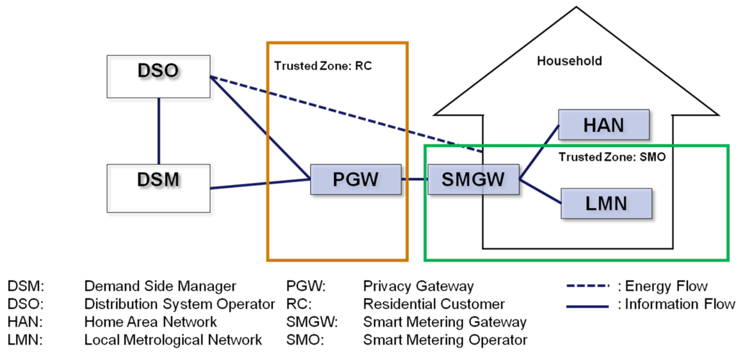


Figure 74: System Structure with PGW

accept any reading in the range $[v - \delta_{max}, v + \delta_{max}]$ as valid. If the SMGW applies a classical signature scheme on v PGW can not tamper with data signed by SMGW without invalidating the signature. An invalid signature would indicate towards the DSO/DSM that the received value is not trustworthy, as it could have been maliciously tampered with in an arbitrary way. Henceforth, we assume that the SMGW will be instructed by the SMGW's operator about the tolerable noise, on behalf of the SG stakeholder. The tolerable noise depends on the required accuracy level for SG stakeholder's application. The actual values depend on the DSO or DSM application needs.

Note that fixing $\Delta = 2\delta_{max}$ in definition 34 allows calculating the maximum differential privacy that can be achieved. The PGW must be instructed by the consumer which level of privacy is tolerable for which optional applications. In this section we assume that the consumer is free to not participate in an application for which his own personal privacy preference can not be achieved, i.e., PGW will not sent privacy-invasive data to a requesting SG stakeholder. However, we are fully aware that some communication must always be allowed for mandatory applications, e.g., net stability. For those mission critical mandatory SG applications we assume that the tolerable perturbation should be fixed by regulators.

4.3.11 Protocol Description

We propose the following phases: Setup, Signing, Adding Noise and Verification.

Setup:

1. Let $\mathcal{RSS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$ be a secure (unforgeable and weakly private) redactable signature scheme.
2. After running KeyGen distribute the keys: SMGW gets a secret signing key *sk* and verification key *vk*, PGW and DSO/DSM get just the public SMGW's verification key *vk*.
3. SMGW is instructed by SMO which amount of noise it tolerates, and which accuracy is required.

Signing:

1. On receiving the actual consumption value v the SMGW calculates a range of discrete noisy values $\mathcal{M} = \{v - \delta_{max}, \ldots, v, \ldots, v + \delta_{max}\}$.

2. SGM signs $\mathcal{M}$ with an $\mathcal{RSS}$: $(\mathcal{M}, \sigma) \leftarrow \text{Sign}(1^\lambda, \textit{sk}, \mathcal{M})$.
3. SMGW sends $(\mathcal{M}, \sigma)$ to PGW.

Adding Noise:

1. On receiving $(\mathcal{M}, \sigma)$ PGW uses its database of historic values and the actual consumption value, which must be at the centre of the range in $\mathcal{M}$, PGW runs the differential privacy algorithms to identify the value n in $\mathcal{M}$ which should be sent to DSO/DSM in order to satisfy $\frac{Pr(k(D_1)\in S)}{Pr(k(D_2)\in S)} \leq e^\epsilon$ where ϵ is a user predefined minimum required privacy parameter. The application execution is denied, if ϵ can not be reached.
2. PGW calculates $\mathcal{R} = \mathcal{M} \setminus n$.
3. PGW obtains a signature on $\mathcal{M}' = n$: $(\mathcal{M}', \sigma') \leftarrow \text{Redact}(1^\lambda, \textit{pk}, \mathcal{M}, \sigma, \mathcal{R})$.
4. PGW sends $(\{n\}, \sigma')$ to the DSO/DSM.

Verification:

1. On receiving $(\{n\}, \sigma')$, DSO/DSM uses the SMGW's verification key *vk* to verify if the signature on n is valid.

The amount of elements in $\mathcal{M}$ depends on the maximum noise and the accuracy, as $\mathcal{M}$ must contain concrete values, e.g., $\mathcal{M} = \{0.99, 1.00, 1.01, 1.02, 1.03, \ldots, 1.48, 1.49, 1.50, \ldots, 1.96, 1.97, 1.98, 1.99\}$ for an accuracy of two decimals, $\delta_{max} = 0.50$ and $v = 1.49$. The $\mathcal{RSS}$ limits the PGW only to redactions based on provided values, e.g., for $\mathcal{M} = \{1.11\}$. The PGW could generate a valid signature facilitating the algorithm Redact. However, the PGW can not generate valid signatures on values outside the range, e.g., $\mathcal{M} = \{0.98\}$ or $\mathcal{M} = \{2.00\}$. To do so would be as hard as forging the signature scheme of the $\mathcal{RSS}$, e.g., breaking the signature scheme like RSA-PSS [20, 198]. To counter replaying or repressing messages, the SMGW can just add a timestamp as an additional element into $\mathcal{M}$ requiring this to be fresh and present during verification.

4.3.12 Security and Privacy Properties

We assume: SM is trusted to perform correct readings, can not be attacked, and transmits the reading securely to SMGW.

Theorem 25. *Our protocol is unforgeable, if the $\mathcal{RSS}$ is unforgeable.*

SG stakeholders can detect any subsequent malicious manipulation of information while it is travelling through the network. Additionally they can use the SMGW's verification key to identify the origin of noisy data.

Theorem 26. *Our protocol achieves the highest differential privacy possible for $\Delta = 2\delta_{max}$, if the $\mathcal{RSS}$ is at least weakly private.*

4.3.12.1 Proof Intuition for Th.25

If the $\mathcal{RSS}$ applied by the SMGW is unforgeable, than neither PGW nor attackers can forge a valid signature on a value $n^* \notin \mathcal{M}_i$, where $\mathcal{M}_i$ denotes all sets signed and sent by the SMGW. Any such forgery would be a forgery in the $\mathcal{RSS}$.

4.3.12.2 Proof Intuition for Th.26

Assume all communication from SMGW will always pass through PGW, see Figure 74. The $\mathcal{RSS}$ allows PGW to be a separate entity acting as instructed by the residential customer. PGW is limited by the range defined within the SMGW's signature but can run the algorithm Redact to select any suitable value out of the range. So seeing a valid $(\mathcal{M}, \sigma)$, which verifies using Verify under the trusted public verification key of a SMGW, that no malicious modification has taken place. Privacy of the underlying $\mathcal{RSS}$ guarantees that attackers can not identify the actual value of removed elements. Hence attackers can not know the actual consumption. We distinguish two cases:
(1) If the $\mathcal{RSS}$ is strongly private, i.e., elements are completely removed during redaction, then the attacker sees a set $\mathcal{M}$ with exactly one element, i.e., $|\mathcal{M}| = 1$.
(2) If $\mathcal{RSS}$ is weakly private, i.e., original values are hidden behind a special symbol ($\blacksquare^r$), then the attacker sees a set $\mathcal{M}$ with exactly one element being an actual value and $2\delta_{max}$ symbols, i.e., $|\mathcal{M}| = 2\delta_{max} + 1$.
Hence, if $\mathcal{RSS}$ is weakly private attackers can infer δ_{max}. However, attackers do never learn the actual values of removed elements, nor their position because its a set. Using the differential privacy mechanism described in Sect. 4.3.6, PGW adds noise within the range guaranteeing a differential privacy of ϵ.

4.3.13 Related Work

Techniques like group signatures [125] are based on the idea to hide the identity of household within a group. This prevents to address customers individually and thus limits potential SG applications to provide energy efficiency recommendations [3]. Another approach applies modifications inside the customers power circuit, e.g., consuming additional or less power from the grid by using a re-chargeable battery [11]. The downside of this approach are sever costs of the battery purchase as well as the maintenance effort. Those types are not optimal, due to the loss of addressing customers individually or the very high costs.

The concept of $\mathcal{RSS}$ was introduced by *Steinfeld* et al. [218] as "content extraction signatures" and almost at the same time by *Johnson* et al. as "homomorphic signatures" [127]. From their initial work many RSS constructions emerged in the last years [51, 163, 164]. Extensions working on more complex structures, e.g., trees [32], have been proposed, but a set is enough for the solution discussed in this section. In [32] *Brzuska* et al. presented a formal security model. Note that according to this model many schemes are not secure, as they do not fulfil their notion of *Privacy* [32, 204]. Also note, that many schemes proposed are also only weakly private, i.e., one can see that a third party redacted something [109, 127, 163, 218, 235]. This generally gives more information to an outsider as already noted in [164]. In this section we will not require transparency, thus we leak the range of noise, but the actual values of redacted elements stay private.

Several works try to identify which privacy relevant information can be inferred by analysing energy consumption values [80, 148, 165]. it is shown that appliances, how the appliances are used and the behaviour of the residential customers can be deduced by the energy consumption values. DR Application data holds additionally information about the incentive sensitivity. PET have been developed to minimise the amount of information which is sent by the SM [125, 211]. To the best of our knowledge only pseudonymization is considered to be applied. The minimisation of information is either spatial or temporal [50, 125]. Temporal data minimisation techniques provide only gross granular data, while spatial based data minimisation do not allow to allocate energy consumption values to certain single households. While pseudonymisation allows to address single households, it is shown that this technique can be sidestepped by linkage attacks [124]. Data perturbation do not minimise data, but tamper it to protect privacy. The downside is the direct and severe impact on the data utility. This concept allows to obtain the differential privacy guarantee for consumption values [1, 75] as well as addressing customers individually.

4.3.14 Summary and conclusion

For any application of smart metering it is vital that the SG stakeholders receive *reliable* and trustworthy information. In this case *reliable* means that the SG stakeholder, e.g., a power grid provider, gets this information as (1) timely and as (2) accurate as needed for the SG application. The exact level of accuracy and timeliness will vary depending on the application itself, but also on the actual contractual, regulatory and installation setting, and is beyond the scope of this section. In our construction the SM operator (SMO) limits the range in which data perturbation, in our case the addition of noise, is considered acceptable by applying a redactable signature ($\mathcal{RSS}$) at the SMGW over a range of the SMO's choosing. Knowing the allowed level of accuracy allows the customer's privacy gateway (PGW) to calculate the differential privacy guarantee that it could achieve using the data perturbation mechanisms it could deploy. With this information the PGW can independently judge if the allowed perturbation is enough to keep a sophisticated level of privacy for the customer.

If not, it can withhold the information until the customer explicitly consents to this leaking of privacy relevant data. If the PGW has enough freedom it will adjust the data accordingly and forward it after the modification. A $\mathcal{RSS}$ allows this alteration of signed data and the SG stakeholder can verify if the change was within his defined limits.

Furthermore, user studies could help to show which loss of privacy is accepted by users and craft privacy endangerment statements depending on several ϵ, e.g., a traffic light system. Finally, we remark that current research barely considers the privacy impact of the input channel to the household.

4.4 Privacy Policy Enforcement Point

As explained in Section 3.2 RERUM contemplates the use of two different PEPs, the pPEP and the sPEP, each of them working independently, but at a high performance cost, because it implies an additional redirection of the request. Taking a look at the internal components of a PEP from Figure 75 from D2.3 [219], it can be seen that a PEP has basically two main parts: an Interceptor, responsible for intercepting requests and forwarding them, and an Authoriser, responsible for authorising (or rejecting) the operation.

To avoid intercepting and forwarding the message twice, the RERUM prototype will use a single interceptor but two instances of the authoriser, one for the Privacy Policies and another one for the Security Policies. And both the pPEP and the sPEP will run against their own Policy files. The following class diagram shows this:

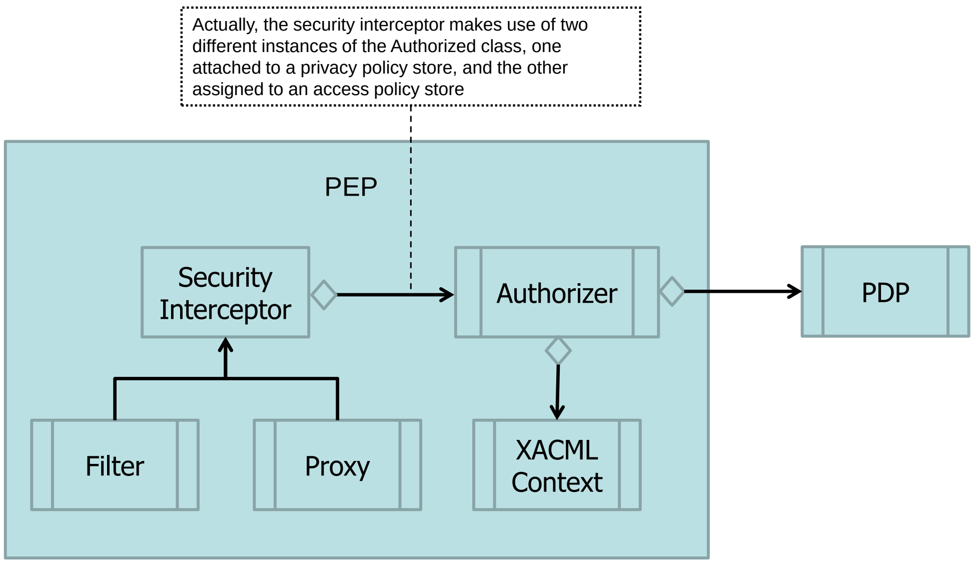

Actually, the security interceptor makes use of two different instances of the Authorized class, one attached to a privacy policy store, and the other assigned to an access policy store
PEP
Security Interceptor
Authorizer
PDP
Filter
Proxy
XACML Context

Figure 75: PEP implementing components

As Figure 75 shows, RERUM implementation has still a single PEP class with a single security interceptor, but this security interceptor is now running two different instances of the Authoriser, one for the sPEP and another one for the pPEP, each of them having their corresponding PDP and XACML context Though, in practice, both pPEP and sPEP will be instances of the same PEP object, there will still some differences with the previous version of the PEP. The first difference is the Policy Retrieval Point (PRP) that originally retrieved the applicable policies now has to deal with different policy repositories, depending on whether it is attached to a pPEP or an sPEP. The second difference is, as the sPEP and pPEP providers might not be necessarily the same ones, then the way they retrieve their respective policy files is conceptually different, even though in our prototype will be the same. For this reason, the PRP will now become an interface instead of a class. The actual PRP object will be created from a Factory class that will create the proper PRP according to the PEP configuration. Though for the RERUM prototype they will be the same class pointing to distinct policy stores, this will allow creating completely different PRPs in the future if needed.

The following diagram in Figure 76 shows the way sPEP and PRP work together

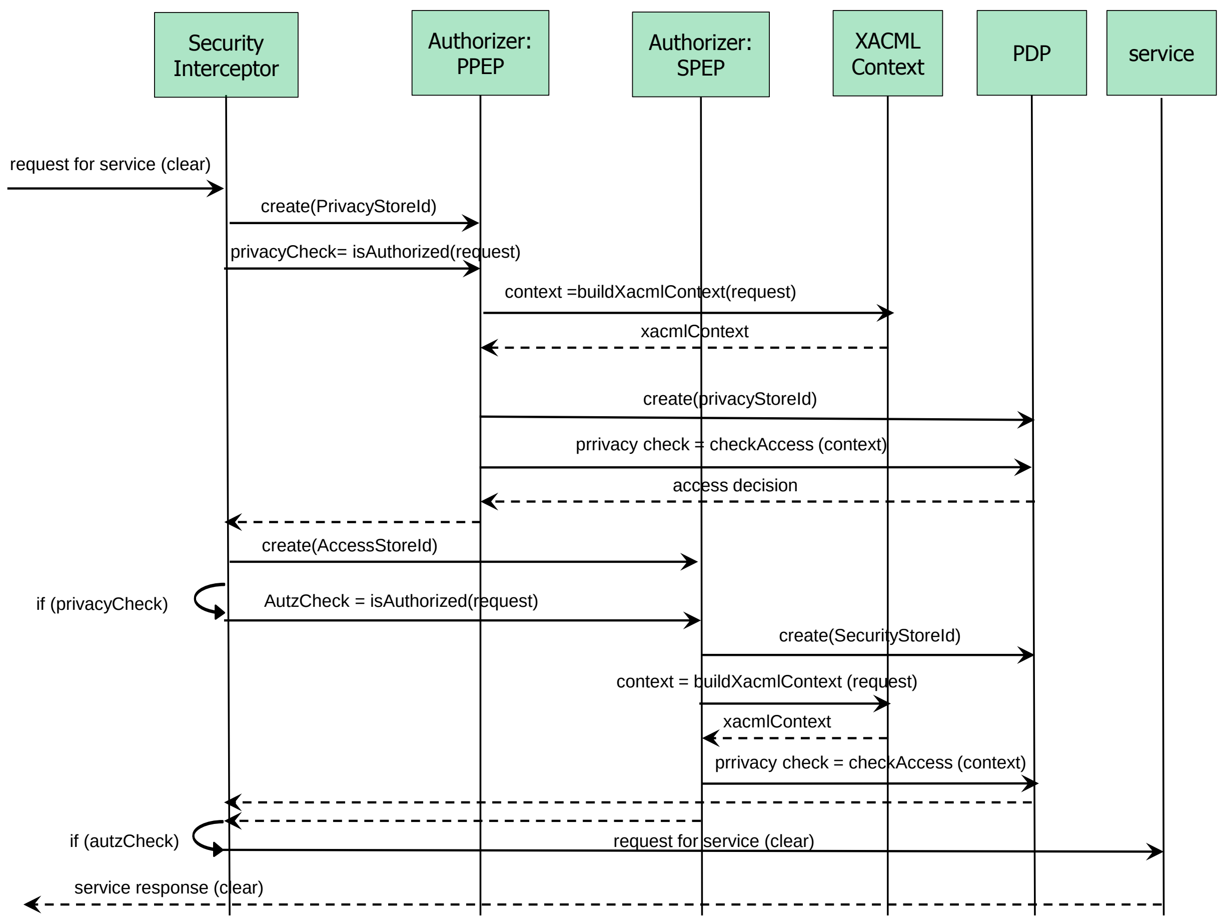

Security Interceptor
Authorizer: PPEP
Authorizer: SPEP
XACML Context
PDP
service
request for service (clear)
create(PrivacyStoreId)
privacyCheck= isAuthorized(request)
context =buildXacmlContext(request)
xacmlContext
create(privacyStoreId)
prrivacy check = checkAccess (context)
access decision
create(AccessStoreId)
if (privacyCheck)
AutzCheck = isAuthorized(request)
create(SecurityStoreId)
context = buildXacmlContext (request)
xacmlContext
prrivacy check = checkAccess (context)
if (autzCheck)
request for service (clear)
service response (clear)

Figure 76: Enforcing privacy and access authorisation

The Security Interceptor, which is part of the PEP, creates internally two different Authorisers, one for the privacy and another one for the access policies, passing a different id for the policy stores, on for the privacy and another one for the access policy stores. This way, this parameter will be available for the PRP when evaluating the policies. Finally, Figure 76 also shows how the evaluation of the access policies and the redirection to the service is only carried out if the privacy checks are passed first. The following Figure 77 shows how the PDPs and the PRPs work together with different policy stores to produce the evaluation of either the Privacy or access policy files:

The sequence diagrams presented in Figure 77 show the process of creating the different PRPs.

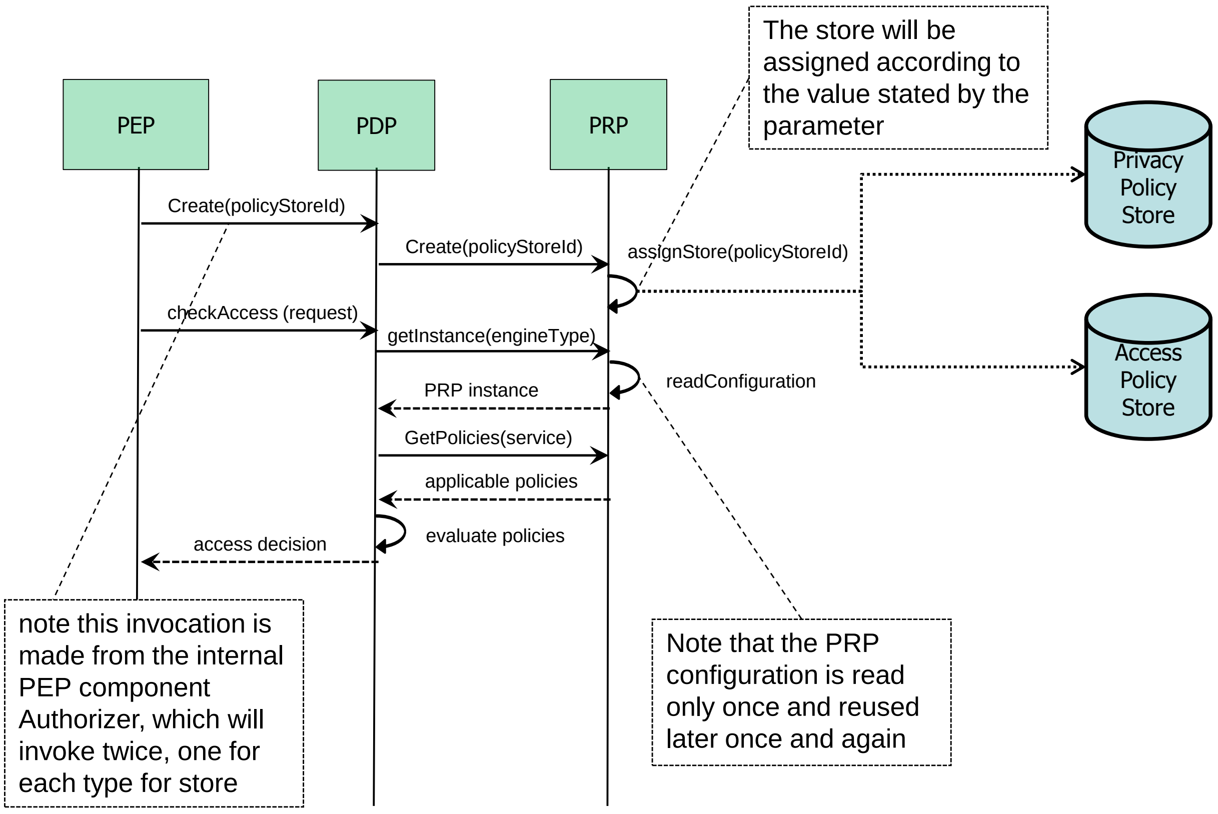

The store will be assigned according to the value stated by the parameter
PEP
PDP
PRP
Privacy Policy Store
Access Policy Store
Create(policyStoreId)
Create(policyStoreId)
assignStore(policyStoreId)
checkAccess (request)
getInstance(engineType)
PRP instance
readConfiguration
GetPolicies(service)
applicable policies
evaluate policies
access decision
note this invocation is made from the internal PEP component Authorizer, which will invoke twice, one for each type for store
Note that the PRP configuration is read only once and reused later once and again

Figure 77: Interaction among PDP, PRP and policy stores

4.4.1 Deploying of privacy policy files

Once the policy files have been generated, they must be deployed on the policy files store so the proper PEP can load and use them for deciding whether to grant access or not to a concrete datum or data-source.

In RERUM, the deployment of a privacy file is designed so it is independent from the generation of the policies themselves. Note that the deployment is not only a matter of moving the file to a concrete location. In RERUM, depending on the type of policy deployed, it can imply checking the existing policies to check if a given policy can no longer be executed or asking for new user attributes in the authentication phase.

As explained in Section 3.2, privacy policies are evaluated on a similar way as access policies, but with the main difference that they are stored in a different repository. For this reason, they are deployed in a similar way as the already presented in D2.3 for access policies, that is, they use the Policy Deployer component, but with an important update. Now the Policy Deployer requires an additional parameter, type, which allows the Policy Deployer knowing what is the type of policy that it has to deploy and so it can deploy in the proper repository and make any additional treatment related with it.

The following Figure 78 shows the process and the classes involved in it.

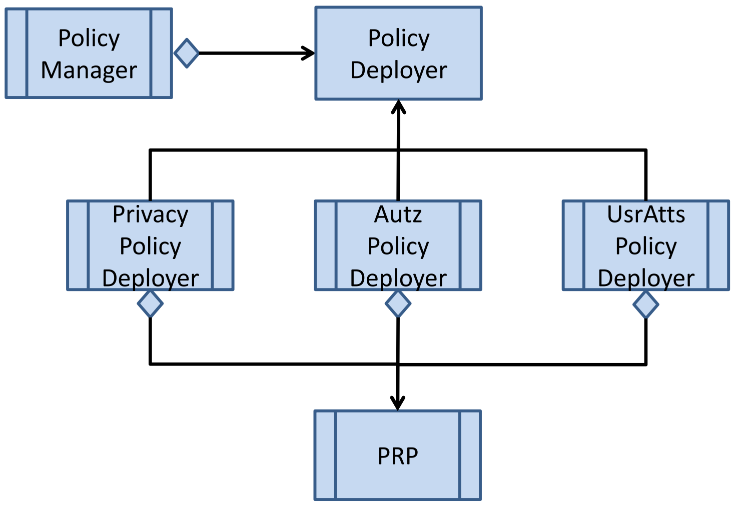


Figure 78: Classes involved in Policy Deployment

As it shows, the Policy Manager uses internally a Policy Deployer, which is built according to the type of deployment to be carried out each time. Each kind of Policy Deployer additionally makes use a of a PRP object to communicate with it and let it do all additional work required.

The following Figures 79 and 80 show how each type of policy is stored in the system, using an appropriate Policy Deployer depending on the type of the policy provided. For privacy policies related to RERUM services this is shown in Figure 79. Figure 80 shows this for privacy files used for accessing user attributes in the authorisation process.

The previously mentioned involved processes have to do with the interaction between the PPR and the Policy Deployer. Actually, deploying a policy is not only a matter of deciding the access logic to a resource or data. This logic if often based on information that needs to be gathered first and, especially in the case of the user attributes, this information is subject to consent as well. Hence, removing consent on some data may result on some features (or all) of RERUM becoming not accessible. In such case, it is desirable to raise a warning regarding this. This warning would be raised from the PRP and propagated to the invoking class (either the Policy Manager or the Consent Manager) in case the interaction with the PRP caused it.

4.4.2 Summary

The authorisation components defined in D3.1 [201] are reused and upgraded in this section to support privacy policies. More specifically, the PEPs are refined into two more specific PEP: a pPEP and a sPEP.

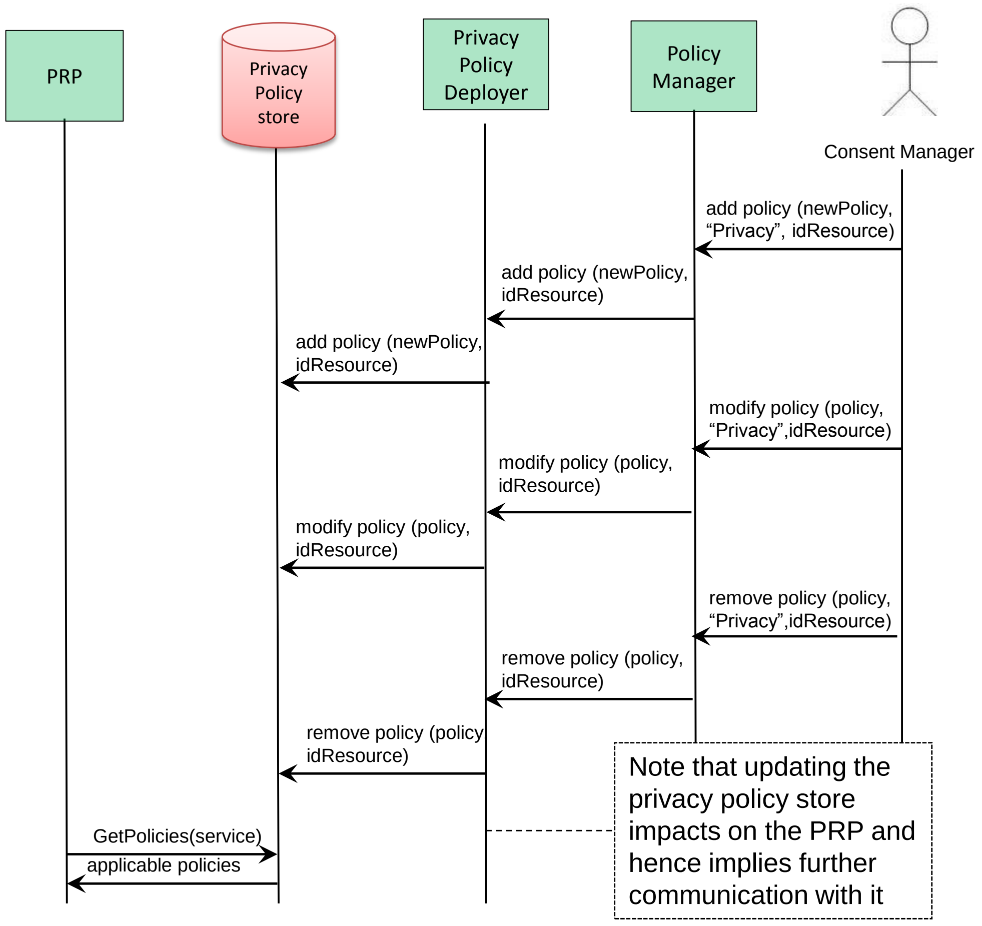

PRP
Privacy Policy store
Privacy Policy Deployer
Policy Manager
Consent Manager
add policy (newPolicy, "Privacy", idResource)
add policy (newPolicy, idResource)
add policy (newPolicy, idResource)
modify policy (policy, "Privacy",idResource)
modify policy (policy, idResource)
modify policy (policy, idResource)
remove policy (policy, "Privacy",idResource)
remove policy (policy, idResource)
remove policy (policy idResource)
GetPolicies(service)
applicable policies
Note that updating the privacy policy store impacts on the PRP and hence implies further communication with it

Figure 79: Deploying privacy policies

The difference between these two PEPs is that they work with a different policy store, one for access policies and another one for privacy policies.

Besides, the PDP and the PRP are additionally upgraded to be able to combine multiple policies, which allows for supporting policies at both local and global policies.

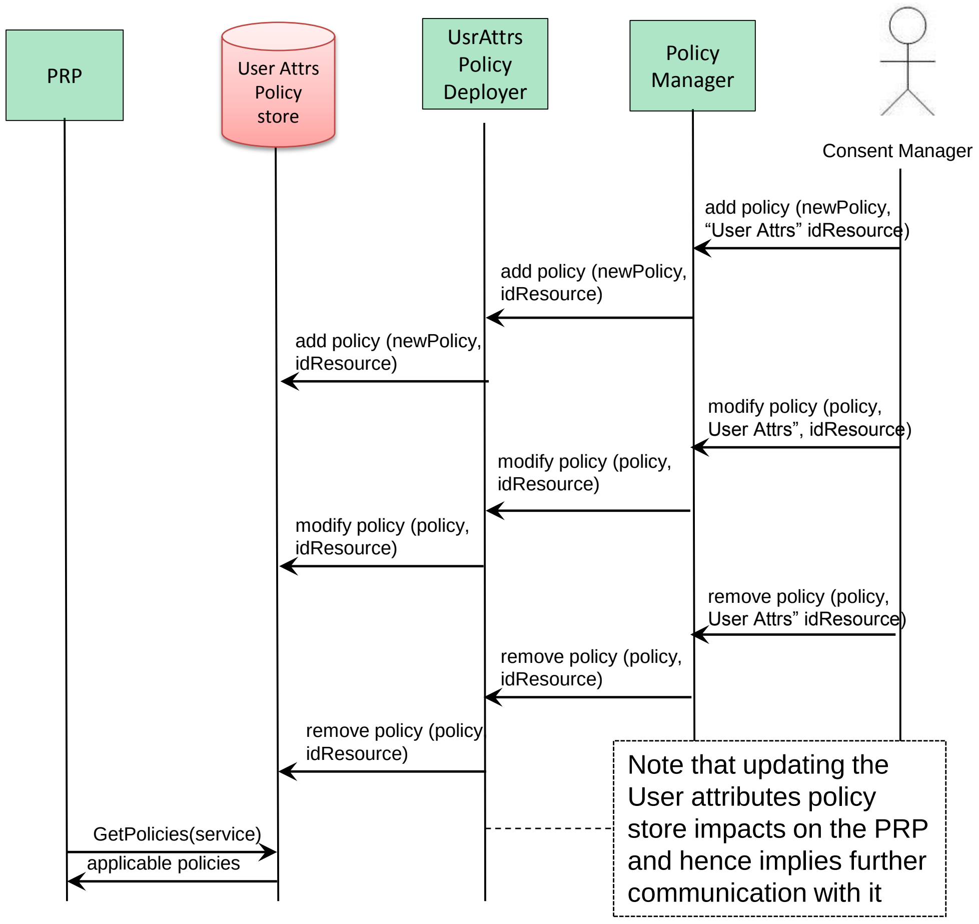

PRP
User Attrs Policy store
UsrAttrs Policy Deployer
Policy Manager
Consent Manager
add policy (newPolicy, "User Attrs" idResource)
add policy (newPolicy, idResource)
add policy (newPolicy, idResource)
modify policy (policy, User Attrs", idResource)
modify policy (policy, idResource)
modify policy (policy, idResource)
remove policy (policy, User Attrs" idResource)
remove policy (policy, idResource)
remove policy (policy idResource)
GetPolicies(service)
applicable policies
Note that updating the User attributes policy store impacts on the PRP and hence implies further communication with it

Figure 80: Deploying policies for accessing user attributes

4.5 Privacy Policy Checker and Attribute Need Reporter

As explained in Section 3.10, RERUM provides two new components, PPC and ANR that work jointly to enrich the security Agent. The ANR selects a initial list of user attributes that the Security Agent need to request to the Identity Provider, and the PPC checks that the Security Agent actually has permissions to do this operation for each attribute included in that list. This subsection detail how each of them work together.

4.5.1 Capabilities of privacy policies for authorisation

Before explaining how the PPC will work with the Identity Agent, it is important to explain what are the criteria that can be covered by the Privacy Policies for user attributes and what cannot be supported:

- Strictly speaking, user attributes may not necessarily be accessed only for authorising requests. It is perfectly legitimate to access these attributes for other purposes. But in the case of the authorisation phase and more especially the Identity Agent (whose purpose is to collect these attributes for the Authorisation components) the only purpose that matters is 'Authorisation'. This does not mean that RERUM will not support other Privacy Policies for other purposes, but their processing is a responsibility of the pPEP. Hence, the PPC will only work with policies whose purpose is assigned to 'Authorisation';
- Privacy criteria are often based on the identity of the RERUM registered user (Data Controller) trying to access the data. For this reason, it is legitimate that those criteria take into account that. However, the goal of this privacy policies for authorisation are to additionally check that it is possible to check any individual field. In other words, the privacy policy for the user attributes cannot be based in others user attributes, except possibly the user-id, because that is precisely what is trying to be retrieved. Because of that, the only criterion allowed to be included in the policy for checking the identity of the RERUM registered user is its user-id provided when trying to authenticate to the system and the corresponding purpose
- These privacy policies must not reference anything that may depend on the requested RERUM services, because these policies will be evaluated only when each RERUM registered user log in the system.

The reasons for these limitations is the very conceptual reason for these policies. These policies are meant to check the access to each individual user attribute before accessing them. Hence, with the exception of the attribute 'user-id', there is no point in basing the decision in the values of the user attributes, because it is precisely the access to these attributes what these concrete privacy policies are for. For instance, if the Identity Agent needs to retrieve the user attribute 'role', there is no point in the Privacy Policy on checking for a provided value of this field, because that is exactly what the Identity Agent is trying to retrieve. And even for the other fields, they have not been requested yet. Note in the concrete case of the user-id, it does not need to be requested to the identity provider, but it has been provided to the Identity Agent when it was invoked, because the Identity agent is meant, among the other things, to be the one to invoke the Identity Provider with that user-id to get the user authenticated.

4.5.2 Interaction among the ANR, the policy deployers and the PPC

Basically, when the set of security policies is changed in the system via a create, remove or update operation in the Authorisation Policies Manager (see section 6.8.1.5 Authorisation Policies Manager of D2.3), the Privacy Policy Deployer calculates how this new policy will impact in the set of needed user attributes and update a configuration file of the Identity Agent accordingly. The sequence-diagram in Figure 81 shows this process.

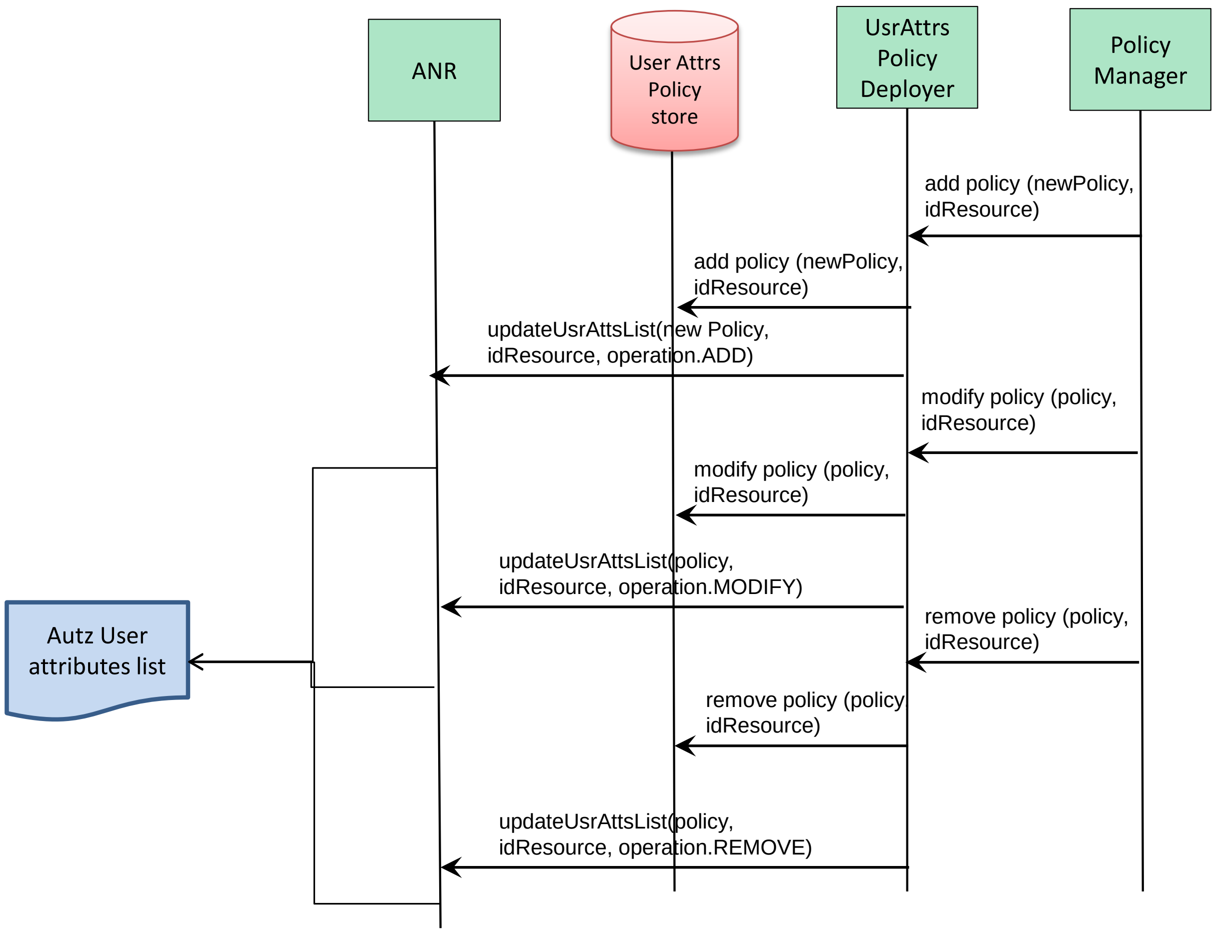

ANR
User Attrs Policy store
UsrAttrs Policy Deployer
Policy Manager
add policy (newPolicy, idResource)
add policy (newPolicy, idResource)
updateUsrAttsList(new Policy, idResource, operation.ADD)
modify policy (policy, idResource)
modify policy (policy, idResource)
updateUsrAttsList(policy, idResource, operation.MODIFY)
remove policy (policy, idResource)
remove policy (policy idResource)
updateUsrAttsList(policy, idResource, operation.REMOVE)
Autz User attributes list

Figure 81: Interaction Policy Deployer with ANR

As it shows, any change in the authorisation policies causes the ANR to recalculate the initial list of attributes that will be provided to the Identity Agent to be collected.

Whenever the Identity Agent needs to authenticate a RERUM registered user (and collect its user attributes for the authorisation), it reads that file for starting with the initial subset of attributes required. For each of these attributes, it invokes the PPC to check whether it is allowed to access them. In case any of them are rejected by the PPC because the Privacy Policies ban it, it marks it in internally as "rejected". Finally, the Identity Agent will ask the Identity Provider only for the fields that are not marked as rejected, setting as value for the rest to a constant whose value is "UNABLE TO RETRIEVE DUE TO PRIVACY

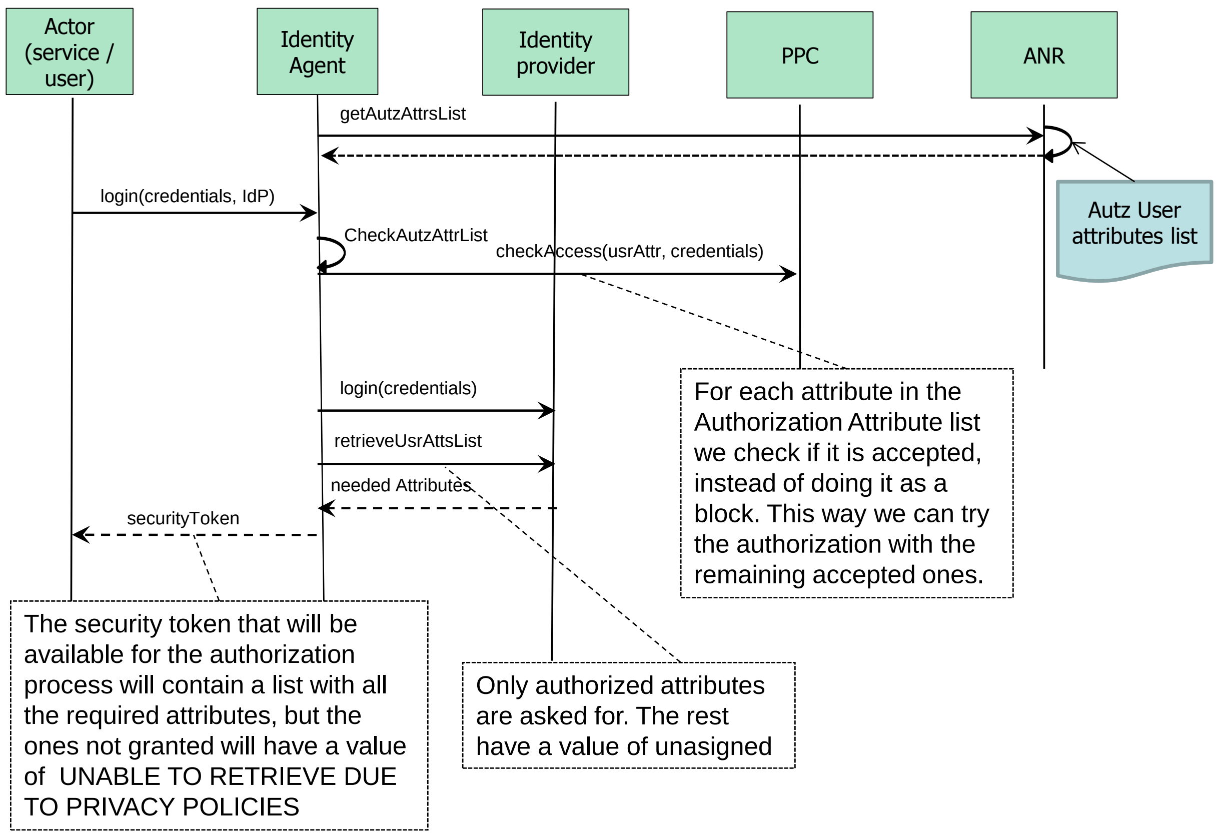

Actor (service / user)
Identity Agent
Identity provider
PPC
ANR
getAutzAttrsList
Autz User attributes list
login(credentials, IdP)
CheckAutzAttrList
checkAccess(usrAttr, credentials)
login(credentials)
retrieveUsrAttsList
needed Attributes
securityToken
For each attribute in the Authorization Attribute list we check if it is accepted, instead of doing it as a block. This way we can try the authorization with the remaining accepted ones.
The security token that will be available for the authorization process will contain a list with all the required attributes, but the ones not granted will have a value of UNABLE TO RETRIEVE DUE TO PRIVACY POLICIES
Only authorized attributes are asked for. The rest have a value of unasigned

Figure 82: Interaction among the PPC, the ANR and the Identity Agent

POLICIES". The sequence diagram in Figure 82 shows the interaction between the ANR, the PPC and the Identity Agent:

It could be argued that actually it should be possible to check the consent for all the user attributes required in a single operation. Indeed, that would be possible, but it would have the cost of not being able to know what values are still available and hence reject any incoming requests. This way, all granted values will be available for the security policies, thus allowing the RERUM registered user to access those services that can be accessed even without knowing the rejected attributes.

The evaluation of the Privacy Policies is a more complex issue because it requires a complete XACML evaluation process and a different subset of policies. For doing it, RERUM reuse again our authorisation PEP, or more exactly, a concrete part of it: The authoriser.

In this concrete case, the PRP of the PDP used will work with its own subset of files, which will be provided by the Privacy Policies Deployer, which is further detailed in Section 3.10. The following class and Sequence diagrams show the classes implicated in this process and their interactions.

It is not a coincidence that the PPC classes are very similar to the one of the PEP. As mentioned, the PPC makes use of the Authoriser of the PEP by simply referring to a different policy store, which in this case, is the user attributes policy store.

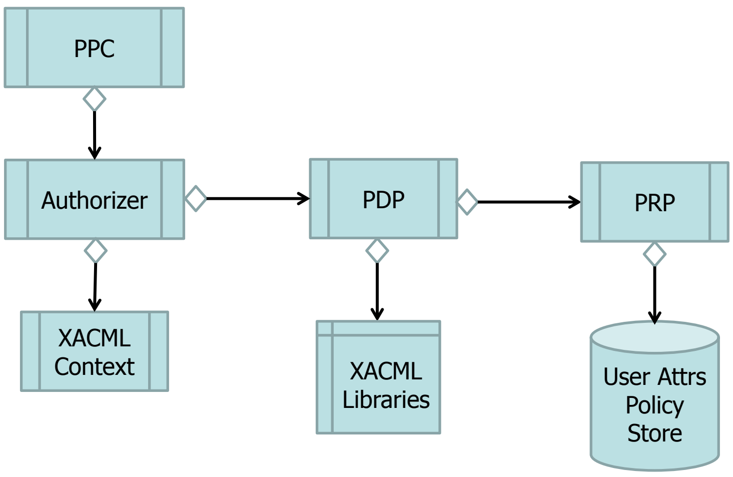


Figure 83: PPC Classes

4.5.3 Consequences for withdrawing access to user attributes

It is feasible that a RERUM registered user that previously gave her consent to grant access on their user attributes to use the application later regrets that decision and decides to revoke the consent to all or part of the attributes he originally granted. That is a legitimate decision, but it has its implications. RERUM should ask for permission for the needed attributes and the right of the RERUM registered user on using the system should be considered to depend on that consent. If the RERUM registered user revokes the consent, the access to the application should be restricted accordingly.

But what happens when any data subject tries to access RERUM services with his valid RERUM registered user whose attributes have been partially restricted to RERUM by the human being that they refer to? The answer is: It depends on what are the attributes restricted and whether they intervene in the concrete approval for each resource. For instance, if the RERUM registered user has two attributes named 'role' and 'age' and bans access to age but not to role, the system will act normally for those resources whose access criteria are based on the role but not on the age, but the access should be rejected to those policies that take into account the age.

4.5.4 Summary

This section explained how the components PPC and ANR work jointly with IdA to provide it the ability to check the privacy of the user attributes referenced in the policies of the system. In short, the steps are:

- Any entity (Administrator user, Consent Manager) invokes the Policy Manager to add, change or modify a policy (security or privacy) in the system;
- The Policy Manager invokes the proper policy deployer to deploy the policy in a suitable policy store and the ANR to recalculate the list of needed attributes to be stored in an intermediate file;
- The IdA invokes the ANR to retrieve the list of needed attributed from the intermediate file and uses it as a starting point for each RERUM registered user that is starting the session;

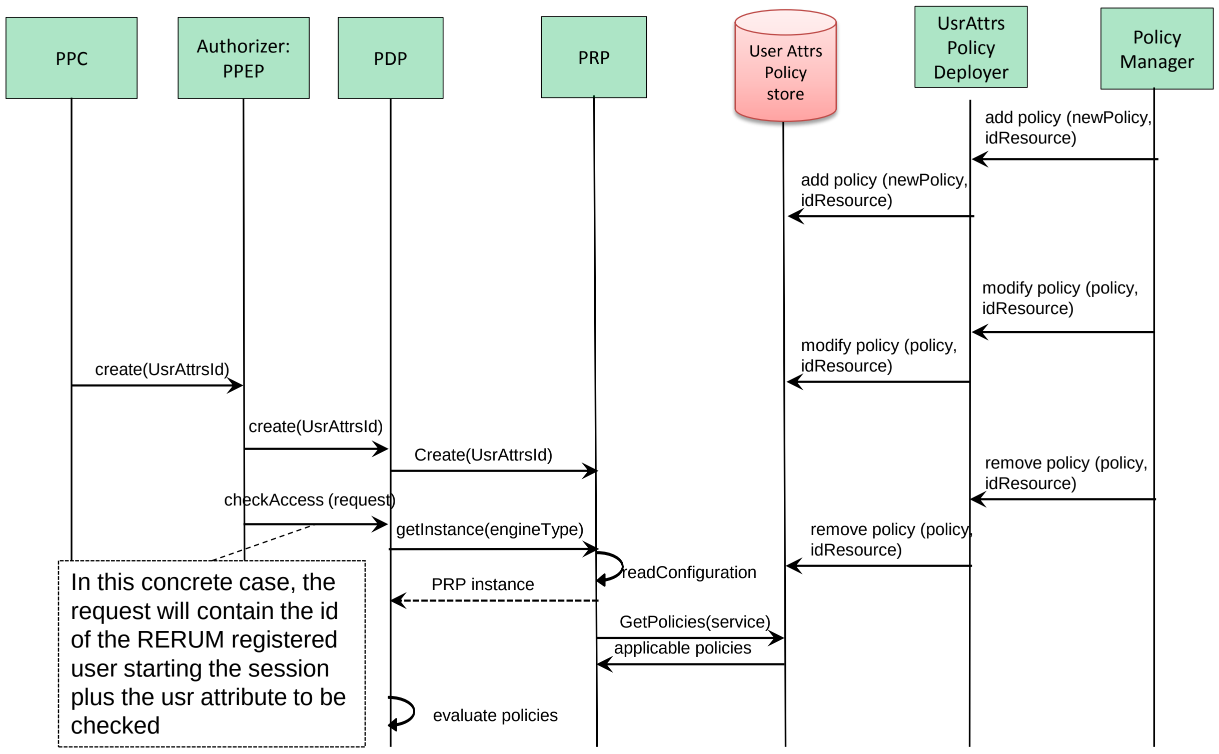

PPC
Authorizer: PPEP
PDP
PRP
User Attrs Policy store
UsrAttrs Policy Deployer
Policy Manager
add policy (newPolicy, idResource)
add policy (newPolicy, idResource)
modify policy (policy, idResource)
modify policy (policy, idResource)
create(UsrAttrsId)
create(UsrAttrsId)
Create(UsrAttrsId)
remove policy (policy, idResource)
checkAccess (request)
getInstance(engineType)
remove policy (policy, idResource)
readConfiguration
PRP instance
GetPolicies(service)
applicable policies
evaluate policies
In this concrete case, the request will contain the id of the RERUM registered user starting the session plus the usr attribute to be checked

Figure 84: Interaction of the PPC with the user attributes deployer

- Whenever a RERUM registered user starts a session, the IdA creates internally a copy of the list of needed attributes and checks the privacy of each of them using the PPC, obtaining a filtered list of needed attributes and
- The IdA takes the filtered list and request the identity provider for those attributes.

4.6 Pseudonyms

The presented pseudonym generation and management mechanism is based on Hash-Trees, similar to those found in the Merkle-Signature-Scheme (see [158]). But in contrast to Merkle-Trees, we propose a top-down approach, which allows generation of practically infinite hash values which we use as pseudonyms.

4.6.1 Existing fundamentals

In the following section we round up existing fundamentals for the creation of top-down hash trees.

One-Way Functions An one-way function is a function *f()*, which takes *x* as an input and computes *y* as an output. Computing *y* as an output is hereby easy, while computing *x* from *y* and *f()* is practically impossible.

Hash Functions A hash-function is a special type of an one-way function *h()*, which takes the input set *X* containing binary coded elements of any length, and produces an output set *Y* of binary coded elements with a certain length n, where following properties apply [169]:

- One-way or non-invertable property: It is virtually impossible to compute $x \in X$ from $y \in Y$ and the hash-function $h()$, where $h(x) = y$.
- Collision resistance: It is very unlikely to find two (or more) inputs $x1, x2 \in X$, where $h(x1) = y$ and $h(x2) = y$.
- Chaos: Even similar inputs generate significantly different outputs. Changing an input by one bit should generate and output that is about 50% different than the output of the unchanged input.

Keyed-Hash Message Authentication Code (HMAC)
We use Keyed-Hash Message Authentication Codes, defined in RFC2104 [137], NIST FIPS 198 [170] and RFC 4868 [132], to describe this technology. HMAC has been designed to have a well understood cryptographic application of hash functions and shared secret material, based on reasonable assumptions on the underlying hash function, see [137]. It should be noted that this is not the only method of how to use hash functions with shares secrets and that the selected hash function method is irrelevant for the rest of the approach.

A keyed-hash message authentication code (HMAC) is a specific construction for calculating a message authentication code (MAC) involving a hash function in combination with a secret key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authentication of a message. Any hash function may be used in the calculation of an HMAC. In RFC2104 [137], an HMAC is calculated the following way:

- $HMAC(K, m) = H(<(K \oplus opad), H(<(K \oplus ipad), m>)>)$

- Where H is a cryptographic hash function. Cryptographic means here, that the function has the properties described in A.2.).

- K is a secret key padded to the right with extra zeros to the input block size of the hash function, or it is the hash of the original key, if it's longer the original key is longer than that block size of the hash function.
- m is the message to be authenticated.
- $<, >$ denotes concatenation.
- $\oplus$ denotes the XOR operation.
- $opad$ is the outer padding (0x5c5c5c…5c5c, one-block-long hexadecimal constant). If K is smaller than the block-size used by the hash-function, this padding extends the key to that length.
- $ipad$ is the inner padding (0x363636…3636, one-block-long hexadecimal constant). This works the same as the opad, but with a different value.

4.6.2 Virtually unbounded generation of values

We propose to use the output y_i from hash-functions as pseudonyms: Due to the one-way property, it is practically impossible to invert *x* from the publicly known pseudonym y and the used hash-function *h()*. Due to the chaos property, it is possible to compute two pseudonyms from a slightly different value *x* and use this outputs again for the generation of other pseudonyms, which allows generating virtually unlimited pseudonyms from one initial value. The generation and coordination of values is based on aforementioned top-down hash-trees:

Figure 85 illustrates the steps needed to create a hash-tree. An initial input x is represented as a binary sequence. It is the seed for the generation of all other values. How the initial input x is obtained, can be very different. It might be from an authenticated diffie-hellman-exchange, a hashed-password known to two or more parties, etc. This is irrelevant for the rest of the approach. The input x is concatenated with one additional bit, "0" and "1", respectively, and given to the hash-function h(). The outcome is two outputs x_0 and x_1 with length n (depending on the hash-function), which in turn are going to be used as inputs for the next branches. The used hash-function and the generated lengths for the outputs x_i can vary; every hash-function with the properties described above (non-invertable, collision resistant, chaotic) can be used for this approach. In the next step, x_0 and x_1 are again concatenated with one additional bit, "0" and "1", respectively. They are used as inputs for the hash-function h(), which again generate two outputs each, namely (x_{00}, x_{01} and x_{10}, x_{11}). By repeating this step several times, a virtual infinite hash-tree can be build. Note: Figure 85 reuses the notion introduced in the explanation of HMACs, where $<, >$ denotes concatenation and $h(< x_i, 0 >)$ denotes, that the concatenated input of x_i with "0" is given to the function h().

4.6.3 Choosing adequate pseudonyms from the hash-tree

As noted above, due to the one-way property of hash-functions, outputs could be used as publicly known pseudonyms, without revealing the input from which they were generated. Once an output is publicly known, it does not qualify as an input for the generation of other pseudonyms. Thus, a path has to be

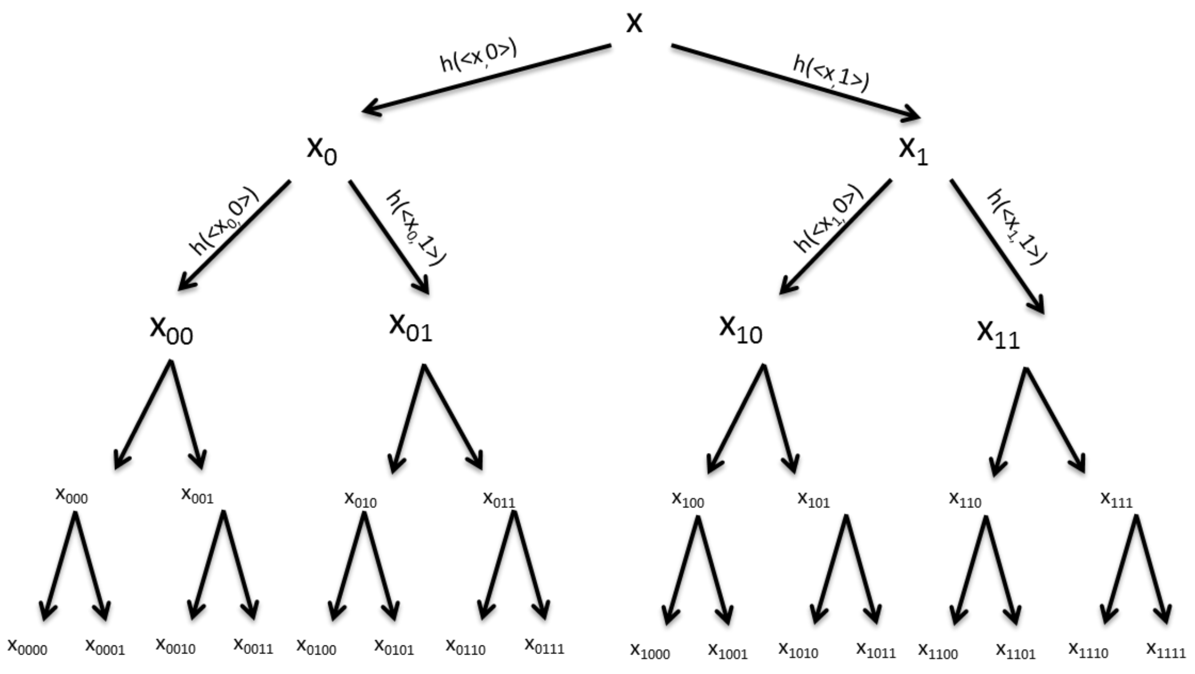


Figure 85: Hash-Tree with an initial input x

chosen, which allows using outputs as new pseudonyms and at the same time allows generating new branches of pseudonyms nonetheless. We propose, as one of many, following path:

- Step 1 – The first, initial value x is used to generate the first two levels of the tree. The first usable pseudonyms are those in the second level, generated by concatenating zeroes, namely ψ_{00} and ψ_{10}. An entity ”A” could now identify itself as ψ_{00} towards a second entity and again as ψ_{10} towards a third entity, instead of using ”A”. These values may not be used to generate further pseudonyms, that means, that the potential branches beneath them may not be calculated, see Figure 86. For the next round of secrets, the parties prepare to “jump” leaves:
- Step 2 – The next input will be the sibling leaf of the last used pseudonym. Assume that ψ_{00} was the last pseudonym, which means that ψ_{01} will be used to generate the next round of outputs. ψ_{01} is now concatenated again with “0” and “1”, respectively. The hash-function computes now two new values, namely the leaves ψ_{010} and ψ_{011}. We use again the output which was generated by concatenating a zero as the new pseudonym, namely ψ_{010}.
- Steps 1 and 2 repeat every time a pseudonym changes. We call these steps the canonical jump.

4.6.4 Definition of path and jump

A path is a bit-string that describes how branches from a hash-tree were (or how they should be) created. Paths generate downward branches by creating descendants of a certain starting leaf. For example, a path 00010 denotes that a hash-tree is generated by following the description of Section 4.6.2 until the leaf ψ_{00010} is reached.

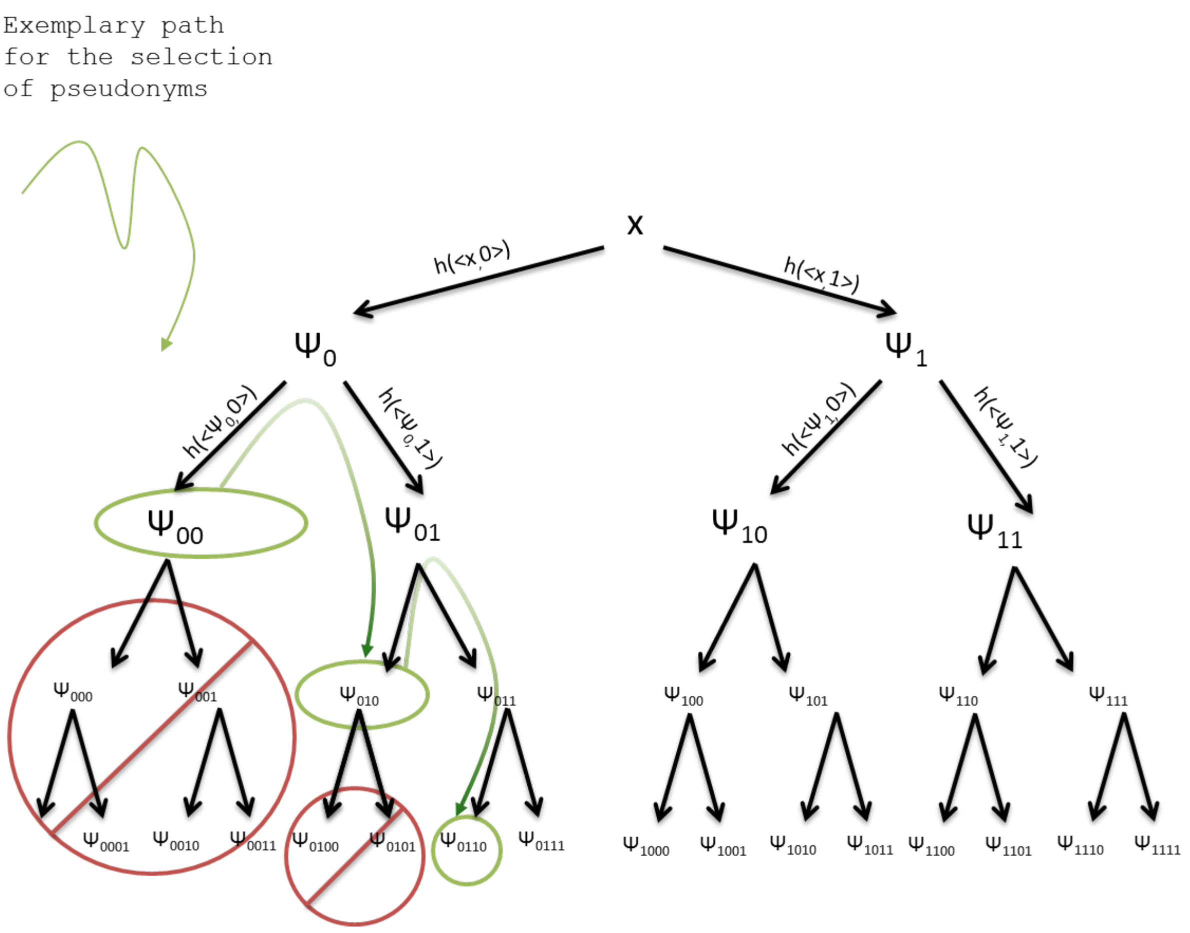


Figure 86: Selection of adequate outputs as pseudonyms

A jump is a form of path, which combines a bit-string with moving directions. A jump firstly moves up from its current leaf (one or several leaves) and then generates or traverses a different branch downwards. The canonical jump for example moves one leaf up, generates the opposing leaf and its left descendant.

4.6.5 Optimization

The canonical leap is just a suggestion to help in choosing adequate leafs as pseudonyms. Another suggestion is the dynamical generation of branches: The hash-tree is not generated entirely, but every branch is generated on demand, after a pseudonym was used. This is done by saving four variables, the root value x, the current input ψ_{i-1} and the current pseudonym ψ_i.

4.6.6 Changing pseudonyms

Generating new pseudonyms is done with the canonical leap. The mechanism is based on hash-functions which are easy and fast computational mechanisms which are very well suited for constrained IoT devices. The question in focus when discussing changing pseudonyms is when to generate new ones.

Pseudonym exchange has been heavily surveyed in vehicular ad-hoc networks, but the results can be transferred to any other system using pseudonyms. Important secure pseudonym exchanging concepts can categorized in spatial concepts, time-related concepts and user-oriented concepts. Spatial concepts are best represented in mix-zones [23], where pseudonyms are exchanged when system participants meet physically, although virtual mix-zones for an artificial pseudonym change have been proposed [151]. Time-related mechanisms propose to change pseudonyms after a certain time, where a secure pseudonym exchange is only possible when the changing participant is not participating in the system any more. One possible solution is a so called silent period [115]. This means that a system participant stops his/her participation for a short time until his/her pseudonym is changed successfully. User-oriented concepts allow the user to decide when he/she wants to change his/her current identity. The decision can hereby be completely subjective, allowing to define own policies and thresholds for the pseudonym change. Such concepts are Swing & Swap [144] and SLOW [41]. Although all of this concepts refer to location based systems, they can be used in IoT scenarios, e.g., where pseudonyms expire and trigger a silent period for data collection. Or where wearable medical devices form a mix-zone and call for a pseudonym change.

The question which of this concepts is usable depends on the type of IoT scenario, as a silent period, a policy based approach or a mix-zone might be or not be possible. We will detail pseudonym change in RERUM's use case UC-11: Home energy management, see Section 3.3.

4.6.7 De-Pseudonymizer

A pseudonym has to be relinked to a system participant in many scenarios, for example when a user wants to access one of his devices and the device's identity is pseudonymized or in case of billing a service or liability of a user in case of damage.

RERUM's proposed mechanism of relinking or de-pseudonymizing is dynamical, the party that re-links the identity and the pseudonym does not have a list of identities and pseudonyms, it generates the pseudonym that an identity must have used. This is possible, if the re-linking party is trusted and has the root secret as described in 4.6.3 and the re-linking party knows the real identity of the system participant.

New Pseudonyms are generated depending on time (see Section 4.6.6) and method (see Section 4.6.3). To demonstrate our de-pseudonymizing mechanism, we assuming a new pseudonym is generated when a predefined timeslot expires and it is generated with the identity of the system participant. In the following example we illustrate the de-pseudonymizing mechanism based on top-down hash trees:

A user's device sends consumption data to a cloud provider. The device protects its identity with pseudonyms generated with top-down hash trees. The user wants to know his consumption data and asks the service provider for the data of his device. He has to generate the pseudonym that the the device has used to retrieve his data.

Figure 87 depicts how both, the device and the user, generate pseudonyms to transmit and retrieve the data from the cloud provider.

The device's root secret x_0 is a sub-secret from the user generated with the user's root secret x and the device's ID. The device changes its pseudonym according predefined periods and transmits its data to the cloud provider. If the user wants to retrieve the data from the cloud provider, he computes the

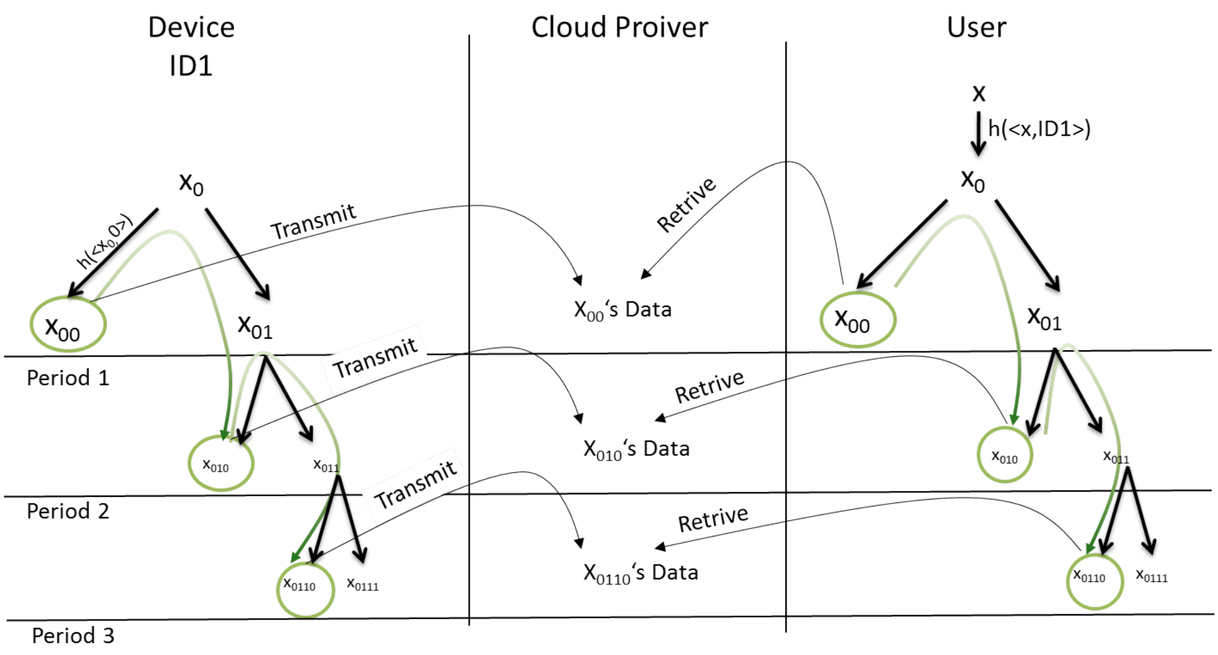


Figure 87: Example of re-linking pseudonyms to identities for data retrieval

device's root secret x_0 and generates the pseudonym according to the period that he wants to retrieve the data from. We assume that the cloud provider saves all data from any pseudonym, as long as the device is able to authenticate itself as a customer of the cloud provider. Anonymous authentication mechanisms have been discussed in D3.1, such as group signatures [53].

In another example, a user could receive a data set from a pseudonym. The data set might be signed with a group signature of one of his devices, such that he can be sure that the pseudonyms is re-linkable. The search algorithm to re-link the pseudonym could be like the following shown in Figure 88.

The user has a limited amount of devices for which he is able to generate pseudonyms of the according time period. He authenticates the incoming data set and reads the period which the data set was generated. The user generates the pseudonyms of the devices that come into consideration (probably not all devices produce this kind of data). It should be noted that the computational capacity of the de-pseudonymizing party is considered to be high and that the computational time does not equal a full binary or n-ary tree search, as the user knows exactly which values of which branches he has to compute. In an optimized version, the user knows which periods are not needed anymore and he can start generating pseudonyms via a a local path.

4.6.8 Summary

The pseudonym data structure is an easy to use, easy to implement, computational and battery efficient pseudonymizing mechanism which allows dynamical access to pseudonyms for RERUM components, fast and secure pseudonym agreement, management and revocation. Furthermore, the mechanism

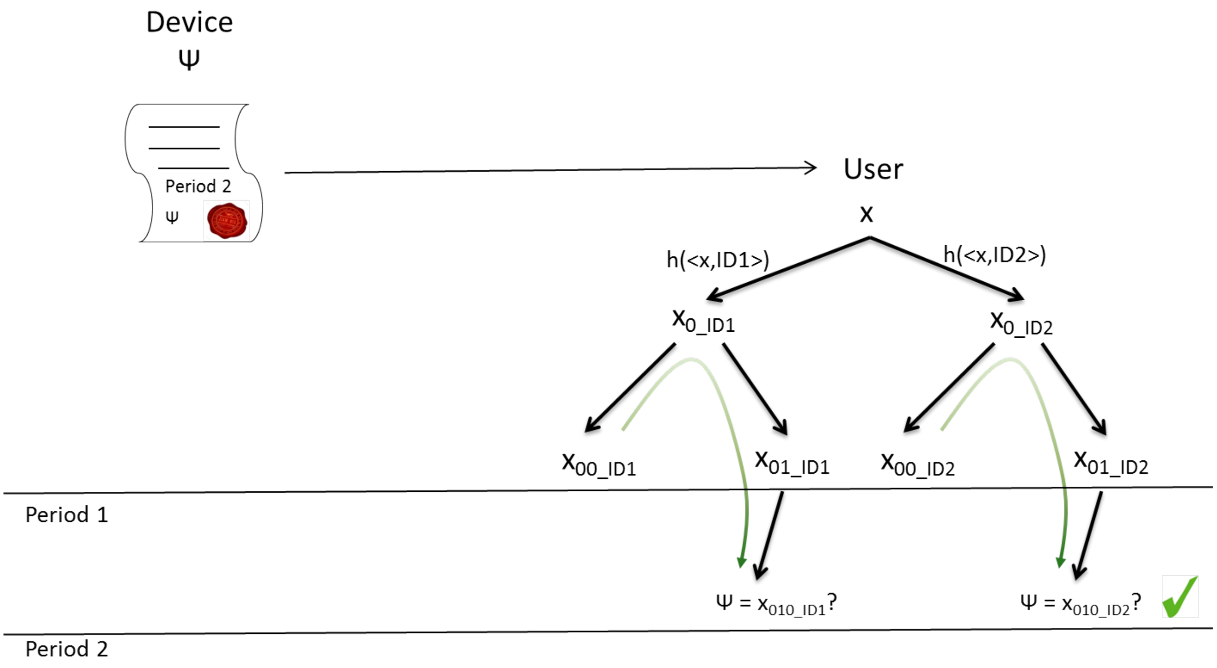


Figure 88: Example of re-linking a pseudonyms to an identity

supports de-pseudonymization without the need for asymmetric cryptography or extensive pseudonym-to-ID tables.

4.7 Consent for authorisation

The IETF has an active working group dedicated to Authorisation in Constrained Environments (ACE). Several of the solutions propose the generation of Authorisation Tokens: Delegated CoAP Authentication and Authorisation Framework (DCAF) (draft-gerdes-ace-dcaf-authorise-02), Fluffy: Simplified Key Exchange for Constrained Environments (draft-hardjono-ace-fluffy-01), Authentication and Authorisation for Constrained Environments Using OAuth and UMA (draft-maler-ace-oauth-uma-00), Two-way Authentication for IoT (draft-schmitt-ace-twowayauth-for-iot-02), Authorisation for Constrained RESTful Environments (draft-seitz-ace-core-authz-00), Object Security for CoAP (OSCOAP) (draft-selander-ace-object-security-02), and The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP) (draft-tschofenig-ace-oauth-bt-01). In general, two main problems can be associated with the proposed solutions (before the IETF meeting Nr 93 in Prague, July 2015): either they are too costly for strongly constrained devices or they leak information that can be used to track clients or users. As the RERUM consortium noticed that the same mechanism used for generating pseudonyms could also be used for generating privacy-enhanced tokens, we submitted an IETF draft and presented it in the meeting mentioned above: Privacy-Enhanced Tokens for Authorisation in ACE (draft-cuellar-ace-pat-priv-enhanced-authz-tokens-00).

All the mentioned IETF drafts can be downloaded via the Datatracker of the ACE WG [116].

The mentioned work is work-in-progress and new versions of the mentioned drafts are soon expected. Also, there is a strong interest for merging the different proposed solutions, including the RERUM draft. For these reasons, we only discuss a high-level view of the current proposal and we expect to describe more refined version later in other Deliverables.

Figure 89 illustrates the steps needed to create and to verify privacy-enhanced tokens. The actors are the following:

Server (S) An endpoint that hosts and represents a CoAP resource.

Client (C) An endpoint that attempts to access a CoAP resource on the Server.

Server Authorisation Manager (SAM) An entity that prepares and endorses authentication and authorisation data for a Server.

Resource Owner (RO) The principal that is in charge of the resource and controls its access permissions. The RO is often the data subject of the protected resource.

We additionally use the following terms:

Server Token (ST) The token which is generated by the SAM for the Server. Besides parameters, which may contain authorisation information that represents RO's authorisation policies for C, it contains a secret, St, called the ST-secret. This one can be used to verify the Authorisation Token and to generate other secrets to be discussed later.

Client Token (CT) The token which is generated by the SAM for the Client. It contains a secret, Ct, which can be used to generate the Authorisation Token, pus some other data used for PoP. Optionally CT may contain authorisation information that represents RO's authorisation policies for C.

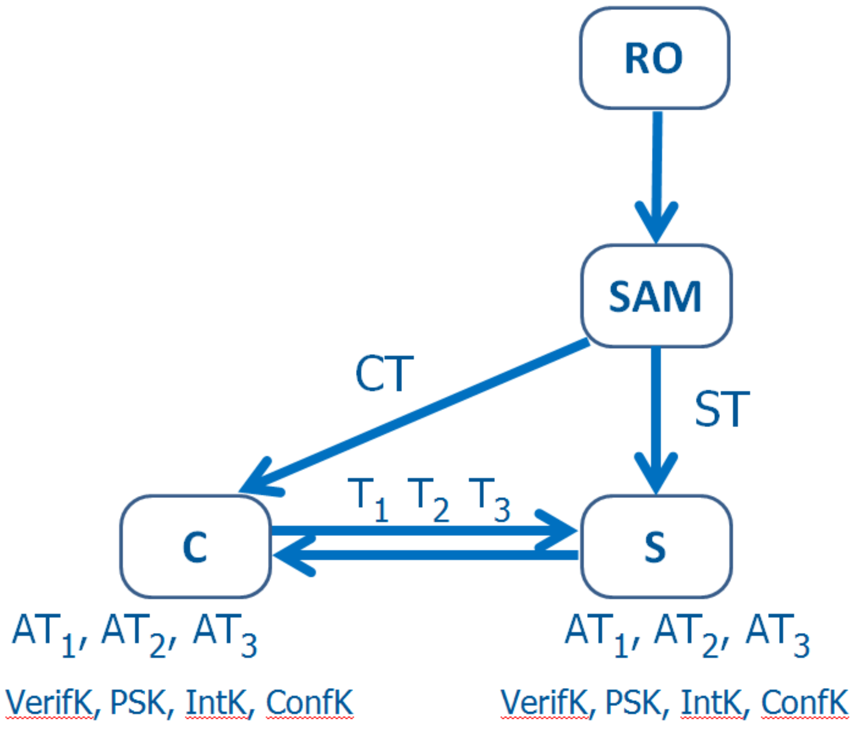


Figure 89: Privacy-Enhanced Tokens: High-Level Overview

Authorisation Token (AT) The token which is generated by the Client and presented by him to the Server. It contains a secret At, which changes regularly (in a similar way to one-time passwords). The AT contains all information needed by the Server to verify that it was granted by SAM.

VerifK, PSK, IntK, ConfK Derived keys between C and S used respectively:

- to verify that they are talking with the intended partner, for the Client C it is used as Proof of Possession of the (current) Authorisation Token
- as Pre-shared Key to establish a DTLS secure channel
- for Integrity protection (in message authentication codes)
- for Confidentiality Protection (to be elaborated in a future version of the document).

Each Server (S) has a Server Authorisation Manger (SAM) which conducts the authentication and authorisation for S. S and SAM are assumed to have a secure channel, probably a DTLS channel, but we do not assume anything about it, except that it is two way secure, preserving integrity and confidentiality.

The Client and Server Tokens may be regarded as key material from which the Authorisation Tokens and the derived keys can be built, using for instance the same way that pseudonyms can be generated form a main secret (details in [63]).

4.7.1 Summary

In summary the proposal has the following advantages:

- The method allows a User, or an Authentication/Authorization Manager on its behalf, to authorise one (or several) client(s) to access resources on a server. The client and/or the server can be constrained devices. The authorisation is implemented by distributing purpose-built Key Material (which we generically call ”Tokens”) to the server and clients.
- The Client Tokens are crafted in such a way that the clients can construct authorisation tokens that allow them to demonstrate to the server their authorisation claims. The message exchange between client and server for the presentation of the tokens may be performed via insecure channels.
- Further, the purpose-built Key material and tokens can be used for establishing a secret shared key between a client and the server, which can be then used to establish a DTLS communication with pre-shared keys.
- The tokens do not provide any information about any associated identities or identifiers of the clients nor of the server. In particular, the method can be used in context where unlinkability (privacy) is a main goal: the tokens convey only the assurance of the authorisation claims of the clients. This means that the payloads of our protocol, and in particular, the Authentication Token secrets used, can be constructed in such a way that they not leak information about the correspondence of messages to the same Client. In other words: if an eavesdropper observes the messages from the different Clients to and from the server, the protocol does not give him information about which messages correspond to the same Client. Of course, other information, like the IP-addresses or the contents themselves of the requests/responses may leak some information in this regard, but that is not information leaked by our protocol and can be treated separately.
- The tokens may be supported by a ”proof-of-possession” (PoP) method. PoP allows an authorised entity (a client) to prove to the verifier (here, the server), that he is indeed the intended authorised owner of the token and not simply the bearer of the token. (Notice that the Authorisation Token may be sent in the clear, and thus, it could be stolen by an intruder. A PoP would hinder the attacker to use the token pretending to be authorised).
- The Key Material can be used to generate and coordinate pseudonyms between C and S and potentially further parties.
- The user (more precisely, the Resource Owner, RO) is able to decide (if he wishes: in a fine-grained way and in real-time) which client under which circumstances may access his data stored in S. This can be used to provide consent (in terms of privacy) from users (again, ROs).

4.8 GeoLocation position hiding

In deliverable D2.3 we first introduced a position hiding mechanism where a traffic participant sends a random number of vectors, which are again determined by random timers.

In this chapter we explain the technical details of this privacy enhancing technology. This privacy friendly approach allows traffic analysis by floating car observation [206] in RERUM use case UC-01: Smart Transportation. The approach allows the adaption of user preferences and temporary opt-out of the data collection. For example, a user might not want to send traffic data for a specific area. But as soon as he passes it, it's fine for him to participate in the data collection again. The presented approach allows this kind of situations.

Additionally, we adopt a privacy-by-default approach and stop the data collection at side-roads, which are less affected by heavy traffic, but at the same time lead to intrusive conclusions of a user's destinations.

4.8.1 GeoLocation PET---generation of vectors

A vector is created the following way: when a user is moving, a timer decides where the starting point of a vector will be, and how long it will take to choose the ending point of the vector. As several vectors may be created at the same time, the traffic participant will have a list of current vectors such as the following:

Table 22: Generation of multiple vectors

Vector	Starting Point	Time Until Stop	Average Speed	Average Driving Time	Endpoint (Elicited at Stop)
A	(X11, Y11)	5 Minutes	...	...	?
B	(X21, Y21)	8 Minutes	...	...	?
C	(X31, Y31)	22 Minutes	...	...	?

The "starting point" and the "time until stop" are chosen at random. The endpoint is measured at the moment when an assigned timer runs out. Afterwards the vector information is sent to a service provider, e.g., the traffic department. While the amount of vectors prevents the knowledge of how many participants are really passing the same route (each vector is transmitted as a unique traffic participant), the attached speed and driving time averages provide information about the overall traffic of each route.

The mechanism can be used to support privacy location policies, such as [61]. The mechanism will stop the data collection as soon as either a policy specified location or a side-road is reached.

We call side-roads and areas defined in privacy location policies opt-out areas, as the user (automatically) opts-out of the generation of vectors and the transmission of traffic data.

We define two actions to support a participation opt-out.

The first action is *stop at geodic/civic location condition*, which stops the data collection and transmission while the participant is in defined area, and, *stop at side-road*, which stops the data collection whenever

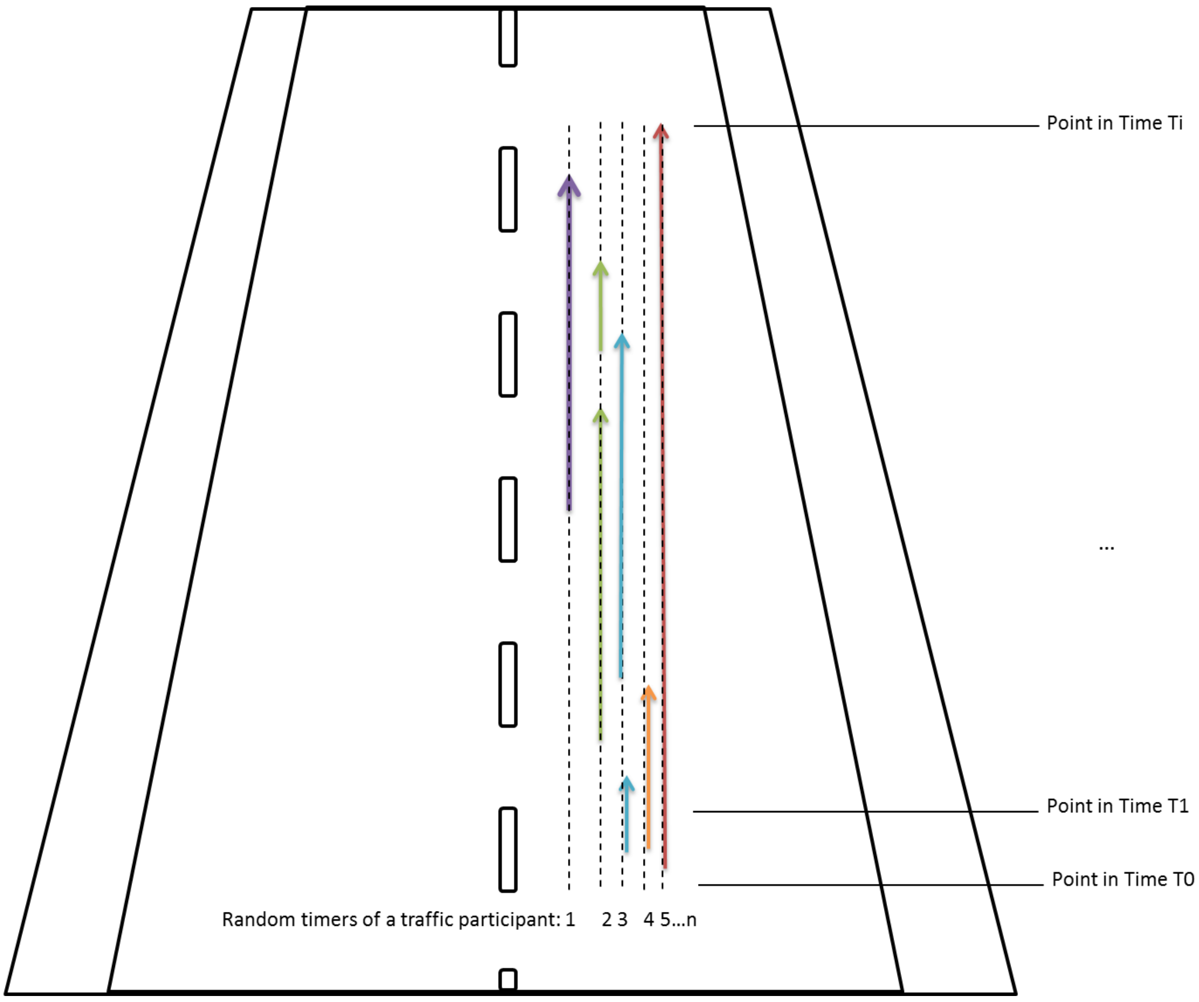


Figure 90: Time Controlled Vectors

a traffic participant exits a main road and enters smaller side roads. Smaller roads lead to a participant's home, working place, etc., and thus are, in our opt-in approach, excluded by default from the analysis.

The stop behaviour is as follows: Several independent vectors are generated and sent to the a service provider at random as usual, with the addition that the generation of vectors will stop when the participant's policies apply or when he enters a side-road. To exemplify the different actions, we assume a traffic participant driving in Regensburg, Germany. The participant has defined policies of a location (green circle in Figure 91), where the data collection system should stop.

4.8.2 Stop at geodic/civic area defined by policy

We assume that a traffic participant has defined some areas where he does not want to send traffic information. One way of defining such policies is by using the geodic and civic location profiles described in RFC6772 [61]. The interpretation of such a policy has been done before, see [72], and is thus not a part of this technology. The traffic participant generates random vectors as described in the section above, see 90; as soon as he reaches the defined area the generation vectors will stop. Active vectors will be sent to the traffic department. Figure 91 exemplifies the reaction of the system when the "stop at geodic/civic area" action is defined. The traffic participant has defined a policy to opt-out when he reaches his residential area around *Friedrich-Ebert-Strasse* (green circle), which is a known area of social flashpoint. His desired route is depicted by the black dotted line; the vectors generated throughout the route are of several colours.

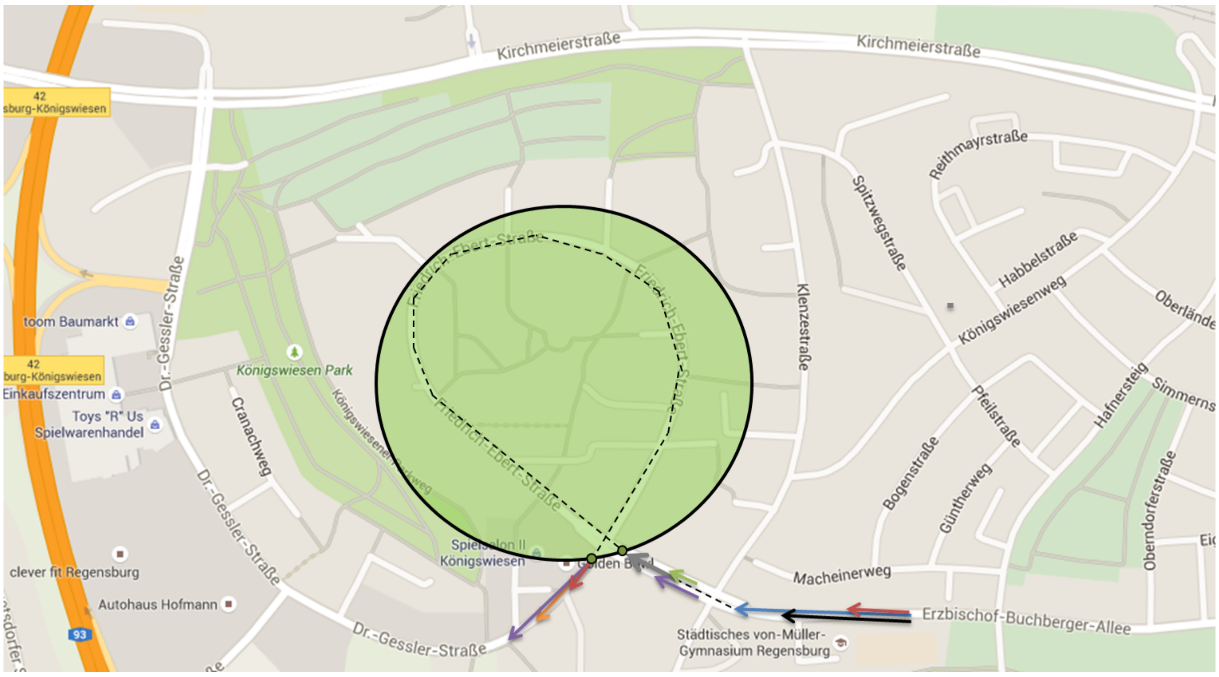


Figure 91: Stop at geodic/civic area

At this point, the data collection will behave as follows: No vectors will be generated starting from this point, and, if any active vectors exist, a common average will be generated and sent as a position somewhere before the entry point to the protected area. A detailed example of how averages can be generated is given in Tables 23 and 24.

4.8.3 Automatic stop at side-roads by default

The user's route is depicted as a black dotted line. The user drives along *Erzbischoff-Buchberger-Allee* and enters *Friedrich-Ebert-Strasse*. He then decides to take detour at a small side-road along (depicted as a a black dotted line traversing Friedrich-Ebert-Strasse). At the point of entrance (small black-framed green circle), the data collection will stop. This means, that no vectors will be generated starting from this point. If any active vectors exist, a common average will be generated and sent as a position somewhere before the entry point of the side-road.

4.8.4 Example of user opt-out with two active vectors

Let's assume two vectors are still active while the user enters the side road (e.g., the purple and green vectors in Figure 92).

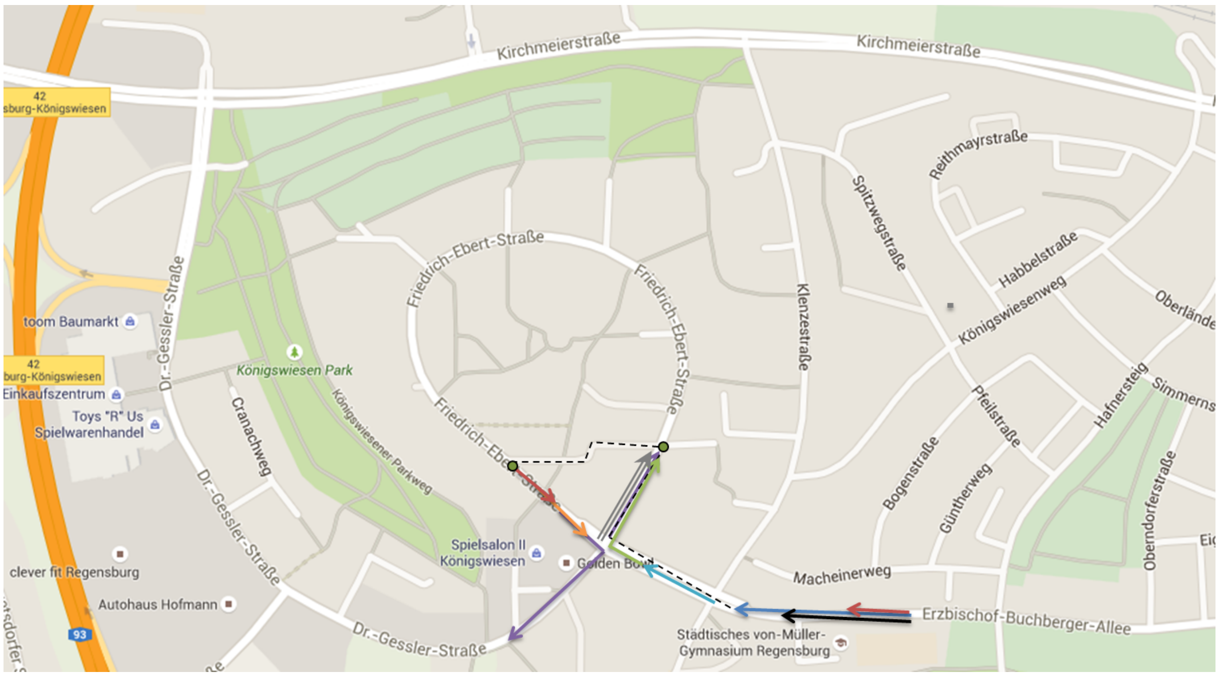


Figure 92: Stop at side roads

Table 23: Multiple active vectors before entering an opt-out area

Vector	Starting Point	Time Until Stop	Average Speed	Average Driving Time	Endpoint (Elicited at Stop)
A	(X11,Y11)	5 Minutes	...	...	?
B	(X21,Y21)	8 Minutes	...	...	?
C	(X31, Y31)	22 Minutes	...	...	?

The two vectors will be averaged, converted to one vector, assigned an endpoint that differs from the entry point of the side-road and sent to the service provider.

Table 24: Averaged vector sent at entrance of an opt-out area

Vector	Starting Point	Time Until Stop	Average Speed	Average Driving Time	Endpoint (Elicited at Stop)
A	(X11,Y11)	Not Required	39,5 Km/h	4:38 Minutes	(X12, Y12)

The new vector (shown as a two-lined gray vector in Figure 92) has an endpoint with a GPS-position somewhere on the dotted black line, before the entry point to the side road. This is the last vector sent before the participant enters the side road. After leaving the side-road, the participant starts sending position data again; this is denoted by the red, orange and purple vectors in Figures 91 and 92.

4.8.5 Avoiding correlation between vectors

To avoid a possible time correlation between the last averaged vector, the driving speed and the new vectors, in case the new vectors are created at the very moment of leaving the opt-out location or side road, a random threshold time until opt-in is suggested. Thus the correlation between the new vectors and the previous driving speed is blurred.

4.8.6 Privacy considerations

As described previously in deliverable 2.3, every vector has to be sent as a unique traffic participant's measurement. This includes hiding the IP-Address from the sender with traffic anonymization techniques, and occasionally adding integrity protection in form of unlinkable group signatures [53]. As seen in [54], merely protecting the sender of GPS-location data is not enough. Additional information, for example by a geolocation system and online social networks, reveal where the traffic participant is heading to and which data subjects are the ones that could have possibly visited those locations. The set of these subjects, or the k-anonymity set of data subjects where each participant is indistinguishable from at least k-1 other participants with respect to a certain GPS-positions, is often very small. The reason for this is, that with every GPS-location sent to a traffic provider and with driving speed and time correlation linking every location, the resulting route becomes very unique.

The generation of artificial vectors enlarges that anonymity set, but without blurring or adding noise to measurements. The artificial vectors provide even more information to the measurements, as every vector has its unique starting and ending point.

4.8.7 Summary

The geo-location privacy component is a novel approach to privacy friendly floating car observation. Related work on vehicular area network has focussed on hiding message routes, while it does not analyse GPS positioning data. RERUM's random vector generation fills this gap and allows an accurate measurement for service providers as well as location and policy based privacy for users.

4.9 Compressive sensing encryption

Privacy and security is of major importance in Wireless Sensor Networks (WSNs), as the miniature sensors can often collect and convey highly sensitive and confidential information (e.g. user location, biometric data). Usually, privacy and security become feasible through the use of encryption for the data exchanged between the communicating parties. Several algorithms based on public key encryption involving public and private keys (e.g. RSA [129]) provide robust encryption against unauthorised users. However, this type of algorithms require advanced resources, in terms of processing power and memory; hence, cannot be easily used by the resource-constrained sensors. On the other hand, symmetric algorithms, like the AES [193], require less computational resources and memory, as they use the same key for encryption and compression. The disadvantage of these algorithms is that a key management scheme is required for key distribution to the communicating parties.

Except the privacy and security requirements of the WSNs, energy efficiency is also important as the sensors are often battery operated, and in many scenarios (e.g. [231]) they are placed in harsh environments where human intervention is difficult or impossible. As many works have shown, energy is mostly spent during the sensor communication (listening or transmitting) over its radio interface. A common method used for the minimisation of the communication overhead is data compression at the application layer. After compression, encryption usually follows for security purposes. At the receiver side, decryption and then decompression takes place. This operation therefore requires two distinct operations: compression/encryption and decryption/decompression.

The last few years Compressed Sensing or Compressive Sensing (CS) has appeared as a new theory that provides encryption and compression in a single step. As shown in [48], if a signal has a sparse representation in one basis, it can be recovered from a small number of projections in a second basis that is incoherent with the first.

Assume that $x \in \mathbb{R}^N$ refers to information collected by a sensor. Suppose that there is a basis Ψ of $N \times 1$ vectors $\{\psi_{i=1}^N\}$ such that $x = \Psi b$, where $b \in \mathbb{R}^N$ is a sparse vector with S non-zero components ($\|b\|_0 = S$). According to CS theory, the information contained in x can be projected using matrix $\Phi \in \mathbb{R}^{M \times N}$, giving $y = \Phi x$, where $y \in \mathbb{R}^M$ is the compressed version of x. As $M \ll N$, the choice of M controls the compression rate of the original data. Furthermore, the compression rate affects the performance of CS, in terms of the reconstruction error. According to Candes et al. [48], an S-sparse signal x can be reconstructed exactly with high probability if $M \geq CS \log(N/S)$, where $C \in R^+$. In any other case, there is a trade-off between the compression rate and the reconstruction error, so, in general, the higher the compression rate is, the higher this error becomes. In general, the compression/encryption using the CS principles is expressed as follows:

$$y = \Phi x = \Phi \Psi b = \Theta b \tag{1}$$

where $\Theta = \Phi\Psi$. The original vector b, and consequently the sparse signal x are estimated using the following ℓ_1 norm convex relaxation problem:

$$\hat{b} = \arg\min \|b\|_1 \quad s.t. \quad y = \Theta b. \tag{2}$$

Observe that the above problem is an under-determined problem with less equations than unknowns as $M \ll N$.

4.9.1 CS encryption strength

As mentioned in the previous section, CS is used to compress a signal $x \in \mathbb{R}^N$, with a compression rate equal to $\frac{N-M}{N}$, by projecting x to matrix Φ. Observe that the formula in (1) is similar to a block cipher used for encryption. For example, the encryption formula of the Hill Cipher [79] (a block cipher) is expressed as:

$$y' = \Phi' x' (mod\, m) \tag{3}$$

where x' is the plaintext, and y' the corresponding ciphertext. In this case, $\Phi' \in \mathbb{R}^{N \times N}$ is the encryption matrix, and both x' and y' have the same length ($\in \mathbb{R}^N$). Consequently, decryption takes place by using $x' = [\Phi']^{-1} y' (mod\, m)$, where $[\Phi']^{-1}$ is the inverse of matrix Φ'. By comparing (1) and (3), observe that CS performs encryption similarly to a block cipher using Φ as the encryption key. A major difference however is that CS performs compression simultaneously with encryption as $M \ll N$. Another difference is that decryption for the block cipher is performed by solving an equation using the inverse of the encryption matrix, where for CS no inverse exists (because $M \ll N$); therefore, an under-determined system has to be solved. Finally, as shown in [88, 175], CS provides robust encryption as it can tolerate fluctuations of Φ, meaning that if a plaintext is encrypted using Φ_1, and then decrypted by a different matrix Φ_2, the reconstruction error remains low if these two matrices are quite similar, however, it is not required to be exactly the same as in block ciphers. Summarising at this point, CS is used for simultaneous encryption and compression using matrix Φ as the encryption key. The reconstruction error depends on the compression rate, as well as on the sparsity of the plaintext.

In this deliverable we mainly consider CS as an encryption algorithm, aiming to study its encryption strength, and propose a technique that advances its strength. Generally, encryption algorithms are studied, in terms of their encryption strength, either by investigating how computationally secure they are against known attacks, or how secure they are from the information theoretic secrecy point of view.

4.9.1.1 Computational secrecy

Consider a scenario where a wireless sensor repeatedly collects sensitive data x, and then by using Φ encrypts these data into ciphertext y. Also assume that an attacker is present that passively monitors the wireless channel; thus, being able to capture the encrypted data y transmitted by the legitimate sensor. The goal of the attacker is to guess Φ by examining the transmitted blocks of y. This attack is usually referred as *known ciphertext attack*. The attacker may try to guess Φ by forcing a brute force attack based on the ciphertexts it has collected, and searching over the values for Φ based on a step size. Then, and for each ciphertext, it creates (guesses) a matrix $\Phi^{'}$, and it reconstructs (decrypts) y to $\hat{x} = \Psi \hat{b}$ using (2). At this point, the attacker can estimate, through the reconstruction process (see [52] for details), the residual error that can be used as a metric of the reconstruction accuracy. If the residual error is larger than a threshold, it retries the same procedure, otherwise, it stops and assumes that it has guessed the correct matrix Φ. Nevertheless, as shown in [175], for this brute force attack to become feasible, the computation cost is in the order of $O(N^{1.2})$, something that makes this process too expensive.

Another type of attack against a CS crypto-system is an attack based on the symmetry and sparsity structure of matrix Φ (described in [175]). This attack is composed of two phases: During the first phase, the attacker tries to estimate the t leading columns of matrix Φ, assuming that x has t non-zero leading coefficients, and the corresponding coefficients in x such that $\Phi_t x_t = y$. A random permutation of the columns of Φ, and of the corresponding positions in x, produces the same values of y. For this reason, during the second phase, the attacker has to determine the appropriate permutation, so as to find a suitable solution for the over-determined system shown in (2). This system has become over-determined as $t < N$. The number of possible permutations requires $C(N,t) \times t!$ possible arrangements that make this attack highly complex.

4.9.1.2 Information theoretic secrecy

Information theoretic secrecy is based on the statistical properties of a crypto-system providing security even if an attacker has an unbounded processing power. Shannon [210] introduced the idea of perfect secrecy, defining that a crypto-system achieves perfect secrecy if the probability of a plaintext conditioned on the ciphertext, is equal to the $\grave{a}$ priori probability of the plaintext, $P(X = x \mid Y = y) = P(X = x)$. Using the mutual information I, this can be expressed as $I(X : Y) = 0$. The mutual information is used to measure both the linear and non-linear correlation. This is usually difficult to measure but it is a natural measure of the dependence between random variables, considering the whole dependence structure of these variables [232]. Mutual information is computed as follows:

$$I(X:Y) = \sum_{x \in X} \sum_{y \in Y} p_{xy}(x,y) log_2 \frac{p_{xy}(x,y)}{p_x(x)p_y(y)}, \tag{4}$$

where p_{xy} and p_x denote the joint probability density function (PDF) and marginal PDF, respectively.

As (1) shows, ciphertext y is a linear projection of plaintext x so, it is expected that no perfect secrecy can be achieved using CS for encryption. For demonstration purposes, we empirically compute the mutual information of a plaintext and its corresponding ciphertext when applying CS, for different plaintext lengths, and for various compression rates. The compression rate is defined as $\frac{N-M}{M} \times 100$, where N and M are the lengths of the plaintext, and the ciphertext, respectively. Observe in Figure 93 that as the compression rate increases, mutual information decreases; hence, higher information secrecy is achieved. This is because when the compression rate increases, ciphertext's length becomes smaller, and linear projections are fewer, so the information leakage from the plaintext to the ciphertext reduces. Furthermore, observe that as the length of the plaintext increases, mutual information increases, because for the same compression rate, more information leakage takes place due to the linear projections.

One would assume that by compressing a plaintext using a higher compression rate, would be the ideal solution in a CS crypto-system. Nevertheless, CS performance, in terms of the reconstruction error, deteriorates as compression increases. The performance is also affected by the sparsity of the plaintext. Figure 94 shows the trade-off between the mutual information and the reconstruction error e of CS, defined as $e = \frac{||x-\hat{x}||_2}{||x||_2}$, where x and $\hat{x}$ are the original and reconstructed plaintexts, respectively. Compressing a plaintext using a higher compressive rate can provide higher information secrecy, however, the reconstruction error increases.

4.9.1.3 CS encryption vulnerability

In the previous sections we described CS encryption strength against brute force and sparsity structure attacks, as well as its strength from the information secrecy point of view. In this section, we highlight the vulnerability of CS encryption in the case of a *Chosen Plaintext Attack* (CPA). Such an attack becomes feasible when an attacker manages to provide specific plaintexts to a CS encryption system, and later

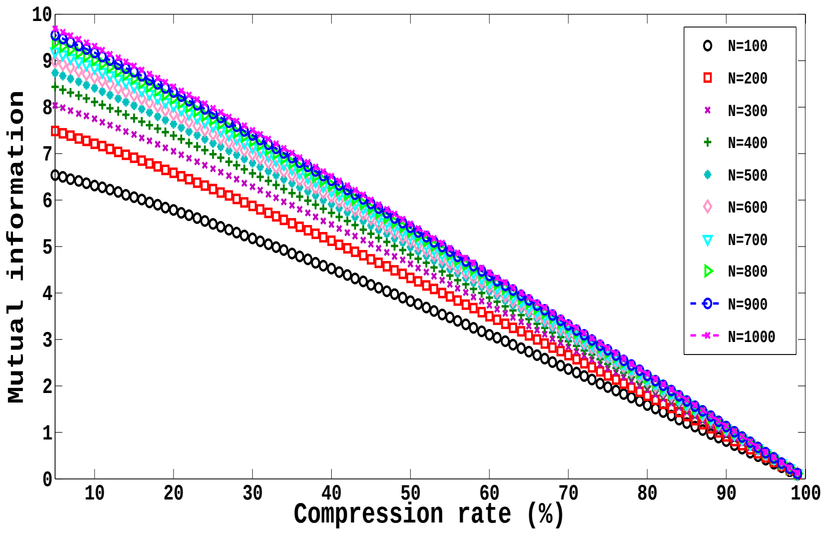


Figure 93: Mutual information for an increasing sample length and compression rate

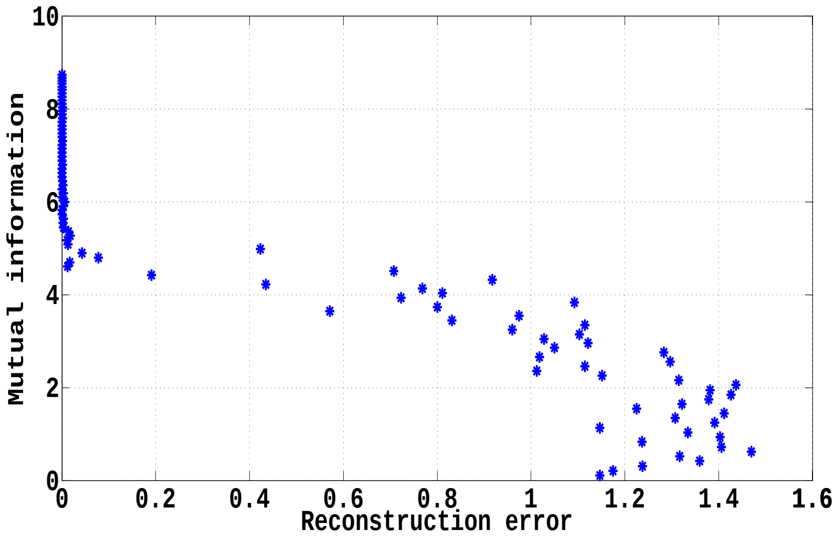


Figure 94: Trade-off between the mutual information and the reconstruction error

on he is also able to capture the corresponding ciphertexts. Recall from (1) that ciphertext y is derived after a multiplication between the measurement matrix Φ and the plaintext x. Supposing an attacker has the ability to launch a CPA, he can provide a plaintext x, where all of its values, except in a specific location (index) j, are equal to zero as shown below:

$$\forall i \in [1, N], x_i = \begin{cases} 0 & \text{if } i \neq j \\ \text{C} & \text{if } i = j \end{cases} \tag{5}$$

where $C \in R_{\neq 0}$. Ciphertext y now has all of its elements equal to zero, except at column j. By repeating this procedure, the attacker can reveal the columns of matrix Φ, one-by-one, using (5).

4.9.2 Enhancing CS encryption using chaos

In general, if a signal $x \in \mathbb{R}^N$ is sparse in some basis $\Psi \in \mathbb{R}^{N \times N}$, then it can be written as $x = \Psi b$. Matrix Ψ is usually referred as the *sparsifying basis*. Typical sparsifying bases commonly used in CS are the Fast Fourier Transform (FFT), and the Discrete Cosine Transform (DCT). An attacker that has successfully launched a CPA using (5), is now aware of matrix Φ, and as $\Theta = \Phi\Psi$, he can solve the optimisation problem of (2) to reveal plaintext x.

Here, we propose a new technique that makes CS immune to CPA. Recall from (1) that the ciphertext is derived by using $y = \Phi\Psi b = \Theta b$; hence, $\Phi = \Theta\Psi^{-1}$, and the general CS measurement model becomes as follows:

$$y = (\Theta\Psi^{-1})x = Ax. \tag{6}$$

Similarly to (2), the reconstruction is performed using

$$\hat{b} = \text{arg min} \, \|b\|_1 \quad s.t. \quad y = \Theta\Psi^{-1}\Psi b = \Theta b. \tag{7}$$

Rather than using a typical measurement matrix Φ to encrypt the plainext, we use $(\Theta\Psi^{-1})x$ as the encryption matrix. Matrix Θ can be generated using typical distributions like the Gaussian [47], or the Toeplitz [12], and Structurally Random Matrices [69] for hardware efficiency. Regarding the sparsifying basis, we create a secret basis, denoted by Ψ_s that is known only to the legitimate users. The CS measurement model becomes as follows:

$$y = (\Theta{\Psi_s}^{-1})x = A_s x, \tag{8}$$

and the reconstruction becomes feasible using:

$$\hat{b} = \text{arg min} \, \|b\|_1 \quad s.t. \quad y = A_s\Psi_s b = \Theta{\Psi_s}^{-1}\Psi_s = \Theta b. \tag{9}$$

4.9.2.1 Secret sparsifying basis using chaos sequences

The equations in (8) and (9) comprise a CS crypto-system where the plaintext x is encrypted using two secret matrices: (i) matrix Θ derived from a typical distribution (e.g. Gaussian), and (ii) matrix Ψ_s. In this section we focus on the creation of the secret sparsifying basis Ψ_s. According to the CS literature, a common basis like the FFT or the DCT provides good results (in terms of the reconstruction error), as most natural signals are sparse in those domains. Let us denote such a common basis as $\Psi = \{\psi_1, \psi_2, ..., \psi_N\}$, where ψ_i denotes the elements of the i_{th} column of Ψ, and N is the total number of its columns.

We create a secret basis Ψ_s by multiplying the columns of the original basis Ψ with a secret sequence $C \in \mathbb{R}^N_{\neq 0}$, as shown below:

$$\Psi_s = \{c_1\psi_1, c_2\psi_2, ..., c_N\psi_N\} \tag{10}$$

A legitimate receiver, in order to successfully decrypt a ciphertext, it has to be aware of both the matrix Θ, and the secret sequence C. The challenge is how to create an appropriate secret sequence that will further provide the secret sparsifying basis, without negatively affecting CS performance. The secret sequence, therefore, has to achieve two objectives: (i) to provide a suitable sparsifying basis, and (ii) to be easily generated in resource-constrained devices like the sensors.

A family of secret sequences that can achieve these objectives is the so-called chaos sequences, produced by a logistic map using a quadratic recurrence equation, as shown below:

$$c_{n+1} = bp \times c_n \times (1 - c_n), \tag{11}$$

where $bp \in R^+_{\neq 0}$ is called as the *biotic potential*. As shown in (11), each value of the sequence is a function of its previous value. A chaotic sequence C denoted by $C(d, k, c_1)$ is fully characterised by three parameters: (i) d that defines the sampling distance from the sequence generated by (11), (ii) k the total length of the chaos sequence, and (iii) c_1 its initial value. A legitimate receiver upon knowing these three parameters, is able to generate the correct chaos sequence; the same sequence the transmitter used to encrypt the data.

The chaos sequence C heavily depends on bp, as this controls how chaotic the sequence can become. Figure 95 shows how bp affects the chaotic sequence generation process (here we use $c_1 = 0.2$). Observe in this figure that when bp exceeds the value of 3.5, the sequence generation starts to become chaotic. The onset of chaos can be better depicted using a bifurcation diagram (Figure 96) that shows all possible distinct values the chaotic sequence can take for different values of bp. As bp increases from 2.5 up to almost the value of 3.5, the sequence takes just a few distinct values, so up to that point it cannot be regarded as chaotic. As soon bp exceeds 3.5, the number of the distinct values increases, and for values over 3.5, the sequence becomes chaotic as the number of the distinct values substantially increases.

When $bp = 4$, the solution for (11) can be written as below [234]:

$$c_n = \frac{1}{2}[1 - \cos(2\pi\theta 2^n)] \tag{12}$$

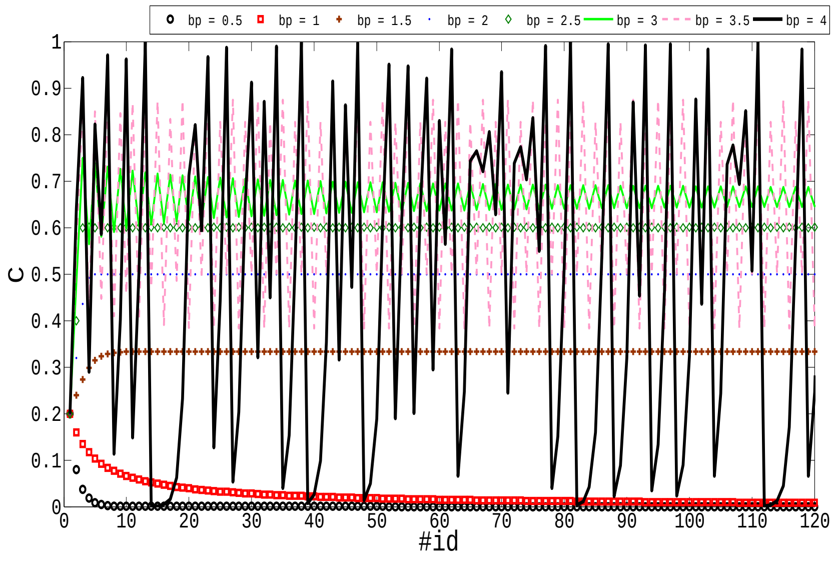


Figure 95: Chaotic sequence generation using different biotic potentials

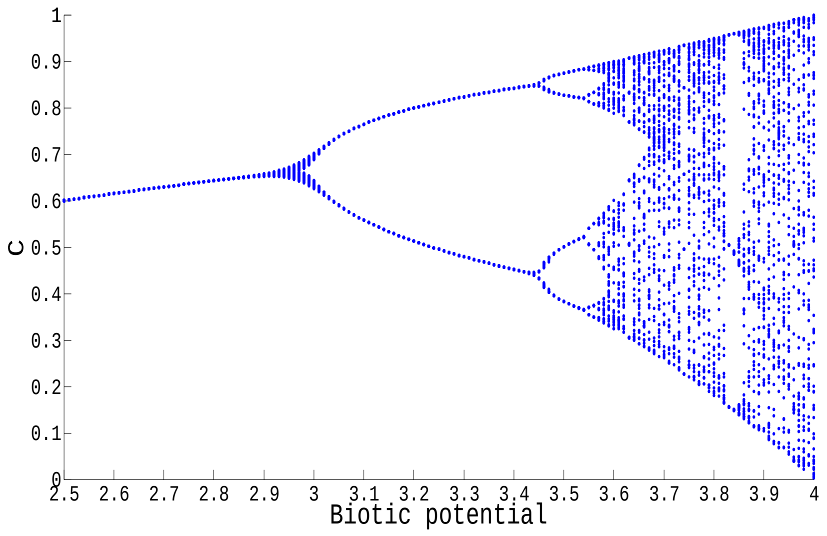


Figure 96: Onset of chaos when biotic potential exceeds a threshold value

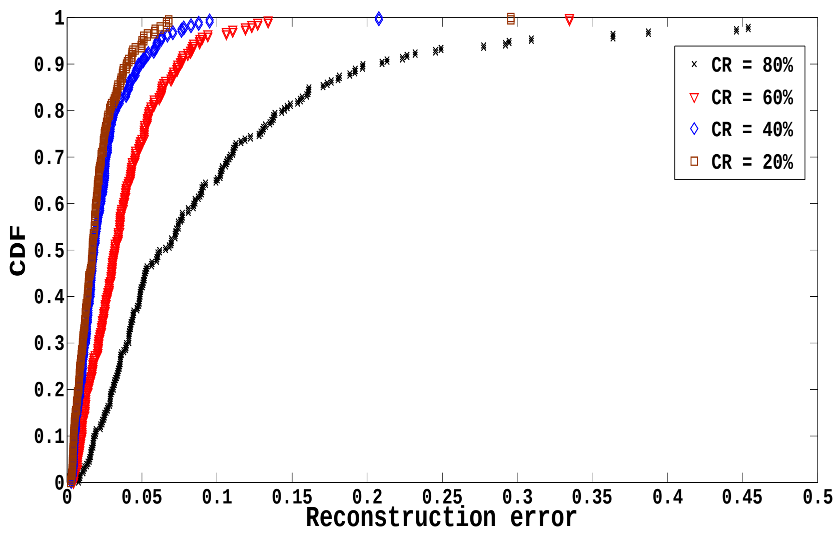


Figure 97: Reconstruction error for different compression rates for the correct sparsifying basis

We generate the secret sparsifying basis Ψ_s of (10) using (12). In order to test the feasibility of using chaotic sequences for creating the secret basis, we measure the performance of CS encryption, in terms of the reconstruction error, when encrypting and decrypting data using different compression rates. The data consist of 200 blocks, each containing 100 values of ambient temperature measurements, provided by [77]. The compression rate varies from 20% up to 80%, while matrix Θ of (8) is created from a Gaussian distribution, and a chaotic sequence $C(15, 1500, 0.2)$ is applied on an FFT basis to derive Ψ_s. Data are decrypted using (9). Figure 97 shows the cumulative density function (CDF) of the reconstruction error for the various compression rates. This error increases, as the compression rate increases, however, and for most of the data blocks, it remains lower than 5% for all cases, except when CR=80% that increases.

Let us assume now that an attacker can sniff the network and capture the ciphertexts transmitted by the legitimate transmitter. This attacker also knows the compression rate used, and that the data are sparse in the FFT domain. He has also performed a CPA, so he is aware of the matrix used for encryption that is $A_s = \Theta{\Psi_s}^{-1}$. Nevertheless, he is not aware of the secret chaotic sequence used to generate matrix Ψ_s; therefore, after capturing the encrypted data, he will try to decrypt them following (9). In this case, he will not be able to correctly decrypt the data as the equality constrain of (9) cannot hold as $y = \Theta{\Psi_s}^{-1}\Psi \neq \Theta b$. So, although the attacker knows $\Theta{\Psi_s}^{-1}$ from CPA, he cannot find the individual matrices Θ and ${\Psi_s}^{-1}$. Figure 98 shows that the reconstruction error this attacker experiences is over 80%, regardless of the compression rate.

4.9.3 Related work

There is a number of significant contributions that study CS encryption strength. The authors in [175] study the robustness of CS encryption for two types of attacks: brute force, and symmetry and sparsity-related attacks. For the former attack, the computational cost is in the order of $O(N^{1.2})$, something

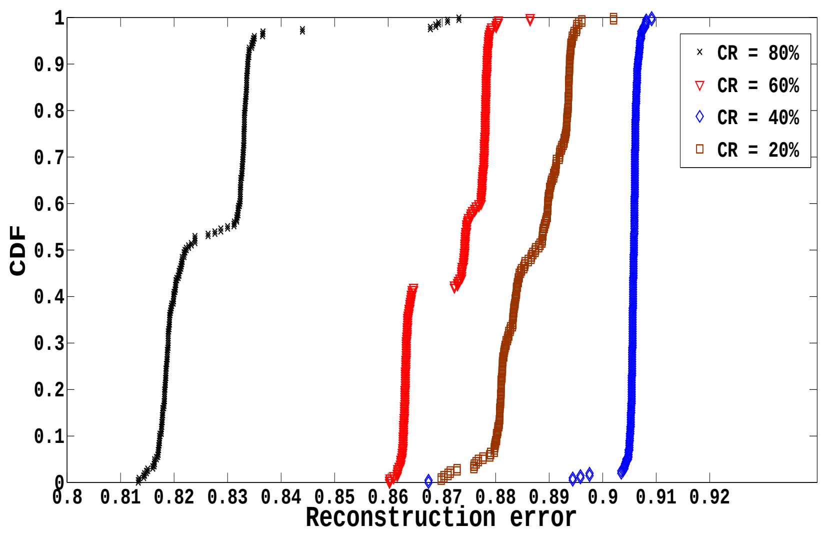


Figure 98: Reconstruction error for different compression rates for a non-suitable sparsifying basis

that make this attack very difficult. For the latter, an attacker has to search over all possible column re-arrangements of the encryption matrix in order to reveal it. This attack is very complex as the number of these re-arrangements is too high. The authors in [192] show that if an attacker decrypts data using a wrong encryption matrix, then the sparsity of the decrypted data is higher than that of the original data.

Cambareri et al. [42] perform a statistical analysis of the encrypted measurements; however without focusing on the computational feasibility of this analysis, showing that CS is not perfectly secure. Their main contribution regards a multi-class encryption scheme where legitimate receivers gain multi-level confidential access to the original data based on different encryption matrices.

All the above contributions study CS encryption strength, but without focusing on its vulnerability on CPA. The only work that focuses on this vulnerability is described in [238]. They follow the same approach we described here by creating a secret sparsifying basis. They, however, use a technique known as *Fractional Fourier Transform* that might be to complex to be used in resource-constrained sensors. On the contrary, we use the chaos sequences that are processing and memory efficient, so easily implemented in sensors.

4.9.4 Summary

CS is a theory that allows the sampling of data far below the theoretic Nyquist frequency if these data are sparse in some domain. A major advantage of CS is that allows lightweight encryption and compression in a single step. From the secrecy point of view, CS although not perfectly secure, it can provide computational secrecy. However, CS encryption is vulnerable to CPA attacks, where the complete encryption matrix can be revealed after a successful attack.

We proposed a method that makes CS immune to CPA attacks. Using a chaos sequence, we generate a secret sparsifying basis that is used as part of the encryption key. The results show that a legitimate receiver can decrypt the data with a small error, depending on the compression rate. On the other hand, an attacker decrypts the data with an enormous error that is over 80%.

4.10 Leakage resilient MAC

In this section we present the leakage resilient Message Authentication Codes (MAC) which, in contrast to the previous proposals in the literature, might be used in practical applications. The content presented in here is getting published in [153].

In many cases it is hard to achieve privacy without applying underlying security mechanisms. These mechanisms are usually supported by cryptographic primitives, schemes and protocols. If not protected against side-channel leakages such as power, EM or timing leakage, cryptographic primitives may leak secret information via aforementioned channels [135] and thus are vulnerable to many different kinds of attacks. The most common approach to mitigate the mentioned leakage issues is to apply countermeasures such as hiding, masking or make the algorithm execution time constant [152]. Finding a generic solution for the leakage problem turned out over the time as a non-trivial, mostly due to a fact that the exact protection techniques might vary depends, i.e., on a particular algorithm, execution platform and implementation details. The common goal of said countermeasures is to reduce an exploitable leakage, but such a reduction is not the only possible scenario. Cryptographic schemes might also tolerate some leakage. In such cases security definitions are enhanced (to include leakage) and security proves are carried in with the presents of leakage. Although theoretically proven, the drawback of these schemes are usually associated with practical implementations, i.e., overheads of leakage resiliency make them impractical. In this section we present the overview of a leakage resilient MAC design [153], which can be consider practical and thus might be included as underlying building block for many cryptographic protocols and thus also support privacy.

A message authentication is a technique that protect message integrity, i.e., it should detect any message (or data) modifications during a communication process between a sender and a receiver. As described in D2.3 Section 6.11.1.1, such the Integrity Generator / Verifier mechanism is essential for the RERUM architecture. It is worth noticing that the integrity protections might be also used as underlying component of D2D Authenticator (as described in Section 3.8.2). In general, the message integrity can be achieved by applying both public and private-key schemes. In the latter case, the message integrity can be achieved with use of digital signatures, in the former case, for the ensuring integrity one can use, i.e., Message Authentication Codes (MAC). A MAC scheme can be defined as a tuple that consists of three algorithms:

$$\mathcal{M} = (\mathsf{MAC.KeyGen}, \mathsf{MAC.Tag}, \mathsf{MAC.Ver}),$$

where:

- MAC.KeyGen is a probabilistic algorithm generating a suitable key K; we denote this by $K \overset{R}{\leftarrow} \mathsf{MAC.KeyGen}()$.
- MAC.Tag is probabilistic algorithm taking as an input key K, message m and generating tag σ; we denote this by $\sigma \overset{R}{\leftarrow} \mathsf{MAC.Tag}(K, m)$.
- MAC.Ver is deterministic algorithm taking as an input secret key K, tag σ, message m and outputting boolean variable whether the tag is correct; we denote this by $b \leftarrow \mathsf{MAC.Ver}(K, m, \sigma)$.

In addition, we require (for correctness) that for all valid keys K the following equation

$$\mathsf{MAC.Ver}(K, \mathsf{MAC.Tag}(K, m), m) = 1$$

is hold.

```
proc KeyGen():
    $K \xleftarrow{\$} \mathbb{G}_1$
    Return $K$

proc Tag(K, m):
    $W \leftarrow H(m)$
    $T \leftarrow e(K, W)$
    Return $T$

proc Ver(K, T, m):
    $W \leftarrow H(m)$
    $T' \leftarrow e(K, W)$
    Return $T' = T$
```

Figure 99: Bilinear MAC scheme $\mathcal{M}$[153]

4.10.1 Bilinear MAC scheme

Before we describe a leakage resilient MAC, we introduce the bilinear maps [91] and the bilinear MAC scheme, on which the leakage resilient MAC is based. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ be a cyclic groups all of prime order p with generators g_1, g_2 and g_3 respectively. The bilinear map is a function $e : GG_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ that holds both bilinearity, i.e, $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2, a, b \in \mathbb{Z}_p : \ e(u^a, v^b) = e(u, v)^{ab}$ and non-degeneracy $e(g_1, g_2) \neq 1$ properties. Following the general definition of the MAC scheme the bilinear MAC is defined as in Figure 99. In the bilinear MAC scheme, the key generation algorithm KeyGen returns a random element of the group $\mathbb{G}_1$, i.e., K. In the second algorithm, i.e., in a tag generation Tag part, the first step hashes the message using a hash function that transforms a message of an arbitrary length into an element of the group $\mathbb{G}_2$, i.e., $H : \{0,1\}^* \rightarrow \mathbb{G}_2$. In the second step of the Tag a bilinear map is applied, in which the first input is the previously generated key K and a hash form the first step. This step generates the desired tag σ on the output. The third algorithm, i.e., the verification algorithm Ver takes as inputs the key K, the tag T, the message m and reassembles the Tag procedure to generate a new tag T'. In the last step Ver algorithm simply checks the correctness of the newly generated tag T'. It can be shown that the above-described definition of bilinear MAC scheme provides Existential Unforgability Under Chosen Message Attacks (EUF-CMA) security, the details of the prove can be found in [153].

In order to understand the leakage resilient MAC, we first provide overview of the key update mechanism which is used in the said MAC. The mentioned key update mechanism consists of four algorithms, $\mathcal{KU} = (\text{Share}, \text{Recombine}, U^{◓}, U^{◒})$ such that:

$$\begin{aligned}
(S_0^{◓}, S_0^{◒}) &\xleftarrow{\$} \text{Share}(K) \\
(S_{i+1}^{◓}, r_u) &\xleftarrow{\$} U^{◓}(S_i^{◓}) \\
S_{i+1}^{◒} &\xleftarrow{\$} U^{◒}(S_i^{◒}, r_u) \\
K_i &\leftarrow \text{Recombine}(S_i^{◓}, S_i^{◒})
\end{aligned}$$

Additionally it is required (for correctness) that for every K the following equation $\text{Recombine}(\text{Share}(K)) = K$ holds.

Initially, the key K is split onto two shares and these shares are further updated. The first share is updated by multiplying by random value, the second share is updated by multiplying it by the inverse of the random value. After each update, the key K_i is recombined by multiplying the two shares together.

```
proc KeyGen():
    $K \xleftarrow{\$} \mathbb{G}_1$
    $S_0^{◓} \xleftarrow{\$} \mathbb{G}_1$
    $S_0^{◒} \leftarrow K \cdot (S_0^{◓})^{-1}$
    Return $(S_0^{◓}, S_0^{◒})$

proc Tag^◓$(S_i^{◓}, m)$:
    $W \leftarrow H(m)$
    $t_i^{◓} \leftarrow e(S_i^{◓}, W)$
    $r_{i+1} \xleftarrow{\$} \mathbb{G}_1$
    $s_{i+1}^{◓} \leftarrow S_i^{◓} \cdot r_{i+1}$
    Return $(S_{i+1}^{◓}, r_{i+1}, t_i^{◓}, W)$

proc Tag^◒$(S_i^{◒}, t_i^{◓}, W, r_{i+1})$:
    $t_i^{◒} \leftarrow e(S_i^{◒}, W)$
    $S_{i+1}^{◒} \leftarrow S_i^{◒} \cdot r_{i+1}^{-1}$
    $T \leftarrow t_i^{◓} \cdot t_i^{◒}$
    Return $(S_{i+1}^{◒}, T)$

proc Ver$(K, T, m)$:
    $W \leftarrow H(m, w)$
    $T' \leftarrow e(K, W)$
    Return $(T' = T)$
```

Figure 100: Leakage Resilient MAC $\mathcal{M}^*$ [153]

4.10.2 Leakage resilient MAC

Having defined the key update mechanism we can now introduce the leakage resilient MAC scheme as it is shown on Figure 100. Following the general definition of MAC, the leakage resilient version consists of three algorithms: key generation KeyGen, tag generation Tag and verification Ver. The key generation mechanism KeyGen generates two shares by applying the following the procedure: firstly, the random element of the group $\mathbb{G}_1$ is generated together with a random share (which is also an element of the group $\mathbb{G}_1$). Furthermore, the second share is calculated by multiplying the preciously generated K by the multiplicative inverse of the first share and both shares are returned. The tag generation procedure Tag can be roughly split onto two main parts, i.e, Tag$^{◓}$ and Tag$^{◒}$. The first sub-procedure Tag$^{◓}$ takes the first share of the key and the message as the input. Then the hash function of the message is calculated, followed by the bilinear mapping, in which as the input the first share and the hash is used. The mapping generates the first share of the tag. The rest part of the sub-procedure preforms the share update, in which the next share of the key is generated. The second sub-procedure of the tag generation. i.e., Tag$^{◒}$ preforms similar steps: a bilinear mapping that takes as the input the second share and the second share update for the next tag generations. The output tag T is a multiplication of share tags obtained in bilinear mapping steps. The verification procedure Ver is the same as the bilinear verification procedure in bilinear MAC. The security definition only allows to leak on Tag and do not allow to leak on key generation, since this would leak the original key. More details about the prove and leakage models can be found in [153].

In the practical implementation the above-described leakage resilient MAC scheme might be realised using the Barreto-Naehrig (BN) [16] family of paring-friendly curves. The provisional evaluation results show that the scheme might be implemented on constrained devices, i.e., on RERUM gateways.

4.10.3 Summary

Message Authentication Codes (MAC) primitives are very important underlying blocks commonly used in cryptographic schemes and protocols. They prevent unintentional and intentional message modifications. Unfortunately, similarly to other keyed cryptographic primitives they might leak secret information via side-channels. Thus considering the mentioned problem, we proposed a novel leakage resilient MAC scheme that is designed to tolerate some leakage. Although other propositions of leakage resilient MACs exist, we argue that our design might be considered for practical applications. When applied it might improve security of employed protocols especially in embedded systems, where side-channels are more likely to be exploited. Improving security of underlying primitives and protocols would also lead to a better privacy for RERUM users.

4.11 Summary

In this chapter we covered the RERUM privacy enhancing protocols and mechanisms in detail.

Sticky Policies provide a soft privacy mechanism that enables service providers to be compliant with and to respect data subject privacy. Sticky policies can be attached to a data set, therefore carrying the data subject's privacy preferences to the data controller (service provider). The data set optionally may be protected until the service provider proves that it fulfils requested privacy requirements. Sticky policies can also be provided to a service provider unknown to the data subject.

We devised three new **Malleable Signatures** schemes specifically suited for RERUM devices. Malleable signatures offer reduced but lower-bounded integrity protection. At the same time they support the co-existence of privacy protecting changes in a cryptographically secure fashion. For the suggested scheme we provided rigorous proofs of their cryptographic security. The new schemes are currently implemented on RERUM devices.

In **Data Perturbation with integrity preservation on the gateway** we describe a mechanism that allows to balance privacy and integrity. We implemented this on an entity called Privacy Gateway. To counter the drawbacks of perturbation of trust towards the gateway we apply redactable signatures. By this the Smart Meter Operation can limit the amount of noise that is considered as acceptable.

We refined the **Privacy Policy Enforcement Point**, and based its implementation the Security Policy Enforcement Point to reduce performance costs. These two PEPs work with a different policy store each, one for access policies and another one for privacy policies. Additionally we upgraded them to combine multiple policies, which allows to support local and global policies.

RERUM provides the Privacy Policy Checker and Attribute Need Reporter. In **Enhanced privacy for user information retrieval** we explained in detail how they act together with the Identity Agent. Jointly they provide the ability to check the privacy of the user attributes referenced in the policies.

We developed a **Pseudonyms** generation and management mechanism based on Hash-Trees presented in Section 4.6. Our approach allows the generation of practically infinite hash values which we use as pseudonyms. It is an easy to use and battery efficient pseudonym data structure that allows RERUM components dynamical access to pseudonyms, fast and secure pseudonym agreement, management and revocation. The mechanism supports as well computationally efficient de-pseudomization.

In **Consent for authorisation** we present a method that allows a user to authorise clients to access resources on a constrained server, where the client can be a RERUM device. This is work-in-progress currently transferred into an IETF internet draft.

We provide a novel, privacy friendly approach for **Geo-location hiding** that allows traffic analysis in floating car observations for RERUM's Smart Transportation use case. RERUM's random vector generation analyses GPS positioning data and allows an accurate measurement for service providers as well as location and policy based privacy for users.

Compressive sensing allows lightweight encryption and compression in a single step. Although not perfectly secure it provide computational secrecy. Standard compressive sensing is vulnerable to Chosen Plaintext Attacks. We presented a method that is immune to this attack. Using a chaos sequence, we generate a secret sparsifying basis that is used as part of the encryption key. This introduces a small error for the receiver, but a significant error of 80% for the attacker.

We present a **leakage resilient Message Authentication Code**, which is resistant to side-channel attacks. This type of attacks should be especially considered in scenarios were embedded devices might be in a possession of untrusted entities. Our MAC design has a low computational overhead compared to other leakage resilient MAC designs presented in the literature and is therefore well suited for RERUM devices strengthening an underlying cryptographic communication protocol.

5 Privacy protection by the RERUM architecture

This chapter explains the interaction of the privacy functional components in order to achieve selected enhancements of the citizen's privacy in several situations. The same we do for those of the RERUM privacy enhancing protocols and mechanisms not part of some privacy functional component or detailing some aspect of such a component. These situations are taken from RERUM's four use cases:

UC-O1 Smart transportation

UC-O2 Environmental monitoring

UC-I1 Home energy management

UC-I2 Comfort quality management

For each of the use cases we state the overall goal that will be achieved and highlight typical privacy problems that these use cases might bring to the citizens. Then we show case how selected functional components, protocols, and mechanisms of RERUM will mitigate the privacy infringement, while still allowing the goals of the UC to be achieved with the data provided by RERUM in consent with the citizen. An overview of which RERUM privacy component, protocol, or mechanism is described to enhance privacy in which RERUM use case is given in Table 25.

Table 25: Privacy components may enhance privacy in the RERUM use cases.

Component/Mechanism	UC-O1	UC-O2	UC-I1	UC-I2
(3.1) User Consent Manager	5.1.1			
(3.2) Privacy Policy Enforcement Point				5.4.1
(3.3) Deactivator/Activator of Data Collection	5.1.2			
(3.4) Privacy Dashboard			5.3.1	
(3.5) Anonymisation and Pseudonymisation				5.4.2
(3.6) De-Pseudonymiser				5.4.2
(3.7) PET Geo-Location	5.1.3			
(3.8) Security functional components as privacy basis				
(3.9) Privacy Enhanced Integrity Generator / Verifier		5.2.1		(5.2.1)
(3.10) User Attribute Minimisation (PPC and ANR)			5.3.3	5.4.3
(4.1) Sticky Policies			5.3.4	
(4.9) Compressive Sensing		5.2.2		
(4.10) Leakage Resilient MAC			5.3.5	

Use case UC-O2 is the environmental monitoring use case. The data from UC-O2, if gathered only at a certain level of granularity, is not critical for privacy. In order to quickly reach a good coverage for

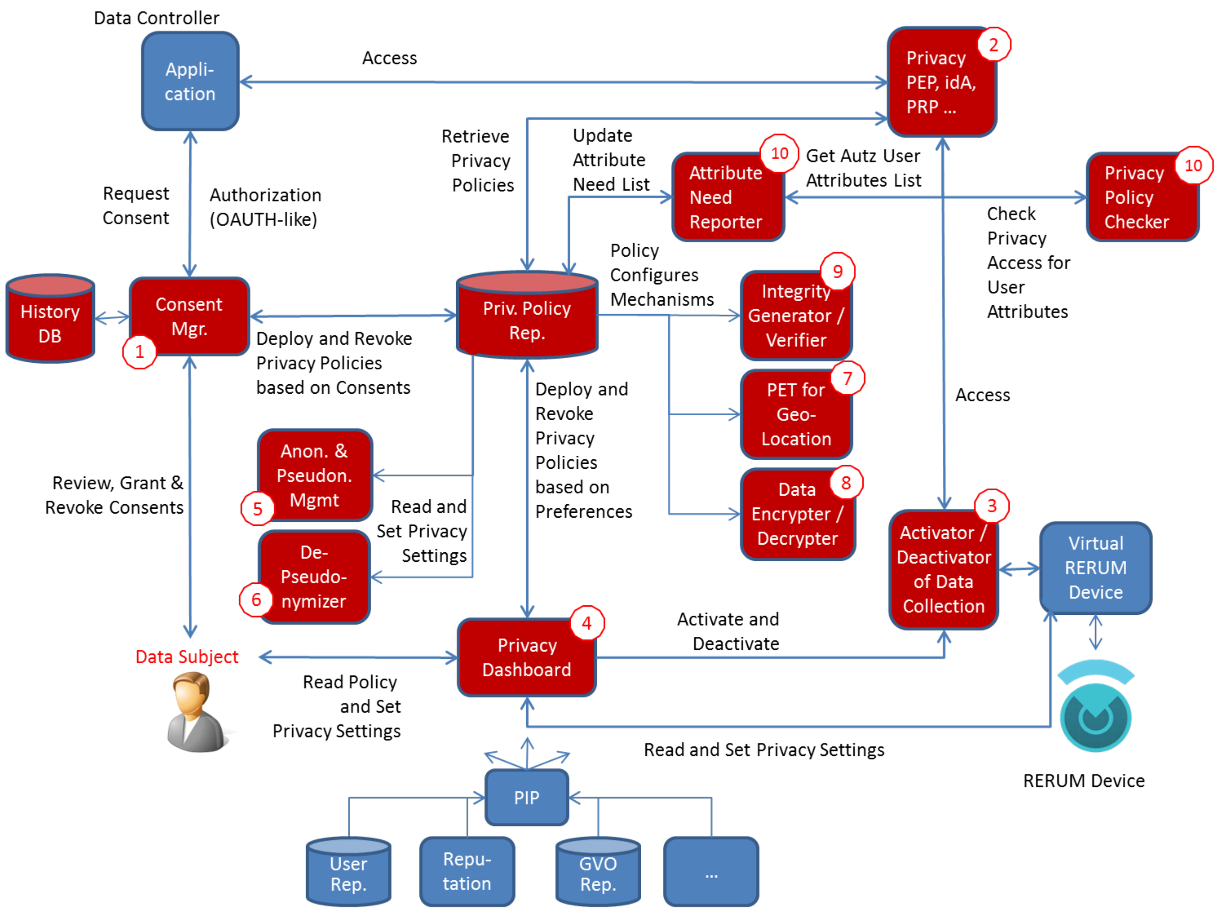

Data Controller
Appli-cation
Access
Privacy PEP, idA, PRP ...
2
Retrieve Privacy Policies
Update Attribute Need List
Attribute Need Reporter
10
Get Autz User Attributes List
Privacy Policy Checker
10
Request Consent
Authorization (OAUTH-like)
Check Privacy Access for User Attributes
History DB
Consent Mgr.
1
Priv. Policy Rep.
Policy Configures Mechanisms
Integrity Generator / Verifier
9
Deploy and Revoke Privacy Policies based on Consents
Deploy and Revoke Privacy Policies based on Preferences
PET for Geo-Location
7
Access
Anon. & Pseudon. Mgmt
5
Review, Grant & Revoke Consents
Read and Set Privacy Settings
Data Encrypter / Decrypter
8
Activator / Deactivator of Data Collection
3
Virtual RERUM Device
De-Pseudo-nymizer
6
Privacy Dashboard
4
Activate and Deactivate
Data Subject
Read Policy and Set Privacy Settings
Read and Set Privacy Settings
RERUM Device
PIP
User Rep.
Repu-tation
GVO Rep.
...

Figure 101: Interaction of RERUM Privacy Components

the monitoring of outdoor measurements on a city wide scale , the municipalities would be required to install a high number sensors themselves. As this is costly, the city council might want to facilitate sensors installed by building owners, e.g. sensors from UC-I2. RERUM facilitates this type of deployments. We will give such a scenario for a privacy preserving usage of UC-I2 sensor data for UC-O2 in Section 5.2.1.

In order to understand how privacy is achieved, let us recall that RERUM's understanding of privacy is based on the need of the data subjects' informed consent to data usage. RERUM adheres to the privacy-by-design principles and we want the RERUM system to be compliant to European data protection law[16]. Figure 101 depicts the ten privacy components and shows that the citizens as data subjects control their own privacy policies. These privacy policies play a central role in configuring and enabling many technical RERUM mechanisms. They allow data subjects to minimise data and control their acquisition.

[16]Note, RERUM's workplan does not foresee an actual legal evaluation of the design. This would however be an interesting interdisciplinary research project of its own.

5.1 RERUM UC-O1. Outdoor: Smart transportation

Goal of the use case is to use a heterogeneous network of sensors and smart objects and perform real time city traffic estimation and prediction. Use case lead is Linkopings Universitet.

The collected sensor data are stored in a central database, also to allow for historical analysis. For prediction a recognition of mobile sensor *pseudonym identity* is needed, which (in case of personal devices) may infringe data subject privacy. The original use case description indicated the intention to collect data *as raw as possible*, which might contradict the data minimisation privacy principle.

The sensors deployed record (and derive) motion, rotation, orientation, and time stamped location and speed. The location of city-owned sensors involves both fixed (placed in traffic lights and lamp posts) and mobile (placed in city vehicles, garbage trucks and taxis) sensors. Here mobile sensors in vehicles may infringe at least on the *driver's privacy*. Tarragona buses are already localised by GPS. Privately owned sensors involve smartphones of volunteers. Citizens and tourists can volunteer to contribute using their smartphones by installing a "RERUM app" on smartphone participates in sensing framework. This involves *profiling of users' movements*. Incentive for volunteers would be real-time traffic viewing and own traffic optimisation.

5.1.1 User Consent Manager

In the context of UC-O1, we are taking into account two types of data subjects that are conscious users of the use case.

Volunteers: This type of data subject participates in UC-O1 by downloading and installing the "RERUM UC-O1 App". Citizens and tourists can volunteer to contribute using their smart-phones. The "RERUM app" on the smartphone participates in the sensing framework profiling of users' movements. As incentives for the data subject real-time traffic viewing and own traffic optimisation have been quoted. The data subject needs to decide whether it is worth the effort and privacy loss.

Bus and Taxi Drivers: This type of data subject regularly drives a vehicle equipped with a "RERUM GPS UC-O1 device" that like the "RERUM app" participates in the sensing framework profiling of vehicle movements. As the drivers of the vehicle are observed by this in their driving habits, the sensor data qualify as personal data and require consent of the driver. Assuming the "RERUM GPS UC-O1 device" is physically connected with the vehicle and there are multiple potential drivers per vehicle, these devices would be multi-data-subject devices.

In the following we describe how the work-flows described in Section 3.1.6 to 3.1.11 may look like when applied to these sub-scenarios of UC-O1. We start with the most basic user: volunteers downloading and installing the RERUM app on their private smartphone.

5.1.1.1 Volunteers

A new data subject (here =user) elects to become a RERUM user. The user downloads and installs the RERUM app and provides the information required by the app. The user is registered in the RERUM user repository and assigned credentials to authenticate with. The smartphone of the user with its sensors becomes a (or several) RERUM device(s) and its (their) virtual counterpart(s) is (are) registered in the GVO repository together with its (their) specification(s). The registration information declares the user as both solitary *device owner and data subject*. A privacy policy is declared and deployed (secure default!) in the privacy policy repository. That privacy policy states that access to the new RERUM device(s) requires consent of the data subject.

As sketched in Figure 14, the data controller, here the city operating the RERUM UC-O1 application, tries to access such a new RERUM device. As the user hasn't granted consent yet, the data controller is redirected to the consent manager. The data controller retrieves the appropriate request for consent for the RERUM UC-O1 app and submits it to the consent manager under "cid". The consent manager with the help of the GVO repository identifies the user who is the data subject and registers the request for consent in that user's to-do-list.

The user, as depicted in Figure 15 logs into the consent manager and decides to review the request for consent "cid". The consent manager in cooperation with the privacy policy decision point analyses whether the hypothetical privacy policy generated from this request for consent would be obscured by current privacy policies derived from the user's privacy preferences. This special work-flow is depicted in Figure 16. For illustration we assume that the user has declared that the device location may only be provided in 100 square metre grids. The request for consent however declares the need for a 10 centimetre grid for proper operation. The consent manager provides a suitably enhanced and display-optimised version of the request for consent including the information about the potential conflict with the user's privacy preference.

The user reviews the request for consent and decides to grant an exception for that city that acts as RERUM UC-O1 data controller. For this, as shown in Figure 16, the user in the privacy dashboard grants this exception and the privacy policy derived from the user's privacy preferences is generated and deployed including the new exception. Then the user in the consent manager grants consent. The consent manager records this in its history DB, generates a corresponding privacy policy and deploys it in the privacy policy repository. Due to the exception in the dominating privacy policy derived from the user's privacy preferences the privacy policy derived from the user's consent can be effective.

The consent manager informs the data controller about the consent. The data controller can now successfully retrieve personal data about the user from the user's smart phone based on consent "cid1".

If we now assume, that the user decides to quit participating in RERUM UC-O1 officially, what would happen then? The user would cancel the RERUM user account. About this action the consent manager and the privacy dashboard would need to be informed, and, as discussed in Section 3.1.10, remove (or archive) their part of the user's account and withdrawing any privacy policy for that user. The device(s) of the user, i.e. the smartphone (or its sensors) need to cease being RERUM devices, their counterpart must be removed from the GVO repository, together with eventual settings made by the activator/deactivator of data collection for these devices on behalf of that user. As the user is solidary owner and data subject of this smartphone and its sensors, this will remove all settings for that device.

5.1.1.2 Bus and taxi drivers

For illustration we assume there is a bus driven regularly by five bus drivers. Preparation of the bus-and-taxi-driver scenario requires some preparation: Bus drivers 1 to 5 are registered as RERUM users 1 to 5 in the user repository. The GPS device is mounted in the bus and registered as RERUM device.

Its virtual counterpart is registered in the GVO repository. There device owner is (presumably) the bus company (or the city). Potential data subjects noted for that GPS device are bus drives 1 to 5. A privacy policy is generated (secure default!) that consent of the current bus driver as the current data subject is required.

The data controller, according to the work-flow shown in in Figure 14, asks the consent manager to accept a request for consent “cid2” for the new GPS device. The consent manager with the help of the GVO repository finds out that users 1 to 5 are potential data subjects of that GPS device and need to be asked for consent. The request for consent is registered in the respective to-do-list of these users.

Following the workflow depicted in in Figure 15, we assume for reasons of simplicity that the test for potential conflicts of the hypothetical privacy policy derived from the request for consent with the dominating privacy policy derived from the user’s privacy preferences does not result in conflicts. For illustration we pretend that user 1 to 3 decide to grant and users 4 and 5 to decline “cid2”. The consent manager records these consents in its history DB and generates and deploys the appropriate privacy policies in the privacy policy repository. If now the data controller tries to access the GPS device on the basis of “cid2”, it will be able to retrieve data if for instance user 2 is the current bus driver and access will be denied, if the current bus driver is e.g. user 4.

Now we assume user 4 decides after some weeks to revoke the denial for “cid2” because the user has reconsidered and now wants to participate in RERUM UC-O1 after all. As discussed in Section 3.1.10, the consent manager removes the corresponding privacy policy and invites the data controller place a request for consent “cid2” again. If the data controller decides to do so, following the work-flow in Figure 14, the request for consent “cid2” is placed on the to-do-list of user 4 again. When user 4, as sketched in Figure 15 grants the consent, the privacy policy is generated and deployed. Next time, user 4 is the current bus driver, the data controller can retrieve data from the GPS device referring to “cid2”.

Let’s pretend a new and hitherto unregistered bus driver now is the current driver of the bus. If the data controller wants to access the GPS device, the privacy policy decision point will find out that for access to the GPS device consent is needed by the data subject but the data subject is unknown. Therefore consent can’t be obtained. So access to the device is denied and no option to ask for consent is available.

If now this new bus driver registers as RERUM user 6 and again the data controller ties to access the GPS device with user 6 the current bus driver, now access is denied with the option to ask for consent following the work-flow in Figure 14. The consent manager will the place the request for consent “cid2” on the to-do-list of user 6, Then user 6 can, using the work-flow of Figure 15, decide whether to grant or reject it. If user 6 grants consent, next time user 6 is the current bus driver, the data controller can access the GPS device based on consent “cid2”.

If user 2 leaves the bus company and will not drive that bus any longer, user 2 is deregistered from the RERUM user repository. As discussed in Section 3.1.10, the references in the GVO repository are removed as well as all privacy policies by that user. The consent manager and the privacy dashboard can mark the account of user 2 as “historical” and maybe after a certain elapse of time remove the corresponding data.

Imagine, that a new bus driver, user 7, is a reserve pool employee and associated to a bus at short notice only. User 7 gets tired at granting consent for each new bus and decides to define consent granting preferences (see Section 3.1.9) to allow the consent manager to grant consent automatically for each new bus (=new data source), provided the request for consent in other respects remains the same as before. Then, when driving a new bus, the data controller would be able to access the GPS data of the new bus without the immediate need for user 7 to grant consent manually. User 7 can afterwards review the automatically granted consent updates in the consent manager.

5.1.2 Deactivator / Activator

In Section 3.3 we presented the Activator / Deactivator of data collection. Following the example of Section 5.1.1, we assume a user wants to quit temporarily the participation in RERUM UC-O1. In case of a permanent withdrawal, the user's account is deleted, the user's policies are removed and the user's devices are deleted from the GVO registry. If the user decides to re-enter participation he has to repeat the definition of policies, registering devices, etc.

The Activator / Deactivator of Data Collection allows a user to temporarily quit the participation. Following the steps depicted in 24 we extend the example of Section 3.3 for the case of a temporary withdrawal from the system.

5.1.2.1 User side deactivation of data collection in RERUM UC-O1

The user is participating in RERUM UC-O1, his consent and policies have been stored as described in the initial example. The RERUM App is collecting location data from the user's smartphone, calculating motion speed, motion vectors, acceleration, etc. (for a detailed description of the collected data see D2.1 section 2.1.3). The user, as the data subject in this use case, has the right of individual participation. He may allow or disallow data collection any time. He opens the Privacy Dashboard, which implements a graphical interface for the Activator / Deactivator of Data Collection (see section 3.4) and deactivates the collection of data for RERUM UC-O1. The setup of the Activator / Deactivator in UC-O1 is displayed in Figure 102.

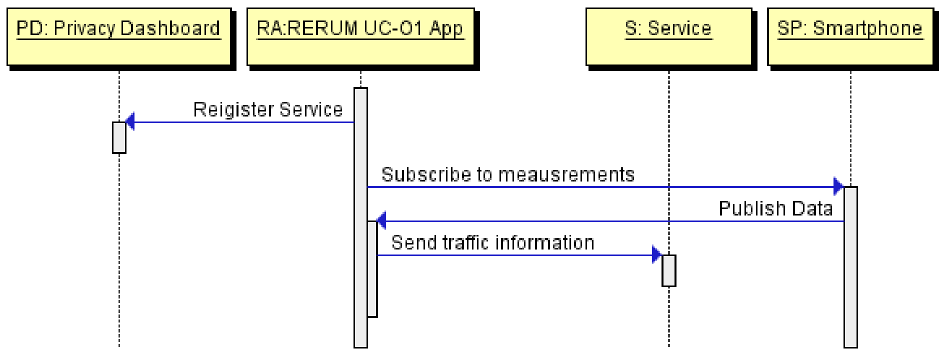


Figure 102: Setup of Activator / Deactivator for UC-O1

The interaction is straightforward, the RERUM App registers the new service in the Privacy Dashboard. The App subscribes to the smartphone, which publishes new locationd ata to the App. The App processes the data and forms traffic information, that is sent to the traffic service.

The sequence in Figure 102 differs hereby from Figure 24 as the data collection is not handled by the RERUM Middleware, but by the RERUM App. The App may directly accept authorized commands from the user's Privacy Dashboard, in contrast to an app developed by external developers. In this case, the data collection will be stopped at the Data Collector as shown in Figure 24.

We assume now that the user gives the command to stop the data collection. The sequence is as follows:

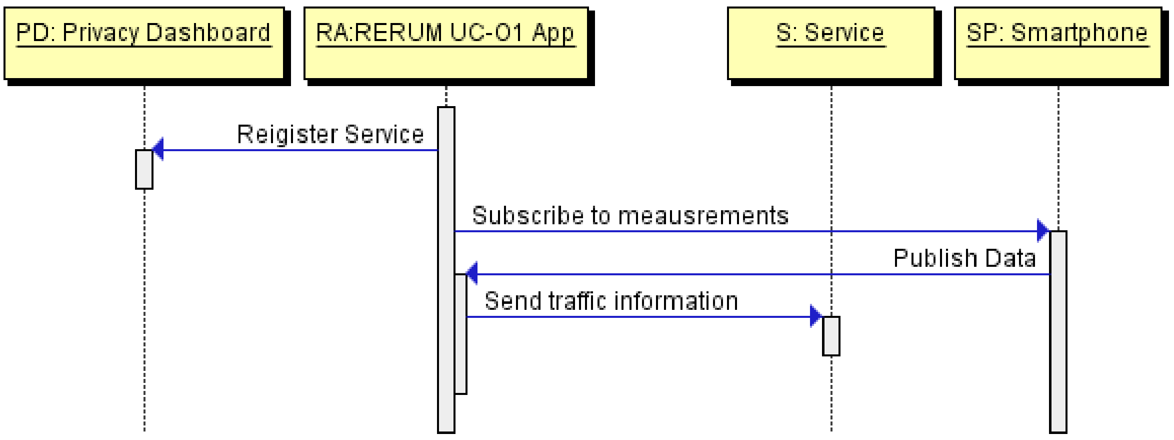


Figure 103: Interplay of Activator / Deactivator for UC-O1

The Privacy Dashboard forms a command, according to the wish of the user to stop the data collection, and sends it to the RERUM App. The App unsubscribes from the publishing service of the smartphone and optionally sends a message to the service provider, that no data is available (the reason for this is described in Section 3.3.2).

In case of reactivation, the would repeat the steps in Figure 23 and re-register itself as an active service and resubscribe to the smartphone's traffic data publish service.

5.1.2.2 Deactivation of data collection for bus/taxi drivers in RERUM UC-O1

We follow again the setup of the example given in Section 5.1.1. Five bus drivers are registered as RERUM users, they drive alternately a bus equipped with a RERUM GPS UC-O1 device. Drivers 2 and 4 have not consented to the participation in RERUM UC-O1, while the others have.

Figure 23, the collection of data does not take place for drivers 2 and 4, until their consent is given. Thus the activator / deactivator of data collection does not appear on their Privacy Dashboard for this use case.

For drivers 1, 3 and 5, the Activator / Deactivator behaves as described as in Figures 102 and 103. The drivers may deactivate the data collection in their respective Privacy Dashboard, which sends a command to the RERUM GPS UC-O1 device (instead of the RERUM App). The collection will be stopped for the respective driver.

A new sequence is given for alternating drivers: We assume driver 1, which has allowed data collection by employing the Activator / Deactivator, leaves the bus and driver 3, which has stopped data collection by means of the Activator / Deactivatior before, enters the bus.

The settings of the Activator / Deactivator of data protection are stored as policies in the policy repository as well (see Section 3.3). Therefore, whenever a new driver enters the bus, the data controller, i.e. the traffic service provider, has to check the privacy policies of the driver. Figure 24 illustrated the relationship between policies, underlining that the Activator / Deactivator policies resemble the user's wish of participation, overriding any previous consent. Figure 104 sums up the sequence for alternating drivers.

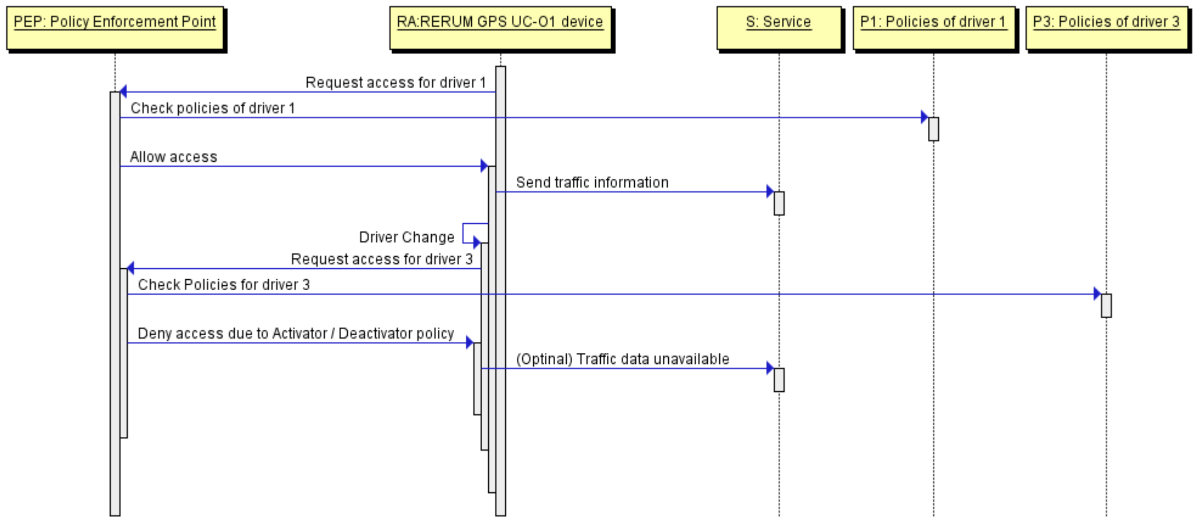

PEP: Policy Enforcement Point
RA:RERUM GPS UC-O1 device
S: Service
P1: Policies of driver 1
P3: Policies of driver 3
Request access for driver 1
Check policies of driver 1
Allow access
Send traffic information
Driver Change
Request access for driver 3
Check Policies for driver 3
Deny access due to Activator / Deactivator policy
(Optinal) Traffic data unavailable

Figure 104: Sequence in case of alternating settings for the Activator / Deactivator in UC-O1

It should be noted that in case of external devices, the RERUM GPS UC-O1 device would be exchanged by the Data Collector of the RERUM Middleware, as described in Figure 24.

5.1.3 PET geo-location

The geo-location privacy enhancing technology is applied on the RERUM App or the RERUM GPS UC-O1 device. A *user* downloads the RERUM App and installs it on his smartphone. The smartphone publishes the GPS position of the user in short periods. The RERUM App converts this data to motion vectors as described in Figure 90. The App handles the random timers and the transmission of vectors. The RERUM App also handles geo-privacy policies as described in Figures 91 and 92.

A *bus or taxi driver* does not have to install or activate the technology. It is available on the RERUM GPS UC-O1 device by default. It is responsible for generating vectors and handling policies.

5.1.3.1 Traffic analysis

The geo-location PET depends on traffic anonymization techniques to hide identifiers such as IP- and MAC-addresses or timing attacks (see Section 4.8.6). For a *user* we assume that the user's smartphone sends messages over an anonymization network. For example, the user could install the Orbot application (see [28]) and either configure the RERUM App to send messages over the Orbot client or the user configures the smartphone operating system to route all traffic over the TOR-network, utilizing the Orbot client. Optionally, the RERUM App could support the user in installing and configuring an anonymous networking client. A anonymous networking client could be integrated in the App as well.

To ensure the same behaviour for the RERUM GPS device, an anonymous networking client has to be integrated in the gateway handling the message traffic of the bus / taxi.

5.1.3.2 The geo-location Privacy Enhancing Technology in RERUM UC-O1

The following sequence diagram explains interplay of components to ensure the privacy for geo-location data. As pointed out above, the same mechanisms apply the cases of both user and the bus / taxi drivers.

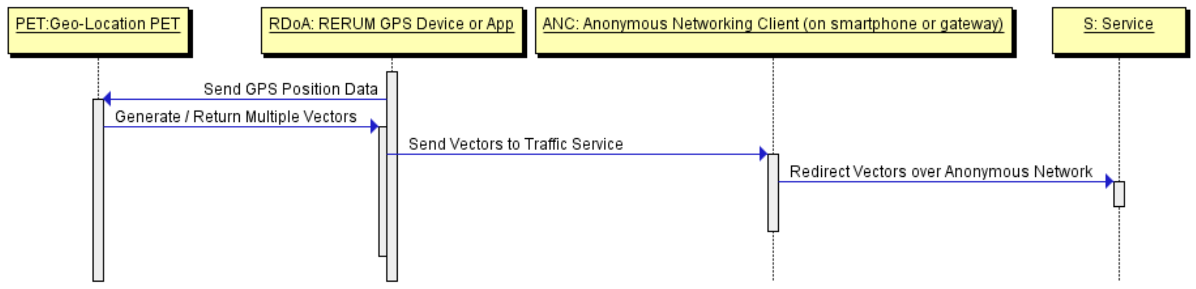

PET:Geo-Location PET
RDoA: RERUM GPS Device or App
ANC: Anonymous Networking Client (on smartphone or gateway)
S: Service
Send GPS Position Data
Generate / Return Multiple Vectors
Send Vectors to Traffic Service
Redirect Vectors over Anonymous Network

Figure 105: Interplay of Gep-Location PET in RERUM UC-O1

The sequence starts after the user grants access to his location data. The RERUM GPS UC-O1 device's or the smartphone's GPS module continuously grab the user's location data. This data is sent to the PET component on the device/smartphone. The PET component generates a data set according to Figure 90 and Table 23. The data set is encrypted and integrity protected by the RERUM GPS device / RERUM App and sent to the anonymous traffic client. The client redirects the data set over the anonymous network to the traffic provider.

The data set is free from identifiers (see Section 4.8.6 for details) but it still provides an adequate level of granularity nonetheless.

Figure 105 also shows that the suer does not have to apply any effort to protect his location data, making the PET transparent and user-friendly for RERUM UC-O1 participants.

5.2 RERUM UC-O2. Outdoor: Environmental monitoring

Goal of this use case is to perform continuous measurements for pollution in city environments, to focus on outdoor environmental measurements, and to put data into graphs and *provide them to public* on a server. Here for instance identifiable sources of environmental impact (like noisy inns) may feel monitored. Use case lead is Zolertia.

Sensors measure temperature, humidity, air pollution, radio pollution (electro magnetic fields), air quality, noise, radiation, weather and many other environmental data. Tarragona is already using the SIRUSA and EMATSA City Council Agencies' sensors. Location of city-owned sensors involves both fixed and mobile ones. Many sensors will placed in parks, squares, congested areas, etc. Mobile sensors will be places on public vehicles (like in UC-O1), however sensing will be performed only when the vehicle is stationary. Heraklion does not intend to publish health related measurements so as not to worry citizens and noise relates values so as not to annoy innkeepers.

5.2.1 Combined hybrid deployment: Authentic UC-I2 sensor data for UC-O2

Assume that the municipality wants to quickly cover large areas of the city to read a certain environmental value, e.g. the temperature. In order to quickly reach a good coverage for the monitoring of outdoor measurements on a city wide scale, the municipalities wants to facilitate sensors installed by building owners, e.g. sensors from UC-I2. RERUM facilitates this type of deployments.

Note, that monitoring the temperature is just an example of a possible measurement, others are possible, e.g. noise or CO_2 emission. For this example we need a large number of trustworthy (e.g. certain grade of reliability and accuracy) sensors that building and home owners had bought off the shelf and installed in their homes. Lets call them HQ sensors for the remainder of this example, which stands for trusted high quality sensor. As there is an increasing number of products for the home on the market, RERUM sees this a a way to quickly and cost effectively cover the city area. Also, the municipalities could, together with a sensor manufacturer, offer the sensors for sale to their citizens.

In general, RERUM facilitates this type of deployments. We had depicted a hierarchical deployment scenario scenario, in RERUM Deliverable D2.3 Figure 83 [219]. In Figure 106 we have in a similar fashion put two UC-I2 indoor deployments that allow the city to access their data. Assume that the sensors are very precise and can push the temperature in the precision of 10^{-2} degrees, e.g. 23.45°C, every 30 seconds. Now this data might be of interest in this granularity and in this high data quality for the building control. In Figure 106 we have highlighted, that the RERUM architecture facilitates access control such that only the authorised owner of the deployment Indoor1 can access this data. The RERUM architecture uses VRDs. At the level of VRDs, a virtual sensor does not require to be fed from a local data source. In the hybrid / hierarchical deployment VRDs like VRD#1.3 gets the data from Service#1.1 which originate from the physical sensor. But not only is RERUMs architecture with its virtualisation of devices and access control capable of doing this.

Regarding privacy, the data quality is much too high. We assume we need far less information, e.g. only one reading per hour and only in the range of -10 to 0 to 10 to 20 to 30. Hence, the principle of data minimisation requires to reduce the resolution before the data leaves the RD. Now of course we can instruct the service to reduce the resolution. This is depicted in Indoor deployment number 2 at service#1.3.

Now if we additionally want to make sure that the municipalities service is only fed with trustworthy data, e.g. data originating from authentic HQ sensors, then we need to ensure integrity and authenticity. To gain this, RERUM has the Integrity Generator. For example, RERUM enables to sign the data on the sensor, which is done on the RD#1.1 for scenario Indoor#1. This way the data is offered the best protection, as it is applied early on. Of course, we can also add a signature later at the MW before the data leaves the service, as depicted in service#n.3 we can also assume that the HQ sensor might be adding a redactable signature to protect the data from arbitrary tampering and to allow to authenticate the origin, e.g. verify that it was indeed a sensor from a certain manufacturer.

To reduce the resolution and enable information blurring, the signed readings need to be redacted. Namely, if the RD#1.1 sensed and signed the temperature value of 23.45°C with a timestamp of $12 : 45 : 39$ (format is HH:mm:ss). A redaction in resolution for the required lower quality of service#1.3 means to do a redaction to 2■.■■°C with a timestamp of 12 : ■■ : ■■. The privacy enhanced Integrity Generator / Verifier with malleable signatures can be facilitated to achieve this.

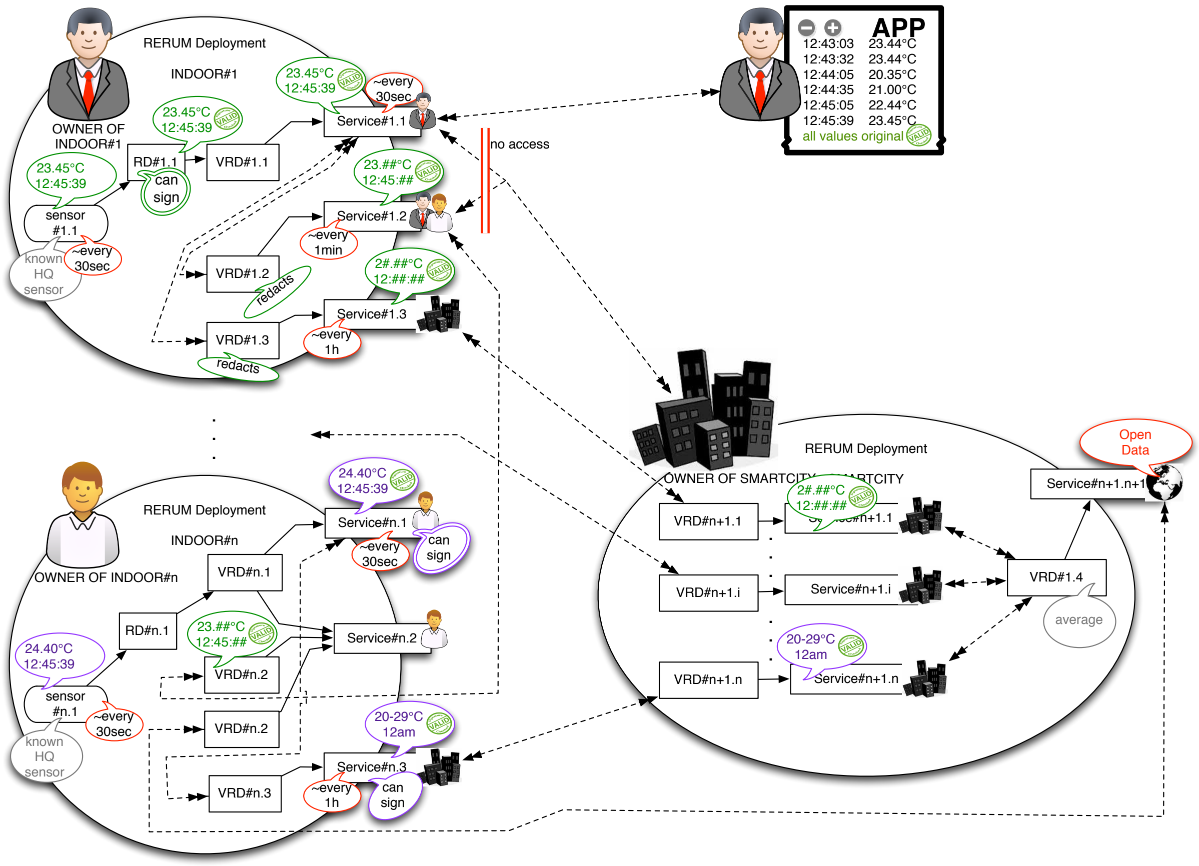


Figure 106: facilitating an RSS to adapt the resolution by redaction of sensor readings to preserve privacy while allowing the authenticity of the blurred information still being verified in a hierarchical deployment

5.2.2 Compressive Sensing

Recall from Section 4.9 that CS offers lightweight encryption and compression in a single step. CS has a good performance, in terms of the reconstruction error, for sparse data, so this makes it suitable for data compression/encryption of environmental data that are usually sparse. The parameters that have to be initialised in this UC are the following:

- the compression rate that affects the trade-offs between the reconstruction error and the transmission energy consumed, as well as the strength of the encryption
- the measurement matrix used
- the initial parameters of the chaos sequence regarding the sparsifying basis (see Section 4.9 for more details)

5.3 RERUM UC-I1. Indoor: Home energy management

Goal of this use case is to *monitor* and control the energy consumption and to reduce energy consumption of several devices in public buildings. Subgoals involve device energy monitoring, gathering of data to databases, development of web services, mobile applications, and portals for services, as well as comparison of energy consumption. Use case lead is CYTA.

The Heraklion trial involves 2 public municipal office buildings, however the use case also potentially to be deployed to *private homes* watching the behaviour of known residents. The buildings to be monitored in Heraklion are the Vikelaia Library (a very new building with own BMS) and the Androgeo building (a very old building). Heraklion is interested in A/C, window status and light control. Potential privacy issues even in public buildings may arise in shared spaces in apartment buildings and private offices in public buildings, resulting in behaviour tracking of staff, occupants, and visitors.

Landlord-owned sensors are attached to high powered devices and to lights. They measure energy consumption, light; also temperature, humidity, water flow, motion, and many other data. Actors may switch on and off lights, regulate A/C, heating, hot water, among other things.

5.3.1 Privacy dashboard

RERUM UC-I1 will allow users to remotely control and monitor their devices, manage energy savings and overall support the efficient operation of the electrical grid. The user interface envisioned in this use case is the user's smartphone, tablet or laptop. Generally, the user interface visualizes the consumption of data, sends commands to smart objects inside the home and process policies to do this automatically. The user interface, often called Energy Management System, is a mash-up of different services and options, tailored to the appliances found in a smart home [39].

This resembles very much the behaviour of the Privacy Dashboard (see Section 3.4), where the EMS allows policy definition for which state which appliances should start or finish when energy prices are high or low. The Dashboard contains policies which define how often, to whom and in which granularity energy consumption data should be published. Extending the mash-up concept, the Privacy Dashboard can be integrated into the RERUM EMS App for UC-I1, with a common policy generation interface.

5.3.1.1 Example

We assume a *user* is a participant of RERUM UC-I1. The user has consent and agreed to the terms of service of a service provider, which takes the user's consumption data, analyses it, and gives energy optimization feedback of which appliances are consuming the most, when to turn them on due to low energy prices, and when to sell energy back to the grid.

The user himself has also own priorities, he doesn't want some appliances to shut down even if prices are high, etc.

The user accesses his EMS App to define the policies for energy consumption. He also defines which appliances are going to be analysed by the service provider. Typical options are:

- Which appliances are subject to analysis by the service provider?
- Should applications be turned on and off following the service provider's suggestions?

- How often should the service provider analyse the energy consumption?

Every of the options above are privacy sensitive. Information about appliances give insight into the users everyday habits (e.g. ”When is he watching TV?” or ”When is he cooking?” or ”When is nothing turned on, meaning that the user is not at home?”). If the service provider’s suggestions are followed, they can complete the information of the energy consumption giving insight on if or if not the user acted accordingly (e.g. ”did the user sell energy stored in his e-car on monday evening?” and the conclusion from the energy consumption ”No his car was plugged two hours later for recharge.”).

The Privacy Dashboard could extend the options above with privacy relevant information, such as:

- Service Provider has been given consent to analyse the following appliances
- The consumption of following appliances are published to the service provider with
  - a granularity ”x”
  - a frequency ”f”
- Following appliances have preference polices

In RERUM UC-I1, device identities are pseudonymised. The correlation of identities and pseudonyms is done by the anonymization and pseudonymization manager, the re-linked identities are shown in the Privacy Dashboard. The flow between service provider, anonymization and pseudonymization manager and privacy Dashboard is shown in Figure 107:

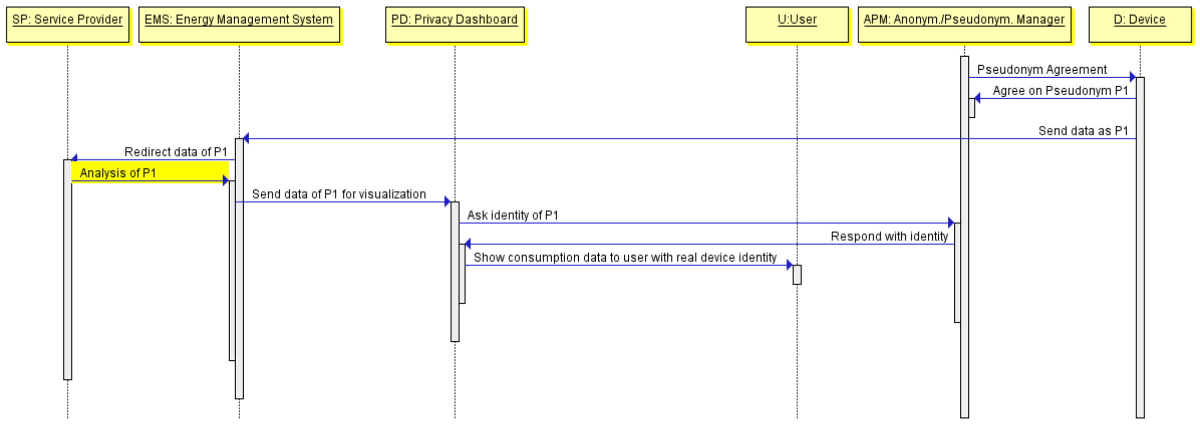


Figure 107: Interworking Privacy Dashboard, EMS and Pseudonym Management in UC-I1

5.3.2 Malleable Signatures

Our published case study in Section 3.9.1 gave an overview of the differences in aggregation, resolution reduction and perturbation of real-life energy consumption data. We gathered the data from the family household of one RERUM participant as raw data, not from some trials. We informed the household inhabitants about the impact and obtained consent from all members of the family. Additionally, we automatically obtained the uptime of certain IP-enabled appliances, e.g., SmartTV, and because the inhabitants kept diaries, we obtained a ground truth to identify which actions correlate to consumption data. The rising quality of the gathered data which increases the sensitivity of the recorded data to be privacy invasive. For this case-study we devised relatively simple threshold driven machine-learning algorithms to extract features about the behaviour from the energy consumption data. Even with the compression property of aggregation or the noise introduced by perturbation the presence detection still works quite accurately ($> 74\,\%$). It is worthwhile to note, that although simple presence detection is still feasible on the processed data set, more detailed inferences requiring higher temporal or energy-level details are clearly aggravated. Thus, removing the fine grained values shall be one goal, as they impose a privacy threat to the residential customer. Examples are too fine-grained energy values that allow detecting appliances within the household [165], detecting the use mode of the appliances [80], or deducting behaviour [148]. Existing de-pseudonymization is feasible when it comes to the Smart Grid as a whole, however pseudonymization is vulnerable to linkage attacks [124].

The loss of data quality that purposefully occurs with all methods from the case study is data minimisation. Thus, it can always be seen as a privacy gain, e.g. we showed that presence detection within an interval of 4 hours is still achievable with far lower data quality. Even more interesting, if you consider our simple extraction algorithms. As a result we conclude, that for each IoT application's well defined purpose — and purpose must be defined to operate within the EU's legal boundaries — you must carefully validate if you could not offer the same service with less data.

While it might be hard to retro fit the actual hardware sensors to give less accurate readings, we think that in the MW or even on the RERUM GW we can reduce the data quality, e.g. current consumption is 2■■■■ mW. While we want to modify the original sensor reading for privacy, we might still want the remaining information to be trustworthy. In the simple example this means that while the consumer used some calculation by some trusted privacy component to perform the data perturbation to protect his privacy, the energy provider would like to base a decision on the current consumption of 2000-2999mW. The main point we would like to raise is that the entity trusted to generate data could be controlled and trusted by a stakeholder other than the data subject, e.g. building manager or energy provider. Their goal might be to gather trustworthy and as fine-grained data as possible, but in general we are not convinced that such a third party will become trusted to maintain the consumer's privacy. Vice versa, the stakeholder will not be able to rely on data gathered by an untrusted consumer-controlled device.

We have published this scenario and the malleable signature based solution in the SmartGridSec workshop at ESSSOS in 2014 [184].

5.3.2.1 Opposing players and different trust in components

Figure 108 depicts the situation of opposing trust in different devices by the different stakeholders in a smart grid scenario. It is taken from our publication.

The figure shows that there is conflicting interests of privacy and integrity[17]. This needs to be balanced. We follow an approach called data perturbation, which is widely used in the field of privacy preserving data mining and differential privacy [75]. We will call this entity the *privacy gateway* (PGW). The downsides of data perturbation are twofold:

- First it obviously must result in a reduced data utility, and
- second the data tampering entity must be trusted.

[17]which here includes accuracy

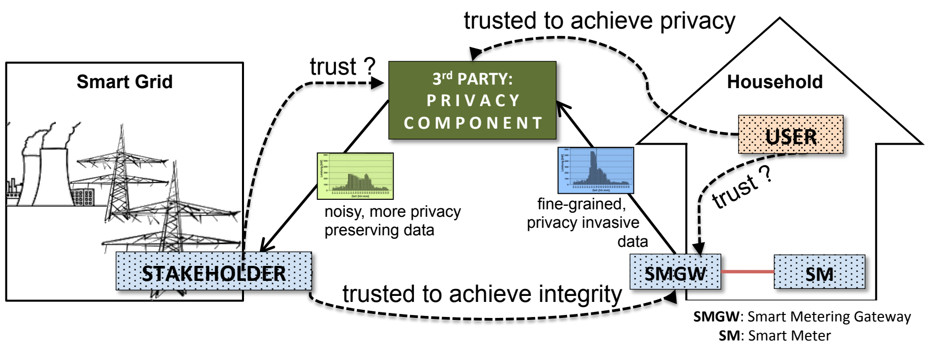

trusted to achieve privacy
Smart Grid
trust ?
3rd PARTY: PRIVACY COMPONENT
Household
USER
trust ?
noisy, more privacy preserving data
fine-grained, privacy invasive data
STAKEHOLDER
SMGW
SM
trusted to achieve integrity
SMGW: Smart Metering Gateway
SM: Smart Meter

Figure 108: Trust towards components between the SG stakeholders and the privacy-aware household [184]

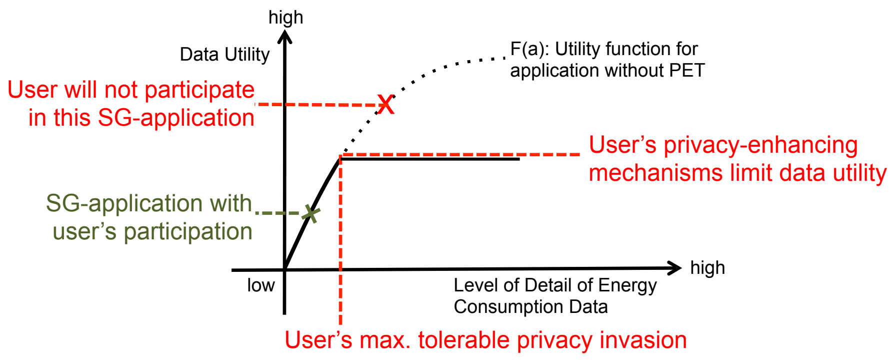


Figure 109: Data-utility might be hindered by PET, but some applications might still be possible [184]

For RERUM we must remain flexible and open towards future IoT applications and their need for data utility. Also, consent can be given on the level of each individual data subject's privacy tolerance. Figure 109 shows that applications are possible, if they require a data quality that is below the citizen's privacy preference, which limits the maximum data utility. The reduction of data quality and the tolerable amount of redaction is different among applicable PET-algorithms and among different application domains. The second downside, however, remains the same across all applications that are in need of performing modifications to the data. Here RERUM offers a solution by applying a redactable signature instead of a classical digital signature at the RERUM device.

5.3.2.2 Contribution of Malleable Signatures

In a nutshell the idea is to let the RERUM device, assumed to be trusted by the stakeholder, sign a *range of values* around actual energy consumption using a redactable signature scheme ($\mathcal{RSS}$), but allow the residential customer's *privacy gateway* (PGW) to tamper with the data by choosing one out of the signed range. As figure 110 shows, just adding a small amount of random noise is not enough. The mathematical concept of differential privacy was given in [75]. In our paper [184], jointly written with Markus Karwe from the iUrban EU project[18], we took the theory of differential privacy [75]. It provides

[18]`http://www.iurban-project.eu/`

mathematical model to give an ad omnia privacy guarantee of privacy by calculating the noise needed to perturbate the data. We will not discuss the mathematical foundations of differential privacy in this deliverable as we see this as one particular suitable PET algorithm for data minimisation, but in general they are out of scope of RERUM.

The advantages we gain are all the advantages of data perturbation combined with that of redactable signatures:

1 data perturbation still allows the stakeholders to address RERUM devices individually allowing for all applications, e.g. they could provide energy efficiency recommendations;
2 data perturbation is allowed and allows PET, e.g. differential privacy, to modify raw data in order to do data minimisation;
3 redactable signatures allow the verifier to gain reassurance that the RERUM device actually signed this value. Hence, the signing RERUM device is instructed by the owning stakeholder to set limits to allowed modifications and thus defines the required data quality;
4 redactable signatures allow the PGW to be an independent third party; the PGW can do the choosing without any interaction with the signing RD; it can be done by any party trusted by the data subject.

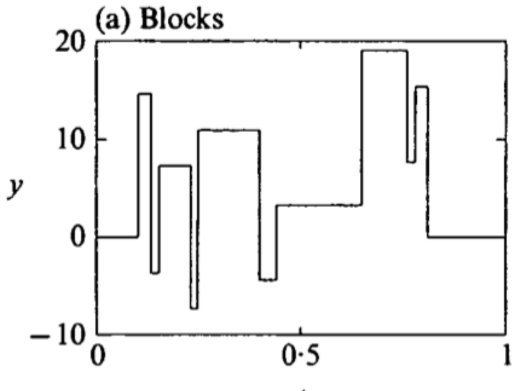
5.3.2.3 Solution Sketch: Signing a range with an $\mathcal{RSS}$

With an malleable scheme like the $\mathcal{RSS}$, we provide the application stakeholders with signed and henceforth trustable sensor values, e.g., energy consumption values. At the same time, we allow the customer to achieve a desired level of privacy, by allowing the energy consumption value to be tampered with, e.g., adding noise. The party running PETs to achieve the consumer's privacy is termed Privacy Gateway (PGW). It is not important where this PGW is running, it could be completely outsourced to a third party.

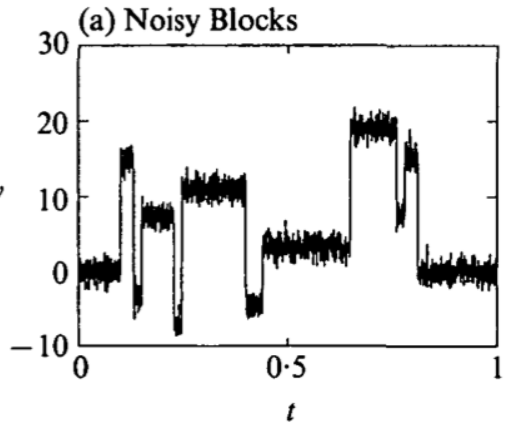
Note that it is the stakeholder who knows and requests a desired level of data utility. This means in case of perturbation by noise to limit the maximum allowed noise. Our solution allows the party doing the addition of noise to be trusted to preserve the customer's privacy, as the customer remains in full control. The task of the PGW is to tamper energy consumption values in order to protect the privacy of residential customers. The task of the RERUM device is to sign the energy consumption values and the maximum tolerable perturbation in order to protect the integrity and trustworthiness of the RD's sensor readings. Both devices act on behalf of different parties: the RD on behalf of the stakeholders and the PGW on behalf of the citizen/consumer. Hence the devices are in different trust zone. Our solution uses redactable signatures to solves this conflict.

For brevity, we will now focus only on the transmission of a consumption value, other information that the RERUM device might be sending alongside, like timestamps, are not considered.

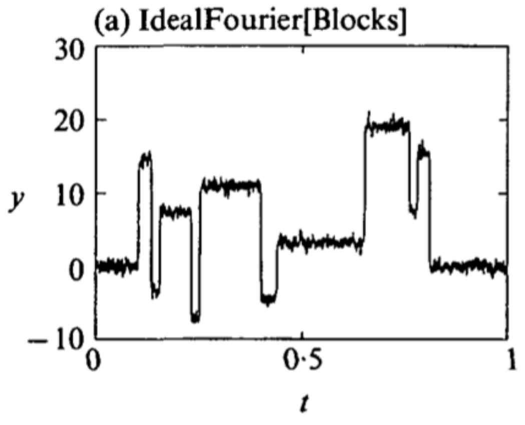
The RERUM device that senses the energy consumption must make sure that values are not tampered in an unauthorized malicious way. Depending on the application the application provider can tolerate a certain level of inaccuracy, e.g., allow that a certain amount of noise degrades their data utility. We denote the maximum amount of noise that can be added to an accurate reading by δ_{max}. Assuming the actual consumption value to be v, then the application provider will accept any reading in the range $[v - \delta_{max}, v + \delta_{max}]$ as valid. An application of a classical signature scheme on v would mean that



(a)



(b)



(c)

Figure 110: a) Original b) Random Noise Added c) Random Noise Removed (all taken from [71])

the PGW tampering with data signed by RERUM device will always invalidate the signature. An invalid signature would indicate towards the application that the received value is not trustworthy, as it could have been maliciously tampered with in an arbitrary way. Henceforth, we assume that the RERUM Device will be instructed by the application about the tolerable noise, on behalf of the stakeholder. This tolerable noise depends on the required accuracy level for the stakeholder's application.

Note that fixing $\Delta = 2\delta_{max}$ in definition 34 allows calculating the maximum differential privacy that can be achieved. The PGW must be instructed by the data subject which level of privacy is tolerable for which optional applications.

5.3.2.4 Protocol Description

We propose the following phases: Setup, Signing, Adding Noise and Verification.

Setup:

1. Let $\mathcal{RSS} := (\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Redact})$ be a secure (unforgeable and weakly private) redactable signature scheme.
2. After running KeyGen distribute the keys: RERUM Device sensing energy consumption gets a secret signing key *sk* and verification key *vk*, PGW and application get just the public RERUM Device's verification key *vk*.
3. RERUM Device sensing energy consumption is instructed by SMO which amount of noise it tolerates, and which accuracy is required.

Signing:

1. On receiving the actual consumption value v the RERUM Device sensing energy consumption calculates a range of discrete noisy values $= \{v - \delta_{max}, \ldots, v, \ldots, v + \delta_{max}\}$.
2. SGM signs with an $\mathcal{RSS}$: $(, \sigma) \leftarrow \text{Sign}(1^\lambda, \textit{sk},)$.
3. RERUM Device sensing energy consumption sends $(, \sigma)$ to PGW.

Adding Noise:

1. On receiving $(, \sigma)$ PGW uses its database of historic values and the actual consumption value, which must be at the center of the range in , PGW runs the differential privacy algorithms to identify the value n in which should be sent to application in order to satisfy $\frac{Pr(K(D_1)\in S)}{Pr(K(D_2)\in S)} \leq e^\epsilon$ where ϵ is a user predefined minimum required privacy parameter. The application execution is denied, if ϵ can not be reached.
2. PGW calculates $\mathcal{R} = \backslash n$.
3. PGW obtains a signature on $' = n$: $(', \sigma') \leftarrow \text{Redact}(1^\lambda, \textit{pk}, , \sigma, \mathcal{R})$.
4. PGW sends $(\{n\}, \sigma')$ to the application.

Verification:

1. On receiving $(\{n\}, \sigma')$, application uses the RERUM Device's verification key *vk* to verify if the signature on n is valid.

Note that the amount of elements in depends on the maximum noise and the accuracy, as must contain concrete values, e.g., $= \{0.99, 1.00, 1.01, 1.02, 1.03, \ldots, 1.48, 1.49, 1.50, \ldots, 1.96, 1.97, 1.98, 1.99\}$ for an accuracy of two decimals, $\delta_{max} = 0.50$ and $v = 1.49$. The $\mathcal{RSS}$ limits the PGW only to redactions based on provided values, e.g., for $= \{1.11\}$. The PGW could generate a valid signature facilitating the algorithm Redact. However, the PGW can not generate valid signatures on values outside the range, e.g., $= \{0.98\}$ or $= \{2.00\}$. To do so would be as hard as forging the signature scheme of the $\mathcal{RSS}$, e.g., breaking the signature scheme like RSA-PSS [20, 198]. To counter replaying or repressing messages, the RERUM Device sensing energy consumption can just add a timestamp as an additional element into requiring this to be fresh and present during verification.

5.3.2.5 Security and Privacy Properties

We assume: RERUM device is trusted to perform correct readings, can not be attacked, and transmits the reading securely to the PGW.

Theorem 27. *Our protocol is unforgeable, if the $\mathcal{RSS}$ is unforgeable.*

SG stakeholders can detect any subsequent malicious manipulation of information while it is travelling through the network. Additionally they can use the RERUM Device's verification key to identify the origin of noisy data.

Theorem 28. *Our protocol achieves the highest possible differential privacy possible for $\Delta = 2\delta_{max}$, if the $\mathcal{RSS}$ is at least weakly private.*

5.3.2.6 Proof Intuition for Th.27

If the $\mathcal{RSS}$ applied by the RERUM Device sensing energy consumption is unforgeable, than neither PGW nor attackers can forge a valid signature on a value $n^* \notin i$, where i denotes all sets signed and sent by the RERUM Device. Any such forgery would be a forgery in the $\mathcal{RSS}$.

5.3.2.7 Proof Intuition for Th.28

Assume all communication from RERUM Device sensing energy consumption will always pass through PGW, see Fig. 74. The $\mathcal{RSS}$ allows PGW to be a separate entity acting as instructed by the residential customer. PGW is limited by the range defined within the RERUM Device's signature but can run the algorithm Redact to select any suitable value out of the range. So seeing a valid $(, \sigma)$, which verifies using Verify under the trusted public verification key of a RERUM Device, that no malicious modification has taken place. Privacy of the underlying $\mathcal{RSS}$ guarantees that attackers can not identify the actual value of removed elements. Hence attackers can not know the actual consumption. We distinguish two cases:
(1) If the $\mathcal{RSS}$ is strongly private, i.e., elements are completely removed during redaction, then the attacker sees a set with exactly one element, i.e., $|| = 1$.
(2) If $\mathcal{RSS}$ is weakly private, i.e., original values are hidden behind a special symbol ($\blacksquare^r$), then the

attacker sees a set with exactly one element being an actual value and $2\delta_{max}$ symbols, i.e., $|| = 2\delta_{max} + 1$.
Hence, if $\mathcal{RSS}$ is weakly private attackers can infer δ_{max}. However, attackers do never learn the actual values of removed elements. Using the differential privacy mechanism described in Section 4.3.6, PGW adds noise within the range guaranteeing a differential privacy of ϵ.

5.3.3 User Attribute Minimisation

UC1 shows a hierarchy of users such as admin, user, guest, et cetera. That is, it will need to include some kind of user attribute 'role' that will be used to authorize the requests to the RERUM services for each service. In a home environment, this attribute seems to be enough to perform the authorisation and hence, no other user attribute is foreseen to be needed for the moment. This means that the value of this attribute will be needed for request sent to the RERUM installation. As explained, querying for user attributes is a time consuming operation. For this reason, the caching capabilities provided in the Identity Agent should considerably speed up the processing the authorization of the requests to the RERUM services.

On the privacy side, the PPC will guarantee that only those attributes approved by the people of the home will be able to be accessible by the RERUM platform, and the ANR will make sure that only those attributes that take part in some authorization process even get to the previous check. Hence, on a typical installation that only takes into account the field role, this components will both optimize the access to this attribute and ensure that no other attribute is ever tried to be accessed.

5.3.4 Sticky Policies

We resume the example of a *user* who is a participant of RERUM UC-I1 and is consuming a service which analyses his energy consumption.

We assume the service is composed of two providers, one manages the customer relation and the overall service workflow, the second performs statistical operations on the customer data.

The service is offered as a free or paid service.

For the free service, customer data is used for targeted advertising. Targeted advertising means, that selected products and brands will be showcased on the customer's evaluation according to his consumption data.

The paid service excludes targeted advertising and offers ad-less evaluation.

The customer has signed up for the paid service, agreeing to give his consumption data for the analysis and feedback of energy saving plans only, excluding the usage of his data for advertisement.

The user installs a plug-in for using the service on his EMS. Upon requesting consumption data from the plug-in, the EMS follows the workflow of 39. At first, a data set is created with the consumption data. It is then encrypted with a key, which was given to the second party upon agreement of processing.

We further assume the service exchanges its analyzing provider and detail how the user's historical data could be handled. The user's data was protected by sticky policies, thus describing how the data maybe processed, when it has to be deleted and so on. A part of the user's data may be therefore deleted already. The remaining data is given to the new provider, which is able to read and request the needed key from the user's EMS. The user may have to give his consent once again to the newly formed service composition and give the needed key for the encrypted data thereafter.

Note: As discussed in Section 4.1 and noted in [166], sticky policies are merely a soft privacy mechanism, which support an accountable and legally compliant relationship between customers and providers. A malicious provider may decrypt encrypted data sets and store the clear text data. It could also process the data for targeted advertising anyway, infringing the user's policies. Therefore, RERUM UC I1 additionally employs hard privacy mechanisms, such as data perturbation and malleable signatures, see 5.3.2.

5.3.5 Leakage Resilient MAC

In the above-described use case, i.e., RERUM UC-I1 home energy management use case, there is a requirement to transfer data between devices and this information, if leaked, might have an impact on an user privacy. In a very likely scenario, an electricity meter will send a gathered power consumption information using dedicated communication channel to a main server (most likely via an appropriate gateway). This communication channel has to be protected by cryptographic primitives and protocols in order to keep the sensitive information secret. There are studies [106] in which the authors show that an energy consumption pattern could be exploited not only in a cores-grained fashion, i.e., by showing whether someone is currently at home or what kind of appliances he or she is using, but also in more fine-grained fashion, i.e., by investigating a power consumption pattern caused by a specific content displayed on a TV screen. In this specific case, a lack of a cryptographic protocol has been exploited in order to collect unencrypted data transferred between an electricity meter and a server, but similar might hold when indeed a cryptographic protocol has been used but attacker compromised the appropriate keys.

Considering the use case description, along above-mentioned cryptographic protocol, one might also need the D2D Authenticator component (as described in Section 3.8.2). A good candidate to achieve these goals is a DTLS protocol, which suitability for RERUM needs was already investigated in D3.1. DTLS protocol can handle device to device authentication using both symmetric and asymmetric keys and a traffic encryption between these devices. Depends on mode of operation, a leak of keys might have different security impacts. For example, using some cipher suites in DTLS, i.e., these based on public-key cryptography, DTLS can achieve a perfect forward secrecy, which means session keys are derived from long-term keys and compromising a long-term key in the future will not compromise a derived session key. Hoverer, this is not true for a private-key mode, where a leak of keys might have much higher security implication.

In general, side-channel attacks are more applicable in the scenario where a device is in the possession of an attacker, as this might be a case of the home energy management use case. Aforementioned DTLS uses MAC to prevent accidental or deliberate data manipulation in the traffic. In order to prevent some side-channel leakage DTLS might be further supported by leakage resilient MAC, which might protect a key better, especially in cipher suits based on private-keys. The integration of leakage resilient MAC in DTLS might not be trivial and might require a slight enhancement of the protocol itself. This is due to the fact that not all keys derived by mechanism in the original DLTS might also be appropriate in a leakage resilient version. We consider our leakage resilient MAC as an option for deployment in the home energy management use case. The benefits and usability of this specific protocol enhancement, especially its efficiency on constrained platforms is considered as the future work.

5.4 RERUM UC-I2. Indoor: Comfort quality management

Goal of the use case is to get a comfort index. This involves *surveillance* of indoor spaces to measure indoor air quality, detect smoke&fire, detect presence, and other data, which may infringe on the occupants of sparsely populated indoor spaces. Use case lead is Zolertia.

The use case is primarily intended for use in public buildings, i.e. museums, computer rooms, etc., but also potentially to be used in *private homes*, where the occupants may feel themselves watched. In the Heraklion trial some usually crowded municipal office spaces and a museum are monitored.

Landlord-owned sensors here involve integration of both wireless and wired sensors to measure for instance air quality (e.g. CO2), noise, radiation, light, humidity, temperature, fire alarm, motion, presence, and many other data. Actors may open and close windows, regulate A/C, and send alarms due to emphbehaviour anomalies (may indicate presence or actions of identifiable data subjects), among various other possibilities. The overall RERUM UC-I2 Ecosystem is depicted in Figure 111.

To reduce complexity for the description of the usefulness of authentic sensor data we used a combined hybrid deployment involving the use of authentic UC-I2 sensor data for UC-O2. Several characteristics similar to UC-O2, among others the same sensors (see Section 5.2.1).

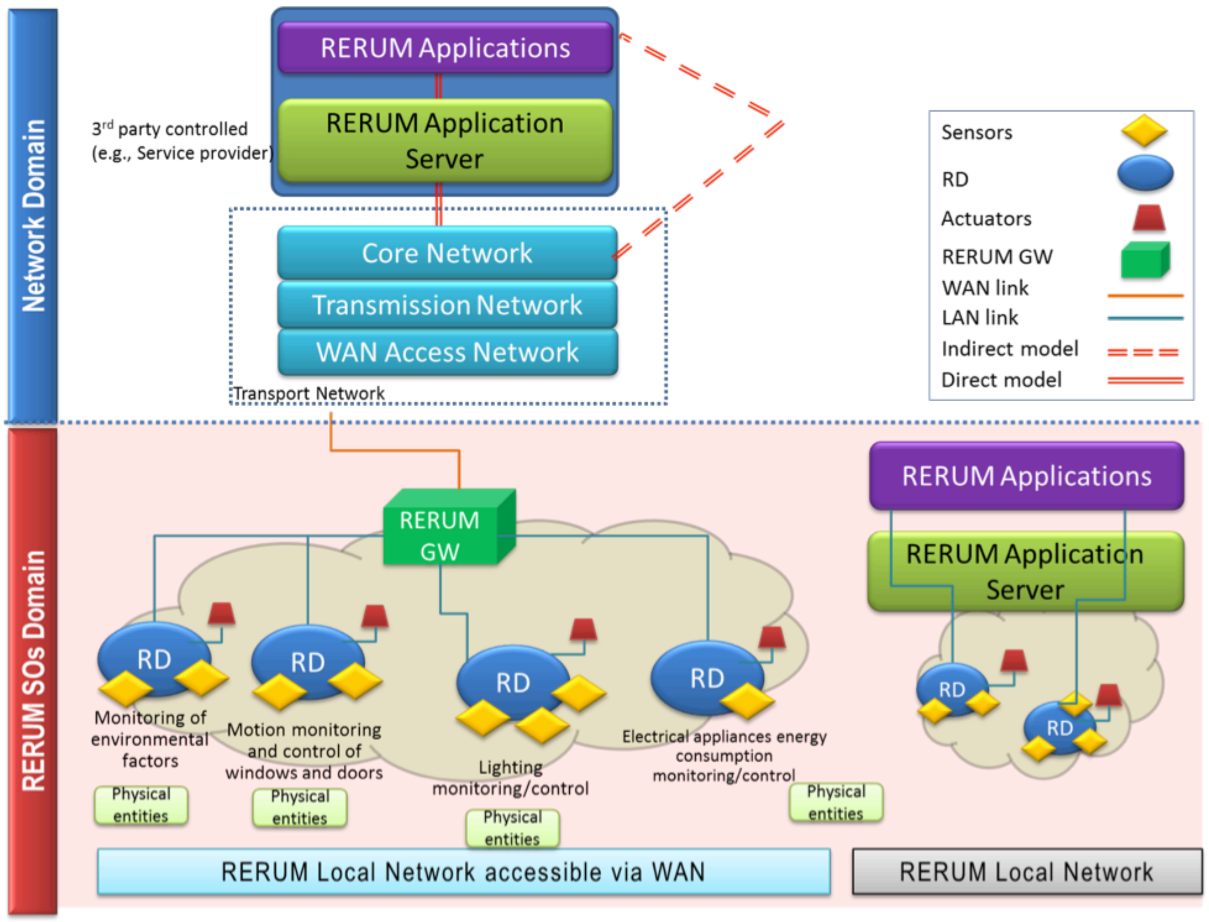


Figure 111: RERUM UC-I2 Ecosystem (by Cyta)

5.4.1 Privacy Policy Enforcement

Privacy in RERUM is based on privacy policies, which are meant to be generated in either the security dashboard or the consent manager. In both cases, these policies are meant to be evaluated and executed for the privacy to be achieved, and the component for doing it the PPEP. That is, there is no privacy in RERUM without the PPEP for any use case, including the UC2. Hence, UC2 benefits from the PPEP because it is the component that can actually evaluate and enforce privacy criteria in the system

5.4.2 Anonymisation and pseudonymisation, incl. de-pseudonymisation

RERUM's UC-I2 is comparable to UC-O2 as indicative measures of the air quality of user's homes are going to study user's actions and give him suggestions of how his home comfort can be optimized for the user, such as increasing the air quality, detecting noise isolation problems and reducing visible contamination that could affect and trigger sleeping problems.

Analysis on user's data and how his actions affect comfort quality maybe done at the user's home or remotely by a service provider. We therefore can follow the example of RERUM UC-I1 described in Section 5.3.1.1 where a user's comfort data is measured and sent to a service provider. The service provider analyses the data and sends suggestions to the user, which can automatically be transformed to policies and followed upon by appliances and actuators in the user's home.

For simplicity, we assume that the user manages the comfort quality in his EMS (i.e., RERUM's UC-I1 EMS App as described in Section 5.3.1.1).

We assume the setup of Figure 87, where the user has given his consent, a plug-in or configurations from the service provider have been installed on the EMS, and customer data can be sent to the service provider. The EMS creates data sets in the following way: every room in the user's home is assigned data for air, noise and visual quality. The EMS utilizes RERUM's pseudonym management to generate pseudonyms for the different rooms in the user's home. The EMS itself signs the data set with a group signature (see [53]) known to the service provider and transmits the data over a telecom provider (see D2.1 [167] Section 2.2.2.3.1). We assume that the service has several customers and that every EMS that is interacting with the service has a group signature, thus creating an appropriate anonymity set. We further assume that the telecom provider has anonymous routing capabilities (as described in 87) to avoid an identification over the packet routes, the IP- and MAC address of the EMS.

Based on the example above, we define the following steps of the interplay of the pseudonym manager in RERUM UC-I2:

The sequence in Figure 112 follows the steps of Figure 5.3.1.1.

- The sequence starts according to the description of the use case in D2.1 section 2.2.2.3, the smart objects measure the environment and send their data to the EMS.
- (Optional) The EMS subscribes to the service and agrees on a group signature and a session key.
- The EMS creates data sets and assigns rooms according to the location of the smart objects. It also adds meta-data to describe if the measurements are for air, noise or visual quality.
- The EMS requests a pseudonym for the rooms. The pseudonym manager creates new pseudonyms based on the room's type, given by the EMS, and the period. For simplicity, we assume that the pseudonym manager changes pseudonyms every day.

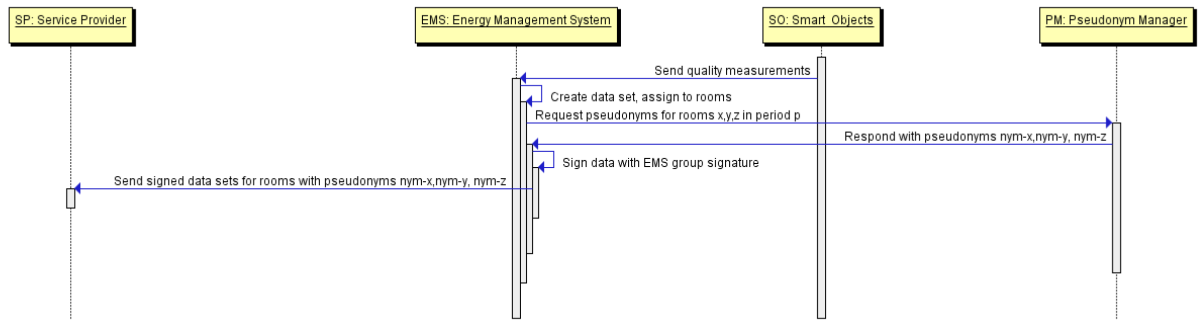

SP: Service Provider
EMS: Energy Management System
SO: Smart Objects
PM: Pseudonym Manager
Send quality measurements
Create data set, assign to rooms
Request pseudonyms for rooms x,y,z in period p
Respond with pseudonyms nym-x,nym-y, nym-z
Sign data with EMS group signature
Send signed data sets for rooms with pseudonyms nym-x,nym-y, nym-z

Figure 112: RERUM UC-I2 Pseudonymization of Quality Comfort Data Sets

- The EMS adds the pseudonyms to the data sets and signs the set with a group signature known to the service provider.
- The EMS sends the data to the service provider. (The telecom provider is not depicted in the figure for simplicity.)

The service provider will now analyse the data, generate a comfort quality report and suggestions on how the user could increase the living quality of his home.

The reports are then requested by the EMS whenever the user wants to see them, or if the EMS has a policy to follow the suggestions of the service. If the data is historical, the EMS has to either remember the pseudonyms or ask the pseudonym manager to dynamically generate the pseudonyms previously used. The service provider will then encrypt the record with the session key that was agreed on (see above, second bullet), so that only the EMS user is able to read the report. This sequence is presented in Figure 113.

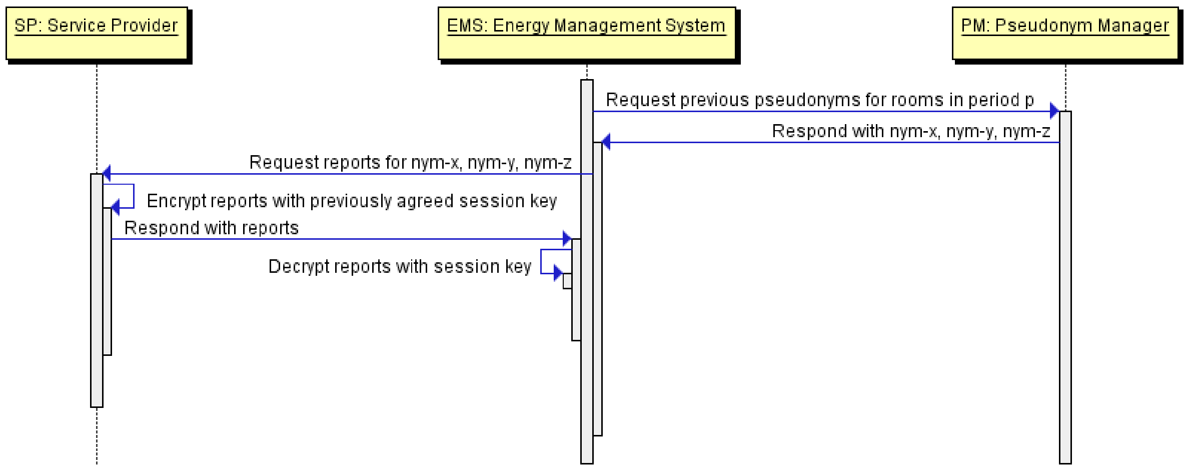

SP: Service Provider
EMS: Energy Management System
PM: Pseudonym Manager
Request previous pseudonyms for rooms in period p
Respond with nym-x, nym-y, nym-z
Request reports for nym-x, nym-y, nym-z
Encrypt reports with previously agreed session key
Respond with reports
Decrypt reports with session key

Figure 113: RERUM UC-I2 Retrieving Pesudonymized Data Sets

5.4.3 User Attribute Minimisation

UC2 defines a list of roles for the stakeholders involved in it and the operation they will be able to perform. That is, this IC defines its access control based on a user attribute ’role’ that will be used to authorize the requests to the RERUM services for each service. Hence, no other user attribute is initially foreseen to be needed for the moment. This means that the value of this attribute will be needed for request sent to the RERUM installation. As explained, querying for user attributes is a time consuming operation. For this reason, the caching capabilities provided in the Identity Agent should considerably speed up the processing the authorization of the requests to the RERUM services.

On the privacy side, the PPC will guarantee that only those attributes approved by the people of the home will be able to be accessible by the RERUM platform, and the ANR will make sure that only those attributes that take part in some authorization process even get to the previous check. Hence, on a tipical installation that only takes into account the field role, this components will both optimize the access to this attribute and ensure that no other attribute is ever tried to be accessed.

Nevertheless, if the system administrator had the intention to add new access policies that demanded further user information, such as the age, the PPC would normally be able to check that the involved RERUM registered user had actually agreed to provide that information. However, due to the lack of a consent manager implementation, current implementation of the PPC will have to accept the lack of privacy policies for the user attributes as a default consent, or otherwise it would be impossible to include new attributes in the policies of the system

5.5 Summary

In this chapter for each of our RERUM use cases, namely UC-O1 (Smart transportation), UC-O2 (Environmental monitoring), UC-I1 (Home energy management), and UC-I2 (Comfort quality management), we stated the overall use case goal and highlighted typical privacy problems of the respective use case. For each use case we showed how selected functional components of RERUM can mitigate the privacy problems, without preventing reachability of the use case's goals.

UC-O1 aims to use a heterogeneous network of sensors and smart objects and perform real time city traffic estimation and prediction. We showed how the User Consent Manager supports volunteers, meaning individual private data subjects installing the RERUM app on their private smart phones, and drivers of public vehicles, namely buses and taxis, in granting or refusing requests for consent. We explained how these RERUM users can comfortably quit temporarily participation in the RERUM UC-O1 with the help of the Activator/Deactivator of Data Collection. The geo-location privacy enhancing technology is applied on the RERUM App or the RERUM GPS module in buses and taxis. We described, how Geo-location PET, responsible for generating motion vectors and handling policies, enhances user privacy via traffic anonymization techniques and the help of a anonymous networking client.

UC-O2 tries to to perform continuous measurements for pollution in city environments, to focus on outdoor environmental measurements, and to put data into graphs and provide them to public on a server. We gave a hybrid scenario for a privacy preserving usage of UC-I2 sensor data for UC-O2, as both scenarios use similar sensor sets. In this hybrid scenario we explained, how the Privacy Enhanced Integrity Generator / Verifier allows to reduce sensor data quality to a level that preserves privacy of the data subject adequately, while still allowing for a reasonable level of accuracy. We also explained which parameters have to be initialised for Compressive Sensing, namely compression rate, measure matrix, and the chaos sequence's initial parameters.

UC-I1's goal is to monitor and control the energy consumption and to reduce energy consumption of several devices in public buildings. In this context, users can also remotely control and monitor their devices, manage energy savings and support efficient grid operation. The Privacy Dashboard allows privacy preferences definition regarding how often, to whom and in which granularity energy consumption data should be published. Device identities are pseudonymised in UC-I1. The correlation of identities and pseudonyms is done by the Anonymization and Pseudonymization Manager. The re-linked identities are shown in the Privacy Dashboard. We made experiments how Malleable Signatures can improve privacy when monitoring energy consumption in private households. User Attribute Minimisation allows here to protect the user's attributes only allowing access to the needed and agreed-on ones. Sticky Policies can be attached to meta data to transport the user's privacy preferences to the data controller. In this use case, an electricity meter will send a gathered power consumption information using dedicated communication channel to a main server.. Here Leakage Resilient MACs strengthen an underlying cryptography protocol thus allow for a more private data transfer.

UC-I2 wants to get a comfort index of indoor spaces. This involves surveillance of these spaces to measure indoor air quality, detect smoke and fire, detect presence, and other data. We explained how the privacy policy enforcement point helps to enforce user's privacy policies for UC-I2. We showed, how anonymisation and pseudonymisation, including de-pseudonymisation protects privacy by using group signatures and a suitable anonymity set. In analogy this can also be deployed to UC-O2. User Attribute Minimisation allows here as well as in UC-I1 to protect the user's attributes.

6 Additional privacy topics, open issues, future research

6.1 Conclusions

User privacy involves the protection of many aspects beyond name and address, like thoughts and feelings, behaviour and action, and location. IoT sensor data qualify as sensitive personal data requiring privacy protection. RERUM favours use of the LINDDUN privacy threat analysis method and the PRIPARE overall privacy engineering process. The RERUM Privacy-by-Design requirements include among others consent and choice also with the possibility of subsequent withdrawal, and purpose legitimacy and specification.

RERUM requires a collection limitation which is adequate, relevant and not excessive, as well as data minimisation, where we explicitly consider use of pseudonymous and anonymous use. Furthermore, data blurring and coarse-grained sensor data generation, as well as local processing, early aggregation, de-personalisation and anonymization of data are also actions of the RERUM system that can significantly enhance the protection of user private information. RERUM also requires accuracy and quality with a right to delete or rectify incorrect data, notice and access of/to collected and processed data, individual participation and transparency> RERUM gives the user the power to handle his own data in the way he wants, allowing him to activate/deactivate the collection of his data in an easy way. Accountability of the person responsible for privacy breaches is also a key part of the RERUM framework.

Some privacy enhancing technologies can support voluntary privacy-policy-compliant behaviour, while others offer privacy-enforcing controls. The components and methods developed and specified by RERUM cover both types. In general, RERUM has built a privacy architecture based on the concepts of privacy by design and privacy by default. As described in this deliverable, RERUM has achieved a significant progress beyond the state of the art in the area of privacy in the IoT, which is an area that had minor interest until recently. The advances of RERUM span in a cross-layer manner, starting from techniques that can run on the devices (even on constraint devices) for providing a first step of privacy preservation, disabling the gathering and the transmission of identifiable information. Then, privacy enhancing techniques are also applied in the intermediate nodes (e.g. at the gateways) for providing another layer of removal of identifiable information. Next, at the RERUM Middleware, several techniques for controlling the applications' access to private information, the management of data collection and the handling of access policies to data are used in order to fine grain the protection of the user data and ensure their unlinkability from the application point of view. That way, it is ensured that the applications will only get the exact data that they need and nothing more that could potentially allow the linking of the data to individuals.

Although RERUM has worked on many key techniques for enhancing the privacy in IoT, it has also identified some areas that need further research for a more holistic and optimised privacy framework. As a final remark in this document and for stimulating the future research, the next subsections provide a discussion on the open research items.

6.2 Improving privacy with unobservable communication in the Internet-of-things

To preserve privacy you need at least to prohibit the leakage of information to unauthorised third-parties. Today, encryption and authenticated channels between authorised parties technically protect the privacy of the message's payload during transmission. However, metadata still leaks details about the communication. It is very hard to estimate to what extent. Metadata can be gathered by network traffic analysis. Among various other information metadata includes identifying endpoints, message timing and location details of the communication. When combined with a-priori knowledge and processed by machine learning algorithms extracted information be so rich that end-to-end encryption can be bypassed. For example it might not be necessary to decrypt the payload at all, because its content can be guessed.

To counter traffic analysis we need to minimise any kind of information leakage due to communication meta-data. Therefore the system should ensure the unobservability of the network communication. This property ensures that messages and random noise are indistinguishable from each other. In terms of network nodes it ensures that their activity goes unnoticeable and that messages cannot be correlated. It is a very powerful property combining unlinkability, unidentifiability, and dummy traffic.

Unobservability = Anonymity + Dummy Traffic with
Anonymity = Unidentifiability + Unlinkability.

These terms are defined in detail in [179]. Unlinkability ensures that neither messages nor network nodes system can be correlated. Unidentifiability ensures that these are indistinguishable, building a so-called anonymity set.

To our knowledge, there is very limited state of the art in the area of unobservable communications in the IoT. RERUM has identified that it is a required issue to solve especially in indoor solutions (e.g. smart home applications) where even by monitoring when messages are sent can disclose personal information regarding the inhabitant. Although some initial work has been done within RERUM to address this issue, it remains an open issue for future researchers.

6.3 User-friendly ways to generate privacy policies

RERUM designs and implements a Privacy Authorisation Engine to evaluate privacy policies and to check the privacy for the user attributes referred in both the access and privacy policies (see Sections 3.2 and 3.10). Both the Security and the Privacy Policy Engine have been implemented as prototypes. Using manually generated privacy policies, RERUM demonstrates that only requests that comply with the manually generated privacy policies are able to pass the Privacy Policy Engine. However some more user-friendly ways to define privacy policies clearly are desirable:

Policy Definition and Administration: Privacy policies (even more than security policies for which we can assume the administrative human user to have some expert knowledge) are in need to be generated in an end-user-friendly manner by lay persons. In certain circumstances they may be generated automatically, based on decisions indicated by the data subjects (humans) in the RERUM Consent Manager and the RERUM Privacy Dashboard. The RERUM Consent Manager and the RERUM Privacy Dashboard are being provided as a design prototype only. It is desirable that they are implemented and tried to provide end-user-friendly tools to generate privacy policies.

Preference Elicitation: Data subjects can specify and adjust the privacy preferences having lead to the unwelcome decisions. Gradually they may arrive at a working set of privacy preferences. Options made available to the data subject and the way they are presented to the data subject requires careful user interface design and needs to be tailored to the actual IoT situation very carefully to avoid misunderstandings and misconfigurations as far as possible. This issue requires further in-depth investigation beyond the scope of RERUM. There may be a preferences assistant to guide the data subject through the process of making useful settings. More research is required on this topic, involving user interface design, and cognitive psychology aspects.

6.4 Measuring and visualising privacy

To reduce consent complexity and visualise consents in a human-user-friendly manner, techniques as described in Section 3.1.3 may be helpful, even if these aren't standards. To offer support for consent automation as described in [99] may also be a promising approach. This however requires further research out of scope of RERUM. Among other topics, a solution to the following may be beneficial to privacy engineering:

Putting a price on privacy: It may be interesting to find suitable approaches to visualising the trade-off between privacy and services in an intelligible way. For this one among other things one has to solve the problem of how do you put a price on privacy.

Measuring data minimisation: Data minimisation has been termed the paramount Privacy-By-Design principle by RERUM. There are many aspects to data minimisation, as discussed, like local processing, early aggregation and anonymization, minimal data collected, etc. However we still lack methods to rate the degree of adherence to this principle. It would be desirable to find metrics and visualisation techniques to figure out to what degree the data minimisation principle has been observed in a given IoT system design and implementation in relation to the officially stated purpose. This would help to discuss privacy improvements, find hidden agendas of the data collector and rate the trustworthiness of the IoT application provider.

Auto-discovery of IoT scenarios: It would be desirable to have (semi-)automatable methods to derive (discover?) and visualise the layout of an IoT scenario for the purpose of explaining the situation to the data subject in order to support informed consent to an IoT application. Based on such a layout the requested sensors for instance could be highlighted and available options could be shown on demand. Also the radius of a sensor could be visualised that way. This requires further research combining at least user interface design and network management.

6.5 Traffic anonymisation

Anonymisation of traffic in IoT networks is another research area that has not attracted much attention up until now. Network anonymisation aims to provide users with anonymity when they are transferring data through the Internet. Although there are several anonymity systems that provide anonymous communications, like Tor and I2P, their applicability in the IoT domain has not been investigate so far.

For avoiding disclosing the original source of a message, existing approaches use either onion routing or layered encryption. Depending on the application(s) that is used on top of the network, different techniques for traffic anonymisation can be applied. For example, when the applications are delay-tolerant, the messages can be gathered at an intermediate node (playing the role of a proxy), grouped altogether and then forwarded to the destination. Of course this approach has the weak point of the proxy being a single point of failure and if it is hacked then all the communications will be affected. For this reason, the onion routing was proposed as a way of using multiple proxies, but this increases the latency of the system because the path from the source to the destination increases with the inclusion of more proxies (which on the other hand increases the privacy of the source). The onion routing ensures the anonymity between the source and the destination, while other approaches focus on improving the anonymity between the nodes of the same network, by using unidirectional tunneling.

However, to our knowledge, none of the above solutions have been adequately addressed or adapted to the requirements of the Internet of Things and especially considering the constrained devices that are involved in IoT networks. Thus, these also remain as open research items.

References

[1] Gergely Ács and Claude Castelluccia. I have a DREAM! (DiffeRentially privatE smArt Metering). In *Proc. of Information Hiding*, pages 118–132. Springer-Verlag, 2011. ISBN 978-3-642-24177-2. doi: 10.1007/978-3-642-24178-9_9. URL http://dl.acm.org/citation.cfm?id=2042445.2042457http://link.springer.com/10.1007/978-3-642-24178-9_9.

[2] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *Lecture Notes in Computer Science*, volume 7194 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2012. ISBN 978-3-642-28913-2. doi: 10.1007/978-3-642-28914-9_1. URL http://link.springer.com/10.1007/978-3-642-28914-9_1.

[3] Hunt Allcott. Social norms and energy conservation. *Journal of Public Economics*, 95(9-10): 1082–1095, October 2011. ISSN 00472727. doi: 10.1016/j.jpubeco.2011.03.003. URL http://www.sciencedirect.com/science/article/pii/S0047272711000478http://linkinghub.elsevier.com/retrieve/pii/S0047272711000478.

[4] Joe Andrieu, Iain Henderson, and Judi Clark. The standard information sharing labe v0.4l. Web Page, 2012. URL http://standardlabel.org.

[5] D. Aranha and C. P. L. Gouvea. RELIC Cryptographic Toolkit, 2015. URL https://github.com/relic-toolkit/relic.

[6] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3679 LNCS, pages 159–177. Springer, 2005. ISBN 3540289631. doi: 10.1007/11555827_10. URL http://link.springer.com/10.1007/11555827_10.

[7] N Attrapadung, B Libert, and T Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. In *ASIACRYPT*, pages 367–385, 2012.

[8] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *LNCS Vol. 7778*, pages 386–404, 2013. doi: 10.1007/978-3-642-36362-7_24. URL http://link.springer.com/10.1007/978-3-642-36362-7_24.

[9] Afkham Azeez, Srinath Perera, Sanjiva Weerawarana, Paul Fremantle, Selvaratnam Uthaiyashankar, and Samisa Abesinghe. WSO2 Stratos: An application stack to support cloud computing. *it - Information Technology—Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 53(4):180–187, 2011.

[10] Chris Babel, Andrew Braccia, Ben Golub, and Jeb Miller. Ensure Privacy Compliance and Build Customer Trust: TRUSTe Data Privacy Management Solutions. Web Page, July 2015. URL https://www.truste.com/.

[11] Michael Backes and Sebastian Meiser. Differentially private smart metering with battery recharging. *IACR Cryptology ePrint Archive*, 2012:194–212, 2014. doi: 10.1007/978-3-642-54568-9_13. URL http://link.springer.com/10.1007/978-3-642-54568-9_13.

[12] Waheed U. Bajwa, Jarvis D. Haupt, Gil M. Raz, Stephen J. Wright, and Robert D. Nowak. Toeplitz-Structured Compressed Sensing Matrices. In *14th Workshop on Statistical Signal Processing (SP 2007)*, pages 294–298. IEEE, August 2007. ISBN 978-1-4244-1197-9. doi: 10.1109/SSP.2007.4301266. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4301266`.

[13] Gianmarco Baldini, Trevor Peirce, Maarten Botterman, and Others. Internet of Things: IoT governance, privacy and security issues. Position Paper Activity Chain 05, IERC - European Research Cluster on the Internet of Things, January 2015. URL `http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Governance_Privacy_Security_Final.pdf`.

[14] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Proc. of Advances in Cryptology (EUROCRYPT '97)*, pages 480–494, 1997. doi: 10.1007/3-540-69053-0.

[15] Barocas, Soltani, and Doty. Parsing Privacy Policies. Catalog, Center for Information Technology Policy at Princeton University, 2015. URL `http://solon.barocas.org/?page_id=200`.

[16] Paulo Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer, 2006.

[17] Uwe Baumgarten. Datenschutz Einwilligungserklaerung, Muster Einwilligungserklaerung. Web Page, July 2015. URL `https://www.datenschutz.tum.de/unterstuetzung/einwilligungserklaerung/`.

[18] A Becker and M Jensen. Secure Combination of XML Signature Application with Message Aggregation in Multicast Settings. In *ICWS*, pages 531–538, 2013.

[19] Karissa Bell. Google unveils a privacy dashboard: All your settings in one place, June 2015. URL `http://mashable.com/2015/06/01/google-privacy-hub/`.

[20] M Bellare and D Micciancio. A new paradigm for collision-free hashing: incrementality at reduced cost. In *Eurocrypt'97*, pages 163–192. Springer-Verlag, 1997.

[21] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: the case of hashing and signing. *Proc. of Advances in Cryptology (CRYPTO '94)*, 839:216–233, 1994. doi: 10.1007/3-540-48658-5_22. URL `citeseer.ist.psu.edu/bellare94incremental.htmlhttp://link.springer.com/10.1007/3-540-48658-5_22`.

[22] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures, 1993.

[23] A.R. Beresford and Frank Stajano. Mix zones: user privacy in location-aware services. *Proc. of the 2nd IEEE Ann. Conf. on Pervasive Computing and Communications Workshops*, pages 127–131, 2004. doi: 10.1109/PERCOMW.2004.1276918. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1276918`.

[24] BMW-AG. BMW TeleServices: Datenschutzrechtlicher Hinweis. Web Page, July 2015. URL `http://www.bmw.de/de/topics/service-zubehoer/bmw-service/teleservices/datenschutz.html`.

[25] BMW-AG. BMW TeleServices: Übersicht Dienste. Web Page, July 2015. URL http://www.bmw.de/de/topics/service-zubehoer/bmw-service/teleservices/uebersicht.html.

[26] Dan Boneh and David Mandell Freeman. Homomorphic Signatures for Polynomial Functions. In *Proc. of Advances in Cryptology (EUROCRYPT 2011)*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168, 2011.

[27] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. of Advances in Cryptology (EUROCRYPT 2003)*, pages 416–432, 2003. doi: 10.1007/3-540-39200-9_26. URL http://link.springer.com/10.1007/3-540-39200-9_26.

[28] Marco Bonetti. Mobile privacy: Tor on the iPhone and other unusual devices, 2010.

[29] C Bormann, M Ersue, and A Keranen. Terminology for Constrained-Node Networks. RFC 7228 (Informational), May 2014. URL http://www.ietf.org/rfc/rfc7228.txthttps://www.rfc-editor.org/info/rfc7228.

[30] E Boyle, S Goldwasser, and I Ivan. Functional Signatures and Pseudorandom Functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[31] BrightFort. EULAlyzer. Web Page, 2012. URL http://www.brightfort.com/eulalyzer.html.

[32] C Brzuska, H Busch, O Dagdelen, M Fischlin, M Franz, S Katzenbeisser, M Manulis, C Onete, A Peter, B Poettering, and D Schröder. Redactable signatures for tree-structured data: definitions and constructions. In *Proc. of the 8th Int. Conf. on Applied Cryptography and Network Security*, ACNS'10, pages 87–104. Springer, 2010. ISBN 3-642-13707-5, 978-3-642-13707-5. URL http://portal.acm.org/citation.cfm?id=1894302.1894310.

[33] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In *Proc. of the 12th Int. Conf. on Practice and Theory in Public Key Cryptography*, pages 317–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00467-4. doi: 10.1007/978-3-642-00468-1_18. URL http://link.springer.com/10.1007/978-3-642-00468-1_18.

[34] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *Proc. of Public Key Cryptography (PKC 2010)*, pages 444–461, 2010. doi: 10.1007/978-3-642-13013-7_26. URL http://link.springer.com/10.1007/978-3-642-13013-7_26.

[35] Christina Brzuska, Henrich C Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In Sabrina di Vimercati and Chris Mitchell, editors, *Proc. of the 9th European PKI Workshop: Research and Applications (EuroPKI 2012)*, volume 7868 of *LNCS*, pages 178–193. Springer, 2013. doi: 10.1007/978-3-642-40012-4_12. URL http://web.sec.uni-passau.de/papers/2012_BrzuskaPoehlsSamelin_NonInteractive-Public-Accountability-for-SanSigs.pdfhttp://link.springer.com/10.1007/978-3-642-40012-4_12.

[36] Karl-Heinz Bueschemann. Datenschutz im Internet: Ausbeutung der #Neuland-Ureinwohner. Web Page, February 2015. URL http://www.sueddeutsche.de/digital/datenschutz-im-internet-ausbeutung-der-neuland-ureinwohner-1.2331926.

[37] Ahto Buldas, Peeter Laud, and Helger Lipmaa. Accountable certificate management using undeniable attestations. In *Proc. of the 7th ACM conf. on Computer and Communications Security (CCS '00)*, pages 9–17, New York, New York, USA, 2000. ACM Press. ISBN 1581132034. doi: 10.1145/352600.352604. URL http://portal.acm.org/citation.cfm?doid=352600.352604.

[38] Bundesamt für Sicherheit in der Informationstechnik. {BSI} {TR-03109} {@ONLINE}, 2011. URL https://www.bsi.bund.de/DE/Themen/SmartMeter/TechnRichtlinie/TR_node.html.

[39] Marianne Busch, Nora Koch, and Santiago Suppan. Modeling security features of web applications. In *Engineering Secure Future Internet Services and Systems*, pages 119–139. Springer, 2014.

[40] Giovanni Buttarelli and Wojciech Wiewiorowski. The european data protection supervisor data protection glossary. Web Page, July 2015. URL https://secure.edps.europa.eu/EDPSWEB/edps/EDPS/Dataprotection/Glossary.

[41] Levente Buttyán, Tamás Holczer, André Weimerskirch, and William Whyte. Slow: A practical pseudonym changing scheme for location privacy in vanets. In *IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2009.

[42] V Cambareri, M Mangia, F Pareschi, R Rovatti, and G Setti. Low-Complexity Multiclass Encryption by Compressed Sensing. *IEEE trans. on Signal Processing*, 63:2183–2195, 2015.

[43] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proc. of Advances in Cryptology (CRYPTO 2002)*, pages 61–76, 2002. doi: 10.1007/3-540-45708-9_5. URL http://link.springer.com/10.1007/3-540-45708-9_5.

[44] S Canard and A Jambert. On extended sanitizable signature schemes. In *CT-RSA*, pages 179–194, 2010.

[45] S Canard, A Jambert, and R Lescuyer. Sanitizable signatures with several signers and sanitizers. In *AFRICACRYPT*, pages 35–52, 2012.

[46] Sébastien Canard, Fabien Laguillaumie, and Michel Milhau. Trapdoor sanitizable signatures and their application to content protection. In *Applied Cryptography and Network Security (ACNS 2008)*, pages 258–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-68914-0_16. URL http://link.springer.com/10.1007/978-3-540-68914-0_16.

[47] E Candes and T Tao. Near-optimal signal recovery from random projections: UNiversal encoding strategies? *IEEE trans. on Information Theory*, 52:5406–5425, 2006.

[48] E Candes and M Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[49] Ann Cavoukian. 7 foundational principles for privacy by design. Web Page, January 2009. URL https://www.privacybydesign.ca/index.php/about-pbd/7-foundational-principles.

[50] T.-H. H Chan, E Shi, and D Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*, volume 7397 of *LNCS*, pages 200–214. Springer, 2012. ISBN 978-3-642-32945-6. URL http://dblp.uni-trier.de/db/conf/fc/fc2012.html#ChanSS12.

[51] Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short redactable signatures using random trees. In *Proc. of the The Cryptographers' Track at the RSA Conf. 2009 on Topics in Cryptology*, CT-RSA '09, pages 133–147. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-00861-0. doi: 10.1007/978-3-642-00862-7_9. URL http://dx.doi.org/10.1007/978-3-642-00862-7_9http://link.springer.com/10.1007/978-3-642-00862-7_9.

[52] Pavlos Charalampidis, Alexandros G Fragkiadakis, and Elias Z Tragos. Rate-adaptive compressive sensing for IoT applications. In *81st IEEE Vehicular Technology Conf. (VTC Spring)*, pages 1–5. IEEE, 2015.

[53] D Chaum and E Van Heyst. Group signatures. In *Proc. of the 10th Ann. Int. Conf. on Theory and application of cryptographic techniques*, pages 257–265. Springer-Verlag, 1991.

[54] Chi-Yin Chow, Mohamed F Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proc. of the 14th Ann. ACM Int. Symp. on Advances in Geographic Information Systems*, pages 171–178. ACM, 2006.

[55] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. Technical report, World Wide Web Consortium, March 2001. URL http://www.w3.org/TR/wsdl.

[56] Stephanie Clifford and Quentin Hardy. Attention, Shoppers: Store is tracking your cell. Web Page, July 2013. URL http://nyti.ms/15eaRaD.

[57] Andrew Couts. Digital trends: terms and conditions. Web Page, March 2014. URL http://www.digitaltrends.com/topic/terms-and-conditions/.

[58] Lorrie Cranor. IE Privacy Bird: Find web sites that respect your privacy. Web Page, 2009. URL http://www.privacybird.org/.

[59] Lorrie Cranor and Rigo Wenning. Platform for Privacy Preferences (P3P) Project. Web Page, November 2007. URL http://www.w3.org/P3P/.

[60] Lorrie Faith Cranor, Pedro Giovanni Leon, and Blase Ur. Bank Privacy Search: A Large-Scale Evaluation of U.S. Financial Institutions' Standardized Privacy Notices. Technical report, Carnegie Mellon University, Cylab Usable Privacy and Security Laboratory, 2013. URL http://cups.cs.cmu.edu/bankprivacy/.

[61] J Cuellar, J Polk, J Morris, and M Thomson. Geolocation Policy: A document format for expressing privacy preferences for location information. Technical report, Internet Engineering Task Force (IETF), January 2013. URL https://www.rfc-editor.org/info/rfc6772.

[62] Jorge Cuellar and Et. al. RERUM Deliverable D2.2 - System requirements and smart objects model, 2014.

[63] Jorge Cuellar, Santiago Suppan, and Poehls Henrich. Internet Draft: privacy-enhanced tokens for authorization in ACE, 2015.

[64] Danezis, Domingo-Ferrer, Hansen, Hoepman, LeMetayer, Tirtea, and Schiffner. Privacy and Data Protection by Design. Technical Report ISBN 978-92-9204-108-3, European Union Agency for Network and Information Security, December 2014. URL https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/privacy-and-data-protection-by-design/at_download/fullReport.

[65] H de Meer, M Liedel, H C Pöhls, J Posegga, and K Samelin. Indistinguishability of One-Way Accumulators (MIP-1210). Technical report, University of Passau, 2012.

[66] Hermann de Meer, Henrich C Pöhls, Joachim Posegga, and Kai Samelin. On the relation between redactable and sanitizable signature schemes. In *Engineering Secure Software and Systems*, volume 8364 of *LNCS*, pages 113–130. Springer, 2014. doi: 10.1007/978-3-319-04897-0_8. URL http://link.springer.com/10.1007/978-3-319-04897-0_8.

[67] Hermann de Meer, HenrichC. Pöhls, Joachim Posegga, and Kai Samelin. Redactable Signature Schemes for Trees with Signer-Controlled Non-Leaf-Redactions. In Mohammad S Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 455 of *Communications in Computer and Information Science*, pages 155–171. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-44790-1. doi: 10.1007/978-3-662-44791-8_10. URL http://dx.doi.org/10.1007/978-3-662-44791-8_10.

[68] Paul DeHert and Dariusz Kloza. EU Project PIAF: a privacy impact assessment framework for data protection and privacy rights. Web Page, October 2012. URL http://www.piafproject.eu.

[69] T Do, L Gan, N Nguyen, and T Tran. Fast and Efficient Compressive Sensing Using Structurally Random Matrices. *IEEE trans. on Signal Processing*, 60:139–154, 2011.

[70] Mathias Doepfner. An open letter to Eric Schmidt from Mathias Doepfner. Web Page, April 2014. URL https://www.axelspringer.de/dl/433625/LetterMathiasDoepfnerEricSchmidt.pdf.

[71] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. ISSN 0006-3444. doi: 10.1093/biomet/81.3.425. URL http://biomet.oxfordjournals.org/cgi/doi/10.1093/biomet/81.3.425.

[72] Nick Doty and Erik Wilde. Geolocation privacy and application platforms. In *Proc. of the 3rd ACM SIGSPATIAL Int. Workshop on Security and Privacy in GIS and LBS*, pages 65–69. ACM, 2010.

[73] Nick P Doty, Mohit Gupta, Jeff Zych, and Rowyn McDonald. Privacy dashboard, a privacy pattern. Web Page, July 2015. URL http://privacypatterns.org/patterns/Privacy-dashboard.

[74] Marcos Duarte. BMC - Detection of peaks in data {@ONLINE}, 2013. URL http://nbviewer.ipython.org/github/demotu/BMC/blob/master/notebooks/DetectPeaks.ipynb.

[75] Cynthia Dwork. Differential Privacy. In *ICALP (2)*, pages 1–12, 2006.

[76] Manfred Dworschak. Im Tollhaus der Zukunft. *Der Spiegel*, (3):118–122, January 2015. URL http://www.spiegel.de/spiegel/print/d-131242928.html.

[77] Ecole Polytechnique Fédérale de Lausanne (EPFL). Sensorscope: Sensor Networks for Environmental Monitoring, 2013. URL http://lcav.epfl.ch/sensorscope-en.

[78] EFF, the Internet Society, and ToS;DR. TOSBack: The Terms of Service Tracker. Web Page, 2015. URL https://tosback.org/.

[79] Murray Eisenberg. Hill ciphers and modular linear algebra. *Mimeographed notes, University of Massachusetts*, pages 1–19, 1998.

[80] M Enev, S Gupta, T Kohno, and S N Patel. Televisions, video privacy, and powerline electromagnetic interference. In *ACM CCS*, pages 537–550. ACM, 2011. ISBN 978-1-4503-0948-6. URL http://dblp.uni-trier.de/db/conf/ccs/ccs2011.html#EnevGKP11.

[81] EU. Directive 95/46/EC On the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data, Official Journal of 23 November 1995, L 281, page 31 - 50, 1995.

[82] EU Article 29 Data Protection Working Party and Aritcle 29 Data Protection Working Party. Opinion 8/2014 on the on Recent Developments on the Internet of Things. Technical Report 14/EN WP 223, European Union, September 2014. URL http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp223_en.pdf.

[83] {European Parliament}. Legislative resolution of 12 March 2014 on the proposal for a regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Prot. Web Page, March 2014. URL http://www.europarl.europa.eu/sides/getDoc.do?type=TA&reference=P7-TA-2014-0212&language=EN&ring=A7-2013-0402.

[84] {European Parliament and Council}. Regulation 45/2001 of the European Parliament and of the Council of 18 December 2000 on the protection of individuals with regard to the processing of personal data by the Community institutions and bodies and on the free movement of such data. *Official Journal*, L 8/1(November 2000):22, 2001.

[85] Alan Fairless. Zero knowledge privacy: learn why privacy matters. Web Page, 2013. URL http://zeroknowledgeprivacy.org/.

[86] Hannes Federrath. JAP—Anonymity & Privacy. *URL http://anon. inf. tu-dresden. de/index en. html. Accessed*, 10, 2007.

[87] Rachel L Finn, David Wright, and Michael Friedewald. Seven Types of Privacy. In S Gutwirth Et al.., editor, *European Data Protection: Coming of Age*. Dordrecht: Springer Science and Business Media, 2013. URL http://works.bepress.com/michael_friedewald/60/.

[88] A Fragkiadakis, E Tragos, and A Traganitis. Lightweight and secure encryption using channel measurements. In *Proc. of VITAE*, pages 1–5, 2014.

[89] Fraunhofer-ISI. EU Project SAPIENT: surveillance, privacy and ethics. Web Page, 2011. URL http://www.sapientproject.eu/.

[90] Friedman, Lin, and Miller. Informed Consent by Design. In *in Security and Usability*, pages 503–530, 2005. URL http://hornbeam.cs.ucl.ac.uk/hcs/teaching/GA10/lec9extra/ch24friedman.pdf.

[91] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[92] Garcia, McDonnell, Troncoso, LeMetayer, Kroener, Wright, DelAlamo, and Martin. Deliverable D1.2: Privacy and Security by design Methodology. Technical Report ICT-610613 D1.2, EU Project PRIPARE, 2014. URL `http://pripareproject.eu/wp-content/uploads/2013/11/PRIPARE_Deliverable_D1.2_draft.pdf`.

[93] Garcia, McDonnell, Troncoso, LeMetayer, Kroener, Wright, DelAlamo, and Martin. Deliverable D1.1: Privacy and Security, Concepts and Principles Report. Technical Report ICT-610613 D1.1, EU Project PRIPARE, March 2014. URL `http://pripareproject.eu/wp-content/uploads/2013/11/D1.1.pdf`.

[94] R Gennaro, S Halevi, and R Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *EUROCRYPT*, pages 123–139, 1999.

[95] Ben Gerber. OECD privacy principles, version 1.1. Web Page, August 2010. URL `http://oecdprivacy.org`.

[96] Giannangelo and Vele. Privacy Policy Generator for Websites and Apps. Web Page, July 2015. URL `http://www.iubenda.com`.

[97] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[98] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. ISSN 0097-5397. doi: 10.1137/0217017. URL `http://epubs.siam.org/doi/abs/10.1137/0217017`.

[99] Gomer, Schraefel, and Gerding. Consenting Agents: Semi-Autonomous Interactions for Ubiquitous Consent. In *UbiComp '14*. ACM, 2014. doi: 978-1-4503-3047. URL `http://eprints.soton.ac.uk/366977/1/mcp_ubicomp_workshop_camera2.pdf`.

[100] Joshua Gomez, Travis Pinnick, and Ashkan Soltani. KnowPrivacy: privacy policy analysis. Web Page, 2009. URL `PrivacyPolicyAnalysis`.

[101] J Gong, H Qian, and Y Zhou. Fully-secure and practical sanitizable signatures. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology*, volume 6584 of *Lecture Notes in Computer Science*, pages 300–317. Springer Berlin / Heidelberg, 2011.

[102] Google. View your data and account activity in the Google Dashboard. Web Page, July 2015. URL `https://support.google.com/accounts/answer/162744`.

[103] G Gottlob, C Koch, and R Pichler. The complexity of XPath query evaluation. In *Proc. of the 22nd Symp. on Principles of Database Systems*, PODS, pages 179–190, New York, USA, 2003. ACM. ISBN 1-58113-670-6. doi: http://doi.acm.org/10.1145/773153.773171. URL `http://doi.acm.org/10.1145/773153.773171`.

[104] Glenn Greenwald. Why privacy matters. Web Page, October 2014. URL `http://www.ted.com/talks/glenn_greenwald_why_privacy_matters/transcript?language=en`.

[105] U Greveler, B Justus, and D Löhr. Identifikation von Videoinhalten {ü}ber granulare Stromverbrauchsdaten. In *Sicherheit*, volume 195 of *LNI*, pages 35–45. GI, 2012. ISBN 978-3-88579-289-5. URL http://dblp.uni-trier.de/db/conf/sicherheit/sicherheit2012.html#GrevelerJL12.

[106] Ulrich Greveler, Benjamin Justus, and Dennis Loehr. Multimedia content identification through smart meter power usage profiles. *Computers, Privacy and Data Protection*, 1:10, 2012.

[107] Gutsol, Sobolev, and Tchebotarev. 500px Privacy Policy. Web Page, February 2012. URL https://500px.com/privacy.

[108] Nils Haag. Einwilligungserklaerungen wirksam formulieren, schwierig, aber machbar! Web Page, July 2012. URL https://www.datenschutzbeauftragter-info.de/einwilligungserklaerungen-wirksam-formulieren-schwierig-aber-machbar.

[109] Stuart Haber, Yasuo Hatano, Yoshinori Honda, William Horne, Kunihiko Miyazaki, Tomas Sander, Satoru Tezoku, and Danfeng Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *Proc. of the 2008 ACM Symp. on Information, computer and communications security (ASIACCS '08)*, page 353, New York, New York, USA, 2008. ACM Press. ISBN 9781595939791. doi: 10.1145/1368310.1368362. URL http://portal.acm.org/citation.cfm?doid=1368310.1368362.

[110] Haduong, Tordillos, and Quintana. Privacy simplified icons. Web Page, May 2012. URL http://yale.edu/self/psicons.html.

[111] D Hardt. The OAuth 2.0 Authorization Framework. Standard RFC 6749, Internet Engineering Task Force (IETF), October 2012. URL http://tools.ietf.org/html/rfc6749https://www.rfc-editor.org/info/rfc6749.

[112] Heise-Online. BMW ConnectedDrive gehackt. Web Page, January 2015. URL http://www.heise.de/newsticker/meldung/BMW-ConnectedDrive-gehackt-2533601.html.

[113] Ralph Herkenhöner, Hermann de Meer, Meiko Jensen, and Henrich C Pöhls. Towards automated processing of the right of access in inter-organizational web service compositions. In *IEEE Int. Workshop on WebService and Business Process Security (WSBPS)*, pages 645–652. IEEE, July 2010. ISBN 978-1-4244-8199-6. doi: 10.1109/SERVICES.2010.56. URL http://web.sec.uni-passau.de/papers/2010_Herkenhohner_Poehls_Jensen_AutomatedRightofAccess.pdfhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5575518.

[114] S Hirose and H Kuwakado. Redactable Signature Scheme for Tree-structured Data based on Merkle Tree. In *SECRYPT*, pages 313–320, 2013.

[115] Leping Huang, Kanta Matsuura, Hiroshi Yamane, and Kaoru Sezaki. Enhancing wireless location privacy using silent period. In *IEEE Wireless Communications and Networking Conference*, volume 2, pages 1187–1192. IEEE, 2005.

[116] IETF. IETF Tools WG Page Datatracker: Authentication and Authorization for Constrained Environments (ace), 2015. URL https://datatracker.ietf.org/wg/ace/documents/.

[117] Fraunhofer ISI. EU Project: PRESCIENT - Privacy and Emerging Sciences and Technologies. Web Page, 2012. URL http://www.prescient-project.eu/prescient/index.php.

[118] ISO/IEC. ISO-IEC 7498-2: Information processing systems Open Systems Interconnection Basic Reference Model. Part 2: Security Architecture. *ISO Geneve, Switzerland*, 1989.

[119] ISO/IEC-JTC-1-SC-27. ISO/IEC 29100: Information technology, Security techniques, Privacy framework. Technical Report ISO/IEC 29100:2011(E), ISO/IEC, December 2011. URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c045123_ISO_IEC_29100_2011.zip.

[120] ISO/IEC-JTC-1-SC-27. ISO/IEC 27018: Information technology – Security techniques – Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors. Technical Report ISO/IEC 27018:2014, ISO/IEC, July 2014. URL http://www.iso.org/iso/catalogue_detail?csnumber=61498.

[121] ISO/IEC-JTC-1-SC-27. ISO/IEC CD 29134: Privacy impact assessment -Methodology. Technical Report ISO/IEC CD 29134 ICS: 35.040 Stage: 30.20 (2015-06-26), ISO/IEC, June 2015. URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=62289.

[122] ISO/IEC-JTC-1-SC-27. ISO/IEC CD 29151: Code of practice for PII protection. Technical Report ISO/IEC CD 29151: ICS: 35.040 Stage: 30.20 (2015-06-30), ISO/IEC, June 2015. URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=62726.

[123] Marek Jawurek. *Privacy in Smart Grids*. PhD thesis, Friedrich-Alexander-University Erlangen-Nuernberg, 2013.

[124] Marek Jawurek, Martin Johns, and Konrad Rieck. Smart metering de-pseudonymization. In *ACSAC*, pages 227–236, 2011.

[125] T Jeske. Privacy-preserving smart metering without a trusted-third-party. In *SECRYPT*, pages 114–123, 2011.

[126] Tobias Jeske. Schwerpunktthema Smart Grid: Datenschutzfreundliches Smart Metering. *Datenschutz und Datensicherheit*, 35. Jahrga(8):530–534, August 2011. doi: 10.1007/s11623-011-0132-9. URL http://www.dud.de/Ausgabe/2011-08.html.

[127] Robert Johnson, David Molnar, Dawn Song, David Wagner, and D.Wagner. Homomorphic signature schemes. In *Proc. of the RSA security Conf. - Cryptographers Track*, pages 244–262. Springer, 2002.

[128] J Jonsson and B Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), February 2003. URL https://www.rfc-editor.org/info/rfc3447.

[129] B Kaliski. The Mathematics of the RSA Public-Key Cryptosystem. *RSA Laboratories*, page 9, 2006.

[130] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *3rd IEEE Int. Conf. on Data Mining (ICDM 2003)*, pages 99–106. IEEE, 2003.

[131] P G Kelley, L J Cesca, J Bresee, and L F Cranor. CUPS privacy nutrition labels. Technical Report CMU-CyLab-09-014, Carnegie Mellon University, CyLab, Usable Privacy and Security Laboratory, November 2009. URL `http://www.cylab.cmu.edu/research/techreports/2009/tr_cylab09014.html`.

[132] S Kelly and S Frankel. Using hmac-sha-256, hmac-sha-384, and hmac-sha-512 with IPSec. Technical report, Internet Engineering Task Force (IETF), 2007.

[133] Eike Kiltz, Anton Mityagin, Saurabh Panjwani, and Barath Raghavan. Append-only signatures. In Luís Caires, Giuseppe Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, volume 3580 of *LNCS*, pages 434–445. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-27580-0. doi: 10.1007/11523468_36. URL `http://link.springer.com/10.1007/11523468_36`.

[134] Marek Klonowski and Anna Lauks. Extended Sanitizable Signatures. In *ICISC*, pages 343–355, 2006.

[135] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proc. of Advances in Cryptology (CRYPTO'99)*, pages 388–397. Springer, 1999.

[136] Hugo Krawczyk and Tal Rabin. Chameleon Hashing and Signatures. In *Symp. on Network and Distributed Systems Security*, pages 143–154, 2000.

[137] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. Technical report, Internet Engineering Task Force (IETF), February 1997. URL `https://www.rfc-editor.org/info/rfc2104`.

[138] Ashish Kundu and Elisa Bertino. Privacy-preserving authentication of trees and graphs. *Int. Journal of Information Security*, pages 1–28, 2013. ISSN 1615-5262. doi: 10.1007/s10207-013-0198-5. URL `http://dx.doi.org/10.1007/s10207-013-0198-5`.

[139] Lanneroe. Fighting the biggest lie on the Internet, Common Terms Beta Proposal. Technical report, Metamatrix AB, April 2013. URL `http://www.commonterms.net/commonterms_beta_proposal.pdf`.

[140] Lanneroe, Jakobsson, Bernstein, Toernquist, Arkestal, Walter, Aspelund, and Carlman. How can the biggest lie be stopped? Web Page, July 2015. URL `"How"athttp://www.biggestlie.com/`.

[141] Lanneroe, Jakobsson, Bernstein, Toernquist, Arkestal, Walter, Aspelund, and Carlman. Let's STOP the biggest lie on the web! Web Page, July 2015. URL `http://www.biggestlie.com`.

[142] Laura Brandimarte. The limits of control and transparency. Technical report, Heinz College, Carnegie Mellon University, March 2015. URL `http://www.meaningfulconsent.org/MCDE_2015/brandimarte.pdf`.

[143] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In *Applied Cryptography and Network Security*, pages 253–269, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi: 10.1007/978-3-540-72738-5_17. URL `http://link.springer.com/10.1007/978-3-540-72738-5_17`.

[144] Mingyan Li, Krishna Sampigethaya, Leping Huang, and Radha Poovendran. Swing & swap: user-centric approaches towards maximizing location privacy. In *Proc. of the 5th ACM Workshop on Privacy in Electronic Society*, pages 19–28. ACM, 2006.

[145] Antonio Liñán and Marc Fàbregas. RERUM Deliverable D5.2 - Smart object and application Implementation, 2015.

[146] S Lim, E Lee, and C.-M. Park. A short redactable signature scheme using pairing. *Security and Communication Networks*, 5(5):523–534, 2012.

[147] Helger Lipmaa. Secure Accumulators from Euclidean Rings without Trusted Setup. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS*, volume 7341 of *Lecture Notes in Computer Science*, pages 224–240. Springer, 2012. ISBN 978-3-642-31283-0.

[148] M A Lisovich, D K Mulligan, and S B Wicker. Inferring personal information from demand-response systems. *IEEE Security and Privacy*, 8(1):11–20, 2010. ISSN 1540-7993. doi: 10.1109/MSP.2010.40. URL http://dx.doi.org/10.1109/MSP.2010.40.

[149] L Liu, B Thuraisingham, M Kantarcioglu, and L Khan. An adaptable perturbation model of privacy preserving data mining. Technical Report UTDCS-04-06, Univ. of Texas at Dallas, 2006.

[150] Hal Lockhart and Bill Parducci. OASIS eXtensible Access Control Markup Language (XACML) Technical Committee. Web Page, July 2015. URL https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[151] Zhendong Ma, Frank Kargl, and Michael Weber. Pseudonym-on-demand: a new pseudonym refill strategy for vehicular communications. In *Vehicular Technology Conf., 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5. IEEE, 2008.

[152] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[153] Daniel P Martin, Elisabeth Oswald, Martijn Stam, and Marcin Wójcik. A leakage resilient MAC. In *Proc. of 15th IMA int. conf. on Cryptography and Coding (IMACC 2015)*, 2015.

[154] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: understanding the Tor network. In *Privacy Enhancing Technologies*, pages 63–76. Springer, 2008.

[155] Ross McGuinness. Terms and conditions may apply - Does anybody read internet Terms and Conditions? Web Page, July 2014. URL http://metro.co.uk/2014/07/01/terms-and-conditions-may-apply-does-anybody-read-internet-tcs-4781976/.

[156] Matthias Mehldau. Iconset for Data-Privacy Declarations v0.1. Technical report, Chaos Computer Club, Berlin, wetter@berlin.ccc.de, May 2007. URL http://netzpolitik.org/wp-upload/data-privacy-icons-v01.pdf.

[157] Stefan Meissner and Et. al. RERUM Deliverable D2.5 - Internet of Things - Architecture IoT-A, 2012.

[158] Ralph C. Merkle. A Certified Digital Signature. In *Proc. of Advances in Cryptology (CRYPTO' 89)*, pages 218–238, New York, NY, 1989. Springer, Springer New York. doi: 10.1007/0-387-34805-0_21. URL http://link.springer.com/10.1007/0-387-34805-0_21.

[159] Merkley, Green, Lee, and Lendl. Creative Common Licenses. Web Page, July 2015. URL `https://creativecommons.org/licenses`.

[160] Microsoft. The STRIDE threat model. Web Page, 2005. URL `https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx`.

[161] Joe Miller. City of London calls halt to smartphone tracking bins. Web Page, August 2013. URL `http://www.bbc.co.uk/news/technology-23665490`.

[162] K Miyazaki, S Susaki, M Iwamura, T Matsumoto, R Sasaki, and H Yoshiura. Digital documents sanitizing problem. Technical report, IEICE, 2003.

[163] K Miyazaki, M Iwamura, T Matsumoto, R Sasaki, H Yoshiura, S Tezuka, and H Imai. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A(1):239–246, January 2005. ISSN 0916-8508. doi: 10.1093/ietfec/E88-A.1.239. URL `http://search.ieice.org/bin/summary.php?id=e88-a_1_239&category=A&year=2005&lang=E&abst=`.

[164] Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *Proc. of the 2006 ACM Symp. on Information, computer and communications security*, ASIACCS '06, pages 343–354, New York, NY, USA, 2006. ACM. ISBN 1-59593-272-0. doi: http://doi.acm.org/10.1145/1128817.1128868. URL `http://doi.acm.org/10.1145/1128817.1128868`.

[165] A Molina-Markham, P Shenoy, K Fu, E Cecchet, and D Irwin. Private memoirs of a smart meter. In *Proc. of 2nd ACM BuildSys '10*, pages 61–66. ACM, 2010. ISBN 978-1-4503-0458-0. doi: 10.1145/1878431.1878446. URL `http://doi.acm.org/10.1145/1878431.1878446`.

[166] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. Towards accountable management of privacy and identity information. In *Computer Security–ESORICS 2003*, pages 146–161. Springer, 2003.

[167] T Mouroutis and Et. al. RERUM Deliverable D2.1 - Use cases definition and threat analysis, May 2014. URL `https://bscw.ict-rerum.eu/pub/bscw.cgi/d14540/RERUMdeliverableD2_1.pdf`.

[168] R Munne and Et. al. RERUM Deliverable D5.1 - Trial scenario Definitions and Evaluation, May 2015.

[169] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. of the 21st Ann. ACM Symp. on Theory of Computing*, pages 33–43. ACM, 1989.

[170] NIST FIPS. 198: The keyed-hash message authentication code (HMAC). *National Institute of Standards and Technology, Federal Information Processing Standards*, page 29, 2002.

[171] Kaisa Nyberg. Fast accumulated hashing. In *Fast Software Encryption*, pages 83–87, 1996. doi: 10.1007/3-540-60865-6_45. URL `http://link.springer.com/10.1007/3-540-60865-6_45`.

[172] OECD. The OECD Privacy Framework, including The Guidelines governing the protection of privacy and transborder flows of personal data. Technical report, OECD, July 2013. URL `http://oecd.org/sti/ieconomy/oecd_privacy_framework.pdf`.

[173] Republic of Mauritius. Mauritius Hosts 36th Int. Conf. of Data Protection and Privacy Commissioners. Web Page, October 2014. URL `http://www.govmu.org/English/News/Pages/Mauritius-Hosts-36th-International-Conference-of-Data-Protection-and-Privacy-Commissioners.aspx`.

[174] Chasey Oppenheim, Jackson Patrick, Gus Warren, Victor Toyens, Eason Goodale, Codi Mills, and Errol Grannum. Disconnect.me: privacy icon crowdsourcing effort. Web Page, 2012. URL `https://disconnect.me/icons`.

[175] Adem Orsdemir, H. Oktay Altun, Gaurav Sharma, and Mark F. Bocko. On the security and robustness of encryption via compressed sensing. In *Proc. of the IEEE Military Communications Conference (MILCOM)*, pages 1–7, 2008. ISBN 9781424426775. doi: 10.1109/MILCOM.2008.4753187.

[176] Siani Pearson and Marco Casassa Mont. Sticky policies: an approach for managing privacy across multiple parties. *Computer*, 44(9):60–68, 2011.

[177] S Peppet. Regulating the Internet of Things: First Steps Toward Managing Discrimination, Privacy, Security, and Consent. Technical report, Colorado Law, University of Colorado Boulder, November 2013. URL `http://www.texaslrev.com/wp-content/uploads/Peppet-93-1.pdf`.

[178] Scott R. Peppet. Tenured and Tenure-Track Faculty. Web Page, 2015. URL `https://lawweb.colorado.edu/profiles/profile.jsp?id=1`.

[179] Andreas Pfitzmann and Marit Hansen. A Terminology for Talking About Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. Technical report, 2010.

[180] H C Pöhls, K Samelin, J Posegga, and H de Meer. Flexible redactable signature schemes for trees — extended security model and construction. In *Proc. of the Int. Conf. on Security and Cryptography (SECRYPT 2012)*, pages 113–125. SciTePress, 2012. URL `http://web.sec.uni-passau.de/papers/2012-Poehls-Samelin-DeMeer-Posegga_SECRYPT12_Flexible-Redactable-Signature-Schemes-for-Trees.pdf`.

[181] Henrich C. Pöhls. Verifiable and revocable expression of consent to processing of aggregated personal data. In L Chen, M D Ryan, and G Wang, editors, *Information and Communications Security*, LNCS 5308, pages 279–293, Berlin, Heidelberg, 2008. Springer. doi: 10.1007/978-3-540-88625-9_19. URL `http://web.sec.uni-passau.de/papers/2008_Poehls_RevocableExpressionOfConsent_ICICS2008.pdfhttp://link.springer.com/10.1007/978-3-540-88625-9_19`.

[182] Henrich C Pöhls. JSON Sensor Signatures (JSS): End-to-End Integrity Protection from Constrained Device to IoT Application. In *Proc. of the Workshop on Extending Seamlessly to the Internet of Things (esIoT), collocated at the IMIS-2012 International Conference (IMIS 2015)*. Conference Publishing Service, 2015. URL `https://web.sec.uni-passau.de/papers/2015_Poehls-JSONSensorSignatures_esIoT.pdf`.

[183] Henrich C. Pöhls and Et. al. RERUM Deliverable D3.2 - Privacy enhancing techniques in the Smart City applicaons, 2015.

[184] Henrich C Pöhls and Markus Karwe. Redactable Signatures to Control the Maximum Noise for Differential Privacy in the Smart Grid. In Jorge Cuellar, editor, *Proc. of the 2nd Workshop on Smart Grid Security (SmartGridSec 2014)*, volume 8448 of *Lecture Notes in Computer Science (LNCS)*. Springer International Publishing, 2014. URL `http://web.sec.uni-passau.de/papers/2014_Poehls_Karwe_RedactableSignaturesToControlTheMaximumNoiseForDifferentialPrivacyInTheSmartGrid.pdf`.

[185] Henrich C Pöhls and Kai Samelin. On updatable redactable signatures. In *Proc. of the 12th Int. Conf. on Applied Cryptography and Network Security*, pages 457–475. In: {ACNS}, 2014. doi: 10.1007/978-3-319-07536-5_27. URL `http://link.springer.com/10.1007/978-3-319-07536-5_27`.

[186] Henrich C. Pöhls and Kai Samelin. Accountable Redactable Signatures. January 2015.

[187] Henrich C Pöhls, Arne Bilzhause, Kai Samelin, and Joachim Posegga. Sanitizable Signed Privacy Preferences for Social Networks. In *Proc. of GI Workshop on Privacy and Identity Management for Communities - Communities for Privacy and Identity Management (DICCDI 2011)*, GI-Edition Lecture Notes in Informatics (LNI). GI, October 2011. URL `http://web.sec.uni-passau.de/papers/2011_Poehls-Bilzhause-Samelin-Posegga_Sanitizable-Signed-Privacy-Preferences-for-Social-Networks_DICCDI-2011.pdf`.

[188] Henrich C Pöhls, Kai Samelin, and Joachim Posegga. Sanitizable signatures in XML signature — performance, mixing properties, and revisiting the property of transparency. In *9th Int. Conf. on Applied Cryptography and Network Security (ACNS 2011)*, volume 6715 of *LNCS*, pages 166–182. Springer, 2011. doi: 10.1007/978-3-642-21554-4_10. URL `http://link.springer.com/10.1007/978-3-642-21554-4_10`.

[189] Henrich C Pöhls, Bendikt Petschkuhn, Johannes Rückert, and Max Mössinger. Aggregation and Perturbation in Practice: Case-Study of Privacy, Accuracy and Performance. In *IEEE Int. Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (IEEE CAMAD 2014)*. IEEE, 2014. doi: http://dx.doi.org/10.1109/CAMAD.2014.7033231. URL `https://web.sec.uni-passau.de/papers/2014_PoehlsMoessingerPetschkuhnRueckert_AggregationAndPerturbationInPractice_CAMAD2014.pdf`.

[190] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. Analyzing the energy consumption of security protocols. In *Proc. of the 2003 Int. Symp. on Low power electronics and design*, pages 30–35. ACM, 2003.

[191] N F Pub. DRAFT FIPS PUB 202: SHA-3 standard: Permutation-based hash and extendable-output functions. *Federal Information Processing Standards Publication*, 2014.

[192] Y Rachlin and D Baron. The secrecy of compressed sensing measurements. In *Proc. of Allerton Conf. on Communication, Control, and Computing*, pages 813–817, 2008.

[193] K Raeburn. Advanced Encryption Standard (AES) Encryption for Kerberos 5. *RFC 3962*, February 2005. doi: 10.17487/rfc3962. URL `https://www.rfc-editor.org/info/rfc3962`.

[194] Lee Rainie and Janna Anderson. The future of privacy. Technical report, Pew Research Center, December 2014. URL http://www.pewinternet.org/2014/12/18/other-resounding-themes/.

[195] Raskin, Betz et.al. No Title. Web Page, June 2011. URL https://wiki.mozilla.org/Privacy_Icons.

[196] S Rass and D Slamanig. *Cryptography for Security and Privacy in Cloud Computing*. Artech House, 2013. ISBN 1-60807-575-3.

[197] Erik Rissanen. eXtensible Access Control Markup Language (XACML) Version 3.0. Standard urn:oasis:names:tc:xacml:3.0:core:schema:wd-17, OASIS, January 2013. URL http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[198] R L Rivest, A Shamir, and L Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/357980.358017.

[199] Hugo Roy. Panel about consent. Web Page, public hearing at the European Parliament on Data Protection, January 2013. URL https://tosdr.org/ep-hearing-201301.html.

[200] Hugo Roy, Michiel de Jong, Jan-Christoph Borchardt, Ian McGowan, Jimm Stout, and Suzanne Azmayesh. Termos of Service; Didn't read. Web Page, June 2012. URL https://tosdr.org/.

[201] D Ruiz and Et. al. RERUM Deliverable D3.1 - Enhancing the autonomous smart objects and the overall system security of IoT based Smart Cities, March 2015. URL https://bscw.ict-rerum.eu/pub/bscw.cgi/d14540/RERUMdeliverableD3_1.pdf.

[202] John Sabo and Gershon Janssen. OASIS privacy management reference model (PMRM). Web Page, April 2015. URL https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=pmrm.

[203] Kai Samelin, Henrich C Pöhls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. On Structural Signatures for Tree Data Structures. In *10th Int. Conf. on Applied Cryptography and Network Security*, volume 7341 of *LNCS*, pages 171–187. Springer-Verlag, 2012.

[204] Kai Samelin, Henrich C Pöhls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. Redactable signatures for independent removal of structure and content. In *Proc. of Information Security Practice and Experience (ISPEC 2012)*, volume 7232 of *LNCS*, pages 17–33. Springer, 2012. doi: 10.1007/978-3-642-29101-2_2. URL http://link.springer.com/10.1007/978-3-642-29101-2_2.

[205] Tomas Sander. Efficient Accumulators without Trapdoor Extended Abstracts. In *ICICS*, pages 252–262, 1999.

[206] Ralf-Peter Schäfer, Kai-Uwe Thiessenhusen, Elmar Brockfeld, and Peter Wagner. A traffic information system by means of real-time floating-car data. In *ITS World Congr.*, 2002.

[207] Schraefel, Gerding, Vlassopoulos, Thomas, Gerding, Tonin, Vlassopoulos, Thomas, Gerding, Tonin, and Vlassopoulos. Meaningful consent in the digital economy. Web Page, February 2014. URL http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/K039989/1http://www.meaningfulconsent.org.

[208] Stefan Schultz. Planungschaos: Regierung vermasselt Start neuer Stromzaehler-Generation. Web Page, April 2015. URL http://www.spiegel.de/wirtschaft/service/energiewende-teures-planungschaos-bei-intelligenten-stromzaehlern-a-1026859.html.

[209] David Searls. EmanciTerm. Web Page, December 2011. URL http://cyber.law.harvard.edu/projectvrm/EmanciTerm.

[210] Claude E Shannon. Communication Theory of Secrecy Systems*. *Bell System Technical Journal*, 28(4):656–715, October 1949. ISSN 00058580. doi: 10.1002/j.1538-7305.1949.tb00928.x. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6769090.

[211] E Shi, T.-H. H Chan, E G Rieffel, R Chow, and D Song. Privacy-preserving aggregation of time-series data. In *Proc. of the Network and Distributed System Security Symp. (NDSS 2011)*. The Internet Society, 2011. URL http://dblp.uni-trier.de/db/conf/ndss/ndss2011.html#ShiCRCS11.

[212] Nicolas Sklavos and O Koufopavlou. On the hardware implementations of the SHA-2 (256, 384, 512) hash functions. In *Proc. of the 2003 Int. Symp. on Circuits and Systems (ISCAS'03)*, volume 5, pages V—-153. IEEE, 2003.

[213] Daniel Slamanig and Stefan Rass. Generalizations and Extensions of Redactable Signatures with Applications to Electronic Healthcare. In *Communications and Multimedia Security*, pages 201–213, 2010.

[214] SmartCitiesCouncil. London dumps its smartphone-tracking trash bins. Web Page, August 2013. URL http://smartcitiescouncil.com/article/london-dumps-its-smartphone-tracking-trash-bins.

[215] SmartCitiesCouncil. Teaming to build the cities of the future. Web Page, 2015. URL http://smartcitiescouncil.com.

[216] Matt Snyder. Youluh in a Nutshell. Web Page, July 2012. URL http://www.x2xroads.com/.

[217] Florian Stahl and Stefan Burgmair. The OWASP Top 10 Privacy Risks Project. Web Page, July 2015. URL https://www.owasp.org/index.php/OWASP_Top_10_Privacy_Risks_Project.

[218] Ron Steinfeld, Laurence Bull, and Y Zheng. Content Extraction Signatures. In *4th Int. Conf. on Information Security and Cryptology (ICISC 2001)*, volume 2288, pages 163–205. Springer, 2002.

[219] E Tragos and Et. al. RERUM Deliverable D2.3 - System Architecture, September 2014. URL https://bscw.ict-rerum.eu/pub/bscw.cgi/d14540/RERUMdeliverableD2_3.pdf.

[220] TRIALOG. EU Project PRIPARE: PReparing Industry to Privacy-by-design by supporting its Application in REsearch. Web Page, 2013. URL http://pripareproject.eu/.

[221] TZ-Online. Stasi-Barbie: Neue Puppe belauscht Ihre Kinder. Web Page, February 2015. URL `http://www.tz.de/multimedia/stasi-barbie-neue-puppe-belauscht-ihre-kinder-4755509.html`.

[222] UN General Assembly. The right to privacy in the digital age. Technical Report A/C.3/69/L.26/Rev.1, Sixty ninth session, Third Committee, November 2014. URL `http://www.un.org/ga/search/view_doc.asp?symbol=A/C.3/69/L.26/Rev.1`.

[223] U.S. Department of Commerces International Trade Administration et.al. U.S.-EU and U.S.-Swiss Safe Harbor Frameworks. Web Page, May 2015. URL `http://www.export.gov/safeharbor/`.

[224] Atul Varma, Gary Kovacs, and Emily Carr. Lightbeam for Firefox. Web Page, July 2015. URL `https://www.mozilla.org/de/lightbeam/`.

[225] John Viega, Matt Messier, Pravir Chandra, Matt Messier, and John Viega. Network Security with {OpenSSL}: Cryptography for Secure Communications. *O'Reily, June*, 2002. URL `http://safari.oreilly.com/059600270X;$\delimiter"026E30F$nhttp://www.oreilly.com/catalog/openssl`.

[226] David von Oheimb. {IT} Security architecture approaches for {smart metering} and {smart grid}. In *SmartGridSec*, pages 1–25, 2012.

[227] W3C. W3C: Privacy enhancing browser extensions. Technical report, W3C, 2012. URL `http://www.w3.org/2011/D1.2.3/`.

[228] Peter Waher. Internet of Things - Sensor Data. Experimental Standard XEP-0323, v0.4, XMPP Standards Foundation, March 2015. URL `http://xmpp.org/extensions/xep-0323.html`.

[229] Watkinson, Hall, Picciafuocco, and Casanova. Docracy: free legal documents. Web Page, July 2015. URL `http://www.docracy.com`.

[230] Rob Waugh. Watch them watching you: Privacy site that gives instant rating for how websites use and abuse your details. *Daily Mail Online*, pages 13:28 GMT, 14 February 2012, February 2012. URL `http://www.dailymail.co.uk/sciencetech/article-2100958/Who-watching-Privacy-Score-lets-EVERY-company-watching-you-visit-webs-sites.html`.

[231] G Werner, K Lorincz, M Ruiz, O Marcillo, J Johnson, J Lees, and M Welsh. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing, Special Issue on Data-Driven Applications in Sensor Networks*, 10:18–25, 2006.

[232] J Williams and Y Li. *Estimation of Mutual Information: A Survey*. Rough Sets and Knowledge Technology, Lecture Notes in Computer Science, Springer, 2009.

[233] Christopher Wolf and Jules Polonetsky. An Updated Privacy Paradigm for the Internet of Things. Technical report, Future of Privacy Forum, November 2013. URL `http://www.futureofprivacy.org/wp-content/uploads/Wolf-and-Polonetsky-An-Updated-Privacy-Paradigm-for-the-%E2%80%9CInternet-of-Things%E2%80%9D-11-19-2013.pdf`.

[234] Wolfram Research. logistic map, 1234. URL `http://documents.wolfram.com`.

[235] Zhen-Yu Wu, Chih-Wen Hsueh, Cheng-Yu Tsai, Feipei Lai, Hung-Chang Lee, and Yufang Chung. Redactable signatures for signed CDA documents. *Journal of Medical Systems*, 36(3):1795–1808, June 2012. ISSN 0148-5598. doi: 10.1007/s10916-010-9639-0. URL http://link.springer.com/10.1007/s10916-010-9639-0.

[236] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. LINDDUN: Unawareness of entity. Web Page, August 2014. URL https://distrinet.cs.kuleuven.be/software/linddun/contentunawareness_E.php.

[237] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. LINDDUN privacy threat modeling. Web Page, July 2015. URL https://distrinet.cs.kuleuven.be/software/linddun/.

[238] L Zhang, K Wong, C Li, and Y Zhang. Towards secure compressive sampling scheme. *CoRR*, abs/1406.1, 2014. URL http://arxiv.org/abs/1406.1725.

[239] H Ziekow, C Goebel, J Strüker, and H.-A. Jacobsen. The potential of smart home sensors in forecasting household electricity demand. In *SmartGridComm*, 2013.

A Privacy Analysis of RERUM Requirements and Use Cases

RERUM claims to focus on "security, privacy, and reliability by design". However there are several objectives, goals and requirements in RERUM itself conflicting essentially with privacy needs of data subjects. Is the citizen really at the centre of attention? "…Things, people, data, and processes are *readable, …, and controllable via Internet …*" (deliverable D2.1, section 1.3) does not really sound privacy-respecting, neither does one of the RERUM objectives that aims to "…investigate adaptation of Cognitive Radio (CR*) technology in smart objects …to minimize wireless interference and ensure the *always connected* concept …". In the following we take a brief look at potentially privacy conflicting requirements in RERUM.

A.1 Application

RERUM requires the possibility for remote control of devices *by the user*, who not always may be the actual data subject recorded by the device's sensors. Approved sensor data can be released to applications. However there should be an adjustable rate of data collection and transmission *by the application*. Here it must be taken care not to allow the application to exceed consented ranges. RERUM also requires time-efficient connectivity for data uploading *meeting application-needs*, which however we need to balance with data subject's needs to meet privacy needs.

A.2 Networking and QoS

RERUM requires that a large number of devices, and network partitioning needs to be supported with *centralised management* in constrained networks. Centralised components however always required extra care regarding privacy issues. The requirement for *ubiquitous connectivity* wishes to ensure that devices may have the option to select any available network operator and technology to connect to the Internet (*CR-support*). This for instance may make it hard for data subjects to control communication of such devices. Also reconfigurable wired and wireless communication interfaces to sensors / actors and to other devices (*for CR-support*) may be hard to control and restrain by data subjects, as does dynamic spectrum management, distributed spectrum selection, and the permission for devices to operate freely in both licensed and unlicensed spectrum bands (also for *for CR-support*). Need for data subjects to control behaviour of devices also may conflict with the RERUM requirement to meet application QoS, and support self-* mechanisms (*zero user interference*).

A.3 Devices, Gateways

RERUM requires sufficient performance, main memory and persistent storage for their IoT devices. They need to be sturdy, power-efficient, preferably with low energy consumption. This also applies to software, where lightweight algorithms are demanded. Devices are to be *OTA-programmable*, e.g. for a remote firmware update, which makes them harder to control for data subjects.

A.4 Virtualization, Middleware

Monitoring and traceability of middleware, devices, services, applications is a RERUM requirement. However extensive monitoring and tracing capabilities allows at least the administrators of an IoT infrastructure for privacy infringing behaviour. There needs to be filtering and decision making, including *accounting support* in the middleware. Basis for accounting is a careful and detailed logging, which may impair privacy of data subjects as well.

A.5 Security

CIA is demanded by RERUM in transit and at rest, attribute-based access control (use of XACML) is recommended. One needs to be aware that security in itself may be privacy violating, if they impair a data subject's capability to repudiate actions, thoughts, feeling and other privacy related aspects. Reputation mechanisms also require *extensive monitoring* to achieve a certain level of trustworthiness.