Dipl.-Ing. Alexander Rech, BSc

Towards Cross-Domain Service Access

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Eugen Brenner

Institute of Technical Informatics

Advisor

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Graz, December 2020

AFFIDAVIT

Date

Signature

Acknowledgments

This thesis has been carried out at Graz University of Technology, Institute of Technical Informatics, in cooperation with the industrial partner CISC Semiconductor GmbH. I want to thank Dr. Markus Pistauer for enabling my PhD studies and allowing me to carry out my research in the course of different industrial projects. Furthermore, I would like to express my gratitude to my advisor Ass.Prof. Dr. Christian Steger for his continuous support and mentorship throughout the entire research project. Additionally, I would like to thank my supervisor Ao.Univ.-Prof. Dr. Eugen Brenner, for his open mind and help during my PhD. I also appreciate the excellent work of the students I have supervised and who were involved in my ideas and projects with great interest and ambition. Additional thanks go to Assoc. Prof. Dr. Damian Dalton, who kindly agreed to serve as a second adviser for my thesis. Finally, I would like to express my sincere gratitude to my partner and my family: Thank you for being a part of my life, always watching my back, and supporting me throughout my journey!

Graz, December 2020
Alexander Rech

Abstract

A smart city is an urban area that consists of a large number of information systems handling services across diverse domains, such as economy, environment, health, living, mobility, and security. These information systems are often realized as client-server systems offering their clients various resources via different service interfaces. Many services such as smart parking, renting or sharing vehicles, placing orders via mobile applications, or accessing online data streams are already part of our everyday life. However, carefully examined, there is low interaction between different client-server systems of contrasting domains, which has resulted in the creation of a massive number of independent applications. Additionally, a lack of standardization, environmental restrictions, and specific use case requirements impede the usage of standard service access methods. Finally, due to an interplay of heterogenous data processing systems, also secure and privacy conform data-processing poses a challenge.

We envision pervasive computing technologies to facilitate seamless interaction between service providers, their users, and the environment. The aim is to address more clients, make services and their resources accessible more efficiently and securely, and save time when booking, redeeming, or merely utilizing services, thus facilitating users' daily activities. This leads us to our main research question: How can we provide people with personalized and secure cross-domain service access? In this thesis, a multi-layered service-oriented platform is presented, paving the way for a paradigm shift towards "smart city as a service". It enables businesses and their pre-existing client-server systems to issue, forward, redeem, and exchange digital services seamlessly. Distinct authentication and authorization mechanisms are integrated into the platform to comply with diverse environmental restrictions and application requirements. Furthermore, a particular focus is on secure data aggregation while preserving the users' privacy. The overall platform is designed, implemented, and evaluated in the course of several industrial projects. Performance measurements as well as acceptance and usability studies have been conducted successfully. Our evaluation results show that our cross-domain service management system achieved low execution times, along with high system acceptance and usability levels.

Zusammenfassung

Eine “smart city” ist ein städtisches Gebiet, das sich aus einer Vielzahl an Informationssystemen aus unterschiedlichen Domänen, wie z.B. Wirtschaft, Umwelt, Gesundheit, Wohnen, Mobilität und Sicherheit zusammensetzt. Diese Informationssysteme werden oft als Client-Server-Systeme umgesetzt, die ihren Clients über diverse Serviceschnittstellen unterschiedliche Ressourcen anbieten. Viele Services wie Smart Parking, das Ausleihen von Fahrzeugen, das Aufgeben von Bestellungen über mobile Anwendungen oder der Zugriff auf Data-Streams gehören bereits zu unserem Alltag. Bei sorgfältiger Prüfung zeigt sich jedoch, dass die Interaktion zwischen verschiedenen Client-Server-Systemen unterschiedlicher Domänen gering ist, was zu einer massiven Anzahl unabhängiger Anwendungen führt. Außerdem erschweren mangelnde Standardisierung, diverse Umgebungseinschränkungen und spezifische Use-Case-Anforderungen den Einsatz von Standardmethoden für den Zugang zu diversen Services. Darüber hinaus stellt sichere und datenschutzkonforme Datenverarbeitung, aufgrund des Zusammenspiels heterogener Systeme, eine Herausforderung dar.

Die nahtlose Interaktion zwischen Service Providern, ihren NutzerInnen und der Umgebung soll erleichtert werden. Es sollen mehr User angesprochen, Ressourcen leichter und sicherer zugänglich gemacht sowie Zeit bei der Service-Buchung, Service-Einlösung oder Service-Nutzung gespart werden. Dies führt uns zu unserer Hauptforschungsfrage: Wie können wir personalisierten und sicheren domänenübergreifenden Service-Zugang ermöglichen? In dieser Dissertation wird eine mehrschichtige, Service-orientierte Plattform vorgestellt, welche den Weg für einen Paradigmenwechsel hin zu “smart city as a service” ebnet. Sie ermöglicht Service Providern und ihren bereits bestehenden Client-Server-Systemen die Ausgabe, Weiterleitung, Einlösung und den Austausch digitaler Services. Entsprechend unterschiedlicher Umgebungsrestriktionen und Anwendungsanforderungen werden mehrere Authentifizierungs- und Autorisierungsmechanismen in die Plattform integriert. Darüber hinaus wird ein besonderer Schwerpunkt auf die sichere Datenaggregation unter Wahrung der Privatsphäre der NutzerInnen gelegt. Das resultierende System wurde im Rahmen mehrerer Industrieprojekte designt, implementiert und evaluiert. Performance-Messungen sowie Akzeptanz- und Usability-Studien wurden erfolgreich durchgeführt. Unsere Evaluierungsergebnisse zeigen, dass unser domänenübergreifendes Service-Management-System sowohl niedrige Ausführungszeiten als auch eine hohe Systemakzeptanz und Benutzerfreundlichkeit erreicht hat.

Contents

List of Figures

List of Tables

List of Abbreviations

ASP average success probability.

A-Token Authentication-Token.

a-ID application-ID.

API application programming interface.

ATT Attribute Protocol.

BLE Bluetooth Low Energy.

CIA Confidentiality-Integrity-Availability.

CSE Connected Services Engine.

DSE Design Space Exploration.

ECC Elliptic Curve Cryptography.

eMaaS electro-Mobility-as-a-Service.

GAP Generic Access Profile.

GATT Generic Attributes Profile.

HSM Hardware Security Module.

HTTP HyperText Transfer Protocol.

HWC hardware-based cryptography.

HWF hardware-based firewall.

ICT Information and Communications Technology.

IoT Internet of Things.

MCU	microcontroller unit.
NFC	Near Field Communication.
O-Token	OAuth-Token.
OSGi	Open Service Gateway Initiative.
PKI	Public-Key-Infrastructure.
P-Token	Privacy-Token.
PKG	private-key generator.
RFID	radio-frequency identification.
SE	secure element.
S-Token	Service-Token.
s-ID	service-ID.
SCOTT	Secure Connected Trustable Things.
SIG	Bluetooth Special Interest Group.
SOA	Service-Oriented Architecture.
SQL	Structured Query Language.
STEVE	Smart-Taylored L-category Electric Vehicle demonstration in hEterogeneous urban use cases.
STIP	Secured Trustworthy IoT Platform.
u-ID	user-ID.
UWB	Ultra-Wideband.
Wi-Fi	Wireless Fidelity.
WSDL	Web Service Description Language.
WSN	Wireless Sensor Network.

1 Introduction

This thesis elaborates on several aspects of providing personal and secure access to heterogeneous services and their related digital and physical resources. The introductory chapter lists the thesis' remarks in section 1.1 and points out motivational aspects within section 1.2. Subsequently, the problem statement (1.3) and the scientific contributions (1.4) are discussed. An outline of this doctoral thesis is given in section 1.5.

1.1 Thesis Remarks

This thesis was carried out at Graz, University of Technology, Institute of Technical Informatics, in cooperation with the industrial partner CISC Semiconductor GmbH and its subsidiary companies COYERO GmbH and COYERO Inc. While the Institute of Technical Informatics offers research and education on modern networked embedded systems, CISC is an international company within the fields of automotive, radio-frequency identification (RFID), and Internet of Things (IoT) based in Klagenfurt and Graz (Austria), as well as Mountain View, California (US). CISC's expertise in the IoT domain lies in secure BLE and NFC application services for smartphones and embedded devices, as well as the design and implementation of distributed cloud-based systems. Being involved in joint R&D activities with major companies in the IoT sector and partners in the area of microelectronic systems, CISC handles the increased system complexity with its widespread system know-how from embedded to the cloud layer. Furthermore, this thesis has profited from the influence of European and Austrian projects and the close cooperation with well-known organizations in the field of IoT and connected services, including partners from universities, municipalities, and industry.

1.2 Motivation

Information and Communications Technology (ICT) plays an essential role in our increasingly connected world. Multiple independent service providers and their heterogeneous client-server systems are distributed across urban areas and provide people with different services. Various methods for accessing services and their resources via interaction with corresponding infrastructural elements are already part of today's cities. Fig. 2.1 depicts a few service examples, spanning from locally available services (e.g., getting access to resources such as parking lots or vehicles) to online services (e.g., ordering food or receiving

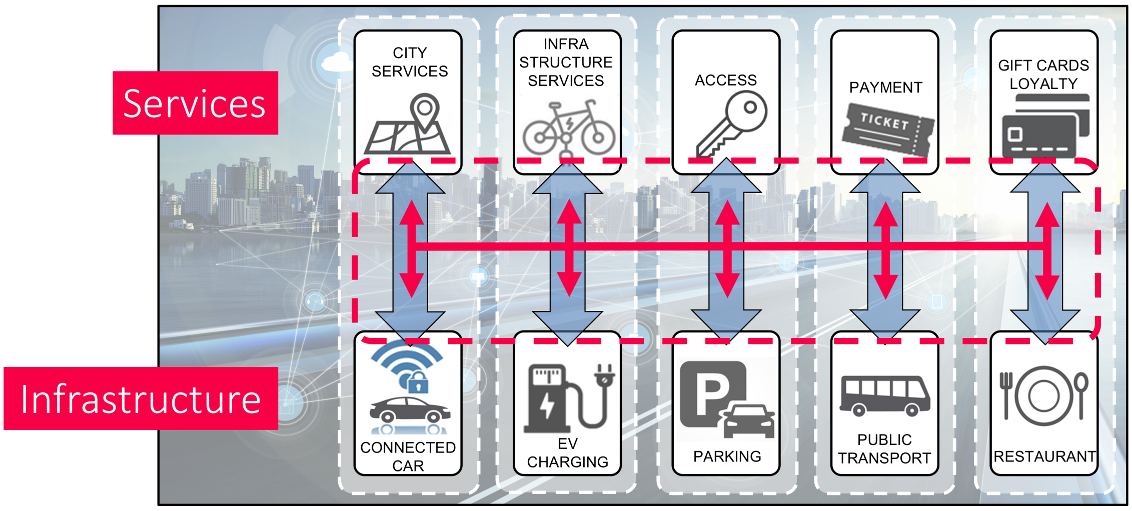


Figure 1.1: "Smart city as a service" – Interconnecting heterogeneous services and infrastructural elements within urban areas is important for providing seamless, secure, and privacy-preserving resource access (adapted from [35])

permission to access sensory data streams). We envision a new paradigm shift towards "smart city as a service"; seamless interaction between service-oriented systems, users, and the environment shall be possible. The primary challenge now is to overcome the diverse jungle of independent service-oriented information systems by defining a common strategy for sharing service-related data across one or multiple applications. This can be seen as an opportunity to extend the functionality of existing client-server applications and facilitate heterogeneous cross-domain service and resource access within metropolitan areas.

1.3 Problem Statement

Providing an interrelationship between heterogeneous client-server systems across different hardware and software layers concerning personalized access is not trivial. Different communication standards, environments, systems and subsystems, and consequently, diverse use cases and ways to access, manage, and redeem services complicate the matter. Complementary to this, security risks and privacy concerns often impede the evolvement of new technologies. Due to these restrictions and issues, this thesis' main research question is: *How can we provide people with personalized and secure access to services and their resources coming from client-server systems of different domains?* This high-level question and its associated challenges can further be subdivided into three coherent subtopics:

1. **Trusted cross-domain service exchange**. Interaction between service providers beyond their domain and application area is difficult to achieve. One reason for this is a lack of *standardization.* While many custom-tailored solutions for specific application areas exist, there is no common strategy for exchanging service-entitlements of distinct domains across different or even single applications. Usually, each service provider can address only a specific set of services and users per application. What can be done from a technical standpoint to facilitate collaboration between different client-server systems? Additionally, in business applications, participants such as different companies may identify but not fully *trust* each other. Enhancing trust in the sense of data integrity, confidentiality, and availability for inter-domain transactions is essential. Last but not least, customers tend to be creatures of habit. Once they are used to specific services and know how to obtain and handle them, they often no longer look around for new services without an *incentive*, a specific reason, or an advantage. Conversely, users often do not know which new services and corresponding offers are available. How can service providers from different domains incentivize users to make use of new services?

2. **Heterogeneous ways to access different resources.** Due to various applications, use cases, and environmental restrictions regarding connectivity, no standard authentication and authorization procedures for heterogeneous resource access is applicable. Nevertheless, different methods to facilitate seamless access to local and digital resources in different environments are required.

3. **Secure and privacy-preserving data collection.** Due to various ubiquitous information processing systems, we have to deal with a growing data privacy awareness of users and regulatory institutions. Increasingly stringent data protection rules enforce the correct use and processing of user-related data. However, since abstraction comes with information loss, it is difficult to determine which abstraction level is required to guarantee user data privacy while still aggregating enough information to let a system perform its tasks correctly. Furthermore, while the transition to a more connected world gives rise to new possibilities, new problems such as security issues arise. Therefore, secure processes are required to establish a trusted relationship between different distributed devices. Different security aspects from the embedded to the cloud level, including secure key management and data distribution, must be addressed.

Overcoming these limitations is critical for providing scalable solutions to interconnecting heterogeneous client-server systems and improving the interrelationship between their services, users, and the environment in a secure and privacy-preserving way.

1.4 Contributions

The overall scientific contributions of this thesis are summarized in the following paragraphs. Each aspect is covered by a dedicated design chapter. Parts of the elaborated concepts were integrated into the distributed service platform COYERO[1]. Its primary system model from 2017, explained in section 2.2, was our starting point.

Service Co-Modality – connect and gamify:
Client-server-based systems are widely used to manage and offer services of different kinds. However, their offers in the form of products and services often have a scope limited to a single company or a small set of vendors. The lack of standardization and the resulting jungle of heterogeneous implementations impede an easy exchange of business-related data with other independent systems. In order to introduce new means of collaboration, the general idea is to provide a common trusted overlay on top of existing client-server systems whose purpose is to anonymize and share services in the form of digital service-entitlements across independent service providers, their systems, and users. The overall platform, its advantages, limitations, and how it can be utilized to map different user and service-related datasets while at the same time preserving the users' anonymity are described in this thesis. Furthermore, we leverage the tamper-resistant properties of the blockchain and smart contracts to protect sensitive transactions against fraudulent manipulation and to introduce service-based agreements to the overall system. Additionally, gamification concepts are taken into account. They often aim towards goals of marketing and can, therefore, be used in special-offer strategies. Especially in urban areas where many independent information systems come together, it is all the more important to provide higher collaboration between different service providers in order to exploit the full potential of an extensive inter-company service-based reward system.

Ubiquitous access; local and online authentication and authorization:
Authentication and authorization are essential functions concerning service and resource access. However, different environmental situations and use cases impede the usage of a single standard resource access scheme. In this thesis, a distributed middleware was elaborated, responsible for issuing, canceling, and redeeming digital tokens. Depending on the environmental circumstances, different token types can enable local or online redemption of digital vouchers to access physical and virtual resources.

Adaptive privacy and end-to-end secured data aggregation:
Given increasingly stringent data protection rules and increased privacy awareness of users, special attention must be paid to how user data is processed. This is especially important when it is passed across multi-layered cloud environments and systems of independent

[1]https://www.coyero.com (last access: 1st December 2020)

service providers. This part of the thesis focuses on an approach for letting users adapt the data aggregation flow during the runtime of a mobile application according to their privacy preferences while still providing mechanisms to anonymize the data collected. Additionally, often data of confidential nature has to be handled. Therefore, end-to-end security methodologies are essential. Various protection schemes on different hardware and software layers have to be taken into account for finding the optimal security design for the desired application. A case study has been conducted based on a design-space-exploration tool that identified a set of potential hardware and software building blocks for secure system design.

1.5 Outline

This thesis is organized as follows. Chapter 2 gives an overview of our terminology, the basic system model, important security attributes and mechanisms, wireless communication protocols, and related research and development projects. Consequently, chapter 3 discusses existing work in the areas of interoperable connected systems and smart access. The subsequent chapter 4 presents our platform's requirements, a system overview, and explanations of its subcomponents. While section 4.4 takes into account the architectural descriptions from papers [88, 89, 90], section 4.5 covers the design concepts elaborated within the work [86]. Eventually, section 4.6 incorporates ideas discussed in [87, 91]. Implementational details are provided in chapter 5, while an evaluation of the overall system is given in chapter 6. Consequently, the summarized ideas beyond the state of the art and information on future work are pointed out in chapter 7. Finally, more details on the published work related to this thesis are presented in chapter 8.

2 Background

This chapter provides definitions of key terms as well as technical and organizational background information related to this thesis.

2.1 Terminology

The following section comprises the essential terms used in this thesis and how they are defined within the scope of this work.

- **Access.** Combination of authentication (the process of confirming that someone is who he/she claims to be) and authorization (the process of verifying if someone is allowed or has the permission to perform a specific action).
- **Service.** In economics, a service is a transaction where no physical goods are transferred from a seller to a buyer. By contrast, in the context of software architectures, the term service refers to reusable software functionalities. In this thesis, the term service addresses software components responsible for the issuance, cancellation, redemption, or generally speaking, the management process of digital entitlements used for accessing physical or digital resources (products, data, etc.).
- **Service Provider.** A service provider manages a set of services. These services are linked to a specific domain. In this sense, a parking-service provider is in charge of services related to parking, and a mobility-service provider provides mobility-specific services.
- **Resource.** Resources can be defined as physically or digitally available goods, such as products or licenses.
- **Token.** Tokens are digital data structures created by software and assigned a certain purpose, e.g., providing authentication or authorization.
- **User.** A user is a person who utilizes a device and its application.

2.2 Basic System Model

Fig. 2.1 shows the basic system model based on the COYERO system that will accompany the reader through this thesis. It consists of a client-server module that enables information exchange between clients (service requester) and servers (service provider). Furthermore, an additional kiosk unit is part of the overall system model for enabling service redemption and consequent resource access.

- **Client.** This entity acts as an interface for requesting and obtaining services.
- **Kiosk.** A kiosk is a validation authority and service redemption unit. It authenticates and authorizes a client to redeem services and access the requested resources.
- **Server.** The server is a central computer that communicates with client and kiosk devices and keeps them synchronized. It is responsible for managing (issuing, forwarding, canceling, etc.) different services.

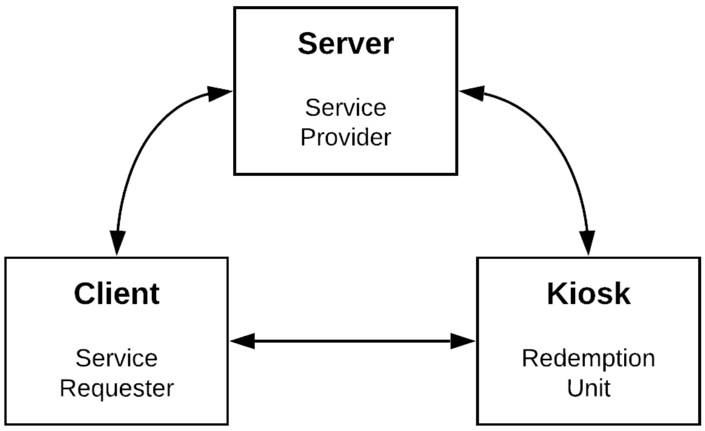


Figure 2.1: Basic system model: service requester (client), redemption unit (kiosk), service provider (server)

2.3 Security Attributes

The Confidentiality-Integrity-Availability (CIA) triad summarizes the most important security aspects of information security. The following attributes are important to ensure an appropriate level of security [13]:

- **Confidentiality.** Information should remain confidential. Only authorized entities should be able to access data.
- **Integrity.** Ability to protect data from fraudulent, unauthorized changes.
- **Availability.** Probability that the system performs its intended function when it is required.

The compliance of secondary security attributes is also important for protecting systems against cyberattacks. The following examples complement the attributes mentioned before:

- **Authenticity.** Integrity of a message (content, time of emission, etc.) and its origin.
- **Nonrepudiability.** Integrity and availability of the identity of the sender or receiver of a message.
- **Accountability.** Integrity and availability of the identity of a person or device that performed a specific operation.

2.4 Distributed Ledger Technology

A blockchain is a type of distributed ledger, i.e., a peer-to-peer network architecture consisting of a distributed database that is shared and synchronized across all network participants. There is no central authority to manage this database, reducing the risk of a single point of failure. Data uploaded to the blockchain network (as a transaction) is aggregated into blocks linked to each other. The first block is called the genesis block. It defines the properties of the blockchain as well as the underlying consensus algorithm. The consensus algorithm specifies under which criteria valid blockchain entries are generated with the involvement of network authorities. No single party has control over the data. Depending on whether the majority votes in favor of accepting a new block containing multiple transactions, the block is appended to the blockchain or rejected. The consensus mechanism should be designed in a way that acting against the rules does not pay off. Blockchains can be public. In this case, information is available in the public domain, and any of the network participants can verify transactions. Alternatively, blockchains can be private. In this case, only a set of authorized individuals can process new transactions. A combination of both methodologies is called a federated blockchain. Here, only people with an explicit invitation are authorized to participate in the process of reading, writing, or verifying transactions. The distributed ledger may support smart contracts for establishing a relationship between different parties. A smart contract is software that runs on the distributed ledger, consisting of inputs, deterministic outputs, custom data structures, and trigger-conditions. It can be utilized to create additional rules according to the requirements of the underlying application. A deployed contract is assigned to a random blockchain address. It is used as a reference to invoke the contract and its functions. Smart contracts are executed by the blockchain nodes due to processing transactions by the network validators, for example. Results are stored as transaction onto the blockchain. Transactions cannot be deleted or altered due to the blockchain's tamper-resistant design relying on cryptographic primitives such as hash functions, asymmetric cryptographic signatures, etc. However, the blockchain also comes with a few disadvantages. Consensus algorithms are highly use case and application area dependent. Additionally, since all transactions or at least parts of them are stored across different machines, and in most cases, the execution of transactions is triggered on all verifying units, this results in an increased overhead regarding memory and computational power. Additionally, public blockchain transactions are generally slower compared to traditional database solutions. These problems aggravate scalable blockchain-only implementations [25, 28].

2.5 Bluetooth Low Energy

Wireless connectivity is often considered the key functionality in IoT devices, but at the same time, it is a vital contributor to power consumption. Moreover, the plurality of available wireless standards generally poses severe challenges concerning the compatibility between collocated networks. Therefore, energy efficiency and interoperability are the significant challenges in connectivity for the IoT.

Bluetooth Low Energy (BLE) is a short-range wireless technology. The Bluetooth Special Interest Group (SIG) defines its specifications. Due to its power-saving design, its widespread use, and its robustness against obstacles, it is used in many IoT applications. The following information about BLE was derived and adapted from [38].

Attribute Protocol (ATT) and Generic Attributes Profile (GATT):
The ATT is a stateless protocol for discovering, reading, and modifying data on a peer device. Depending on the use case and the Bluetooth unit's offered features, a device can act as a client or server. Data is always stored on the server and can be accessed and modified by the client. When a client wants to read an attribute or write to it, it will be addressed by its handle. Subsequently, the server will respond with the attribute value or an acknowledgment. The GATT is built on top of the ATT and uses it as its transport protocol. The GATT defines a framework for transferring data between devices using concepts like services, characteristics, and descriptors:

- **Services.** Attributes on a GATT server are grouped into read-only services. A GATT service breaks up data into logical entities.
- **Characteristics.** Characteristics are data containers that are included in a service. They consist of at least the characteristic-declaration and the characteristic-value.
- **Descriptors.** Characteristics may include descriptors. A client may read a descriptor to get additional information on the characteristics and their values. Alternatively, a client may also write to a descriptor in order to configure a specific characteristic. Depending on its value field, notifications or indications can be switched on or off.

Generic Access Profile (GAP)
The GAP is the highest level of the BLE stack and provides a framework consisting of roles, modes, and procedures that allow BLE devices to discover each other, broadcast data, establish secure connections, and perform other operations according to the predefined standard. The GAP roles define the system's topology. A broadcaster only broadcasts data without requiring any confirmation or acknowledgment. It does not support connections to other devices. By contrast, the observer role listens for advertising packets from broadcasters without actually connecting to the device. Similar to BLE broadcaster and observer roles, a peripheral sends advertising packets that may be received by a central device. Additionally, the central role is designed to initiate and manage multiple connections, whereas the complementary peripheral role is used by devices that wait for others to connect to it. After establishing a connection, further data can be transferred.

2.6 Related European and National Projects

Parts of this thesis have been performed within different funded research projects described in the following paragraphs. Among these are the European Union's Horizon 2020 projects *Smart-Taylored L-category Electric Vehicle demonstration in hEterogeneous urban use cases (STEVE)* under grant agreement No. 769944 (see subsection 2.6.1) as well as *Secure Connected Trustable Things (SCOTT)* under agreement No. 737422 (see subsection 2.6.2). Additionally, other research aspects have been elaborated within the research program *Secured Trustworthy IoT Platform (STIP)*, funded by the Austrian Federal Ministry for Digital and Economic Affairs (BMDW) under the program COIN-Programmlinie "Netzwerke" under agreement No. 867914 (see subsection 2.6.3).

2.6.1 STEVE

The H2020 project STEVE[1] is composed of 21 partners – a network of cities, industry, and research facilities. The primary idea of STEVE is to design, implement, and evaluate a human-centric approach to electro-Mobility-as-a-Service (eMaaS). The goal is to provide low-cost and financially sustainable electric vehicle solutions and "gamified" services combined with a dedicated mobile, universal access solution to enhance users' awareness and engagement, as well as vehicle energy efficiency (TRL 5-8). The developed technologies are pilot-tested during extensive demonstration phases in the four STEVE lighthouse cities Turin and Venaria in Italy, Calvià in Spain, and Villach in Austria.

[1]http://www.steve-project.eu (last access: 13th May 2020)

2.6.2 SCOTT

SCOTT[2] is an H2020 project with 57 key partners from twelve countries. Its focus lies in providing efficient solutions for wireless, end-to-end secure, and privacy-preserving connectivity (TRL 6-7). SCOTT focuses on sensor/actuator networks and wireless communication in mobility, smart infrastructure, and health, thus addressing challenges regarding trustable automated vehicles and Industry 4.0. SCOTT is based on fifteen industrial use cases emphasizing cross-domain applications and heterogeneous environments.

2.6.3 STIP

The deployment of new applications and services is often hindered by technology barriers such as security problems regarding communication and data access, complex privacy management, or lack of interoperability due to various standards. The Austrian BMDW funded project STIP[3] addresses the development of an open technology framework for end-to-end secured IoT solutions. Five Austrian partners work on secure data exchange solutions from the chip level to the application level with product personalization features and secure key management. Furthermore, cloud-based secure data collection, data analysis, and selective decision support for secure communication with the application are addressed within this project.

[2]https://scottproject.eu (last access: 13th May 2020)
[3]http://stip.tech (last access: 13th May 2020)

3 Related Work

The following paragraphs mark the entrance to our related work chapter giving an overview of the different topics the thesis discusses. The ubiquitous usage of connected objects and their interoperability aspects, for example, are essential to establish seamless cooperation between different distributed systems within a smart city (see section 3.1). Complementary to this, secure access methodologies, including aspects such as authentication, authorization, and data privacy, play a crucial role in emerging technologies (see section 3.2).

3.1 Interoperable Connected Systems in Smart Cities

We are living in the era of interoperable ICT systems. According to [73], interoperability is the "ability of things to interact for a specific purpose, once their differences have been overcome". Another definition of interoperability states that only if cooperation between all systems is ensured and solutions are made to eliminate the discrepancy between all components can interoperability regarding efficient information exchange be achieved within the resulting system of systems [22]. The ubiquitous interaction of interoperable, connected systems – in short IoT [12] – is an integral part of the technological progression we are experiencing. In 2017, an increase of 31% of IoT devices was reported compared to the previous year [39]. This trend is expected to lead to around 75 billion devices by 2025 [100]. Due to the advancement of the IoT, the usage of various digital applications is growing tremendously [64, 69, 113]. In further consequence, the implementation of digitalization strategies can also be observed within cities, with the aim of enhancing the citizens' quality of life and optimizing their resources. This is important, given that in 2014 already 54% of people worldwide lived in cities. According to predictions, the urbanization level will reach 66% by 2050 [105]. Various areas of life are affected by this development: food processing, transportation, education, agriculture, healthcare, energy, or waste management, for example, gradually transforming cities into smart cities. According to Bakici et al. [15], a smart city is a high-tech intensive and advanced city that connects people, information systems, and city elements using new technologies to create a sustainable, greener city, competitive and innovative commerce, and an increased life quality. Other smart city definitions, also including platform-designs for a ubiquitous data exchange and enhanced collaboration between different services, exist [2, 24, 36, 94, 115].

Wireless Sensor Networks. Sensory devices, handhelds, and particularly Wireless Sensor Networks (WSNs) play an important role in the growth of the IoT and directly contribute to the Big Data age due to their data collection and forwarding capabilities. As their hardware has become cheaper, more powerful, and less power-hungry in the last few years, wireless sensor networks are widely deployed [19, 83, 114]. For example, Zanella et al. [116] implemented a distributed sensor platform, in which standard protocols and data formats allow interoperability among different systems. By contrast, paper [5] discussed a distributed cross-layer protocol for facilitating efficient retrieval of distributed sensory data in the context of a smart city via WSNs.

The proposed architecture of this thesis' platform was designed, taking into account the usage of embedded and mobile devices (e.g., edge nodes, gateways, smartphones) to gather, process, and forward data. In contrast to the work listed above, not only the data transfer on embedded layers is discussed, but also the data flow in conjunction with different server and cloud systems.

Service Distribution. Efficient data aggregation and exchange with sensory devices form the basis of many Service-Oriented Architecture (SOA) approaches. The SOA elaborated within paper [7] provides efficient distribution of internet services. Access nodes act as service publishers on which an unstructured peer-to-peer (P2P) overlay network is created to provide geo-localized lookup functions. The work proposes an implementation using a distributed hash table for efficiently forwarding services based on the geographical position of resources and users. Another paper [8] discusses a SOA composed of several layers that support mobile applications and services. On the one hand, the infrastructure layer contains all network and location-based service systems. On the other hand, the information layer acts as an information repository. Furthermore, the service layer offers location-based services to users on the stakeholder layer. Finally, the business layer communicates vertically with all layers and applies rules to participating applications and systems. Efforts to increase the interoperability of heterogeneous service-oriented systems have also led to the creation of lightweight web services that can be used to manage sensory data [32]. The elaborated framework allows users to connect to multiple data sources using different data adapters and query data from them. These sources may consist of existing sensors and IoT platforms, different databases, and data located on local machines or clouds. By contrast, Petrolo et al. [84] worked on increasing horizontal interoperability between different vertical IoT platforms. Their VITAL platform uses virtualized unified access interfaces with different connectors and drivers. It relies on the domain-independent W3C SSN ontology to harmonize the semantics of different IoT systems and provides the possibility to use service-queries across different platforms. Dedicated object handlers point to physical resources that can be selected, filtered, and provided to the user.

In contrast to the work discussed above, we not only use a lookup table for efficient service forwarding but also apply privacy-preserving transformations on services and users of different client-server systems. The SOA of this thesis is also composed of different layers for better code-reusability and separation of concerns. We designed a software overlay with dedicated service, data, and service provider layers to increase the interoperability of heterogeneous services. Easier software integration and access to services of different platforms are achieved via middleware and a RESTful interface. Additionally, a web-based interface provides service providers with a more comprehensive service distribution and management view.

Mobility Services. Most SOAs provide generic approaches to handle city data and services, while others are more specialized. The SMARTY platform described by [6] focuses on mobility-related services, such as route planning, realtime traffic monitoring, parking reservation, car- and bike-sharing, carpooling, etc. Physical sensors are deployed in the urban area (e.g., CO_2 sensors, temperature sensors) to provide SMARTY with environmental data. It also collects user-related data from (i) smartphone apps (e.g., position, service utilization) communicating to the SMARTY cloud and from (ii) social media posts and messages. By contrast, paper [74] proposes a mobility management platform for connected cities. It allows city governments to formulate mobility policies that should positively influence the travel behavior of people. The authors integrate their framework into existing applications that enable the utilization of city resources. Data on mobile devices is collected and evaluated to adapt route planning for users, for example.

The examples mentioned above provide concrete mobility-related implementations. The service handling concepts described in this thesis are embedded within one mobility app for accessing mobility-related data and services. On top, our platform provides an interface for securely uploading data collected from the environment. However, not only mobility services but also other services can be managed by our platform. Furthermore, customizable policies between service providers enable condition-dependent service-creation and related incentivization strategies.

Development Layers. Some SOAs also provide an additional development layer for defining new applications, such as the work described by Wu et al. [111]. Users can create queries to retrieve sensory data stored on both Structured Query Language (SQL) and NoSQL database clusters for scalability reasons. The data can be visualized, and users may develop applications on top of it. A task scheduler takes the resulting web-based services. Communication with other systems is established via a RESTful application programming interface (API). Another example is described by the authors of [21]. They worked on an Open Service Gateway Initiative (OSGi) based middleware platform that offers an API for building web-based applications. Communication between IoT devices is leveraged by creating and handling generic data structures for event channels (publish/subscribe mechanisms), states, content, and web-services.

In this thesis, we do not provide a fully programmable development layer; however, context-aware conditions can be defined on our cloud layer for introducing automated service handling based on transactions and collected datasets.

Big Data Processing. Other scientific SOA publications focus on the influence and impact of Big Data handling strategies. Authors of [33] worked on a Big Data based platform. It embeds different data processing modules for data collected in cities by smart devices. IoT-agents connect to the IoT middleware and serve as a gateway to data-collecting devices. IoT-brokers act in between IoT agents, the Big Data repository, and the CityModel Server. The Big Data processing module utilizes Apache Spark to process the Big Data repository (e.g., data aggregation or data mining). A dedicated API allows applications to perform queries and subscribe to data streams. The platform of Vilajosana et al. [107], for instance, focuses on data management and service hosting. Big Data techniques are applied to collect data streams and analyze data. Predictions and inferences are derived from the data. The platform provides an API that enables third-party applications to access the data stored within the framework. The paper [97] includes a presentation of a comprehensive platform for a smart city. It executes tasks related to data collection, storage, analysis, system control, and user interaction. The platform is a cooperative system where different communities (administrators, experts, urban service providers, city managers) can share data and information. Lee and Rho [61] worked on a spring-framework-based three-layered architecture towards Big Data processing. Next to a presentation layer that visualizes data and controls devices of the device layer, there is also a processing layer for analyzing, aggregating, and filtering data. It consists of middleware that provides a data acquisition interface for sensors and processing devices. The evaluated data is collected for service management, including autonomic service discovery and deployment.

Also our platform provides software components for data collection, processing, and interpretation, distributed across different embedded devices and cloud-layer units. However, our primary focus was not on data mining and handling of diverse datasets. Compared to the publications described above that put little effort into security, we investigated end-to-end security mechanisms. We conducted a case study to introduce secure yet efficient data collection and forwarding at early development phases. The outcome of our analysis phase was applied to devices and communication protocols on different IoT-layers.

Semantic Webservices. The idea of encapsulating a system's functionality within an appropriate interface and advertising it as services is not new [30]. Semantic web service paradigms have been trying to enhance automated service discovery, service access, service combination, and web services management for years. Both industry and academia are working on mechanisms to provide machine-understandable representations of services and their behavior [18, 31]. The smart city platform elaborated by Girtelschmid et al. [48] uses semantic technologies for enhanced flexibility in system configuration and adaptation. The performance loss associated with reasoning tools is mitigated by also adding Big Data processing methods. The paper [72] describes how service-based middleware can provide techniques to implement and deploy smart city services. The service description information is based on Web Service Description Language (WSDL). An interface between cloud services (mining, analytics, optimization, simulation) and fog services (storage, caching, streaming, processing, configuration, monitoring, measurement) is proposed. While a global broker service maintains all services in the environment, local broker services manage information about the fog's available services. Invocation services include the messaging system between different services, and location-based services contain all connected devices' positions. Finally, security services mainly include information regarding authentication and authorization.

We work on a service cloud layer for online and offline redeemable services across client-server systems of different domains. While the papers described above elaborate semantic web-service handling techniques, we focus on more strict service syntax rules for fast data handling on the server-side and straightforward interpretation for the system's users. Nevertheless, our service-related generic data structures are extended with a dynamically definable subpart that can be used for applying semantic methodologies.

Interoperable Clouds. Interoperable cloud networking is also becoming increasingly important. Different approaches already exist that provide APIs for managing virtual objects or combining existing collaboration tools [27, 99]. Complementary to these, other approaches focus on combining heterogeneous cloud systems for a more standardized sharing between different datasets across geographically distributed resources. Zou et al. [118] describe how a federated marketplace can provide participants with access to different resources regarding data, memory, and computational resources. The design and implementation are based on federated Comet spaces. These are distributed shared memory that all users and providers can access and observe for enabling information sharing. A management space is used to handle different resources and exchange any operational messages to discover resources, announcing changes, or routing users' requests to the appropriate site(s). Besides, execution spaces are created on-demand to satisfy the computing needs of the users. Authors of [1] work on a cloud abstraction and service marketplace platform backed by several heterogeneous service clouds. The service consumer module provides a conversational interface to consumers. It queries the service knowledge base for service solutions, while the service provider module serves to simplify the on-boarding of new service providers and the update of marketplace services. Finally, the service marketplace module manages various kinds of metadata of the service knowledge base obtained from any service characterization. Among other things, it comprises marketplace support services, including service composition tools (service registry, graphical tooling, service descriptor exporter).

In our case, an additional service-overlay acting as a ubiquitous service-entitlement marketplace is introduced for issuing, accessing, and enabling the exchange of datasets and services. Compared to other approaches listed above, an advantage of our system is that the real, human-readable user data remains at the base system, enabling an increased anonymity level. Furthermore, since shared data is prone to malicious changes, we introduced an additional layer relying on distributed ledger technology for tamper-resistant transaction handling.

Hybrid Blockchain Approaches. Blockchains can be used to increase transactional security. Like the projects discussed in this paragraph, many of today's developed prototypes describe a hybrid approach between a traditional cloud environment and a blockchain part. Authors of [119] combine blockchain and off-blockchain storage to construct a personal data management platform. The collected data is encrypted and sent to an off-blockchain key-value database. Only a hash value is retained, pointing to the data on the blockchain. The data can be queried by using the key associated with it. Every time a user subscribes to a service, a new transaction specifies the access permissions, and another contains the hash of the data. Another blockchain-based implementation, discussed by [77], supports data accountability and provenance tracking. A data subject (person) authorizes a data controller (organization) to access personal data. Consequently, the data can be forwarded to a data processor (organization). The subject subscribes to the data controller using smart contracts that manage the data usage policy in question, store data-hashes, and transfer it to the controller. For each new contract, the subject uses a new blockchain address to prevent the linkability of the contracts created with each controller. Furthermore, the subject must maintain a list of all addresses and the respective nonce established with each controller or processor. Further work focuses on managing electronic medical records distributed across different data providers [14]. In this case, the hybrid blockchain approach gives patients an immutable log and access to their medical information across providers and treatment sites. Hashed data pointers are stored on the blockchain to guarantee that the data cannot be altered. Additional query strings are intended to be executed on the corresponding data provider's server to retrieve the data again. The authors of paper [63] describe a procedure for secure drone communication, focusing on data assurance and resilience. Again, a distributed blockchain storage is used together with a traditional cloud-based system. The system forwards the hashed data records collected from drones to the blockchain network and generates a blockchain receipt for each record stored in the cloud.

In this thesis, the elaborated cloud overlay is extended with an additional permissioned blockchain/smart contracts part to profit from (i) increased execution speed compared to a blockchain-only approach and (ii) enhanced protection against fraudulent changes. The interface between the central and the decentralized blockchain storage is established via smart contracts.

3.2 State of the Art Access Methodologies

An increased level of connectivity may not only lead to advantages but also to issues in the areas of security, privacy, and ethics [11]. As stated in [41], especially security is one of the main challenges engineers have to face when implementing successful IoT solutions. Every poorly secured device connected online can serve as a potential entry point for cyber-attacks, compromising the system and exposing data [93]. Slack security methodologies can also lead to unsafe or even lethal scenarios (e.g., automotive hacks described in [71, 106]). Therefore, paving the way for robust and trustable ICT becomes all the more important.

Security by Design. IoT layers that consider different levels of secure data processing are being researched, like for example: data collection [42] by sensor nodes, data forwarding via gateways [102], and data storage and analysis of cloud services [23]. Other work focuses on analyzing specific security aspects, e.g., hardware devices and security threats, as discussed in [81] and [92], respectively. Effort has also been made to describe the whole end-to-end data flow of such connected systems [49]. Design Space Exploration (DSE) tools can be applied to enhance a system's quality in the early development phases. They are used to help designers consider system components related to their power consumption, performance, and security requirements [59]. These tools can generally be classified through their applicability in either individual component integration [54, 59], or overall system design [101, 117]. Analyses at a later stage of system development are also possible. Authors of [66] performed a security and privacy study. Each design platform can be divided into functional units that are then evaluated on predefined security cases. The analysis is done on real-world applications and security attacks.

The system that we present handles the security levels mentioned above as a one-entity framework, enabling secure interfaces implicated on different IoT-levels. A novel DSE tool with specified security constraints and formulated attack scenarios presented as Bayesian attack graphs has been used in the course of a case study to amplify the design phase of our SOA's data-collection part.

Access Control. Security and privacy are important metrics for many smart city applications and platforms [16, 53, 55]. Especially authentication and access control mechanisms are crucial for enabling trusted communication for the IoT [3, 4, 76]. Different scientific contributions utilize digital tokens for authentication and authorization procedures. In the research paper [37], for example, the authors propose a token-based authentication procedure for IoT devices. The tokens are used to access specific resources for a predefined time. The protocol relies on lightweight operations such as XOR and hash functions, reducing the computational effort. Other approaches also use hash and XOR techniques for establishing authenticated communication, especially for computational-constrained devices [17, 108]. The authors of the paper [20] also discuss a token-based authentication scheme. In their case, it is applied together with the MQTT protocol to remain compatible with constrained devices. First, publishers/subscriber send their username/password credentials to a token authentication server. A JSON Web Token is issued, forwarded to the publisher, and stored locally. It is used for further authentication instead of periodically triggering a re-authentication with the original credentials and requiring the MQTT broker to keep track of all sessions. The Stack4Things architecture from [26] provides authentication, authorization, and delegation mechanisms. Authentication is enforced via the identify service Keystone. It validates a set of credentials (username, password, API-key, for example) supplied by a client. In case the credentials are validated successfully, an authentication token is issued. The client can use this token for further API calls. The issued tokens are merged with roles assigned to a client. Agents validate them to check if the client is authorized to execute a specific operation on an IoT node. Each API call has a corresponding line in a policy JSON file that determines which level of access applies. Ouaddah et al. [79] proposed an access control framework for IoT environments. Their model uses RESTful web services to enforce different security policies. Organizations may determine which published resources they want to protect. Consequently, these resources are assigned to a URI. Other organizations can contact the resource holder and define authorization rules via an access control policy.

This thesis covers multiple authentication and authorization aspects. Compared to the approaches mentioned above, we worked on a more comprehensive approach, providing various token types for condition- and use-case-dependent online and offline authentication and authorization methodologies. Our authorization handling does not only provide access to resources available online but also offline. For each resource, dedicated entitlement objects define granular authorization rules.

OAuth2.0 Authorization Tokens. Regarding distributed authorization, one way to enable it across different systems and applications is with the OAuth 2.0[1] protocol. Companies like Google or Facebook utilize it to handle authorization across different web services, for instance. However, it is also part of scientific research papers. Examples include distributed privacy-preserving authorization handling in e-Health applications where different pseudonyms are introduced for each user [103] as well as the use of OAuth 2.0 to increase the security for API access in order to provide more robust protection against fraudulent users [67]. OAuth 2.0 can also be applied in the context of resource-constrained IoT devices. The authors of the paper [96] describe a gateway-approach that collects information provided by smart devices. It controls the access requests to the datasets via OAuth 2.0 authorization. Another example is discussed by paper [34]. It describes an IoT setup and proposes a framework for enabling IoT devices to utilize an external OAuth 2.0 authorization framework based on HTTP/CoAP without implementing the protocol handling on each device. Additionally, paper [47] demonstrates how IoT protocols such as

[1]https://oauth.net/2/ (last access: 13th August 2020)

MQTT can be (i) integrated into existing Web API management models and (ii) enhanced with a software API gateway for access control, monitoring, etc. Additionally, an OAuth 2.0 and OpenID Connect part are included for client registration and token handling. Finally, a Mosquitto MQTT broker provides an open-source messaging broker.

We extended our token management system that is addressable via a RESTful interface with a dedicated OAuth 2.0 support layer. This enables client devices to receive data from virtual entities while only relying on one central authentication and authorization center. While the approaches described above primarily focus on authorization aspects, we also took different authentication mechanisms into account.

Local Authentication. Efforts have been made to study local authentication mechanisms. The work from [95] provides an approach for identity authentication in cities. A single access key is created that combines identity, payment, access, and ticket related information. The access key is stored locally on a specific computing device such as a smart card or smartphone and is used to authenticate the user via interaction with a backend access system. Dmitrienko et al. [43] discussed the topic of secure offline access using a mobile car-sharing application as an example. The user needs to utilize two authenticators to pass the authentication step of the car's lock. The first authentication factor is a user key created during the user's registration, while the second factor is an access token issued and downloaded to the client device during the online booking procedure. On the client-side, the user key is hosted by the secure-element-provider. The token itself is stored within the host environment that, in this case, is a smartphone app.

We provide local authentication with authentication-tokens issued and cryptographically signed by a trusted central party. In combination with service-based-tokens, they allow not only offline authentication but also enforce local authorization policies.

Certificate-based Authentication. There is also significant research interest in certificate-based authentication mechanisms. However, since X.509 certificates may be hard to apply on resource-constrained IoT devices, Park et al. [80] propose a concept for simplifying certificate structure to make them more suitable for IoT devices. Panwar et al. [60] worked on a security mechanism for the IoT using digital certificates with datagram transport layer security (DTLS). Authentication is achieved via digital certificates that are issued by a certificate authority. The client/server authentication procedure includes mutual exchange and verification of certificates. Piro et al. [85] elaborated a lightweight service access strategy between service-publisher and service-user devices. A device may advertise an interest-packet, including information about the content that should be shared. Any interested node may respond with the device's public key/certificate (checked by authority), the cryptography algorithm intended to be used for additional encryption, and other security-related information. Authenticity is achieved via cryptographic signatures. The publisher signs the hashed data packages. Finally, a node verifies the authenticity of the data with the publisher's public key.

All anonymized tokens created by our platform are based on a custom public key infrastructure (PKI) that forms the security basis of the platform's access methods. The tokens are issued and signed by a trusted central authority and validated before being redeemed against the resource in question.

Distributed Authentication. Further research efforts have been invested into distributed authentication mechanisms applying variants of the Kerberos protocol [70]. Periera et al. [82] propose an access control framework on service level for power-constrained devices. It merges the idea of Kerberos and RADIUS access control systems. There are two steps: authentication and access control. The users are first authenticated based on their credentials like a shared key, password, or another validator. On successful authentication, the CoAP-NAS is informed of the users and their permissions, the timestamp of a ticket, etc. The CoAP access server sends a ticket to the user for future requests. In the access control step, the server will only respond with the correct message if the request message has a valid ticket; otherwise, it will generate an error response. Complementary to this, authors of [46] designed a grid authentication model based on Kerberos and Hierarchical ID-based Cryptography (HIBC). The overall model is divided into various trust domains with different parameters. The first level trust domain applies the Kerberos mechanism, while the second level trust domain applies the HIBC. It enables a private-key generator (PKG) to distribute their workload to a lower-level PKG. Therefore, user authentication and key delivery can happen locally. The authentication method between different domains is Kerberos-based since the Kerberos ticket can deliver the second level trust domain parameters.

A primary advantage of such distributed registration/login mechanisms is that authentication can be granted without exposing critical user and device data. Therefore, we rely on a Kerberos-based registration procedure for connecting client devices to our central service-overlay over an already trusted server unit. By doing so, we keep user data transfer to a minimum and allow privacy-preserving registration of clients.

Data Privacy. Due to increasingly stringent data protection rules, like the European General Data Protection Regulation (GDPR) [45] or the Data Security Circulation Convention from China Academy of Information and Communications Technology (CAICT) in 2016, data privacy has become ever more important. Data is constantly collected. Organizations analyze and optimize it or create new services based on it. However, people often have little control or knowledge of what data is stored or shared. According to the overview given by [29], different anonymization procedures were developed, such as ℓ-diversity [68], t-closeness [62], δ-presence [78], ε-differential privacy [44], etc. However, too strong anonymization may make the original data useless for many applications. K-anonymity techniques are often applied to reduce the risk of identification while retaining enough data to work with [65, 98, 109, 112]. This is achieved using generalization (attributes are generalized to reduce specification) and suppression (attributes are completely removed) methods. The elaborated model from [29] includes an effective k-anonymity algorithm combined with a k-means clustering algorithm to increase anonymized data diversity. The k-means algorithm clusters a group of data into a predefined k value. Initial cluster centers are selected randomly. The process keeps reassigning the data objects in the dataset to the cluster center based on the distance between the data object and cluster centers until a predefined condition is met. Complementary to these anonymization techniques, researchers work on different approaches to assess and manage data privacy within connected systems [104]. Paper [58], for example, analyzes the personal information flow through telematics systems inside cars. It distinguishes between an embedded approach where cars connect directly to the internet and an integrated approach relying on mobile devices for accessing different services. The approach from Apolinarski et al. [9] provides context-aware security and a privacy module for enabling users of mobile applications to manage their privacy settings. Acquisition and sharing of sensitive data (e.g., user location) can be avoided or limited. Furthermore, collected data may be obfuscated for limiting data precision according to user-defined privacy policies. The policies interact with a semantic data management unit deployed on mobile phones as part of a smartphone app. The initial privacy policy can be derived from context sharing settings in social networks, for example, and adapted afterward. Regarding data exchange, encryption protocols can be integrated into the communication process as BASE plugins. A shared secret is exchanged for symmetric encryption. PIKE is used to exchange shared secret keys using Facebook, or Google Calendar [10]. Finally, the shared data is made available to a data discovery registry. While the techniques mentioned above focus on anonymizing data, other approaches try not to store any privacy-critical data at all. The system described by Mylonas et al. [75] enables developers to write Android-code and deploy it to Android devices, alongside the distributed SmartSantander platform. They assign a unique identifier to smartphones and only store the device model and Android version but do not collect other privacy-critical user data for the registration.

Summarized, also the federated data and service management approach discussed in this thesis avoids the storage of confidential data in the first place. The transfer of user data is bypassed due to a distributed registration mechanism. Furthermore, a dedicated cloud-based abstraction layer is in charge of creating new anonymized service objects and efficiently mapping them to the real data that remains on the connected client-server systems.

4 The Distributed Platform's Design

This chapter gives an overview of the technologies that resulted in this thesis' platform. First, different requirements are defined in sections 4.1 and 4.2. Consequently, starting from the basic system model of section 2.2 an extended system model is explained in section 4.3. Its functionality is elaborated in more detail in the subsequent sections 4.4, 4.5, and 4.6.

4.1 Functional Requirements

Taking into account the problem description of section 1.3, functional requirements (FRs) were derived:

FR1 **Cross-domain service management.** First, a SOA shall be elaborated for managing digitally and physically available resources and their corresponding service-entitlements. Second, while many custom-tailored solutions for specific application areas exist, there is no common strategy or standard interface for enabling digital service-entitlement exchange between different client-server systems. This exchange shall be enabled between service providers, service providers and clients, and clients by employing a virtual service marketplace layer.

FR2 **Data management.** The SOA shall provide mechanisms for client and server applications of different businesses to upload and edit service and user-related datasets. Also, service-related permission settings shall be editable.

FR3 **Cross-domain rewarding system.** Users tend to be creatures of habit. Once they are used to services and are familiar with obtaining the corresponding resources, they often no longer look around for new services without a specific reason or an advantage. It should be possible for service providers to define different strategies for motivating users to access new services based on their service-usage behavior and preferences.

FR4 **Ubiquitous resource access and control**. Different services and their resources shall become accessible by client users. Regarding the redemption procedure of service-entitlements, not only the access to digitally but also to physically available resources should be handled via application and use-case dependent authentication and authorization schemes.

FR5 **External Access.** External applications should be able to access the service-oriented platform via dedicated software libraries in combination with an online reachable API. Furthermore, a web-based registration and control mechanism should be provided to avoid time and cost consuming integration procedures.

4.2 Non-Functional Requirements

Next, non-functional requirements (NRs) of this thesis' platform are derived. They are discussed in the following paragraphs:

NR1 **Interoperability.** Different platforms and systems must operate with each other. The interexchange of services between independent service providers shall be improved by providing a common federated service overlay that can be accessed via an online reachable interface. It shall consist of an abstraction layer for abstracting and equating different datasets and a service and data exchange layer.

NR2 **Context Awareness.** Context-aware middleware for clients shall be elaborated. In this sense, according to the environmental setup and the actual service, an appropriate redemption method should be applied, choosing from local and online applicable wireless procedures. Additionally, it shall be possible to define condition/context-dependent rules for the automatic creation of service-entitlements.

NR3 **Privacy.** The platform collects and manipulates different datasets. Due to increasingly stringent data protection rules and increased privacy awareness amongst users, privacy-preserving data exchange is crucial for client-server systems that intend to exchange user data with each other. User data privacy shall be protected while still providing means to collect enough non-anonymized information to perform the system's tasks. Furthermore, the clients' privacy awareness shall be increased by providing an interface to overview the current privacy policy. Complementary to this, privacy by design shall be enabled by supplying the technical means to adapt the clients' data aggregation flow according to the users' preferences.

NR4 **Security.** Interfaces to other systems may lead to security risks such as disclosure of confidential data, fraudulent data manipulation, repudiation scenarios, or system failures, especially when secure authentication and authorization are not provided. Dedicated security mechanisms are required to enable trusted communication between different server and client-server applications. On the one hand, end-to-end secured communication shall be provided for *confidential* data. On the other hand, also *data integrity* has to be taken into account. There should be the possibility to protect obtained and redeemed services and transaction-related data against changes. In regards to *data availability*, the risk of data loss shall also be avoided.

NR5 **Scalability.** The overall SOA shall be able to scale with a growing number of users, services, and transactions.

NR6 **Configurability.** Service-entitlements shall be adaptable and re-configurable in order to be compatible with different authentication and authorization schemes.

4.3 System Model Overview

A basic system model composed of client, kiosk, and server devices was described in section 2.2. It was extended with new concepts and ideas in this thesis, resulting in the system depicted by Fig. 4.1. An overview of the most important communication participants is given in the following, while each part's design aspects are discussed on the subsequent pages.

- **Client.** The client is a mobile device running an application and representing a user. It offers the possibility to obtain digital service-entitlements. They can be redeemed in different ways by communicating with kiosk units for accessing locally or online available resources. Depending on the client's requirements and use case, it may run different applications, e.g., a mobile parking app for booking parking tickets, a restaurant app for ordering food, or a web app for obtaining and visualizing sensory data.
- **Embedded Kiosk.** Embedded kiosks are physically available devices within reach of clients. Service entitlements can be redeemed by communicating to them and the Connected Services Engine (CSE). In case of a successful authentication and authorization procedure, client users are granted permission to access physically available resources. Depending on the underlying application, an embedded kiosk can be an embedded device (e.g., part of a parking control unit for a gated parking use case) or a mobile device (e.g., smartphone used by an officer for an attended parking use case).
- **Virtual Kiosk.** If the unit responsible for triggering the redemption process is part of a server, it is called a virtual kiosk. Clients can redeem their service entitlements directly online (including secure authentication and authorization) without local interaction with other devices, e.g., to get access to data stored online.
- **App Server.** An application/business server communicates with client and kiosk devices and provides them with application-dependent functionality (e.g., login/registration, visualization of user and service data). User and product relevant data is stored on a dedicated database. If an app server wants to communicate to other app servers for achieving cross-domain service exchange, interaction with the CSE is required. In this case, the CSE enters a one-to-many relationship with connected app servers.

- **Connected Services Engine (CSE).** The heart of the CSE is a cloud part, also referred to as the core server or core layer. It communicates to clients, kiosks, and app servers. Furthermore, the CSE provides dedicated middleware that enables local communication between clients and kiosks and online communication to the core server via a RESTful interface. The CSE manages cross-domain service-entitlements. It is a *token issuer* and *token handler.* It provides mechanisms for issuing different tokens for online and local authentication and authorization. Furthermore, it is the highest authority that decides if a redemption procedure is valid or not (*online redemption handler*). To exchange services between client-server systems of different domains, a service *mapping* functionality is part of the CSE that can be utilized together with service *offer and agreement* procedures. Moreover, the CSE provides an inter-system *transaction watchdog* that keeps track of past transactions. An additional *blockchain* and *smart contracts* layer enforces transaction security. Last but not least, a *front-end management tool* facilitates the docking process of other systems and provides further data-management capabilities.

- **Blockchain** The blockchain emerged as a tamper-resistant tool with excellent integrity protection and traceability features. A smart contracts based interface enables the interaction between the blockchain part and the other systems of the platform.

- **Sensor Node.** An embedded device that collects sensory data securely in constant intervals. Other tasks include the processing of this data.

- **Gateway.** A trusted communication participant that applies different local and online communication techniques for exchanging data between sensor nodes and virtual kiosks.

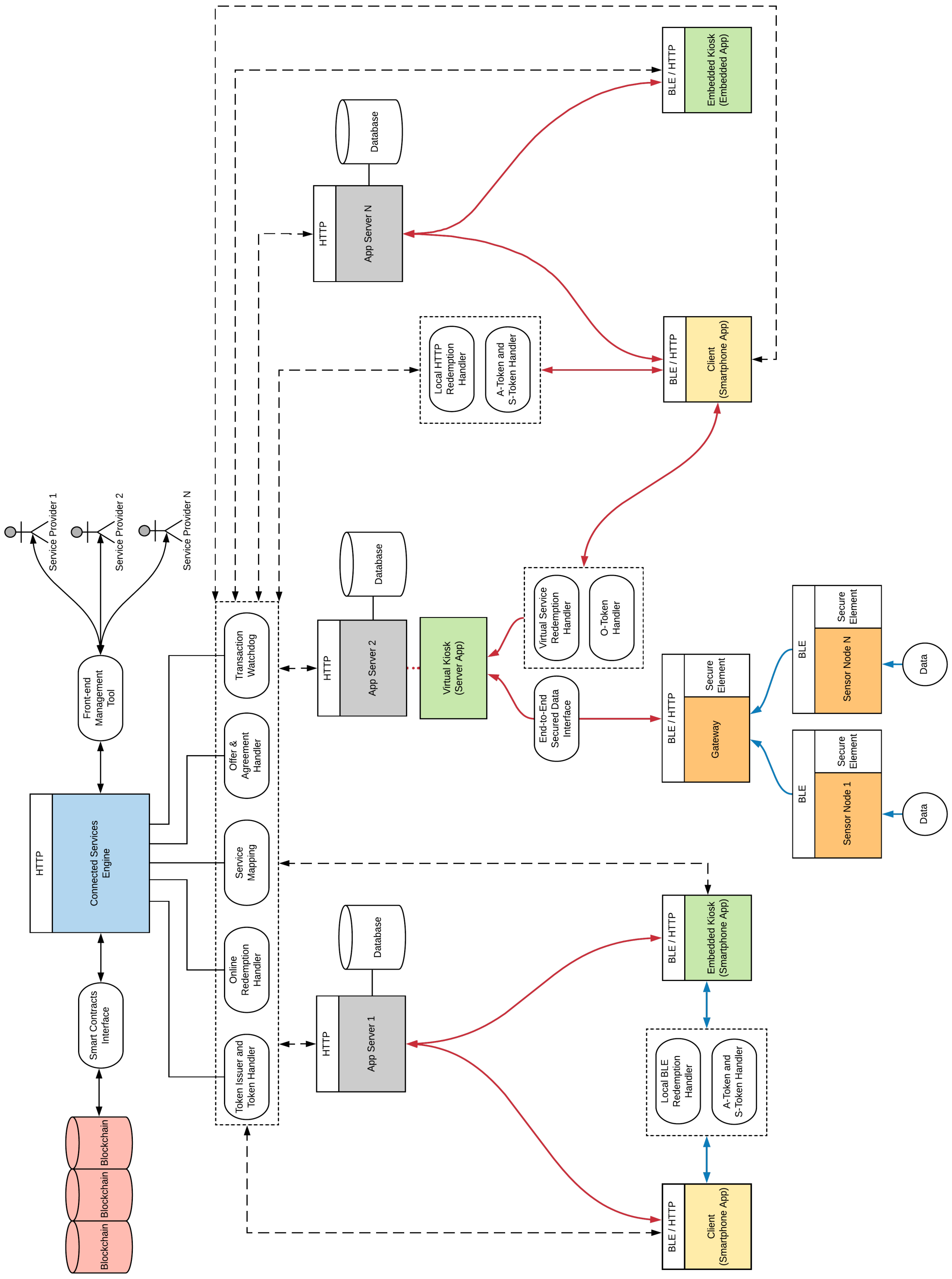

Service Provider 1
Service Provider 2
Service Provider N
Front-end Management Tool
HTTP
Connected Services Engine
Smart Contracts Interface
Blockchain
Blockchain
Blockchain
Transaction Watchdog
Offer & Agreement Handler
Service Mapping
Online Redemption Handler
Token Issuer and Token Handler
HTTP
App Server N
Database
HTTP
App Server 2
Database
HTTP
App Server 1
Database
Virtual Kiosk (Server App)
Local HTTP Redemption Handler
A-Token and S-Token Handler
Virtual Service Redemption Handler
O-Token Handler
End-to-End Secured Data Interface
BLE / HTTP
Embedded Kiosk (Embedded App)
BLE / HTTP
Client (Smartphone App)
BLE / HTTP
Secure Element
Gateway
BLE
Secure Element
Sensor Node N
Data
BLE
Secure Element
Sensor Node 1
Data
BLE / HTTP
Embedded Kiosk (Smartphone App)
Local BLE Redemption Handler
A-Token and S-Token Handler
BLE / HTTP
Client (Smartphone App)

Figure 4.1: Extended system model overview

4.4 Service Co-Modality

This section presents a generic service handling and exchange concept for client-server systems. The general idea is to enable service providers and their systems to publish their services on a common layer and access other cross-domain services through it. Besides, their clients should profit from various service-entitlements and access the underlying digitally and locally available resources via context-aware service redemption methods further explained in section 4.5. Fig. 4.2 provides a high-level overview of this.

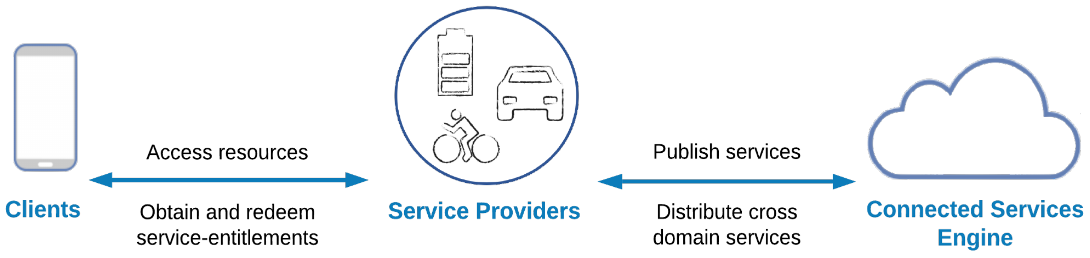

Access resources
Publish services
Clients
Obtain and redeem service-entitlements
Service Providers
Distribute cross domain services
Connected Services Engine

Figure 4.2: Clients access resources by redeeming service-entitlements from service providers that publish and distribute their services via the Connected Services Engine

The resulting service-oriented architecture is composed of two major sublayers.

Application/Business Layer:
An application or business layer is a self-contained system composed of different subsystems that are all part of the same business. By default, it has no direct interfaces to systems outside of its domain or application area. In our scenario, all components of a typical client-server system are grouped into a business layer. In the simplest case, a business layer is composed of a client device and an application server. Based on our basic model of section 2.2 also an optional kiosk unit for redeeming services may be part of this layer. A business layer's functionality is based on the respective application area of the service provider. For establishing a connection to other business layers, a common trusted layer, the core layer, is introduced.

Core Layer:
The core layer or core cloud is the server part of the CSE. It seamlessly interconnects services of different business layers with each other. From an architectural perspective, it is attached on top of existing business layers, abstracts user and service-related data, and handles different resource access methods. Communication with clients, kiosks, and application servers of a business layer is established via a RESTful interface. The core layer is mainly composed of two sublayers, the tokenization layer (4.4.1) and the federation layer (4.4.2). Additionally, see 4.4.3 for an associated distributed ledger part and 4.4.4 to get an overview of possible gamification principles provided by the core layer.

4.4.1 Tokenization Layer

Fig. 4.3 gives an overview of the core layer. The left part depicts the tokenization layer. This layer represents the entry point for devices on the business layer. It not only provides access control functionality but also takes care of abstracting user and service data.

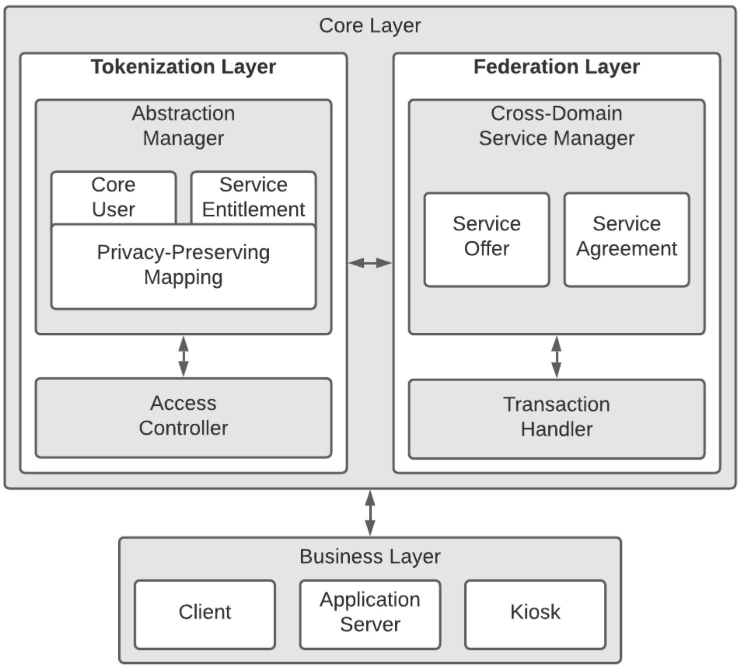


Figure 4.3: The core layer and its subparts: tokenization layer with access control and data abstraction and federation layer with transaction and cross-domain service handling

4.4.1.1 Access control

Direct communication between clients and the core layer, as well as application servers and the core layer, can be established via API-key authenticated RESTful HyperText Transfer Protocol (HTTP) requests. The API-keys are included in the HTTP header (Authorization field) and encoded in Base64. The core server issues them after a distributed registration mechanism is completed.

Integration between the application server and core layer:
A service provider may register at the core layer to create an authenticated connection between it and his business layer. Service provider specific information (e.g., email, password, company name, address, etc.) and a client-app specific identifier (e.g., Android app-ID) need to be submitted as well as a REST-endpoint that can be used by the core-layer to

contact the application server. Subsequently, an account is created, and the following datasets are generated automatically on the core layer:

- *application-ID (a-ID).* Identifies services of a specific service provider. Each service provider has a unique a-ID.
- *API-key.* Key used for authenticating the application server when interacting with the core layer over a RESTful HTTP interface.
- *Wallet.* In order to increase the level of trust of transactional data, an optional blockchain layer is introduced. In this case, each business server is involved in the consensus procedure of service-based transactions. Please refer to subsection 4.4.3 for more details on the blockchain approach.

Integration between client and core layer:
The integration of the application client into the overall platform foresees several steps. It is assumed that (i) the client and application server can communicate over an authenticated channel and (ii) the service provider has already conducted the core layer's registration procedure described above and is, therefore, able to access the core layer's REST interface.

Eventually, a distributed registration procedure can be applied between the client, application server, and core server, according to Fig. 4.4. As soon as the client and application server are authenticated, the client device may generate an Elliptic Curve Cryptography (ECC) keypair and trigger the distributed registration procedure by sending its public key to the application server. Subsequently, the application server requests a ticket generation on the core layer by submitting the client's public key. Next, the core server issues a one-time registration ticket bound to the client device via its previously sent public key. The response is received by the business server and forwarded to the device. Consequently, the ticket can be redeemed by the client at the core cloud. An anonymized core user entity is created identifiable via a dedicated user-ID (u-ID). Eventually, the client receives a device and user-bound API Key to authenticate all further REST-based calls to the core server. Furthermore, an Authentication-Token (A-Token) is created for local authentication to kiosk devices. More information on the A-Token and other token types can be found in subsection 4.5.1. Finally, the client device may directly communicate to the core layer and authenticate itself by submitting the previously received API-key.

4.4.1.2 Data Abstraction

The tokenization layer enforces data abstraction. Dedicated core layer objects are created, and a privacy-preserving data mapping between the business layer and core layer is used to address the right user, service, and application on the business layer. The distributed registration mechanism pointed out in Fig. 4.4 triggers the creation of an anonymous

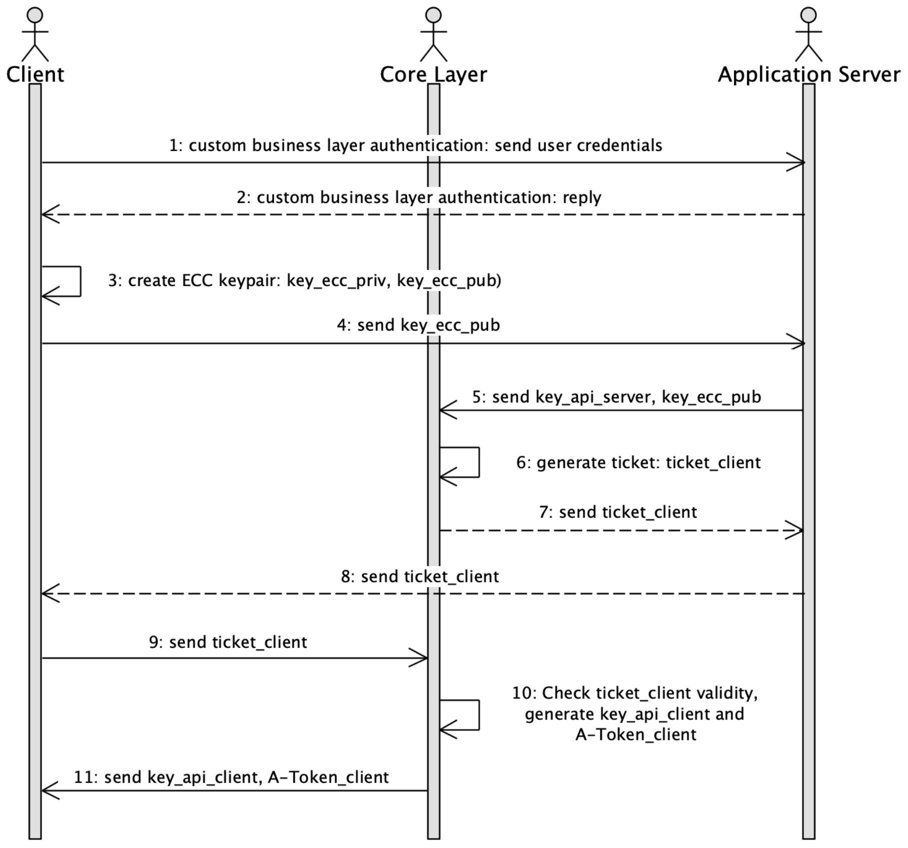

Client
Core Layer
Application Server
1: custom business layer authentication: send user credentials
2: custom business layer authentication: reply
3: create ECC keypair: key_ecc_priv, key_ecc_pub)
4: send key_ecc_pub
5: send key_api_server, key_ecc_pub
6: generate ticket: ticket_client
7: send ticket_client
8: send ticket_client
9: send ticket_client
10: Check ticket_client validity, generate key_api_client and A-Token_client
11: send key_api_client, A-Token_client

Figure 4.4: Distributed registration procedure between client, application server, and core server

core user entity. Any user-related data, such as names or emails, remain at the business layer. By contrast, services on the core layer are stored as **service-entitlements**, digitally representing an item, product, license, or another resource. Service-entitlements are linked to the business layer's application identifier (a-ID), the identifier of the original service on the business layer called service-ID (s-ID), and the core user entity (over a u-ID) who is entitled to utilize or consume the service.

A **privacy-preserving mapping** is deployed for linking service-entitlements to the intended users on the business layer. Fig 4.5 depicts a mapping table stored on the core server. It contains the anonymized user identifier of the application layer (e.g., email or username). It is hashed together with a cryptographic salt, mitigating password attacks like rainbow tables. Additionally, the core user and a unique app identifier are stored in this table for addressing the right client app. There are two ways how entitlements are issued for clients. On the one hand, when a client obtains a service on the business layer, this triggers the entitlement generation on the core layer. On the other hand, a service

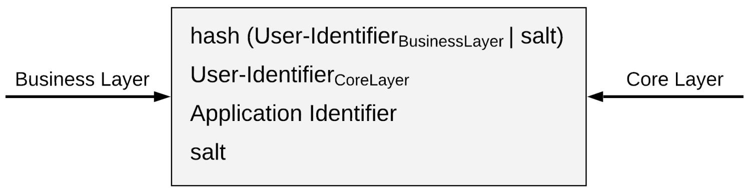

hash (User-Identifier$_{BusinessLayer}$ | salt)
User-Identifier$_{CoreLayer}$
Application Identifier
salt
Business Layer
Core Layer

Figure 4.5: Privacy preserving mapping between business layer and core layer [90]

provider may trigger the creation of entitlements via the federation layer and link it to one of his clients (see 4.4.2). After a service-entitlement has been created, dedicated tokens can be derived and passed to the client device. These tokens act as digital tickets for authorizing clients to access available resources. This topic is elaborated in more detail in section 4.5.

The exchange or sharing process of a service-entitlement from a business-layer$_{\text{origin}}$ to a business-layer$_{\text{destination}}$ can be enabled if its u-ID field is edited. By adding a new identifier or simply changing it, the service-entitlement is linked to another core user. Thus the service also becomes accessible to another user on the business-layer$_{\text{destination}}$ due to the mapping described before. While the core server provides the exchange/sharing interface, the procedure itself must be triggered by a device on the business layer. Which service-entitlements are allowed to be forwarded from business-layer$_{\text{origin}}$ to business-layer$_{\text{destination}}$ has to be specified by the corresponding business layers.

4.4.2 Federation Layer

The federation layer is the second sector of the core layer (see the right part of Fig. 4.3). It can be considered a virtual cross-domain service marketplace for distributing service-entitlements to client and kiosk devices and creating conditions on how these services are handled. The federation layer is in constant interaction with the tokenization layer. As soon as service providers have completed the registration process described in 4.4.1.1, they are authorized to publish federated services.

The creation of federated services and, in further consequence, the possibility to issue corresponding service-entitlements for clients can be enabled via direct interaction of the service provider with the web-front-end of the core layer. Alternatively, the application server's API-key can be used to trigger a service-synchronization procedure between the application server and the federation layer. During this process, the primary service identifier (s-ID) for mapping each service between business and core layer, as well as descriptive information (e.g., name, type, price) in the form of key/value pairs for higher flexibility and platform independence, are submitted.

All service-related transactions involving the core layer are logged via a **transaction registry**. Service transactions differ in their type, distinguishing between issuing, canceling, forwarding, and redeeming service-entitlements. Each transaction consists of a timestamp and refers to the corresponding service-entitlement that contains information about the application (a-ID), the core users on the client and the kiosk side (u-ID), and the service (s-ID) in question.

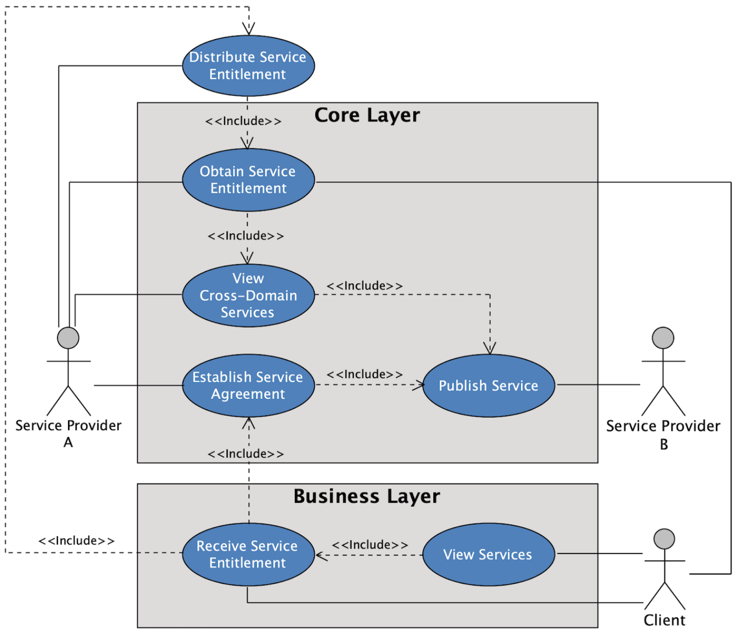


Figure 4.6: Overview of the core layer's federation features (adapted from [90])

Registered service providers can manage their service-entitlements on the federation layer. The use case diagram of Fig. 4.6 gives a functionality overview. Summarized, we distinguish between two ways of federated service exchange – the service offer and the service agreement approaches.

Service Offer:
Service providers are authorized to *publish* service-entitlements on the federation layer. Other service providers can *view* services or *obtain* corresponding cross-domain service-entitlements on demand. It is also possible to *distribute* them to core-users of their domain. The transaction registry and the anonymized business-core mapping form the basis for relating cross-domain service-entitlements from core users to the rightful users on the business layer.

Service Agreement:
Registered service providers may also *establish service agreements*. A service agreement extends the service offer functionality by automatically issuing entitlements in a context-aware manner. Fig. 4.7 depicts the agreement creation procedure. An agreement is created

when a service provider subscribes to an offer and specifies a trigger condition. For example, the condition may specify how many service-entitlements of a service provider need to be acquired or redeemed before a rewarding service-entitlement is issued. The information, whether the condition is met or not, can be retrieved by evaluating specific datasets, such as the core layer's transaction table, for instance. In this sense, after each update of the transaction table, the core layer queries all corresponding agreements and checks if one of the conditions is met by equalizing them with the transaction history. As soon as a condition is fulfilled, the rewarding service-entitlement specified inside the agreement is obtained by the host service provider's system. Receipts are generated to inform participating service providers about the issuing process of a new service-entitlement. Finally, it is linked to the corresponding user that fulfilled the condition.

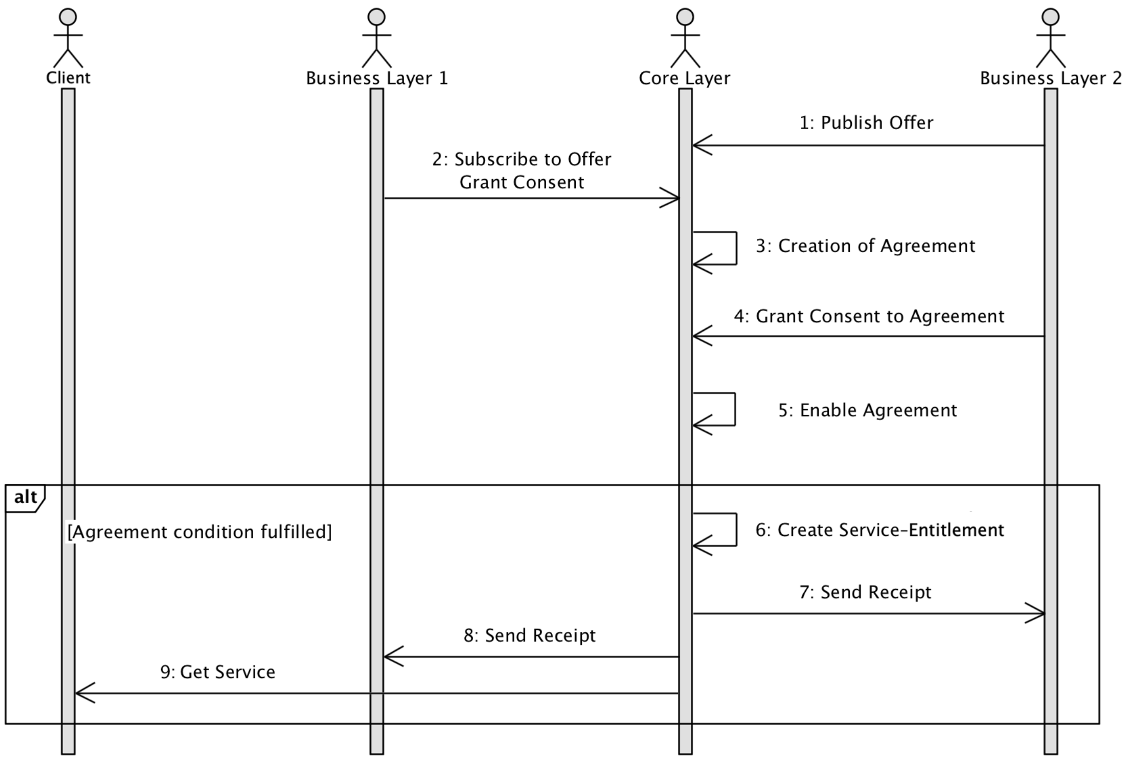


Figure 4.7: Agreement creation flow between a client, two different business layers, and the core layer

Further examples elaborated in combination with user incentivization techniques are discussed in section 4.4.4.

4.4.3 Trusted Traceability

The design approaches of the previous chapter describe how service-entitlements can be made accessible to different business layers on-demand or in an automated, condition-dependent way. Now the Blockchain technology is integrated into the core layer as an additional software building block for enabling tamper-resistant service exchange between different service providers. Service transactions shall be uploaded to the blockchain via smart contracts. This prevents duplicate transactions, non-repudiation of the redemption process, and other fraudulent manipulations since data added to the blockchain can no longer be modified. Additionally, we have advantages in terms of data integrity and data availability due to the blockchain's decentralized nature. General information and explanations about blockchain technology can be found in section 2.4.

Ethereum:
Ethereum has been chosen for our blockchain approach. It is a popular blockchain and offers different consensus mechanisms and smart contract capabilities. It is composed of miners, full nodes, and light nodes. Light nodes rely on full nodes for security and can validate states by downloading and verifying block headers. Full nodes comprise the whole blockchain database. A subset of these validate all blocks and execute the smart contracts. Ethereum comes with different consensus protocols [110]. The most common are Proof of Work, where miners put work into solving a puzzle, and Proof of Stake, where stakeholders bet on whether a particular block is added or not. By contrast, with Proof of Authority, a set of approved authorities decides whether a transaction is valid or not.

Blockchain Design Decisions:
We defined the blockchain to be semi-private. Such a blockchain is called **consortium blockchain**. This has additional advantages in terms of speed, network scope, and authorization handling compared to completely public approaches. Eventually, the consortium Ethereum blockchain has been selected in combination with the **Proof of Authority** algorithm **Clique**. This setup is beneficial for systems where the participants (e.g., different businesses) may identify but not fully trust each other. In this sense, nobody is allowed to participate in the process of reading, writing, or verifying transactions without an explicit invitation. Each application server owns a private/public keypair for blockchain interactions. The blockchain wallet containing a cryptographic keypair is generated during the registration procedure of a service provider and its system (see 4.4.1.1 for more details). The public key is used to generate a blockchain address that serves as a public user identifier. It also identifies deployed smart contracts. Furthermore, each application server on the business layer participates as a sealer in the consensus mechanism. Fig. 4.8 gives an idea of which datasets from the core layer are uploaded to the blockchain via smart contracts. In this context, data accessible by multiple business layers is stored on the distributed network, such as federated services available for the service offer ap-

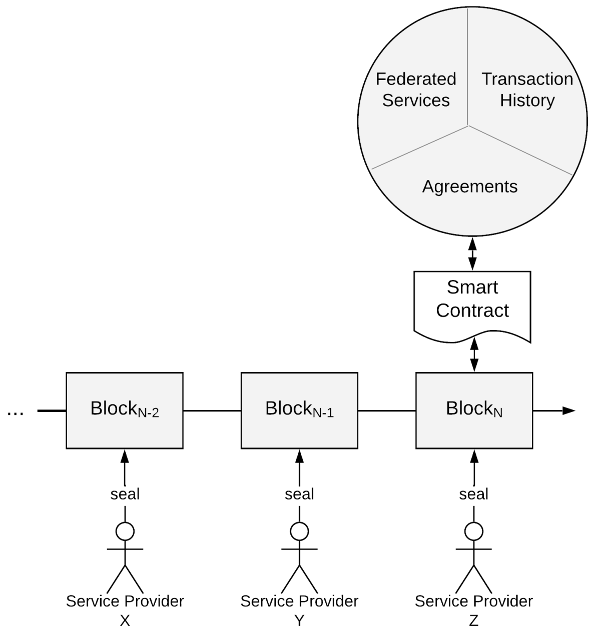


Figure 4.8: Smart contracts are used for storing cross-domain related service data onto the blockchain, such as federated services, service agreements, and transaction history entries [89]

proach, service agreements, as well as transaction-related history entries (e.g., creation, redemption, cancellation, forwarding of service-entitlements).

For each service transaction, a key-value pair is uploaded to the blockchain through a smart contract. While the key field is composed of the a-ID and a timestamp, the value field consists of the hashed transaction entry. Additionally, a blockchain receipt is generated and forwarded to the corresponding application server. This enables the validation and tracking of past transactions while ensuring their integrity. A set of transactions is merged to create a new block that is evaluated by the sealers. If the block is accepted, it will be integrated into the existing Ethereum blockchain. Due to the blockchain's cryptographic functionalities, data uploaded once cannot be altered anymore, which is advantageous if non-repudiation is a vital requirement, but may be problematic for handling service agreements (4.4.2) that should expire or need to terminate. Therefore, a one-time cancellation-request functionality has been included that can be used to set the agreement's validation state to false. This procedure disables the agreement permanently, without deleting it. The described blockchain layer can be attached optionally to the core layer. Please note that besides the advantages mentioned above, it produces a higher computational overhead due to the business layers' constant interaction for reaching consensus.

4.4.4 Service Incentivization

Gamification concepts are a possibility to integrate playful elements to non-game contexts. They often aim towards marketing goals and can therefore be used to apply special offer strategies [40, 56]. Especially in urban areas, many independent information systems are available. Incentivizing users to consume different kinds of services via unique offers and a dedicated loyalty system may benefit users and participating companies. The core layer provides service providers with services of different companies and the possibilities to (i) define condition-dependent rewarding strategies or (ii) directly distribute service-entitlements to their users. These concepts can be used to apply different gamification techniques. We will name a few examples on the basis of use case descriptions.

Example Service Offer:
Loyal customers can be rewarded with the goods and services of other companies via the service offer approach. Fig. 4.9 supplements the following example use case.

> *A client user drives to a shopping mall by car. He successfully parks it in the underground garage and enters the mall. After a while, he also enters a restaurant. Both merchants (restaurant and parking service providers) cooperate over the core layer of the CSE. They know about the other provider's services and are authorized to issue cross-domain vouchers (service-entitlements). When the client orders food, an employee or the restaurant manager may forward him a parking service-entitlement to reward him for his loyalty. The service-entitlement can be used to exit the shopping mall for free or for better parking conditions [90].*

Summarized, service provider A (restaurant) forwards a digital voucher (service-entitlement) to one of his customers that authorizes him to use the service (parking) of service provider B (garage) without applying time and cost consuming mutual integration procedures.

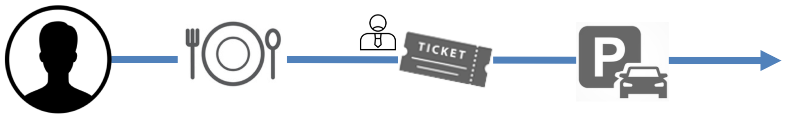


Figure 4.9: Incentivation example – service offer approach [90]

Example Service Agreement:
The service agreement approach enables service providers to define rules for reacting to users' specific app-usage and purchase behavior. The reward is issued automatically without having to claim it when the rewarding condition is met. Conditions can be specified according to the data accessible by the core layer. Fig. 4.10 summarizes the service agreement approach with a service-consumption example.

Condition A (service-consumption): A client user rents e-cars from a car rental agency regularly. As part of an e-mobility campaign, e-bike services are advertised. A new service agreement between the car and bike providers is established. As soon as a client user has rented an e-car a few times, a rewarding service is automatically unlocked. He receives an e-bike-rental service-entitlement that can be redeemed for a free e-bike ride.
Condition B (service-usage): A client user such as a tourist, for example, downloads and installs a mobility app. He uses it to rent a car for the weekend and is now on the way to his destination. In case the application's location services are enabled, and location-specific data is shared, he may receive additional services based on his location. For example, if the user is near a museum with his vehicle, he could get a discount for the museum admission.

Due to the global transaction registry of the federation layer, it is possible to trace which services were initially obtained and consumed by whom. Only the business layer knows the user's true identity due to the privacy-preserving processing of the core layer. If more data is available (see section 4.6) also specific service-usage conditions could be created. The core layer's generic gamification approach can be extended with additional features on the business layer, like an agreement tracker. If this is the case, the service providers can keep track of the completion state of all agreements their users are involved in. The same is true for clients if the interface of the business layer is adapted accordingly.

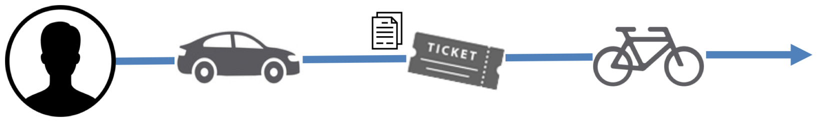


Figure 4.10: Incentivation example – service agreement approach [90]

Example Custom Rewarding Mechanisms:
Each business layer may use the generic core layer as a basis to create customized, rewarding strategies. Services of different service providers could be provided for a single app, e.g., a mobile application that provides a route planner using different mobility services (public transportation, e-cars, etc.) as depicted in Fig. 4.11. This approach could be extended with a point-based system, like in the following use case example:

The client user wants to reach a specific destination. One of the city's goals is to reduce CO_2 emissions. Therefore the city transportation app suggests different transportation schemes for reaching his destination. The user may be rewarded for traveling with environmentally-friendly transportation means, making trips with such vehicles more attractive. The more environmentally friendly the chosen vehicle is, the more points he would get that could again be exchanged for other (federated) services.

Figure 4.11: Incentivization example – custom rewarding mechanism

4.5 Ubiquitous Authentication and Authorization

Different methods for accessing locally and digitally available resources are part of today's smart cities. However, the problem is that due to environmental and use case restrictions, no standard access method can be applied. This part of the thesis complements section 4.4 by defining different methods for secure authentication and authorization between client and kiosk devices and the corresponding server infrastructure. The concepts are based on the issuing, validation, and redemption processes of digital tokens (see 4.5.1). Depending on the service type and the process of how to access the underlying local or digital resources, we distinguish between distinct access methodologies explained in subsections 4.5.2 and 4.5.3, respectively. Table 4.1 gives examples of services, their types, and accessible resources.

Table 4.1: Examples of services, types, and resources

Service	Type	Resource
Unlock e-bike	Local	Bike
Enter parking lot	Local	Parking Spot
Start a charging cycle	Local	Charging Station
Get permission for reading datastream	Digital	Sensory Data

4.5.1 Token Types for Authentication and Authorization

The core layer described in the previous sections is now extended. It becomes a certification authority in charge of managing digital tokens. Tokens are digital objects that act as tickets for authenticating devices and entitle them to consume resources of different kinds, available locally or digitally. First, tokens are issued and signed by the core server. The elliptic curve secp256r1 (prime256v1, NIST P-256) is used for cryptographic signatures.[1] After the tokens have been created, they are assigned to a core-user entity. Due to the

[1]According to the 2018 ECRYPT-CSA recommendation (https://www.keylength.com/en/3/; last access: 8th June 2020) this type of curve is recommended at least until 2028.

established interface between the client and the core server, tokens can be forwarded to the client. During the redemption process, they are transferred to kiosk entities, where they are validated. Eventually, after the client passed all authentication and authorization checks, access to the resource specified by the corresponding service-entitlement is granted. A more detailed explanation of different token types can be found on the following pages.

4.5.1.1 Authentication-Token (A-Token)

An A-Token is core-user-bound and responsible for securely authenticating a client device when communicating with another device over a local channel. It is created in the course of a distributed Kerberos-based mechanism during the client's registration explained in 4.4.1.1. Subsequently, it is forwarded to the client and stored locally. It can be transmitted to other devices via local data transmission protocols without an active internet connection, for example. An A-Token object is composed of the following elements (see the left part of Fig. 4.12):

- *ID:* Unique device-bound identifier.
- *Public Key:* Public key of the device that requested the A-Token.
- *Signature:* Cryptographic signature of the core server.
- *Attributes:* Additional attributes for storing application-related datasets.
- *Validity Period:* Timestamp providing information on the token's validity period.

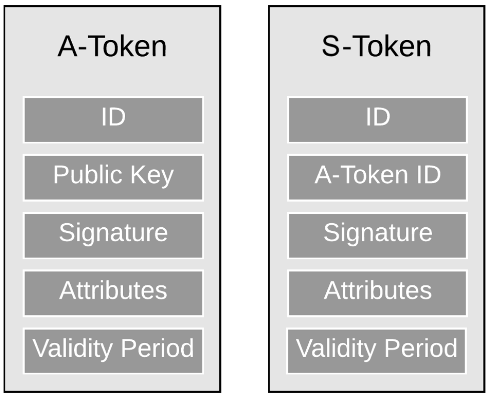


Figure 4.12: Overview of the A-Token for local authentication and the S-Token for local authorization (adapted from [86])

4.5.1.2 Service-Token (S-Token)

All business and service relevant data is transformed into a service-entitlement object stored online on the core layer, as discussed in section 4.4.1.2. In case a local copy of it is required for enabling offline authorization, a Service-Token (S-Token) is derived from the entitlement-object and core-server signed. In this case, the unique IDs from the entitlement-object that identify the service (s-ID) and the corresponding application (a-ID) are embedded into the S-Token. Each S-Token is mapped to one A-Token and authorizes a client device to access a certain resource. Summarized, the S-Token consists of the following elements (see the right part of Fig. 4.12):

- *ID:* Unique device-bound identifier.
- *A-Token ID:* Identifier used as a reference to the corresponding A-Token.
- *Signature:* Cryptographic signature of the core server.
- *Attributes:* Additional attributes for storing application-related datasets, e.g. (i) (s-ID) and (a-ID) to identify the corresponding service and application, (ii) redemption type for local BLE or HTTP redemption (see subsection 4.5.2), (iii) required local redemption range, (iv) entitlement-state (valid, invalid, redeemed, canceled), (v) entitlement-flag (consumable, persistent, non-redeemable).
- *Validity Period:* Timestamp providing information on the token's validity period.

4.5.1.3 OAuth-Token (O-Token)

The previously mentioned tokens are used to access locally available resources involving a client and an embedded kiosk device. By contrast, the O-Token – based on the OAuth 2.0 protocol[2] – is utilized to authorize digital data exchange between a client and a virtual kiosk (backend). In both cases, the central core server is responsible for authorization management. Also, the O-Tokens are issued by the core server. Their structure is defined as follows (see Fig. 4.13):

- *Scope:* The scope determines which data the receiver of the token is allowed to access.
- *Validity Period:* Timestamp providing information on the token's validity period.
- *Value:* The value field is the token identifier.
- *Token Type:* It tells how the token is going to be used to access the resource. Our concept foresees the usage of bearer type tokens; thus, access to the resource is only given to the token's bearer.

[2]https://oauth.net/2/ (last access: 13th August 2020)

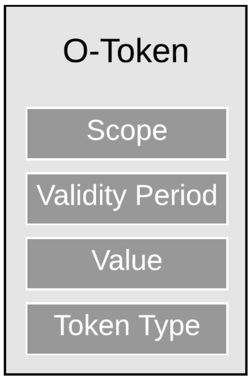


Figure 4.13: Overview of the O-Token for online authorization procedures [86]

The OAuth 2.0 protocol specifies several grant types that define how the OAuth access token can be retrieved. In our case the **client credentials** flow[3] is performed between client and core layer. The initial client request for obtaining an O-Token contains a client-ID, a client-secret, an access-token-URI (specifies where the O-token is being requested from), and the scope-information (defines which resources are being requested).

4.5.2 Local Service Redemption

Concerning local service redemption, clients, the core server, and embedded kiosks interact with each other. As discussed in section 4.3, an embedded kiosk is an application running on a mobile or a stationary device embedded into the infrastructure. During local service redemption, (i) clients are authenticated, and (ii) their entitlements are validated. If all checks are passed, the client is authorized to access a locally available resource. Redemption channels can be established via BLE (4.5.2.1) or HTTP (4.5.2.2), thus foreseeing different checks to confirm the client's physical presence. In both cases, the client's S-Token is transmitted to the core server after the client has passed an authentication procedure. Subsequently, the entitlement validation phase starts where several checks on the token and the corresponding service-entitlement are conducted (e.g., signature check, validity period check, affiliation check).

4.5.2.1 Local BLE Redemption

The communication between the client and the embedded kiosk (redemption unit) is established via BLE. Mutual authentication is provided with a dedicated challenge-response mechanism that counters eavesdropping and replay attacks. It is depicted in the blue part of the sequence diagram in Fig. 4.14. Both A-Tokens (in this example A-Token$_{\mathrm{C}}$ and A-Token$_{\mathrm{EK}}$) are exchanged, checked for their validity, and verified with the server's public

[3]https://oauth.net/2/grant-types/client-credentials/ (last access: 13th August 2020)

key. Furthermore, the client generates a random number (challenge $\mathrm{C_C}$) and challenges the embedded kiosk to sign it with its private key before it is sent back to the client. The received signature is verified against the random number $\mathrm{C_C}$ by using the kiosk's public key embedded into the previously exchanged A-Token$_{\mathrm{EK}}$. If this verification is successful, the ownership of A-Token$_{\mathrm{EK}}$ is proven. The same method is also applied to A-Token$_{\mathrm{C}}$. This challenge-response protocol is part of our middleware and can be used independently of the underlying communication protocol (BLE, Near Field Communication (NFC), etc.), enabling a higher level of modularity and reuse.

If the mutual challenge-response protocol is passed, the authorization part starts, pointed out in the green block of the sequence diagram of Fig. 4.14. First, the embedded kiosk checks if the S-Token$_{\mathrm{C}}$ belongs to the rightful user by comparing it to the A-Token$_{\mathrm{C}}$. Second, the S-Token's signature is verified with the core server's public key. Consequently, the embedded kiosk triggers the online redemption by involving the core server via a RESTful HTTP interface. The kiosk authenticates itself with its API-key and submits its and the client's A-Tokens and the client's S-Token. Eventually, the core server conducts different checks. The most important checks are discussed in the following:

1. *Validity period:* It is checked if the S-Token$_{\mathrm{C}}$ has expired by comparing its timestamp field with the current date and time.

2. *Service State:* Emphasis is laid on whether the state of both Entitlement$_{\mathrm{C}}$ and its corresponding S-Token$_{\mathrm{C}}$ are valid or not. Additionally, it is verified if the entitlement is redeemable according to its specified properties.

3. *Device State:* The devices of the client and the kiosk are derived from their A-Tokens. Their state has to be valid.

4. *Relation User and Entitlement:* This check verifies if the client device's user is the same as the entitlement-holder.

5. *Relation Business Layer:* Since different business layers can be connected to the core layer, this check assures that both client and kiosk devices are part of the same business layer.

Finally, the core server performs an update of the transaction-database and informs the embedded kiosk and the client about the successful redemption procedure. This part of the program sequence of Fig. 4.14 has been visualized in a simplified form with a "Redemption Success" reply.

4.5.2.2 Local HTTP Redemption

Depending on different technology and use case requirements, communication over a direct local channel between client and embedded kiosk is not always an option. In the local

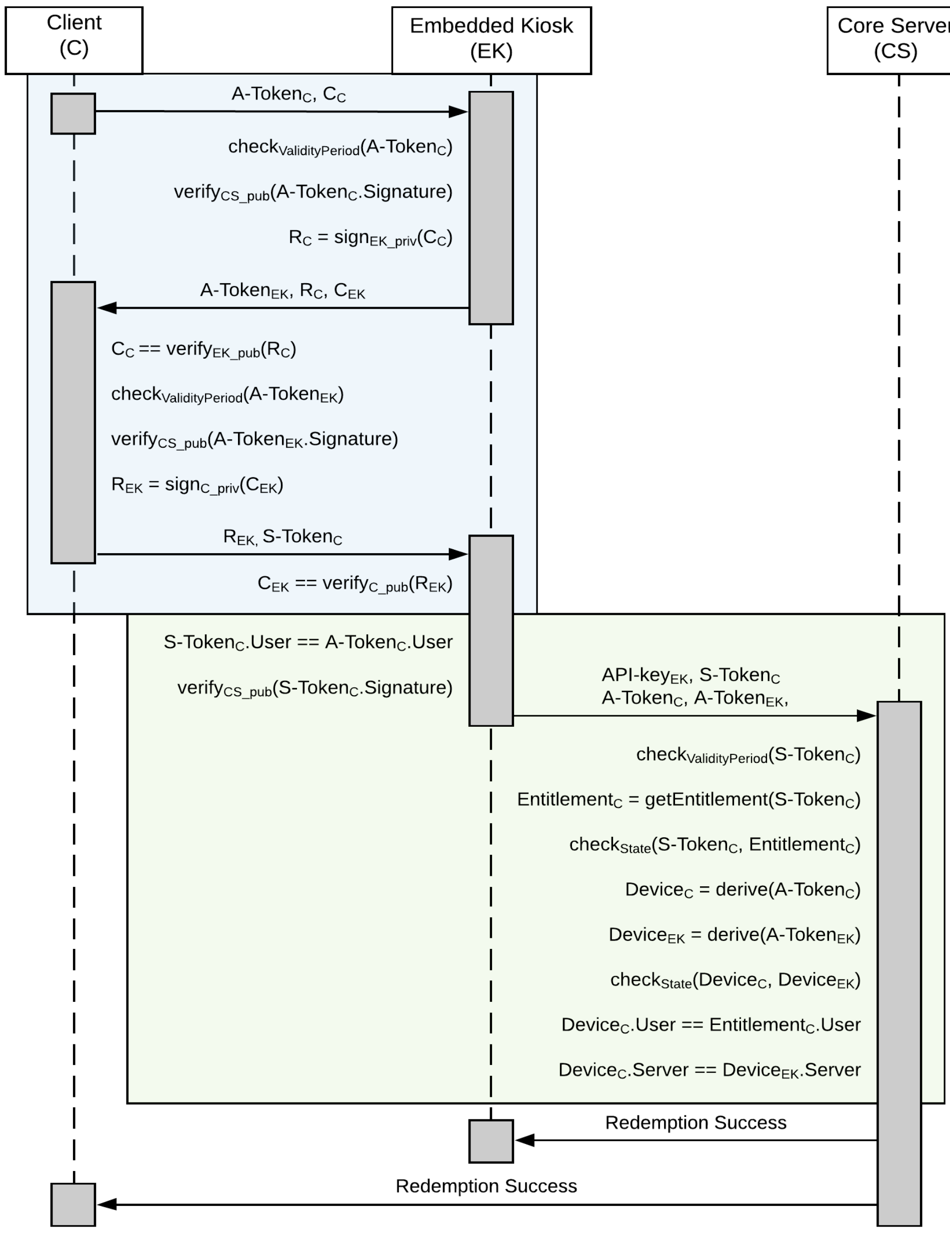

Client (C)
Embedded Kiosk (EK)
Core Server (CS)
A-Token_C, C_C
check_ValidityPeriod(A-Token_C)
verify_CS_pub(A-Token_C.Signature)
R_C = sign_EK_priv(C_C)
A-Token_EK, R_C, C_EK
C_C == verify_EK_pub(R_C)
check_ValidityPeriod(A-Token_EK)
verify_CS_pub(A-Token_EK.Signature)
R_EK = sign_C_priv(C_EK)
R_EK, S-Token_C
C_EK == verify_C_pub(R_EK)
S-Token_C.User == A-Token_C.User
verify_CS_pub(S-Token_C.Signature)
API-key_EK, S-Token_C A-Token_C, A-Token_EK,
check_ValidityPeriod(S-Token_C)
Entitlement_C = getEntitlement(S-Token_C)
check_State(S-Token_C, Entitlement_C)
Device_C = derive(A-Token_C)
Device_EK = derive(A-Token_EK)
check_State(Device_C, Device_EK)
Device_C.User == Entitlement_C.User
Device_C.Server == Device_EK.Server
Redemption Success
Redemption Success

Figure 4.14: Local BLE redemption flow between client, embedded kiosk, and core server

HTTP redemption case, the core server takes care of the authentication, validation, and redemption of the client's tokens instead of the embedded kiosk. Compared to the BLE redemption case (see 4.5.2.1) the client requires an active internet connection for carrying out RESTful HTTP calls.

Fig. 4.15 points out the overall program sequence. First, the client's position data consisting of latitude and longitude coordinates are retrieved. GPS and network lookups are conducted for a more accurate and error-resistant estimation of the client's physical location. While for GPS the location information is retrieved using satellites, in the other case, the location is based on cell tower availability and Wireless Fidelity (Wi-Fi) access points. The client submits its location information to the server together with its A-Token and S-Token. Authentication is provided with its API-key. Consequently, the core server performs several validation steps similar to the checklist of the local BLE redemption method discussed in 4.5.2.1. Complementary to it, the submitted coordinates are also checked against the possible redemption range of the S-Token. If all tests are passed, the core server contacts the corresponding locally available kiosk device via a push notification. This completes the redemption process by granting the client-user access to the requested resource.

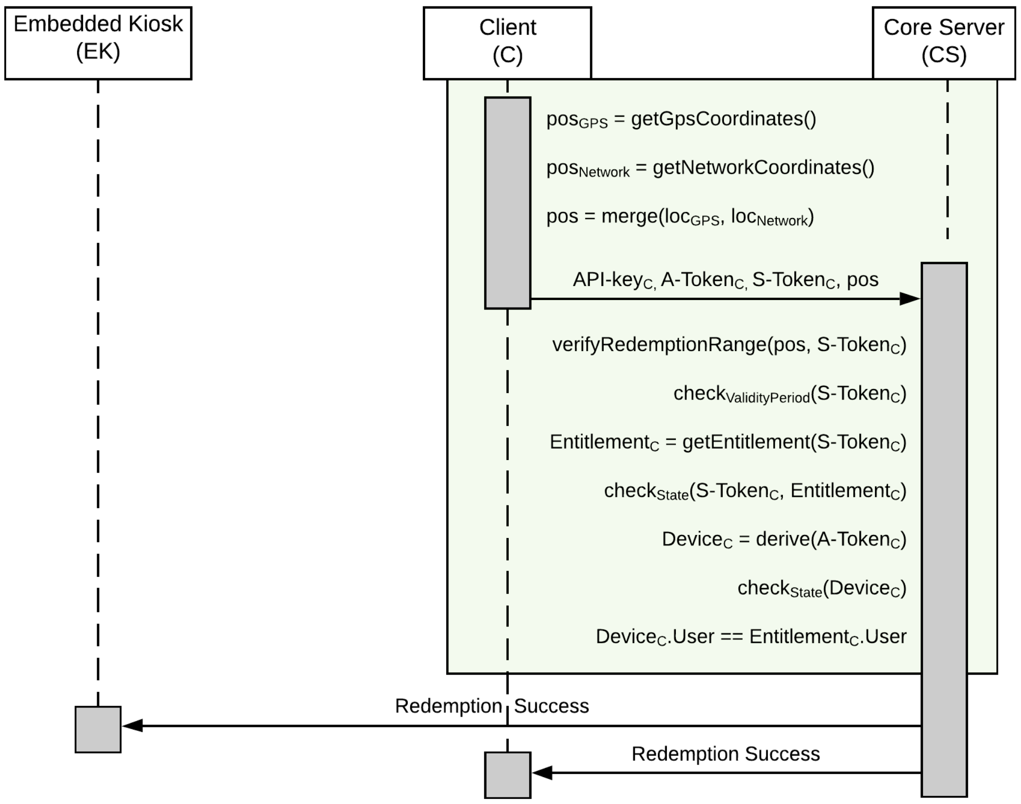

Embedded Kiosk (EK)
Client (C)
Core Server (CS)
pos_{GPS} = getGpsCoordinates()
$pos_{Network}$ = getNetworkCoordinates()
pos = merge(loc_{GPS}, $loc_{Network}$)
API-key_C, A-$Token_C$, S-$Token_C$, pos
verifyRedemptionRange(pos, S-$Token_C$)
$check_{ValidityPeriod}$(S-$Token_C$)
$Entitlement_C$ = getEntitlement(S-$Token_C$)
$check_{State}$(S-$Token_C$, $Entitlement_C$)
$Device_C$ = derive(A-$Token_C$)
$check_{State}$($Device_C$)
$Device_C$.User == $Entitlement_C$.User
Redemption Success
Redemption Success

Figure 4.15: Local HTTP redemption flow between client, embedded kiosk, and core server

4.5.3 Virtual Service Redemption

If the redemption process aims to obtain digital data packets stored online, virtual kiosks are used. A virtual kiosk is a server-side application that holds data (e.g., sensory or mobility data) of one specific service provider. It also acts as a redemption place where clients can obtain the data in question by redeeming digital tokens over a REST-based interface. The authorization procedure of a virtual kiosk is based on the open-standard protocol OAuth 2.0 [57]. The central core server is responsible for managing the authorization process between clients and virtual kiosks. It issues O-Auth-Accesstokens (O-Tokens) derived from service-entitlements for client devices (see 4.5.1.3). This grants the clients access to specific datasets when validated and redeemed at virtual kiosks (e.g., distribution of vehicles used in a specific city, temperature value stream). An overview of the program sequence is given in the following paragraph and by Fig. 4.16:

1. A client wants to access data stored online. This data is owned by a service provider running a virtual kiosk that trusts the core server. The virtual kiosk only shares the data with whoever has the permission (a valid O-Token) to access it. Therefore, the client requests an O-Token from the core server.
2. An authentication request is triggered by the core layer.
3. The client authenticates itself via its API-Key and sends the requested credential data to the core layer. The O-Token generation process follows the OAuth 2.0 Client credentials flow.
4. The O-Token is issued by the core layer and sent to the client.
5. The O-Token is forwarded from the client to the virtual kiosk. It specifies which resource the client would like to access.
6. Before data is exchanged, the validity of the O-Token has to be verified. The virtual kiosk does not know or explicitly trust clients, but it trusts the core server, which receives the token.
7. The core layer validates the O-Token and sends the validation results back to the client device.
8. If the O-Token is validated successfully, the virtual kiosk shares the requested resource with the client.

With this distributed approach, it is possible to store application-related data on the service provider side only. Clients and virtual kiosks can communicate directly without a trust relationship between them by involving the core layer as a central authorization center. In case confidential data should be exchanged, there is the possibility of using

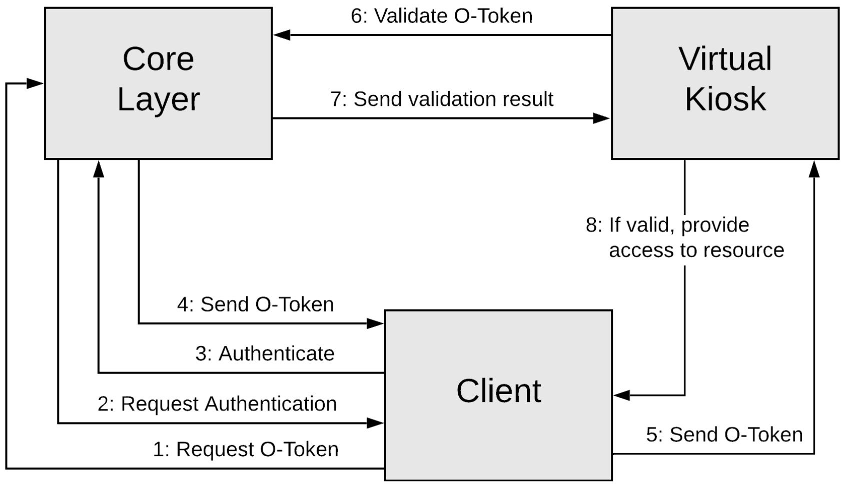


Figure 4.16: Virtual redemption flow between client, virtual kiosk, and core server [86]

multiple virtual kiosks: the data itself stored on one virtual kiosk and the decryption key on another virtual kiosk. This way, the described concept also supports end-to-end encryption methodologies since clients and virtual kiosks would be the only entities that can access the data.

4.6 Privacy Preserving and Secure Data Aggregation

In the previous sections, we discussed a platform that enables the exchange of various services. It also provides different means to authenticate and authorize users to access various resources. Complementary to that, this chapter focuses on the privacy-preserving aggregation of data and how it is processed and postulated throughout the overall platform. For more details please refer to subsection 4.6.1. Additionally, a design-space-exploration driven approach for end-to-end secured data collection is pointed out in subsection 4.6.2.

4.6.1 Adaptive Privacy

As already explained in section 4.4, the user data stored on the business layer is never forwarded to the core layer. Only anonymous IDs linked to the user and services of the business layer are stored on the core layer. Privacy is a very individual topic since what is considered highly private by one person might not be considered private by another. Therefore, it is essential to provide a mechanism to let the user have a say in what data is shared. On the one hand, from the **perspective of a client user**, there should be the possibility (i) to get an overview of the data that is collected during the usage period of a mobile application and (ii) to adapt the data flow according to the user's privacy preferences. From the **perspective of a service provider** on the other hand, access to the stream of shared anonymized data would help (i) to optimize their services and (ii) to issue customized offers. Dedicated rewarding mechanisms (see subsection 4.4.4) may incentivize users to share more data. In this sense, the more permissive the privacy settings are, the more services may be unlocked.

According to the user's privacy preferences, a new concept is proposed for sharing data collected during the runtime of a client's application in a privacy-preserving way. The data shall be shared with the federation layer (4.4.2) and made accessible to all participating service providers connected to it for context-aware service-entitlement creation.

4.6.1.1 Privacy Token

A new generic data-structure is introduced – the Privacy-Token (P-Token). It provides information about the type of data the client user shares during the runtime of the application. Just as in the case of the A-Token (4.5.1.1), S-Token (4.5.1.2), and O-Token (4.5.1.3), the P-Token is issued by the core server. It is used to approve data sharing of clients through local and online communication channels. The P-Token is depicted by Fig. 4.17 and consists of the following entries:

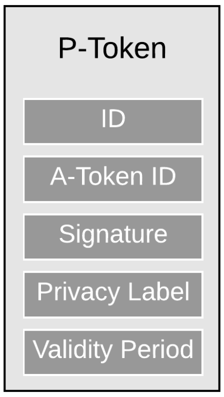


Figure 4.17: Overview of the P-Token acting as interface for managing the privacy of clients

- *ID:* Unique P-Token identifier
- *A-Token ID:* Reference to the client's A-Token. Via this link the corresponding device, its public key, and the core user are known.
- *Signature:* The entire object is signed by the core-server.

- *Privacy Label:* A privacy label is subdivided into a category and a privacy level. One P-Token may consist of multiple privacy labels. More details are discussed in 4.6.1.2.
- *Validity Period:* Indicator how long the P-Token is valid. The data shared by users is visible to other participating service providers for as long as the validity period is not exceeded, or clients do not change their privacy settings.

4.6.1.2 Privacy Labels

In order to cope with different privacy requirements, various privacy labels are specified by the core layer. They consist of privacy categories and privacy levels and are part of each P-Token. On the one hand, **privacy categories** refer to the type of data being collected and are influenced by the application. Categories may be but are not limited to areas such as health, mobility, entertainment, education, shopping, and social data areas, for instance. On the other hand, the **privacy level** defines to which extent personal data is shared. Level A is the most strict level regarding data aggregation, while higher levels are more permissive. Applying this generic concept to a mobility app, for example, results in the following setup. Shared data may consist of mobility (e.g., destination address, ticket expiration date) and personal datasets (e.g., gender, age) without uniquely identifying a person. For example, the privacy levels for mobility data could be defined as follows:

- *Level A:* The mobility services offered by the application can be used. However, no mobility data is shared with the core layer.
- *Level B:* Vehicle-based data is collected (e.g., type of vehicle, usage period) and shared on the core layer.
- *Level C:* In addition to vehicle-based data, trip-based data (e.g., GPS, destination address) is collected during the driver's/rider's journey and shared with participating service providers.

Fig. 4.18 summarizes the overall concept. The datasets submitted via the P-Token may be used by service providers of different domains (App-Server$_A$, App-Server$_B$, App-Server$_C$) to create new offers and agreements. Since the P-token is linked to the user's A-Token, it is possible to issue service-entitlements for the right clients without knowing their real identity. While some data is required to guarantee the application's correct functioning and thus the minimum privacy level is set by the application by default, other data may be optional for optimizing service offers (4.4.2) for the client. Suppose the user decides upon a higher privacy level and thus consents to the application to share data with the core layer. In that case, the P-Token may be submitted directly to the core server at specific trigger points of the client application (e.g., obtainment, usage, redemption of services) or locally to kiosk devices. If the validation of the P-Token is successful, and

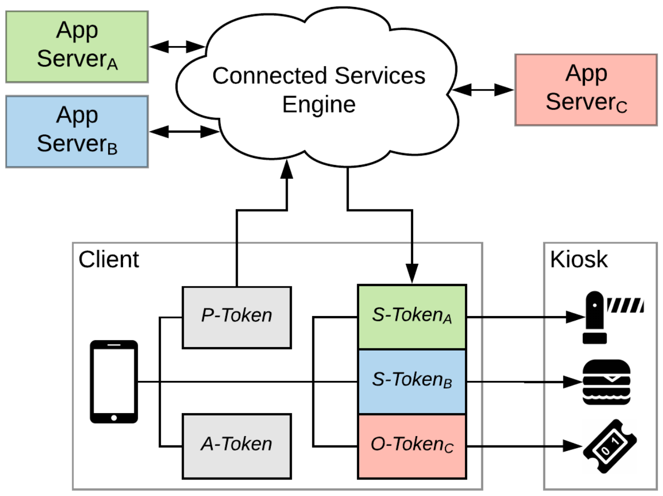


Figure 4.18: P-Token application scenario (adapted from [87])

if the privacy levels allow it, the core server accepts the transferred data. Last but not least, the data is stored inside the federation layer of the core server, where other service providers can access it.

4.6.2 End-to-End Secured Data Transfer Study

Often data of confidential nature has to be exchanged; thus, end-to-end security strategies should be taken into account. It is important to consider protection schemes on different hardware and software layers at early development phases to find the optimal design for the application and avoid problems such as inappropriate security levels, performance issues, and longer time-to-market cycles.

The virtual kiosk was described in subsection 4.5.3. Now the cycle of data collection and redemption is completed by explaining a mechanism of aggregating data and securely forwarding it to a virtual kiosk. A case study has been conducted based on a security-aware DSE framework [50] that takes our system definition as input and identifies a set of potential hardware and software building blocks with specific security and performance properties for key management and distribution. Based on these exploration phase results, a proof of concept was integrated into the overall platform. The data collection system is oriented towards the system architecture of common IoT systems, depicted by Fig. 4.19 and described in the following:

- **Edge Node – Data Collector.** A power and computational resources constrained device. It collects data via sensors from the local environment at constant intervals. It encrypts all data it retrieves before forwarding it to other devices.

- **Gateway – Device Connector.** The gateway acts as a connecting unit between edge devices and the virtual kiosk. It receives data from locally available edge devices, optionally extends the data bundle by adding additional information, and forwards it to the virtual kiosk. The gateway is a trusted communication participant but cannot read the received encrypted data.
- **Virtual Kiosk – Data Repository.** This server is the direct point of contact for the gateway device and receives the secured data package. The virtual kiosk can decrypt the received data. Furthermore, it can communicate to the core layer, which acts as an authorization unit for distributing data services to interested systems, as described in subsection 4.5.3.

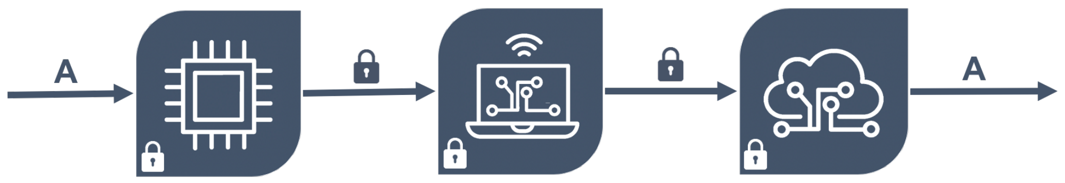


Figure 4.19: Communication participants for end-to-end secured communication. Edge nodes send encrypted messages to a gateway that forwards the data to a virtual kiosk unit for further processing

This basic high-level system is taken into account for the following design-space-exploration driven procedure, consisting of three phases: definition (4.6.2.1), exploration (4.6.2.2), and selection (4.6.2.3).

4.6.2.1 Definition Phase

The DSE framework takes various information as input. In our case, the system's functionality was modeled as a task graph, and possible hardware components were defined. Furthermore, attack scenarios were modeled as Bayesian attack graphs (BAGs) on the whole end-to-end communication path. In each attack graph, the designer had to define certain attack goals an attacker aims to reach, adding to each goal a threshold to limit the goal's attack success probability [91]. Different security options were also considered as inputs for the framework. They comprise symmetric and asymmetric cryptographic algorithms (different keys and lifespans), task encapsulation, and tamper-safe storage options to secure data packets on the **application layer**. Regarding the **transport layer**, the transmitted data packets must be confidential and authentic. The edge nodes should send their measured sensory data to the gateway using BLE, while the gateway communicates over Wi-Fi. Finally, different scalable **provisioning** schemes were considered by the framework: (i) key injection at the time of manufacturing; (ii) key fetch by the edge node from the cloud via the gateway; (iii) direct key fetch from the cloud.

4.6.2.2 Exploration Phase

The DSE tool calculates the secure system partitioning and task mapping. It determines the set of security constraints using the attack graphs and the given task graph. The security constraints state that only those solutions are considered secure for which the attack goals' success probabilities do not exceed their individually set thresholds. Depending on its configuration, the DSE tool provides different solutions from which the designer may choose depending on his performance and security requirements. Each solution represents the whole system and is depicted as a point in the scatter plot in Fig. 4.20. The solutions are ordered according to their average attack success probability and system performance and normalized to the fastest solution's performance. They are colored according to the number of attack goals reached by the attackers; *green* indicates secure and *red* less secure solutions. Solution-examples are pointed out in the following:

Edge node and gateway solutions:

> *"The framework proposes for a security optimal edge node to place all functions (except the BLE packet transmission) on the SE with the highest Common Criteria security certification.[4] Considering the performance optimal yet secure solution, the framework suggests the implementation of the functionality on an MCU supporting HWC and HWF. It proposes placing a certificate on a SE with the highest-rated security mechanisms and deriving a session key for the symmetric encryption and authentication of the BLE packets. For the gateway, the framework proposes the usage of hardware components providing the same security mechanisms. For the gateway, it also suggests the usage of session keys derived from a certificate placed on the SE. "* [91]

Virtual kiosk / cloud solutions:

> *"The framework suggests as the most secure solution for the cloud service, an HSM exclusive implementation. For the performance optimal secure solution, the framework proposes the implementation of the service's functionality on a server platform offering HWC and HWF and the storage of certificates and key material on an HSM extension. The framework suggests securing the communication between the gateway and cloud service using symmetric cryptography."* [91]

Provisioning solutions:

> *"[...] the framework selects approach (i) to be the most secure. Approach (ii) is the least secure one, caused by potential intrusion attacks on the gateway capable of disclosing the key material during provisioning. In contrast, approach (iii) is a good candidate for achieving both security and flexibility under the premise that the edge device can connect to the internet directly."* [91]

[4]https://www.commoncriteriaportal.org/ (last access: 29th August 2020)

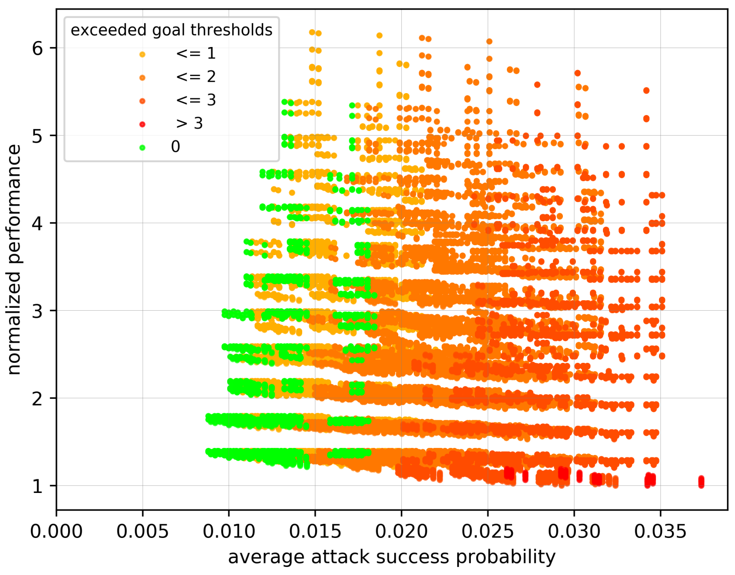


Figure 4.20: Design space exploration solutions for the overall system design. Each point represents a solution and indicates how many attack goal thresholds have been exceeded. The solutions are sorted by the normalized performance and the average attack success probability [91]

4.6.2.3 Selection Phase

According to the solutions depicted by Fig. 4.20 (i) the *most secure* solution comes with an average success probability (ASP) of 0.0074, (ii) the *least secure* with an ASP of 0.0376, and (iii) the *fastest secure* solution with an ASP of 0.0148. The latter solution has been chosen from all solutions calculated by the DSE tool due to a very low average success probability on the attacker side. In this case, the edge node and the gateway use a microcontroller unit (MCU) and a secure element (SE) for storing the sensitive key material and for providing hardware-based cryptography (HWC) and task encapsulation. For the provisioning of the edge node devices with key/certificate material, the direct injection during device manufacturing by the OEM was selected. The injected credentials are of asymmetric nature. Furthermore, the SE is responsible for creating session keys. The edge nodes communicate with the gateway using BLE. In addition to our application-layer encryption, BLE Security Mode 2 / Security Level 2 is used to enforce authentic communication. Mutually authenticated pairing is included in this process. For the secure realization of the virtual kiosk, a server platform that supports HWC, hardware-based firewall (HWF), and a Hardware Security Module (HSM) extension for the secure storage of the server's certificates and secret keys has been chosen. Communication is established via an HTTPS channel, secured via the mutual TLS protocol.

5 Implementation

This chapter gives an overview of the technologies and software projects that resulted in the SOA discussed in this thesis. The descriptions comprise information on communication aspects (5.1), the mobile applications (5.2), the service redemption and data collection procedures (5.3), and the application and core servers (5.4). Each implementation building block is introduced with a short *context and use case preview* illustrating the concepts described in chapter 4.

5.1 Communication

Regarding device to device communication methodologies, we built upon standard wireless communication protocols (e.g., BLE, HTTP). We added dedicated connection establishments, mutual security checks, and a custom payload format on top. This avoided us the implementation of larger communication stacks from scratch while maintaining a high interoperability level.

Local communication is established over BLE. All **data sent over BLE is encoded in TLV-format**. TLV elements can be placed inside the message body in any order since they can be identified through their Type/Tag byte. Another advantage compared to static encoding mechanisms is that data of variable length can be transmitted because the length information is part of the encoded package. Finally, the TLV format depicted in the following causes little overhead despite the additional tag and length bytes.

- **Type:** Unique byte-code of 1-byte length indicating the kind of data following next.
- **Length:** Holds the size of the value field. Since the length has a size of 1 byte, the maximum possible length of a data-fragment is 255 bytes.
- **Value:** Actual data bytes.

By contrast, **HTTP requests and responses are JSON-encoded**, a generic data-exchange encoding scheme. Since its syntax is small and light-weighted, this has advantages on memory consumption and execution speed. JSON is built on two generic structures that can also easily be interpreted by software and people:

- **Collection of name/value pairs**, e.g., object, record, struct, dictionary, hash table, keyed list, array.
- **Ordered value-list**, e.g. array, vector, list, sequence.

5.2 Mobile Apps

Client and kiosk apps were developed as Android applications with a minimum API level of 26 that corresponds to the Android version 8.0 or newer. The Android apps' application logic was programmed in Java and Kotlin, while parts of the user interface were elaborated within XML files. The middleware and the common libraries that enable the authentication and authorization procedures described in section 4.5 were also programmed in Java and Kotlin. This enables a higher level of compatibility with the mobile Android applications and the software of the server logic. The implemented cloud concepts of subsections 4.4.1 (access control, data abstraction) and 4.4.2 (service exchange) were used to integrate and handle different mobility services.

One app – different services: the app (client) acts as a service wallet combining services of different service providers. It has been realized within the H2020 project STEVE (2.6.1) and combines different cross-domain mobility services within one mobile application.

Light-electric vehicles, so-called quadricycles, were provided by the company *JAC-Italy Design Center SRL*[1]. Their proprietary solutions were adapted by *Politecnico di Torino*[2] and *Infineon Technologies Austria AG*[3]. A telematics unit and dedicated control logic has been added for providing location and connectivity features, thus enabling vehicle-to-infrastructure communication and bi-directional data exchange to an application server from *Vem Solutions S.r.l.*[4] over the public mobile network. Additionally, e-bikes and corresponding charging stations were adapted by *SYCUBE GmbH*[5] to be usable for a seamless micromobility experience for clients. Two application layers, one for the quadricycle service provider and the other for the e-bike service provider, were introduced.

The client and kiosk apps became part of each application layer communicating to the corresponding service provider's application server. Furthermore, both application layers were connected to the core layer, enabling the creation of local and online service access mechanisms and easier data exchange across the application servers. Figures 5.1 and 5.2 show the client app and the integrated mobility services of Villach in Austria as well as of Turin in Italy. As soon as a service is *booked/reserved*, a service-entitlement is created, and a corresponding token is issued and forwarded to the user's service wallet (see Fig. 5.3) where it can be managed. This includes redemption mechanisms subdivided into *locking* and *unlocking* of the vehicle as well as *cancellation* of the service-entitlement. More about the implementation of our redemption mechanisms is discussed in section 5.3.

[1]https://www.mesap.it/associati/jac-italy-s-r-l/ (last access: 30th July 2020)
[2]https://www.polito.it (last access: 30th July 2020)
[3]https://www.infineon.com/cms/austria (last access: 30th July 2020)
[4]https://www.vemsolutions.it (last access: 30th July 2020)
[5]http://www.sycube.at (last access: 30th July 2020)

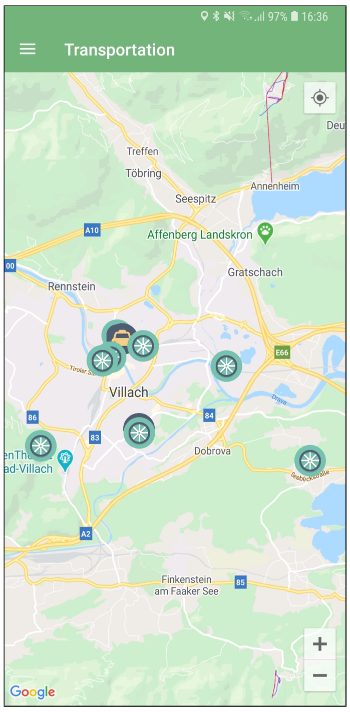


Figure 5.1 Quadricycle and e-bike services in Villach (AT)

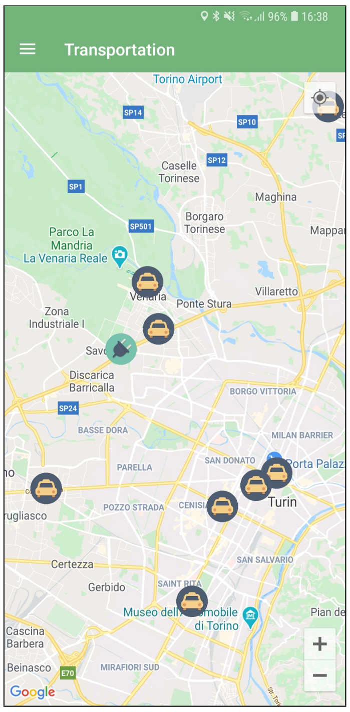
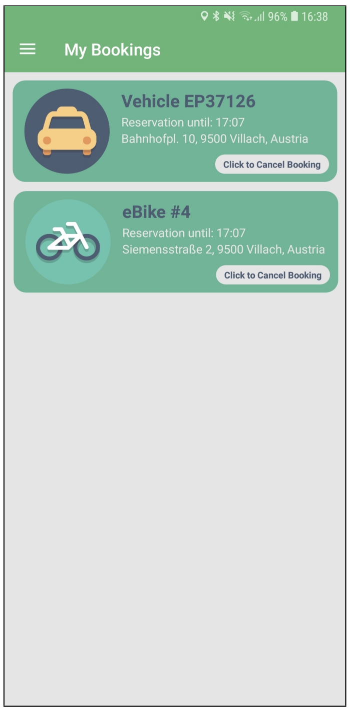


Figure 5.2 Quadricycle and e-charging services in Turin (IT)


96% 16:38

My Bookings

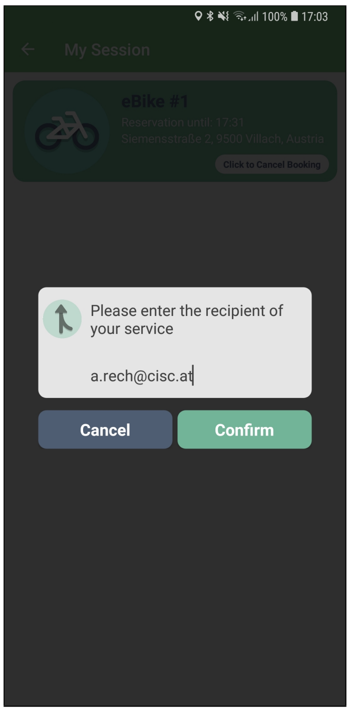
Vehicle EP37126
Reservation until: 17:07
Bahnhofpl. 10, 9500 Villach, Austria
Click to Cancel Booking

eBike #4
Reservation until: 17:07
Siemensstraße 2, 9500 Villach, Austria
Click to Cancel Booking

Figure 5.3 Service wallet – interconnection of different services

Different apps – different services: The interconnection of service platforms enables sharing mechanisms of services across different client and kiosk devices and applications.

A client$_{\text{origin}}$ user may forward obtained services to another client$_{\text{destination}}$ user according to the mapping principle described in section 4.4. In this implementation of the client app, the wallet view of Fig. 5.3 is used to trigger the forwarding feature. The client$_{\text{origin}}$ user has to press and hold the service entry that should be shared. According to Fig. 5.4, this opens a dialog for entering the recipient's email address. In case the recipient is a registered client$_{\text{destination}}$ user of a compatible business-layer, the forwarding process is carried out. The email address is used as an identifier for the business-layer to retrieve the user's$_{\text{destination}}$ core identifier. The same mechanism can also be used by the kiosk device to forward a rewarding service to a client$_{\text{destination}}$ device, for instance. Please note that in case the client devices are in close proximity, BLE could be used to directly transfer the user's$_{\text{destination}}$ core identifier.



Please enter the recipient of your service

a.rech@cisc.at

Figure 5.4: Implementation of the service forwarding feature

5.3 Service Redemption and Data Collection

Based on the explanations of section 4.5 different token-usage scenarios were implemented. According to section 4.6, also data collection mechanisms were taken into account.

The digitalization and generic handling of service-entitlements throughout their lifetime facilitates the realization of different applications and use cases in regard to service access and redemption.

The core layer enables different functionalities, such as service obtainment, redemption, and cancellation. These generic mechanisms were adapted to the STEVE client mobility app's requirements for accessing light-electric vehicles (quadricycles) and e-bikes. Four service-categories are distinguished to access a mobility-resource: (i) **book** (reserve the resource for a predefined time, in our case 30 minutes), (ii) **cancel** (release the booked resource), (iii) **start** (unlock the resource and start a session), and (iv) **stop** (lock the resource and finish the session).

The following tables depict how the overall functionality is split across all communication participants. Table 5.1 shows the services for the quadricycles in Turin. For unlocking or locking a quadricycle with the STEVE client app, the session view shown in Fig. 5.5 can be used. The start button triggers the unlocking procedure, while the stop button is responsible for requesting the locking mechanism. Consequently, the connection to the mobile kiosk app of Fig. 5.6 (local redemption unit) is established. For more details regarding the local BLE redemption procedure, please refer to 4.5.2.1.

Table 5.1: Service functionality for managing and accessing quadricycles in Turin split across client & kiosk, app server, and the core server

Category	**Client & Kiosk**	**App Server**	**Core Server**
book	getVoucher()	obtainResource()	createEntitlement()
cancel	discardVoucher()	cancelResource()	cancelEntitlement()
unlock	localRedemption$_{\text{BLE}}$()	startTicketing()	redeemEntitlement()
lock	localRedemption$_{\text{BLE}}$()	stopTicketing()	redeemEntitlement()

By contrast, Tables 5.2 and 5.3 summarize the main actions for the quadricycle and e-bike use cases in Villach. Due to the implemented context-awareness, the client handles these vehicles via the local HTTP redemption procedure described in 4.5.2.2. Fig. 5.7 shows the client app's session view for the e-bikes that can be used to trigger the redemption flow in case the location checks are passed successfully.

Table 5.2: Service functionality for managing and accessing quadricycles in Villach split across client & kiosk, app server, and the core server

Category	**Client & Kiosk**	**App Server**	**Core Server**
book	getVoucher()	obtainResource()	createEntitlement()
cancel	discardVoucher()	cancelResource()	cancelEntitlement()
unlock	localRedemption$_{\text{HTTP}}$()	startTicketing()	redeemEntitlement()
lock	localRedemption$_{\text{HTTP}}$()	stopTicketing()	redeemEntitlement()

Table 5.3: Service functionality for managing and accessing e-bikes in Villach split across client & kiosk, app server, and the core server

Category	**Client & Kiosk**	**App Server**	**Core Server**
book	getVoucher()	obtainResource()	createEntitlement()
cancel	discardVoucher()	cancelResource()	cancelEntitlement()
unlock	localRedemption$_{\text{HTTP}}$()	startRide()	redeemEntitlement()
lock	localRedemption$_{\text{HTTP}}$()	stopRide()	redeemEntitlement()

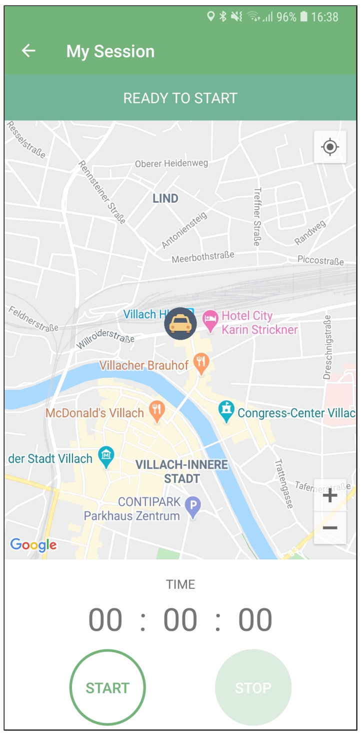


Figure 5.5 Client device – quadricycle session management

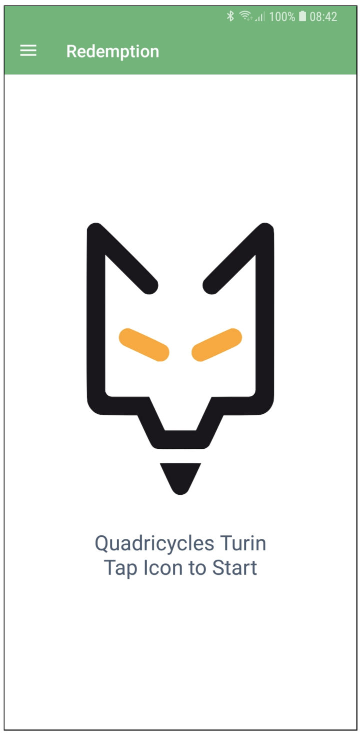


Figure 5.6 Kiosk device – mobile redemption unit

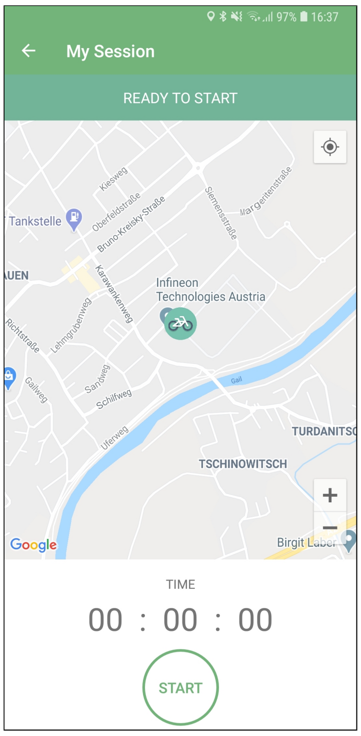


Figure 5.7 Client device – e-bike session management

In addition to different local redemption methods, the virtual redemption described in 4.5.3 was applied in the course of the STIP project (2.6.3). In this case, the redemption process aims to obtain digital data packets stored on virtual kiosks. Table 5.4 shows all communication participants and their interaction. The core server is used as an authorization center that allows trusted third parties (clients) to obtain data stored on virtual kiosks.

Table 5.4: Service functionality for accessing data stored online split across client & kiosk and the core server

Category	Client & Kiosk	Core Server
request	getVoucher()	createEntitlement()
cancel	discardVoucher()	cancelEntitlement()
receive	virtualRedemption()	redeemEntitlement()

Complementary to the virtual redemption mechanism, an end-to-end secured data collection strategy from embedded devices to virtual kiosks was designed. Subsequently, a multi-layered data collection system for virtual kiosks was implemented, considering the results of the design-space-exploration procedure of 4.6.2.3.

As an edge node, a *Raspberry Pi 3 Model B* device with an integrated *DHT22 temperature and humidity sensor* was used. The *secure element EdgeLock SE050* was attached to the edge node. It works as an HSM and increases the host controller's security concerning cryptographic key and certificate management and distribution. The SE050 connects and communicates to the host MCU using the chip's I2C interface. The host MCU runs the application logic and controls all cryptographic operations (encrypting, signing, hashing, etc.). The latter was achieved through software integration of the SE050 support package[6]. Regarding wireless connectivity, the Raspberry Pi's integrated BLE chip was utilized to establish local communication to the gateway. The gateway itself is a *Raspberry Pi 4 Model B* controller that supports BLE and has an integrated Wi-Fi module and Ethernet port used for HTTPS calls to the virtual kiosk server. The virtual kiosk server was realized as an Amazon EC2 instance and extended with the AWS Cloud-HSM. The server's application connects to the HSM using mutually authenticated TLS channels established by the HSM client software. Fig. 5.8 shows the system setup of the embedded layer consisting of two edge nodes (two devices on the right side) and a gateway unit (left part).

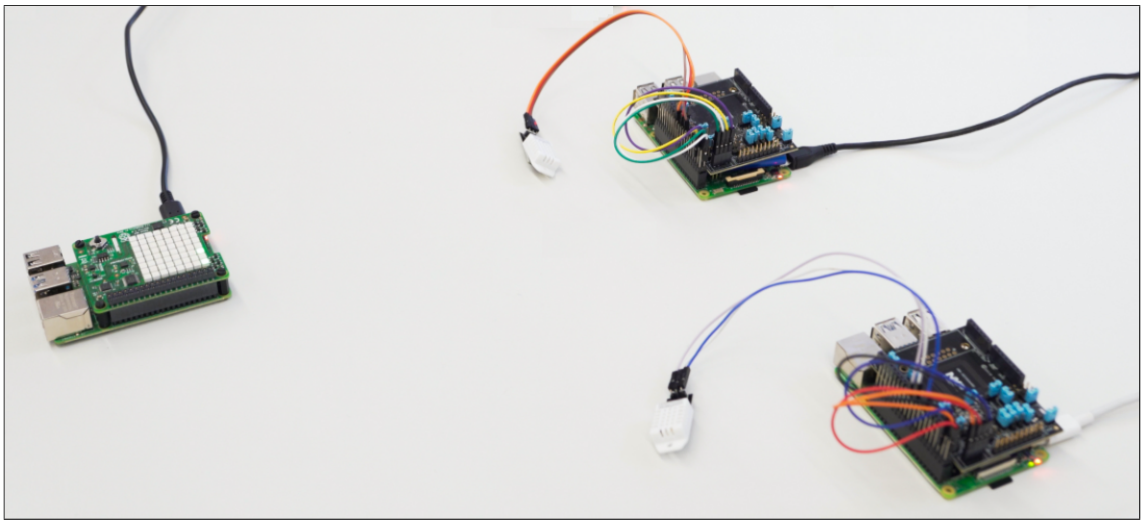
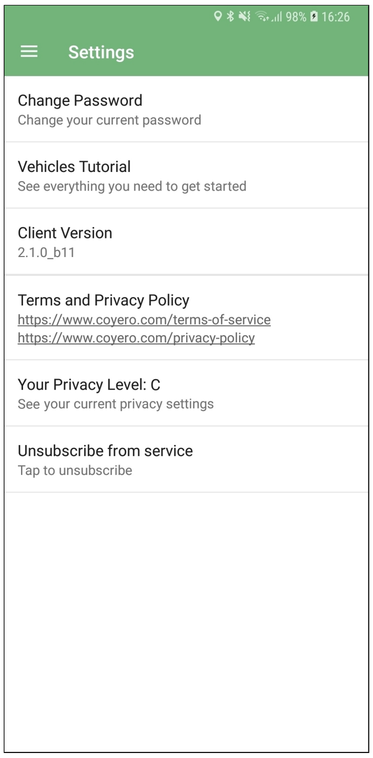


Figure 5.8: End-to-end secured data aggregation system setup: the host controllers are equipped with the secure element EdgeLock SE050

[6]https://www.nxp.com/webapp/Download?colCode=SE050-PLUG-TRUST-MW (last access: 22nd July 2020)

On top of the anonymization of the users' identity, users should be allowed to see and choose which kind of data should be shared on the cloud layer, resulting in new service-offers or service-agreements.

According to subsection 4.6.1 different privacy levels were defined in the course of the SCOTT project (2.6.2). The user can decide which type of data is shared globally with participating service providers while still masquerading his true identity. A dedicated user interface was elaborated within the settings view of the mobile client app. In the menu's penultimate line, the privacy settings can be changed (see Fig. 5.9). A privacy label example has been discussed in 4.6.1.2. The proof of concept depicted by the figure below is based on it. In this context, a *C-privacy-level* means that mobility data (vehicle-based and trip-based data) is shared with the core layer. The shared data can be visualized via the web-front-end shown in section 5.4. Consequently, service providers can create service-offers or service-agreements, e.g., issue a digital voucher to a museum if the user is nearby with his rented e-bike.



Change Password
Change your current password

Vehicles Tutorial
See everything you need to get started

Client Version
2.1.0_b11

Terms and Privacy Policy
https://www.coyero.com/terms-of-service
https://www.coyero.com/privacy-policy

Your Privacy Level: C
See your current privacy settings

Unsubscribe from service
Tap to unsubscribe

Figure 5.9: Privacy settings within the mobile client application

5.4 Cloud – Application and Core Servers

The implementation of our servers (e.g., application and core) are based on the Spring Boot framework[7] and are programmed in Java and Kotlin. The servers were realized as Amazon EC2 instances[8] in combination with the AWS Elastic Beanstalk[9] service. Besides, the relational database service[10] was used to store user, service, and token data inside a SQL database. Additionally, the concept of sharding was applied for better scalability. By using these tools, our cloud layer provides elasticity for all deployed services.

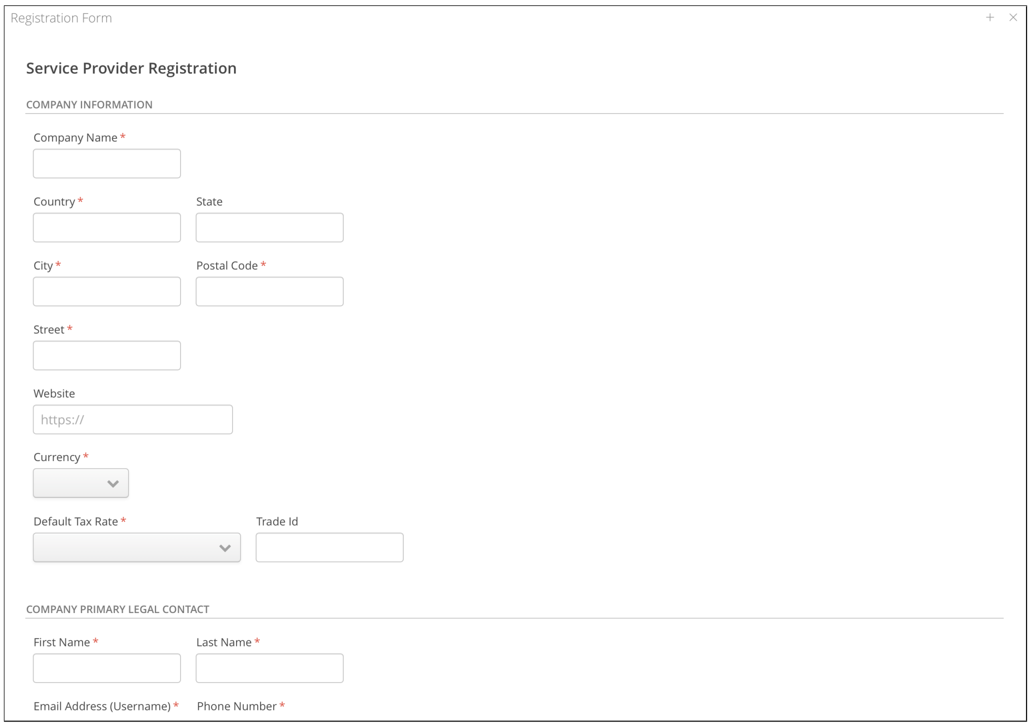
A web-front-end/dashboard was implemented for (i) enabling an easier registration for service providers and their business layers to the core server of the CSE and (ii) providing a management user interface.

The user interface of the web-front-end was developed with Vaadin, an open-source platform for developing web applications[11]. Fig. 5.10 depicts parts of the registration form. After the registration process, service providers are authorized to connect their mobile and server applications to the core layer. Consequently, federated services can be managed via the web-front-end (see Fig. 5.11 as an example). Furthermore, the web-front-end handles agreement specifications defined in subsection 4.4.2. It provides the possibility to specify trigger conditions and resulting actions for automated service creation. There is also the possibility of sending a push notification to the service receiver. Fig. 5.12 shows the action trigger dialog. Additionally, a service provider may also create and forward federated services manually. With these approaches, incentivization-techniques as described in subsection 4.4.4 can be implemented.

The core server provides secured interfaces to devices on a business layer, among these, clients, kiosks, and application servers.

The application servers and the application clients/kiosks access the core server's functionalities through a RESTful HTTPS interface. Basic HTTP authentication is applied to every call using a secret personal API key that is passed as a username within the authorization header. The only exception is the client and kiosk registration process on the core server. In this case, the registration-ticket is used for identifying the device until the corresponding API-key is issued. Regarding token management (see 4.5.1 and 4.6.1), the overall platform relies on Public-Key-Infrastructure (PKI) mechanisms used together with ECDSA as signature algorithm and SHA256 for hashing data.

[7]https://spring.io/projects/spring-boot (last access: 5th September 2020)
[8]https://aws.amazon.com/ec2 (last access: 4th November 2020)
[9]https://aws.amazon.com/elasticbeanstalk/ (last access: 4th November 2020)
[10]https://aws.amazon.com/rds/ (last access: 4th November 2020)
[11]https://vaadin.com (last access: 5th September 2020)



Registration Form

Service Provider Registration

COMPANY INFORMATION

Company Name *

Country * | State

City * | Postal Code *

Street *

Website: https://

Currency *

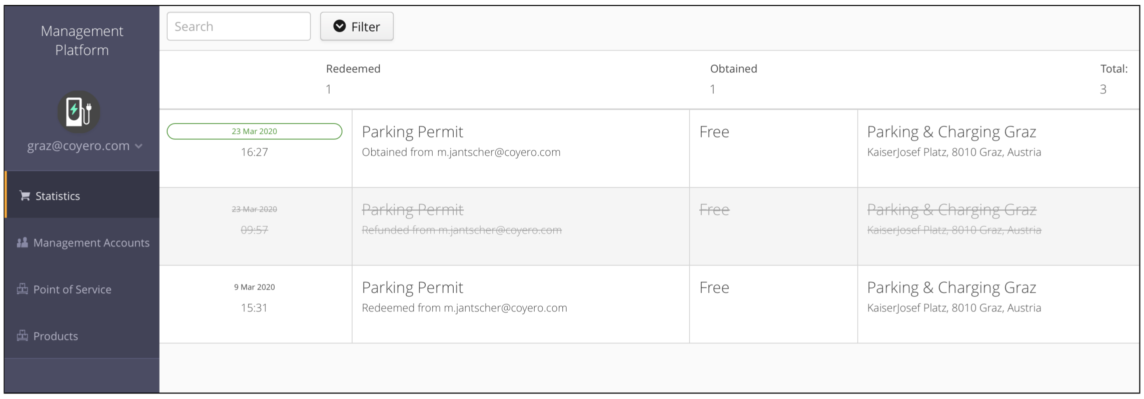
Default Tax Rate * | Trade Id

COMPANY PRIMARY LEGAL CONTACT

First Name * | Last Name *

Email Address (Username) * | Phone Number *

Figure 5.10: Dashboard registration view

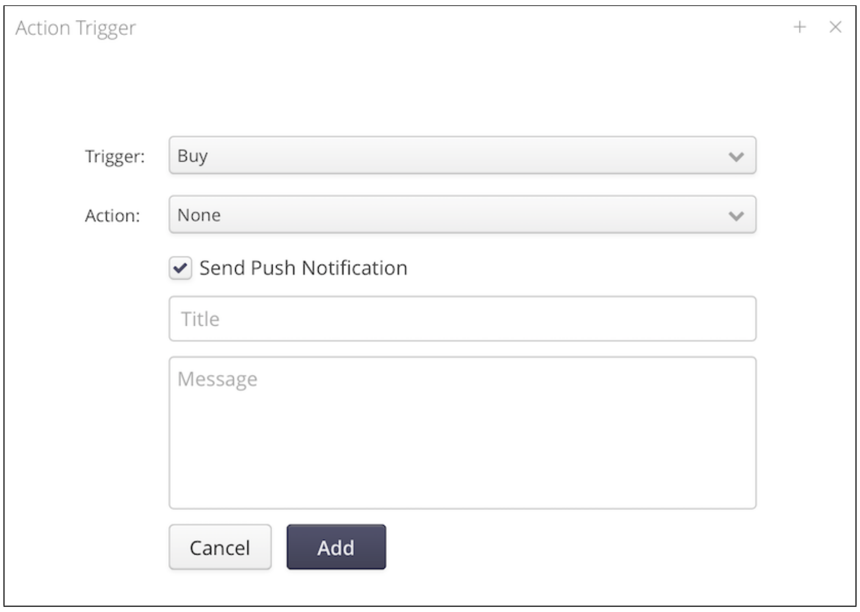


Redeemed	Obtained		Total:
1	1		3

23 Mar 2020 16:27	Parking Permit Obtained from m.jantscher@coyero.com	Free	Parking & Charging Graz KaiserJosef Platz, 8010 Graz, Austria
~~23 Mar 2020 09:57~~	~~Parking Permit Refunded from m.jantscher@coyero.com~~	~~Free~~	~~Parking & Charging Graz KaiserJosef Platz, 8010 Graz, Austria~~
9 Mar 2020 15:31	Parking Permit Redeemed from m.jantscher@coyero.com	Free	Parking & Charging Graz KaiserJosef Platz, 8010 Graz, Austria

Figure 5.11: Dashboard services overview

The tamper-resistant properties of the blockchain, combined with smart contracts, were leveraged and resulted in new software building blocks. They can optionally be used to protect sensitive transactions against fraudulent manipulation as described in subsection 4.4.3.



Trigger: Buy

Action: None

[✓] Send Push Notification

Title

Message

Figure 5.12: Action trigger user interface

For a proof of concept, a local Ethereum **blockchain** was installed on a Linux-based computer, including the PoA Clique consensus algorithm. The Go Ethereum protocol – available as terminal client Geth[12] – was utilized to create all blockchain nodes, accounts, and the genesis block. A boot node running on a static IP address has been initialized. Participating nodes can use it as a reference to find other nodes despite their dynamic IP addresses.

A **smart contract** interface between the Java/Kotlin-based application code and the Ethereum blockchain was established via the web3js library[13]. Smart contracts were programmed in Solidity [14]. For Ethereum, smart contracts are a collection of code (functions) and data (states) stored at specific addresses. Once the network approves a smart-contract-transaction, it is installed on all nodes. In order to embed the contract into the server application, it was compiled using Solidity's solc-compiler. The resulting output files are transferred into Java wrapper classes with the Epirus command line interface[15] and integrated into the application.

[12]https://geth.ethereum.org (last access: 2nd September 2020)
[13]https://web3js.readthedocs.io (last access: 2nd September 2020)
[14]https://solidity.readthedocs.io (last access: 2nd September 2020)
[15]https://docs.epirus.io/sdk/cli/ (last access: 2nd September 2020)

6 Evaluation

In this thesis, a SOA has been designed and implemented for handling service-entitlements of different domains in a secure and privacy-preserving way. Different aspects of the resulting distributed system, mainly consisting of client, kiosk, and server units, are discussed and evaluated in this chapter. Regarding our evaluation process, we have applied a four-step approach. First, we resolve our functional and non-functional requirement definitions within section 6.1 to get insights into the functionality achieved by the final system. Second, we evaluated the performance of the implemented solution through time measurements, focusing on the most critical processes (6.2). Third, based on the findings of our time measurements, a quantitative user acceptance survey has been conducted. This was done to estimate the impact of the CSE on the systems communicating with it (6.3). Fourth, we prepared and supervised usability analyses involving different probands. In this context, the usage-effort to complete specific goals has been assessed (6.4). This final part extends our acceptance section's results by revealing the new system's advantages and limitations compared to the original one.

6.1 Requirements Resolved

Within this section, functional and non-functional requirements are resolved. First, we will discuss our resolved functional requirements (FR), that were listed in section 4.1.

- **FR1 resolved – cross-domain service management:** A service-oriented architecture has been elaborated for issuing, obtaining, canceling, exchanging, and forwarding service-entitlements across client-server systems. Service-entitlements are digital objects that form the basis of resource access mechanisms for client and kiosk devices. They specify which resource can be accessed by whom and under which circumstances. The resulting SOA consists of a common service overlay that, additionally, acts as a service-marketplace where federated service-entitlements, due to their generic nature, can be created and shared across client-server systems of different domains.

- **FR2 resolved – data management:** Service-entitlements are created via interaction with the connected client-server system over a RESTful interface or directly via a web-front-end-view provided by the SOA. With these interfaces, service-entitlements, and generally speaking data of the core layer (e.g., core user or service provider data), can be managed.

- **FR3 resolved – cross-domain rewarding system:** Cross-domain service rewards can be obtained through service offer and service agreement approaches. In the first case, a service provider is authorized to obtain federated service-entitlements on demand and link them to their users. In the latter case, the service offer functionality is extended by providing conditions for issuing service-entitlements automatically. These conditions are based on transactional data or other datasets collected during the usage period of the underlying application.

- **FR4 resolved – ubiquitous resource access and control:** Regarding resource-access, service-entitlements and the core-user-entities are transformed into different tokens responsible for authenticating and authorizing clients to access digitally or physically/locally available resources.
  Local resource access is achieved via the redemption of S-Tokens between clients, embedded kiosks, and the core server. Two approaches are distinguished. First, communication between the three components can be established via a custom redemption protocol based on the local wireless communication protocol BLE. Second, also HTTP redemption is possible. In this case, the core server takes care of the authentication, validation, and redemption of the clients' tokens instead of the embedded kiosk. Furthermore, a check of the client's physical location is conducted via the interpretation of GPS signals and a network lookup.
  Online resource access by clients is enabled via virtual kiosks and the core server through the redemption of O-Tokens. Virtual kiosks are data holders and act as redemption units where the data of interest can be obtained.

- **FR5 resolved – external access:** Firstly, the CSE introduces a RESTful API for enabling client-server systems to communicate to the core cloud. Secondly, the CSE consists of dedicated middleware for easier integration of the overall functionality. It enables local communication between clients and kiosks and online communication to the core server. Thirdly, a web-based dashboard provides a management user interface for service providers.

Next, non-functional requirements (NR) of section 4.2 are resolved.

- **NR1 resolved – interoperability:** From an architectural perspective, the core layer of the CSE is put on top of different business layers that may consist of client, kiosk, and server units. It enters a one-to-many relationship with them that is established via a RESTful interface. The core layer of the CSE is subdivided into two major submodules, the tokenization layer (4.4.1) and the federation layer (4.4.2). The first represents the entry point for external applications and provides access control and abstraction features. It normalizes incoming datasets for more efficient and standardized communication between different business layers. The latter enables mechanisms for sharing and issuing cross-domain service-entitlements. These services can be exchanged by first storing data related to the business layer into an obfuscated mapping table. Second, the mapping table can be used to relate service-entitlements to other business layers.

- **NR2 resolved – context-awareness:** The CSE acts as authorization centre. Different use cases are possible, depending on the available resources. On the one hand, clients can access local resources by (i) directly communicating to locally available kiosk units (4.5.2.1) or (ii) indirectly by first talking to the core cloud that contacts the locally available kiosk in case all checks are passed (4.5.2.2). On the other hand, digital goods can be obtained at virtual kiosks utilizing an OAuth-based approach (4.5.3). This context awareness is possible since our CSE allows granular permission settings for each service. Furthermore, in conjunction with the core cloud, the agreement approach enables condition-dependent creation of service-entitlements based on transactional data. These conditions could also be based on aggregated application-data as long as it is made available to the core layer.

- **NR3 resolved – privacy:** The client's registration process on the core layer foresees a Kerberos-based distributed login procedure that also involves the server of the business layer (4.4.1.1). This procedure ensures that only anonymized user data is passed to the core, from which core-user and core-device objects are derived. The link between real and anonymized information is revocable at any time. Subsequently, all tokens are linked to the core objects, while the business layer remains in complete control of the real user data (4.4.1.2). Furthermore, the topic of privacy labels has been elaborated. In our context, they define what kind of information the user can expect to be shared by using a client application. According to us, a more standardized definition of privacy labels will form the basis for standardization authorities, companies, and users to evaluate the respective application. The definition of standardized privacy labels exceeds the scope of this thesis. However, the concepts discussed in subsection 4.6.1 provide information on privacy-preserving data aggregation and transfer and take example-privacy-labels as inputs. This was realized by introducing a new data structure called P-Token. According to the users' preferences and the mobile application in use, privacy labels can be viewed and adapted, as well as the associated data sharing flow.

- **NR4 resolved – security:** A confidential data collection strategy has been elaborated based on a security-aware design space exploration method. Following the exploration phase's results, an example system was implemented where data collected by embedded devices is sent to a server-based virtual kiosk in an end-to-end secured manner (4.6.2). Regarding resource access, tokens are responsible for authenticating and authorizing client devices. They are not encrypted but obfuscated. This results in faster data transmission and less computational overhead. A token can, in any case, only be redeemed by clients when the authentication checks by

(i) kiosks and (ii) the core server (API-key based authentication procedure) were successful. Regarding data integrity, the tokens discussed in this thesis are based on a PKI with the core server as the root of trust. All tokenized data is signed via ECDSA by the core server and verified in the course of the redemption process between clients, kiosks, and the core server (4.5). Additionally, the integrity of the core layer's transactions can be protected with a blockchain and smart contracts software building block added to the CSE. Due to the blockchain's tamper-resistant properties, data, once uploaded, cannot be changed anymore (4.4.3). The resulting system ensures that service providers control their transactions and have transparency over all transactions involved. Finally, since transaction-based data is stored decentralized on the blockchain network, a single point of failure and data losses are avoided, benefitting data availability.

- **NR5 resolved – scalability:** The number of active server instances is adapted according to the computational needs of the core cloud. Regarding storage, the central core cloud relies on an SQL-based database because of properties such as atomicity, consistency, isolation, and durability. The concept of sharding is applied to provide scalability for growing database scenarios. When a query is received, the database can quickly refer to the desired shard index without scanning the entire database. Additionally, multiple database instances are used to reduce data availability failures. Transactional data is also saved on the distributed ledger.

- **NR6 resolved – configurability:** Service-entitlements are represented as generic objects assignable to clients. They can be revoked but also transformed into tokens for applying different online and offline redemption procedures. Furthermore, specific parameters were integrated to dynamically define authentication and authorization behavior according to the underlying application's requirements.

6.2 Performance Measurements

We measured the execution times of the distributed registration, token creation, and service redemption processes (actions) involving different communication participants: a client device (client), an embedded kiosk device (kiosk$_{\text{embedded}}$), a virtual kiosk unit (kiosk$_{\text{virtual}}$), the application server (server$_{\text{app}}$), and the core server (server$_{\text{core}}$). Table 6.1 depicts the measured timings of our system, including average and standard deviation (Time$_{\text{avg}\pm}$), 90 percentile (Time$_{90\%}$), and maximum (Time$_{\text{max}}$) timing values in milliseconds (ms). Each action was performed 100 times. A Xiaomi Mi Mix 3 smartphone with Android 8.0.0 (client) and a Sony Xperia Z5 with Android 7.1.1 smartphone (kiosk$_{\text{embedded}}$) have been used. All server units were set up as separate Amazon EC2 server instances, as described in section 5.4.

Table 6.1: Average and standard deviation ($\text{Time}_{\text{avg}\pm}$), 90 percentile ($\text{Time}_{90\%}$), and maximum ($\text{Time}_{\text{max}}$) measured timing values in milliseconds (ms) of the distributed registration, token creation, and redemption procedures

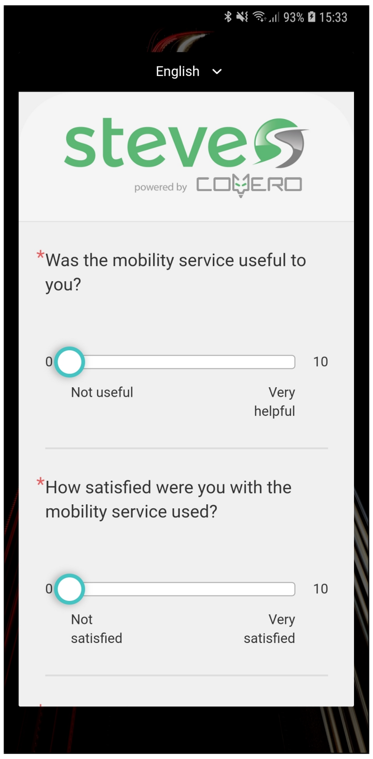
Action	**Devices**	**$\text{Time}_{\text{avg}\pm}$ [ms]**	**$\text{Time}_{90\%}$ [ms]**	**Time_{max} [ms]**
Distributed Registration	$\text{client} \rightleftarrows \text{server}_{\text{app}} \rightleftarrows \text{server}_{\text{core}}$	2670±146	2818	2978
Creation/Obtainment A-Token	$\text{client} \rightleftarrows \text{server}_{\text{app}} \rightleftarrows \text{server}_{\text{core}}$	1097±86	1202	1252
Creation/Obtainment S-Token	$\text{client} \rightleftarrows \text{server}_{\text{app}} \rightleftarrows \text{server}_{\text{core}}$	1233±55	1293	1348
Creation/Obtainment O-Token	$\text{client} \rightleftarrows \text{server}_{\text{core}}$	542±12	557	560
Creation/Obtainment P-Token	$\text{client} \rightleftarrows \text{server}_{\text{core}}$	890±15	908	914
Local BLE Redemption	$\text{client} \rightleftarrows \text{kiosk}_{\text{embedded}} \rightleftarrows \text{server}_{\text{core}}$	1634±105	1726	1749
Local HTTP Redemption	$\text{client} \rightleftarrows \text{server}_{\text{core}} \rightleftarrows \text{kiosk}_{\text{embedded}}$	978±64	1051	1055
Virtual Redemption	$\text{client} \rightleftarrows \text{kiosk}_{\text{virtual}} \rightleftarrows \text{server}_{\text{core}}$	493±29	516	553

As Table 6.1 shows, the client's distributed registration procedure, which also involves the $\text{server}_{\text{app}}$ and the $\text{server}_{\text{core}}$, took less than three seconds on average. A part of this registration involves the creation of an A-Token. The token's issuing procedure includes the ticket creation triggered by the application server. The client redeems the ticket at the core server, where, in further consequence, user and device entities are created. As long as an A-token does not exceed its validity period, this procedure does not have to be repeated. Next, there is the S-Token creation triggered by a client device. Each time a service on the application layer is obtained, an S-Token is issued. The client communicates to the application layer, which contacts the core layer for creating the S-Token. It is linked to the right user and his A-Token and pushed to the corresponding client device. The creation of the O-Token and P-Token are faster due to direct communication between the client and the core server. Furthermore, Table 6.1 shows the local and virtual redemption procedures' timings, also taking into account the authentication steps as described in our design section 4.5. Our SOA does not have any rigid real-time requirements. However, it is still essential that the overall performance does not negatively impact system acceptance and usability due to prolonged reaction times. Summarized, the measured execution timings are in the range of seconds. These measurements suggest that the overall system performance contributes to a seamless system-usage and user experience with its short waiting times. These findings will be substantiated and further discussed in the following section 6.3 via a dedicated acceptance survey.

6.3 Acceptance Evaluation

Another evaluation goal of this thesis was to measure the user acceptance level of the CSE by observing its indirect impact on the client application. A quantitative usability study has been conducted to retrieve the provided services' acceptance level. For this purpose, an in-app questionnaire was created in cooperation with *Carinthia University of Applied*

Sciences[1]. The questionnaire evaluates if the provided services were (i) useful, (ii) satisfying, and (iii) if the users would recommend the utilized services. For the implementational aspects of the survey, we integrated the tool Zoho-Survey[2] as web-view into our mobility app discussed in chapter 5. Fig. 6.1 shows parts of the mobile in-app questionnaire.



*Was the mobility service useful to you?

0 10
Not useful — Very helpful

*How satisfied were you with the mobility service used?

0 10
Not satisfied — Very satisfied

Figure 6.1: In-app questionnaire view

The mobile app was used during the STEVE project's demo phases in Villach (Austria) and partially in Venaria and Turin (Italy). Fig. 6.2 gives an overview of the active Android app users from 1st May 2020 to 8th November 2020[3]. In Italy, 23 surveys were filled out (quadricycle drives) during the first project test phase, while in Austria, a total of 181 in-app questionnaires (104 quadricycle drives, 77 e-bike rides) were completed in the course of the second test phase. During these demo phases, the mobile app asked the user to complete the survey after using a mobility-related service. The questionnaire's goal was to quantify the system's acceptance criteria based on three service-related questions. The questions and their results are discussed on the following pages.

[1]https://www.fh-kaernten.at (last access: 10th September 2020)
[2]https://www.zoho.com/survey (last access: 10th September 2020)
[3]Statistic downloaded from https://play.google.com/console (last access: 30th November 2020)

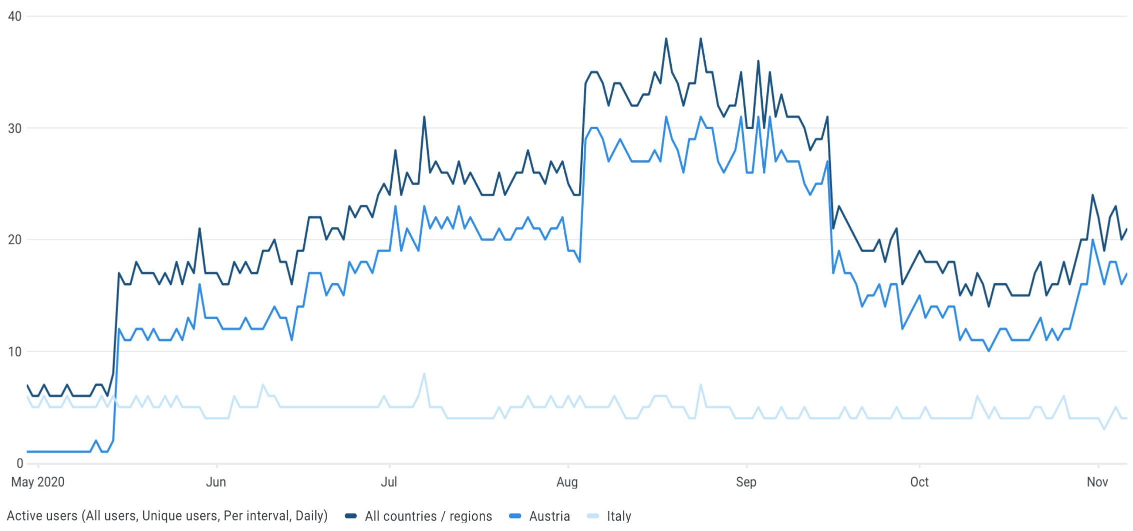


Figure 6.2: Active Android devices running the mobile client app in Austria and Italy from May 2020 to November 2020

Question I – Was the mobility service useful to you?
This question aims to ascertain whether the used systems and their related mobility services benefit the user. The app-user could choose between eleven options, ranging from 0 (not useful) to 10 (very helpful). Fig. 6.3 depicts the results collected in Italy (quadricycles) and Austria (e-bikes and quadricycles) as single stacked bars, giving information on the distribution of the results in %. The legend at the bottom of the figure maps the results to the percentage of users who voted for it. The diagrams show that the implemented service had a very high usefulness rate. For all cities and mobility services, more than 72% of all testers have graded the usefulness level of the used services with 7 points or higher.

Question II – How satisfied were you with the mobility services used?
The idea behind this question was to find out to what extent the app users' expectations correspond to the actual service usage. For this question, the input modalities, as well as the format and type of the responses, were the same as for question I. The resulting diagrams of Fig. 6.4 show the question's results that underline a high satisfaction rate of the implemented services: More than 73% of all testers graded their satisfaction level with 7 points or higher.

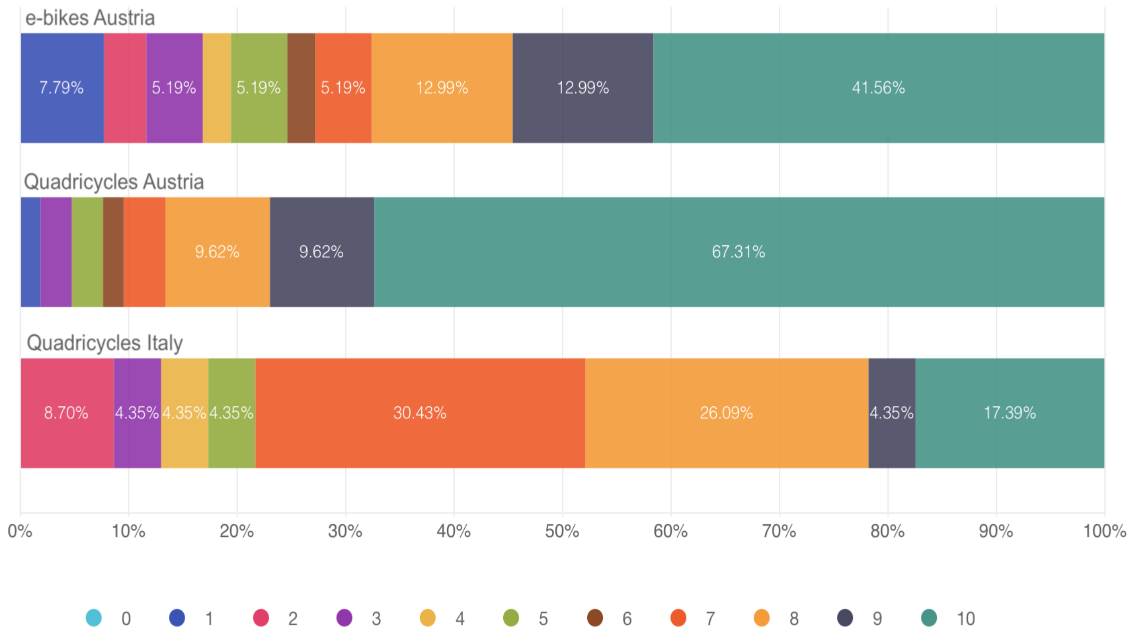


Figure 6.3: Client survey results – distribution of the usefulness level in %

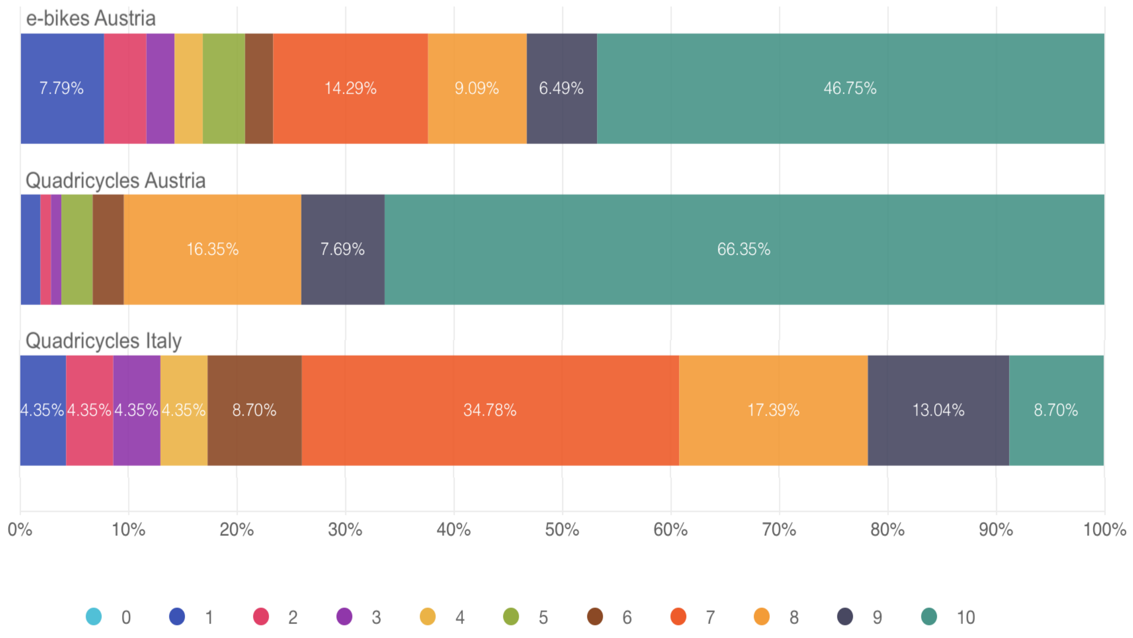


Figure 6.4: Client survey results – distribution of the satisfaction level in %

Question III – Would you recommend the mobility service you used?
This final question depicts the users' willingness to recommend the utilized app and its services, providing feedback about the overall system's acceptance level. In this case, the app-user could choose between two options, "rather yes" or "rather no". Fig. 6.5 summarizes the outcome of all results collected in Austria and Italy via pie charts. The three diagrams show that most users (87,01% - 91,30%) recommend the utilized system and its offered services.

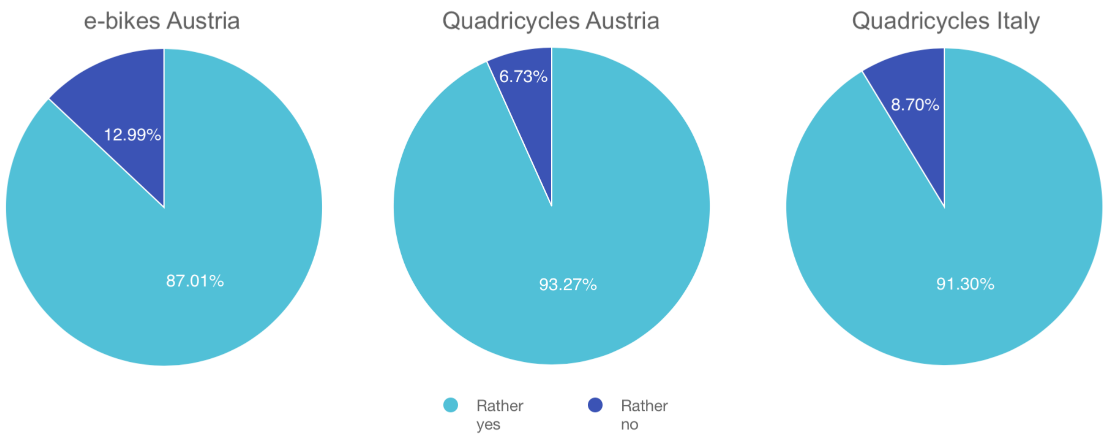


Figure 6.5: Client survey results – distribution of the recommendation readiness in %

Our quantitative user acceptance study revealed a high acceptance rate of the used system. This leads us to the conclusion that our implementation and the associated low waiting times illustrated in the previous section 6.2 had a positive impact on the overall system's acceptance level.

6.4 Usability Assessment

Usability testing cycles have been conducted to assess the effects of the CSE's implemented features on the underlying systems. By doing so, we were also able to deduce the effort associated with reaching specific goals while using our system. In concrete terms, those systems acting as user interfaces, such as the mobile app and the web-front-end, have been assessed by a diverse set of people.

Our probands were selected based on the specification of our *target group*. According to our definition, the target group comprises technophile people with a particular interest in trying out new technologies, such as new smartphone apps, but are unwilling to use inconvenient software that requires a lot of effort or engagement. According to our target group's requirements, we chose ten probands to evaluate our approaches, including employees from *CISC Semiconductor GmbH*, *Graz University of Technology*, and people with a non-technical background. The evaluation criteria were the quality of the implementation regarding application-handling and the corresponding usage-effort associated with completing specific goals while using the provided applications. The system's results elaborated in the course of this thesis were compared with the distributed COYERO framework and its applications as they were back in 2017.

The comparison between the new and old applications was not a trivial matter since several software-aspects and service handling processes had been changed and enhanced in this thesis. Therefore, the following evaluation does not claim to provide a holistic comparison of all designed and implemented features. Nevertheless, we were able to get valuable indications of the new system's benefits. The involvement of users was thoroughly investigated and validated for the following parts of the mobile application.

1. **Service obtainment and cancelation.** Users were asked to obtain two different mobility service-entitlements and to cancel one of these subsequently. Services for handling resources of different service providers can be accessed within one application by using the new system. By contrast, different accounts have to be created in case of the previous system, and different apps to be used to complete this task. Due to the additional federation layer of our new SOA faster cross-domain service obtainments can now be achieved. While users obtained one service-entitlement just as fast as in the original version, our updated core layer showed its advantages in the case of multiple service obtainments, enabling the average user to reach the goal *62% faster* compared to the prior system.

2. **Service sharing.** The service sharing and forwarding functionality has been embedded as a new functionality into the core server of the SOA. To achieve the same functionality, the old system has to trigger the service-entitlement cancellation logic, followed by a new service-entitlement obtainment request. In the new system, this can be done significantly faster due to our cross-domain service mapping. Summarized, the average user was *185% faster* by using the new system for forwarding a service-entitlement.

3. **Service redemption.** The system of 2017 was specialized on local BLE redemption. By comparing only this redemption method with our updated BLE redemption procedure, users were on average *14% slower* with the new system. This slowdown results from the fact that the new redemption building block became context-aware and now supports multiple modes, increasing the overall complexity and thus also slightly increasing the initial waiting times before the right redemption method is initiated. Next to the local BLE redemption, the new redemption building block provides additional local HTTP and OAuth 2.0 based redemption procedures.

Consequently, our probands were asked to try out the system from the perspective of a service provider. Again, the users tried to solve a few tasks using the prior and the new systems. For both usage scenarios, the probands underwent a tutorial first.

1. **Service provider registration.** A dedicated dashboard view connected to the core system of the CSE can now be used to facilitate the registration process of a new service provider and its system. While merchant-related data must still be entered manually, the remaining registration is processed automatically regarding the creation of the business layer API key or the REST-endpoint setup, for instance. In the original platform, these tasks involve several manual steps in order to achieve the same goal. Therefore, the average user was *379% faster* with the new system compared to the prior one.

2. **Gamified service-usage.** The goal of these test runs was to issue a rewarding service for a specific user. Since an automated service creation is not part of the older system, several manual adaptions have to be done. On average, users were *245% faster* with the new system since services could be generated in the form of service-offers directly via the dashboard. Additionally, the new system provides the possibility to set automated service-creation triggers for enabling entitlement-issuing when predefined conditions are met. Furthermore, it should be noted that the original system is only capable of addressing users of one business layer at a time, providing no cross-domain interrelationship.

3. **Service Management.** The specific task was to create, edit, and delete a specific service-entitlement. Since the user interface has been improved, the average user was *85% faster* when using the new system.

Despite a higher level of complexity, the new SOA was significantly faster on average compared to the prior system. Our results underline the high usability aspect of the developed system.

7 Conclusion

This chapter concludes this doctoral thesis by briefly summarizing all scientific contributions (7.1) and emphasizing potential future work and research (7.2).

7.1 Summary

Everyday life is becoming ever more connected to the digital world. Information services and applications run on an extensive set of different systems and constantly interact with people and their environment.

This thesis presented the design, implementation, and evaluation of a SOA with a particular focus on secure and privacy-preserving service-exchange and service-access methodologies. In our case, services were represented as generic digital entitlement objects used to access physical or digital resources (products, data, etc.). We introduced a service interlay for client-server systems – the CSE – that we subdivided into a tokenization layer for access control and abstraction as well as a federation layer for exchanging cross-domain service-entitlements. Third-party service providers can connect their client-server systems via a RESTful API and dedicated middleware and upload their services to a ubiquitous ecosystem, where service consumers are allowed to acquire, forward, and redeem service-entitlements. Regarding context-aware resource access and the overall redemption procedure, different tokens were proposed. While the A-Token enables mutual offline authentication, the S-Token is responsible for authorizing a client to access a physically available resource. Complementary to these, the O-Token is used for getting access to distributed online-resources. Finally, the P-Token has been introduced for managing the client user's privacy settings. The thesis also considered privacy-preserving data aggregation and sharing and how the data is processed and postulated throughout the overall platform. Different privacy labels can be specified by the CSE to cope with diverse privacy requirements. Finally, a security-design study for end-to-end secured communication completed the value chain of online data collection and redemption.

The evaluation of our SOA was subdivided into four parts. First, all requirements explained at the beginning of this thesis were resolved. Second, the timings of the most critical processes were measured. Third, the acceptance evaluation results, which have been conducted based on an in-app survey, were discussed. Fourth, the overall usability and the associated usage-effort have been evaluated compared to the prior system via usability testing cycles with different probands. Our findings revealed low waiting times and high acceptance and usability levels of our developed system.

7.2 Future Work

This doctoral thesis contributes to state-of-the-art SOAs, providing promising solutions for distributed service-access procedures. However, it does not claim to be exhaustive or provide a holistic view of smart city management platforms. The following paragraphs show the current limitations of the overall platform and propose ideas concerning future work.

Related to context awareness, many platforms and their applications adapt their behavior according to external conditions to achieve fault-tolerance, choose the most suitable program execution flow, or improve efficiency regarding data processing and storage, for instance. As discussed, our platform provides different context-aware service access and redemption methodologies. BLE, as elaborated within subsection 4.5.2 has been one of the main actors for enabling local authentication and authorization procedures. Our communication protocol was designed in a generic way to be used independently of the underlying communication protocol. Nevertheless, full integration of other communication techniques, such as Ultra-Wideband (UWB) or NFC, would extend our set of service access and redemption procedures. Also, privacy-preserving data handling was one of the topics of this thesis. The elaborated CSE does not directly utilize user-specific information of the connected business layers and focuses on a distributed, anonymized service mapping (see 4.4.2). We also proposed aggregation techniques for confidential data as part of section 4.6, without explicitly linking it to the business-layer entities. Nevertheless, additional anonymization techniques, as described in our related work section, could be considered for the storing process of the collected data. Complementary to privacy-preserving data collection, additional techniques related to the efficient processing of massive amounts of data could be taken into account. We already implemented methods to store and interpret data in a scalable manner. However, since the data collection techniques discussed in this thesis provide diverse datasets, semantic data interpretation for better extracting the meaning of information would be advantageous and could be included in the overall platform. Last but not least, smart city platforms often provide a set of tools for granular development, maintenance, and management of custom applications. Therefore, a future add-on for our SOA could provide service providers with a sandboxed development environment for a more granular specification of service-agreements and their service and condition handling.

8 Publications

In the course of this doctoral thesis, scientific essays were published in several highly rated domain-specific conferences, book chapters, and workshops. Most notably in chronological order:

A. Alexander Rech, Markus Pistauer, and Christian Steger. “Increasing Interoperability Between Heterogeneous Smart City Applications.” In: *11th International Conference on Internet and Distributed Computing Systems*. Tokyo: Springer International Publishing, 2018, pp. 64–74. ISBN: 9783030027377. DOI: 10.1007/978-3-030-02738-4_6. URL: https://www.springerprofessional.de/en/increasing-interoperability-between-heterogeneous-smart-city-app/16204508

B. Alexander Rech, Markus Pistauer, and Christian Steger. “A Novel Embedded Platform for Secure and Privacy-Concerned Cross-Domain Service Access.” In: *IEEE Intelligent Vehicles Symposium, Proceedings*. Vol. 2019-June. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1961–1967. ISBN: 9781728105604. DOI: 10.1109/IVS.2019.8814123

C. Alexander Rech, Christian Steger, and Markus Pistauer. “A Decentralized Service-Platform Towards Cross-Domain Entitlement Handling.” In: *2nd IEEE Conference on Blockchain*. IEEE, 2019. ISBN: 9781728146935. DOI: 10.1109/Blockchain.2019.00069

D. Alexander Rech, Christian Steger, and Markus Pistauer. “Gamifying Connected Services Patterns.” In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, 2019. ISBN: 9781450362061. DOI: 10.1145/3361149.3361163

E. Alexander Rech, Markus Pistauer, and Christian Steger. “A Distributed Framework Towards Local and Online Service Access.” In: *24th IEEE Conference on Emerging Technologies and Factory Automation - Workshop on Trustable Wirelessly Connected Industrial IoT* (2019), pp. 1715–1721. DOI: 10.1109/ETFA.2019.8869305

F. Alexander Rech, Lukas Gressl, Christian Seifert, Christian Steger, and Andreas Sinnhofer. “Multi-Layered IoT System Design Towards End-to-End Secure Communication.” In: *46th Annual Conference of the IEEE Industrial Electronics Society* (2020), pp. 2213–2220. DOI: 10.1109/IECON43393.2020.9254556

Additionally, this thesis is supplemented by the following peer-reviewed papers:

G. Lukas Gressl, Alexander Rech, Christian Steger, Andreas Sinnhofer, and Ralph Weissnegger. "Security Based Design Space Exploration for CPS." in: *Proceedings of the ACM Symposium on Applied Computing* I (2020), pp. 593–595. DOI: 10.1145/3341105.3374058

H. Lukas Gressl, Alexander Rech, Christian Steger, Andreas Sinnhofer, and Ralph Weissnegger. "A Design Exploration Framework for Secure IoT-Systems." In: *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment, Cyber SA 2020* (2020). DOI: 10.1109/CyberSA49311.2020.9139631

Fig. 8.1 depicts all publications and maps them to the corresponding design sections discussed in this thesis.

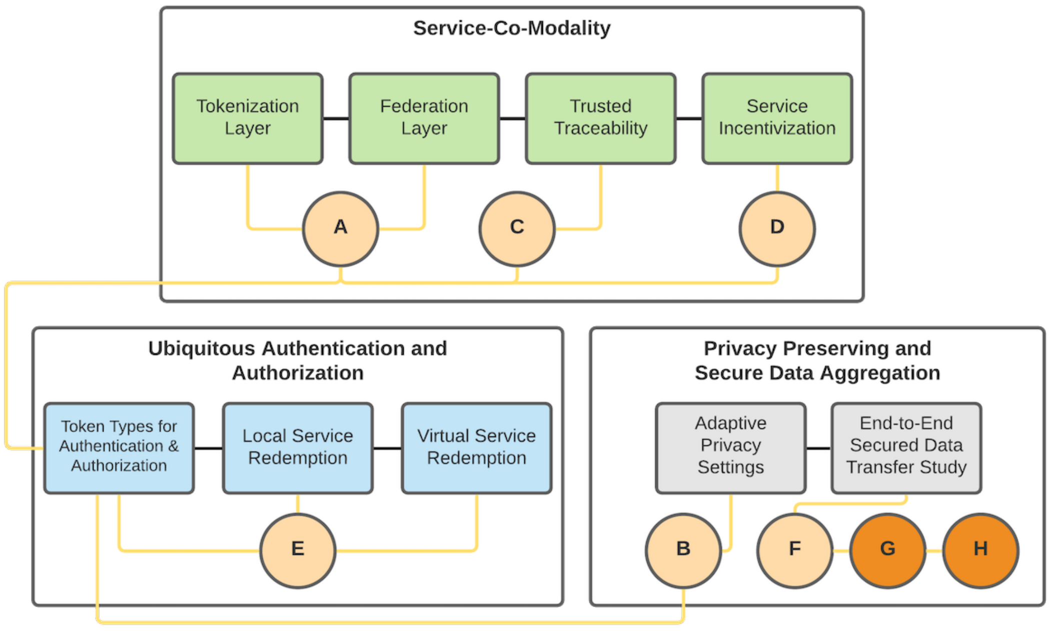

Service-Co-Modality
Tokenization Layer
Federation Layer
Trusted Traceability
Service Incentivization
A
C
D
Ubiquitous Authentication and Authorization
Token Types for Authentication & Authorization
Local Service Redemption
Virtual Service Redemption
E
Privacy Preserving and Secure Data Aggregation
Adaptive Privacy Settings
End-to-End Secured Data Transfer Study
B
F
G
H

Figure 8.1: Overview of contributions and related publications of this thesis

Increasing Interoperability Between Heterogeneous Smart City Applications

Alexander Rech[1](✉), Markus Pistauer[1], and Christian Steger[2]

[1] CISC Semiconductor GmbH, 9020 Klagenfurt, Austria
{a.rech,m.pistauer}@cisc.at
[2] Graz University of Technology, 8010 Graz, Austria
steger@tugraz.at

Abstract. Due to the increasing need for networked systems we can observe a rapid advance of IT-solutions in various sectors. However, most of the developed systems are custom-tailored solutions for specific problems and application areas, leaving us with a set of diverse frameworks. The resulting jungle of heterogeneous systems makes it difficult to find common interfaces for interconnecting the underlying businesses with each other, especially in regard to Smart City concepts. We envision a new paradigm shift towards "Smart City as a service" fueled by increased interoperability between different services with an additional emphasis on privacy-preserving data processing. This would contribute to a new level of connectivity between the environment, service providers, and people, facilitating our daily activities and enhancing the level of trust of the users. In order to achieve interoperability in the context of smart, connected cities, we propose the design of a generic, platform-independent novel architecture for interconnecting heterogeneous systems, their services, and user pools.

Keywords: Interoperability · Connected services · Data privacy Smart City

1 Introduction

The need for stronger interrelationships between different services is constantly increasing. However, when interoperability between independent systems has to be established, we face several issues. One major problem is that systems tend to be tailored to a specific application area, so boundaries are often rigid and inflexible and only users of one specific application are addressed. Second, combining two or more systems can be a considerable challenge and, if integration is indeed attempted, integration methodologies can be very time- and cost-consuming which in turn results in longer time-to-market cycles. Third, when different parties have to share user-related datasets, special attention has to be paid to the correct use and processing of the data, especially in view of the increasingly stringent data protection laws.

Y. Xiang et al. (Eds.): IDCS 2018, LNCS 11226, pp. 64–74, 2018.
https://doi.org/10.1007/978-3-030-02738-4_6

A Smart City is a place where many different information systems come together. At first glance, some of these systems are already embedded into our daily lives. Smart services such as managing parking facilities, renting e-bikes or placing orders via mobile applications are already feasible in many cities. However, closely examined there is poor or almost no interaction between services of different kinds, as for the above mentioned, for instance. Therefore, in order to cope with the increasing amount of heterogeneous services in a Smart City, pervasive computing technologies should offer collaboration methodologies among businesses and their users to enable benefits for all participants involved. This, in further consequence, would contribute to a new level of connected services within cities, enabling businesses to collaborate more easily with each other. Due to this cooperation, more users could be addressed, resulting in additional advantages in terms of higher revenue. In contrast, users are provided with easier access to different, independent services. More specifically, users may save time and profit from additional offers, which in the end facilitates their daily activities.

We call into question the current situation where businesses are limited to a predefined scope regarding their services and users. We want to increase collaboration between independent entities through a federated solution, so that multiple parties do not have to completely adapt to each other's requirements in order to achieve mutual benefits. In further consequence, to accomplish a paradigm shift towards "Smart City as a service" we are working on a generic, platform-independent architecture that gives businesses and their applications access to a predefined set of services of other participating service providers. To avoid high integration costs, the functionality to enable this federated solution shall be offered as additional software layer to pre-existing systems. Additionally, since data needs to be shared across multiple parties, the privacy of the users shall be protected by abstracting their real identity.

This paper is organized as follows. Section 2 describes related work while Sect. 3 discusses design choices of our architecture and gives an overview on its components. Last but not least, Sect. 4 summarizes our ideas and offers suggestions regarding future work.

2 Related Work

Everyday life is becoming evermore connected to the digital world. Information services and applications run on an extensive set of different systems and are in constant interaction with people and their environment. Federated concepts are essential for increased system and business relationships. In order for these systems to communicate seamlessly with each other and to overcome the heterogeneity between them, interoperability is becoming increasingly important. Especially due to the increasing need for networked services from different businesses, interoperability is required ever more. According to IEEE the term interoperability is the "ability of two or more systems or components to exchange information and to use the information that has been exchanged" [1]. A more generic definition of interoperability is given by [2]. It is the "ability of things to interact for a specific purpose, once their differences have been overcome". This definition tries to expand the concept where a system is simply composed of single components, by saying that a system may also consist of multiple diverse autonomous subsystems. Only if cooperation between all systems is ensured and solutions are made to eliminate discrepancy between all components, can interoperability be achieved within the resulting system of systems [3].

In this context, a Smart City can be viewed as a larger system overarching diverse subsystems, which on their own act as autonomous components that should work together seamlessly. New concepts for information sharing among different services as well as for enhancing collaboration between Smart City services are being analyzed and elaborated [4,5].

The connection of different services is also a topic that is being elaborated by the Horizon 2020 STEVE project. The focus of the project is to implement and test a human-centric approach to electro-Mobility-as-a-Service (eMaaS), by connecting different e-vehicle solutions and "gamified" services for enhancing users' awareness, engagement and vehicle energy efficiency [6].

The idea of encapsulating an organization's functionality within an appropriate interface, advertising it as services, and trying to connect it with other similar services is not new [7]. In this sense, semantic web service-paradigms have been trying to enhance automated discovery, access, combination, and management of web services for years. Both academia and industry are researching ways to provide machine understandable representations of services, their properties, capabilities, and behavior [8,9].

Furthermore, interoperable networking is becoming increasingly important to concatenate different backends. There already exist a few approaches on how to combine heterogeneous cloud systems for sharing different datasets across geographically distributed resources and for better cooperation between different services [10–12].

Especially in the electronic marketplace area, the notion of interoperability is often closely associated with the term integration. From the perspective of service providers, pure integration methodologies may be advantageous: If businesses merge into clusters they often have a decreased management overhead with less setup and maintenance costs for instance. However, the downside is that such integration technologies and frameworks are often associated with high adaption costs, especially for businesses with rigid standards [13].

In our opinion, we do not only need new concepts for merging multiple systems and their services, which often proves to be difficult, costly or time consuming. More importantly, we need to emphasize on technologies that give multiple independent businesses the possibility and freedom to interconnect each others' services with negligible adaption costs for maximized mutual benefit. A collaboration of different service providers also means that each company may address more users. In this context, the more companies work together and exchange data, the more important data privacy protection rules become, especially in light of increasingly stringent data protection rules like the European General Data Protection Regulation (GDPR) [14]. Therefore, increased interoperability should not only offer advantages such as easier cooperation between multiple independent businesses, but also adequate data protection for end users.

3 Design of Architecture

This section is about the design of our proposed architecture. First, requirements are discussed and an overview on our architecture is given in 3.1. Afterwards, the architecture's business layer components and the core layer components will be described more detailed in Sects. 3.2 and 3.3.

3.1 Architectural Overview

Our proposed architectures addresses the following topics:

– *Interoperability.* Collaboration between different independent businesses and service providers shall be improved. Nowadays we do not have much cooperation between independent businesses, e.g. parking provider with a restaurant, or a museum with a car-rental service. No matter which product or which service a company offers, it would be beneficial if companies were able to utilize services from other businesses for increased collaboration between them as well as for addressing more users, and for offering more services to them. An example use case would be the following: A user authenticates himself at a parking gate using his smartphone for getting access to the shopping mall. As soon as he enters the garage, he receives a new service in form of a digital restaurant voucher, pushed onto his phone. In this example, two independent businesses, a garage provider and a restaurant, collaborate by sharing their users and services. The reverse case is also possible: a user getting a parking voucher as reward for eating in a restaurant.
– *Integration cost.* It is time consuming to define agreements between companies and to elaborate common interfaces. Therefore, the platform shall provide ways to facilitate this agreement process. Pre-defined interfaces shall be provided for faster integration and shorter time to market cycles.
– *Data Privacy.* When data, especially user-related data needs to be passed between different parties for enabling collaboration, privacy aspects become more important. In this context, only anonymized user data shall be passed between companies and the link between real and anonymized data shall be revocable at any time in order to be compliant with data protection rules.

With these requirements in mind we now present and discuss the structure of our architecture, which will be described more precisely in 3.2 and 3.3. Figure 1 provides a high-level overview. The overall architecture consists of at least one business layer as well as a core layer, both of which are further subdivided into several components.

Each business layer is managed by a company and contains service and user related datasets. Generally speaking, it consists of client and vendor applications as well as of a corresponding server unit that offers application specific functionality. While the client application is adapted to the user's needs, such as giving an overview on current services, the purpose of the vendor application is to provide and manage these services. Both applications operate in constant interaction with the business layer which provides them with application specific functionality and keeps the data synchronized across all devices.

Regarding the core layer's position in the proposed architecture, it is placed on top of a business layer. It can be reached by the components of the business layer via a RESTful interface. The idea behind the concept of the core layer is to provide a common trusted layer responsible for abstracting products and services from various business layers in order to establish an interrelationship between multiple heterogeneous businesses. Even if a company already utilizes its own business layer and corresponding applications, it should be feasible to communicate to the core layer in order to create new means for interacting with other service providers or rather other business layers. As interrelationships between different businesses involve a lot of time and effort for all participating businesses, we see the need for enhancing this integration and cooperation process in regard to speed and costs. Therefore, the entities of the business layer shall be extended by dedicated software libraries defining how to interact with the core and its subcomponents: the Tokenization and the Federation layer.

In order to interconnect different platforms and to anonymize datasets coming from different business applications, we introduce the Tokenization layer. Its task is to abstract application specific data of the business layer such as product and user data for bringing down all datasets to a common denominator, and increasing the level of privacy of the end users. After this abstraction process a mapping between the anonymized and the real data is created and stored at business side. This means that only the business layer which triggered the tokenization process of the data in the first place, may access the derived data from the core layer. Last but not least, there is an additional module called Federation layer. It is responsible for creating and managing services between different vendors. Furthermore, it is able to arrange trusted agreements between independent parties. This makes it possible to automate the issuing process of services as soon as predefined conditions are met.

3.2 Business Layer Components

The business layer provides users with features based on the respective application area of the system and the corresponding service provider. In summary, it consists of the three key components described below. Companies who already utilize their own business layer, may establish a link to the core via the RESTful interface.

Client Application. The mobile client app is used by customers of a specific vendor, shop or company. Together with application specific features it offers a user interface for acquiring services, e.g. parking app for booking parking tickets, restaurant app for ordering food. Regarding the redemption process of

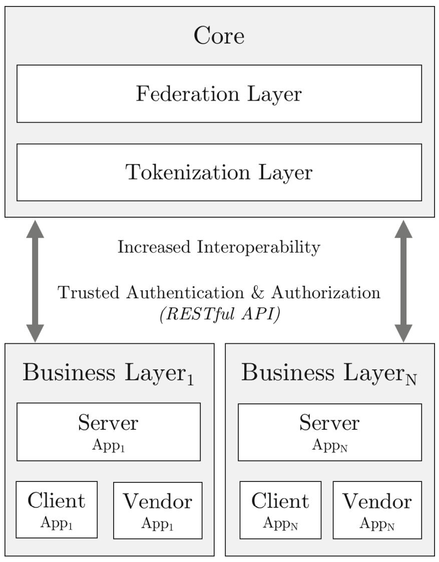


Fig. 1. Architectural overview

the acquired service we distinguish between two cases, depending on the specific application area and the service type in question. On one hand, the client application may redeem the service by directly communicating to the vendor application. In a parking use case, the drivers may redeem their parking voucher directly at the embedded vendor application of the parking gate. On the other hand, services can also be redeemed directly via an online interface without additional interaction of other devices (online food delivery).

Vendor Application. The vendor-side application functions as front-end for service providers to manage both their user pools as well as their services. Depending on the use case, the vendor application decides on how the services should be redeemed, e.g. online via server requests or by first communicating locally to a specific device. Just as with the client application, the vendor application works closely with the application server.

Application Server. The application server communicates with client and vendor devices and keeps them synchronized. On one hand, it is responsible for managing the users of the client application. On the other hand, it offers the possibility to manage shop- and service-relevant data on vendor side. If a business decides to cooperate with other service providers, interaction with the core layer, which is explained in the following, is required.

3.3 Core Layer Components

The goal of this layer is to seamlessly interconnect different services, thus contributing to a new level of interrelationships in the context of Smart City applications. The idea is to attach it on top of an existing system as an additional trusted ubiquitous layer that can be reached via a predefined set of REST calls. It is subdivided into two parts which will be described in the following. Figure 2 gives an overview on the most important components of the core layer, while Fig. 3 describes the high-level-interrelationship between the most important datasets of the different layers.

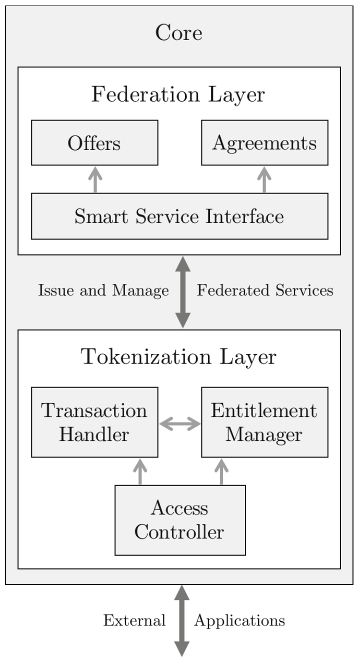


Fig. 2. Overview of core layer subdivided into Tokenization and Federation layer

Tokenization Layer. This layer provides the entry point for external applications. In the first place, the business layer of a system may use it for enabling cooperation with other applications. Since each company usually has its own implementation, this layer is mainly there to abstract the datasets coming from different businesses. This abstraction process converts the information received from the application layer into a mapping of several IDs that uniquely identify businesses, their services and users. Additionally, the Tokenization layer keeps log of transactions via the *Transaction Handler*. Each transaction refers to the participating users on client and vendor side and the acquired or redeemed services in question. According to the specified use case, the Tokenization layer may also communicate with the Federation layer to issue new services in form of entitlements or goods. The Tokenization layer distinguishes between two different token types:

- *AccessToken.* The user- and device related AccessTokens are managed by the *Access Controller*. They are issued in the course of a distributed Kerberos-based authentication procedure and signed by the core layer. Furthermore, they are responsible for providing user devices with authentication when communicating with the core layer and for countering eavesdropping and replay attacks.
  The real user data stored on the business layer is never forwarded to the core layer, but converted into anonymous IDs which are linked with the AccessTokens. Only the business in charge of the creation of the user account knows the user data, thus anonymizing the user's identity for all other participating businesses.
- *ServiceToken.* All business and service relevant data is transformed into a signed ServiceToken by the *Entitlement Manager*. Just as for the user data, also in case of service data, unique IDs are derived, identifying the service and the company. Each ServiceToken belongs to one AccessToken, authorizing a specific user to utilize a certain service. Furthermore, it can be determined which entity purchased or redeemed which services, and how many of them, via the *Transaction Handler*.

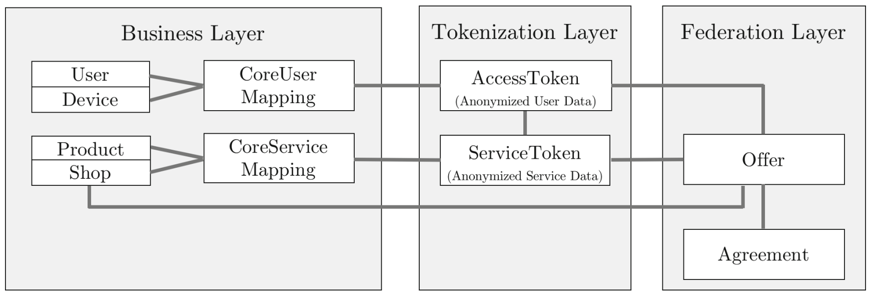


Fig. 3. Overview of interrelationship between datasets of business, Tokenization and Federation layers

Federation Layer. The Federation layer is an interface for interconnecting services from different businesses. It can be considered as federated service market place where businesses can place their products and services for special conditions or purchase services from other participants. Consequently, this cooperation enables companies to offer a wider range of services. Combined with gamification methodologies, goods and services of other companies can be forwarded as rewards to customers, enhancing both user satisfaction as well as revenue for the participating service providers. The Federation layer operates in interaction with the Tokenization layer and utilizes the tokenized data, including AccessTokens and ServiceTokens, to address services and users. In contrast to the actual user data which is only stored on business layer side, some product and company relevant data has to be forwarded from the Business layer to the Federation layer to be utilized by other interested businesses. In this sense, obligatory fields such as the name of the company and its address, have to be submitted together with optional key value pairs identifying the corresponding company's products and services. In summary, the Federation layer handles the services under discussion in two different ways:

– *Offer.* A service provider utilizing the Federated layer is authorized to publish specific services in this domain, stored as offers. Other providers are able to search for offers and acquire them. This layer functions as a market place where products and services can be found according to specific key words such as name, price, or category, for instance. Providers may acquire them and a new ServiceToken would be generated inside the Tokenization Layer. In further consequence, the link between the ServiceToken and the user's AccessToken will be established, enabling the user to utilize the service. In this sense, loyal customers can be rewarded with goods and services of other companies, e.g. a user who rented an e-bike gets an additional voucher for visiting a museum.
– *Agreement.* The idea behind agreements is that there should be the possibility to issue entitlements, aka ServiceTokens in an automated way. As soon as businesses subscribe to an offer, it becomes an agreement and is extended by specific datasets identifying the interested party as well as a condition that specifies under which circumstances the execution of the agreement should be triggered. A condition specifies how many goods or services of a company need to be acquired by a user before a reward is issued, or which specific service needs to be redeemed in order to trigger a reward, for instance. In any case, every time a customer acquires new services or redeems them, new transaction history entries will be generated inside the *Transaction Handler* of the Tokenization layer. Subsequently, it will be determined if the predefined condition is met through interaction between the Tokenization and the Federation layer. If so, the good or service specified inside the agreement block, will be acquired by the subscribed vendor and forwarded to the customer who triggered the current transaction. Furthermore, customers and vendors can keep track of the service-completion-state by utilizing the entries of the *Transaction Handler* as reference. An additional advantage for customers is that, even if they do not actively track the completion state, they can still receive rewards in an automated way.

4 Conclusion and Future Work

On account of poor integration methodologies, high integration costs, and rigid system interfaces, the widespread use of ubiquitous Smart City applications is still in its inception. In order to overcome these problems we are currently working on a generic software platform for increasing the cooperation between participating heterogeneous systems, service providers and user pools. In this paper we gave an overview on our proposed architecture, mainly subdivided into a business and a core layer. The core layer can be seen as additional layer that can be put on top of existing networked systems and enables them to exchange their services. Furthermore, in regard to user-data privacy, especially in view of strict data protection laws, we introduced a Tokenization Layer which task it is to reconcile user- and business-related datasets coming from arbitrary application servers. The user data itself is masqueraded and saved in a privacy-preserving way. In contrast, the Federation Layer provides means for service-based collaboration between independent companies.

The design of the architecture presented in this article is a starting point for us. Parts of the framework discussed in this paper will be evaluated within the European Union's Horizon 2020 project STEVE. Future work will concentrate on a research on semantic service oriented architectures to extend the interfaces between the business and core layer in a more dynamic way. Additionally, we are currently investigating on how to increase the level of trust between different entities. Blockchain in combination with smart contracts could therefore be suitable for our plans.

Acknowledgments. This work is supported by CISC Semiconductor and Graz University of Technology. A part of this work has been performed in the European Union's Horizon 2020 STEVE project under grant agreement No. 769944.

References

1. IEEE standard glossary of software engineering terminology. IEEE Std 610.12-1990, pp. 1–84 (1990)
2. Motta, R.C., De Oliveira, K.M., Travassos, G.H.: Rethinking interoperability in contemporary software systems. In: IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS), pp. 9–15 (2017)
3. Boardman, J., Sauser, B.: System of systems - the meaning of of. In: IEEE/SMC International Conference on System of Systems Engineering, pp. 6–11 (2006)
4. Cledou, G.: A virtual factory for smart city service integration. In: Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance, pp. 536–539 (2014)
5. Antonić, A., Marjanović, M., Žarko, I.P.: Modeling aggregate input load of interoperable smart city services. In: Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems, pp. 34–43 (2017)
6. The European Commission: Smart-taylored L-category electric vehicle demonstration in hetherogeneous urban use-cases (2017). http://www.steve-project.eu
7. Cardoso, J., Sheth, A.: Introduction to semantic web services and web process composition. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 1–13. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30581-1_1

8. Benatallah, B., Nezhad, H.R.M.: Interoperability in semantic web services. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 22–25. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30581-1_3
9. Cardoso, J., Miller, J., Su, J., Pollock, J.: Academic and industrial research: do their approaches differ in adding semantics to web services? In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 14–21. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30581-1_2
10. Bavier, A., et al.: GENIcloud and transcloud: towards a standard interface for cloud federates. In: Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, pp. 13–18 (2012)
11. Zou, M., et al.: Collaborative marketplaces for eScience: a medical imaging use case. In: International Conference on Collaboration Technologies and Systems (CTS), pp. 500–506 (2014)
12. Akolkar, R., et al.: Towards cloud services marketplaces. In: 8th International Conference on Network and Service Management (CNSM) and Workshop on Systems Virtualiztion Management (SVM), pp. 179–183 (2012)
13. Guo, J.: Business interoperability on E-marketplace. In: Xu, L.D., Tjoa, A.M., Chaudhry, S.S. (eds.) Research and Practical Issues of Enterprise Information Systems II. ITIFIP, vol. 254, pp. 257–267. Springer, Boston, MA (2007). https://doi.org/10.1007/978-0-387-75902-9_26
14. European Parliament and European Council: Regulation 2016/679 on the protection of natural persons with regard to the processing of personal data. Official J. Eur. Union (L119) **1**, 1–88 (2016)

2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

A Novel Embedded Platform for Secure and Privacy-Concerned Cross-Domain Service Access

Alexander Rech[1], Markus Pistauer[2], and Christian Steger[3]

***Abstract*— Connected driving is a hot topic in the automotive industry and a leverage to push new Mobility as a Service (MaaS) methodologies, making vehicles an essential part of the Internet of Things (IoT). However, these new technologies often lead to security risks and privacy concerns, especially due to the increasing number of datasets exchanged between vehicles, drivers, and local infrastructure. Furthermore, the possibilities for vehicles to access heterogeneous services offered by different service providers are often limited due to rigid system boundaries. In this paper we present a novel federated service management concept for increased interoperability across distinct services in the field of Smart Mobility and Smart Cities. Our approach provides secure authentication and authorization between cars, their drivers, and other information systems, while retraining the level of privacy according to the users' preferences. The scalability and dynamic configurability of the solution and the elaborated proof-of-concept will set it apart from application-centered gateways to an embedded generic platform by virtue of its modular software design.**

I. Introduction

In addition to different car-to-car communication concepts car-to-x methodologies have now also become increasingly important. New smart mobility approaches are being elaborated, fueled by the increasing need of new soft- and hardware methodologies and stronger interrelationships between different services. In the automotive sector we can also find more and more subsystems exchanging data with the outside world. As a result, cars are becoming "things" in the IoT and thus also a part of a Smart City, where many different information systems come together. However, more data exchange between a vehicle and the local infrastructure also implies more concerns in terms of security and privacy. Moreover, systems today tend to be tailored to a specific application area, with the result that boundaries are often rigid and inflexible preventing the building of synergies between heterogeneous service providers, their services and users. We now face a critical need for innovations, as secure authentication and authorization to different heterogeneous Smart City services as well as personalized data access become more and more important for cross-domain use cases, especially in the fast growing car-to-x sector.

Personalized access. In this paper we propose a concept for managing user- and service-related data in automotive application scenarios and address the lack of privacy control. The concept enables drivers to adapt the data flow during their driving session according to their privacy preferences. The more permissive they are, the more services may be unlocked, tailored to the driver's data.

Connected services. We call into question the current situation where systems are limited to a predefined scope regarding their services and users. Based on the privacy-centered approach, an additional federated service concept is introduced for accessing and redeeming heterogeneous Smart City services according to the users' privacy settings. In this sense, services of different providers shall be accessible via a trusted cloud-based approach.

Overcoming the offline transition phase. In order to overcome the current non-connected, internet-less transition period in the automotive sector, the communication interface of the embedded proof-of-concept includes the usage of the proximity-based wireless technology Bluetooth Low Energy (BLE). On one hand, it is used to connect the vehicle to the driver's mobile phone which acts as gateway to the cloud via a RESTful interface. The phone is responsible for the initialization process of the hardware including the actual user binding and adaption of service- and privacy-related data. On the other hand, BLE is utilized to enable communication to local infrastructure for redeeming services obtained.

Embedded proof-of-concept. The proposed software concepts were integrated into a hardware module which fulfills the demand for increased flexibility, robustness and configurability. The embedded demonstrator reflects the ongoing trend for property sharing – in our case car sharing – and focuses on personalized access. Additionally, the framework enables the vehicle to access other services, related to parking, ordering food (drive-in-restaurant), or leisure events (cinema, museum), for instance. Our work shows that a connected unit inside a car offers possibilities to bring the connected Smart City paradigm to the automotive sector and thus contributes to a new level of connected mobility within cities, while preserving end user privacy aspects.

This paper is organized as follows. Section II shows related work while section III discusses the design choices of our approach. Section IV consists of implementational aspects and a performance evaluation. Finally, section V summarizes our ideas and gives information on future work.

II. Related Work

Everyday life is becoming ever more connected to the digital world. Not only are information services and applications running on an extensive set of different systems and

[1]Alexander Rech is with CISC Semiconductor GmbH, Graz, 8010, Austria and Graz University of Technology, Department of Technical Informatics, Graz, 8010, Austria a.rech@cisc.at

[2]Markus Pistauer is with CISC Semiconductor GmbH, Klagenfurt, 9020, Austria m.pistauer@cisc.at

[3]Christian Steger is with Graz University of Technology, Department of Technical Informatics, Graz, 8010, Austria steger@tugraz.at

are in constant interaction with people and their environment, also the number of wireless connections is growing exponentially around the globe. It is estimated that nearly 25 billion devices will be connected to the internet by 2020 and 50 billion devices by the year 2050. Most of these devices will offer wireless communication interfaces leading to the expansion of wireless areas and the development of novel wireless methodologies [1]. New wireless concepts and frameworks are being developed, such as DEWI. This proposes a locally adaptable and trusted wireless communication bubble for the IoT, where many wireless communication standards are integrated [2]. API-based approaches are a common strategy to interconnect different IoT systems, spanning from access control and area network systems to complex systems that bring together different vendors' solutions. This can be done by connecting IoT endpoints to IoT platforms or gateways which enable the transport of data between multiple devices that would otherwise be unable to communicate with each other. Due to the widespread use of smartphones and their wireless interfaces (Wi-Fi, Cellular, BLE, NFC) they can be used as gateways to connect devices to the internet. BLE is an especially good candidate when power-constrained embedded devices come into play. The work illustrated by paper [3] presents a smartphone-based gateway solution responsible for retrieving data from wearable sensors over BLE as well as storing it to a central cloud storage in real-time. An even more generic gateway approach is shown by the project fabryq presented in the paper [4]. It addresses the problem of increased complexity which arises when developing systems responsible for establishing communication between embedded devices and servers over gateways. By contrast, other approaches involve mobile phones to enable offline communication to cars for secure access in car-sharing application scenarios, by leveraging the phones' local wireless capabilities [5]. Other mobility concepts focus on connecting different e-vehicle solutions. The Horizon 2020 project STEVE for instance, designs and implements a system for connecting different mobility and "gamified" services [6]. Complementary to this approach is this work [7], which focuses on a more generic method for interconnecting different Smart City services via a trusted cloud-based concept. While security considerations are not new in the context of connected systems, many implementations present new security challenges. Every poorly secured device or subsystem that is connected online can serve as a potential entry point for cyber-attacks, compromising systems as well as exposing data [8]. Slack security methodologies can have especially severe consequences in the automotive sector. Since this sector is becoming increasingly connected (75% of European cars will be connected by 2020 [9]), security vulnerabilities can lead to unsafe or even lethal scenarios, as described in [10], [11]. An increased level of connectivity may not only lead to security problems, but may also evoke major privacy concerns for drivers. As a result, a reluctant connected vehicle user / buyer group of 25% will persist in the next few years, according to the survey "My Car My Data" [12], leaving one out of four cars unconnected. This makes paving the way for robust, dependable and trustable transport systems all the more important. The paper [13] analyzes how personal information flows through typical telematics systems, distinguishing between an embedded approach where cars connect directly to the internet and an integrated approach relying on mobile devices for accessing different services. Connected cars are able to receive, process, and send large amounts of data. Since this data may not only be related to the vehicle, conclusions about the drivers and their habits (e.g. their destinations, daily routines, etc.) may be derived. Consequently, this causes potential privacy implications for drivers, especially when data is aggregated from several data-sets and shared across various companies, such as car dealers, car rental companies, insurance companies or mobile device and service providers [14]. For a higher level of trust, privacy preserving data management techniques have been studied extensively [15]. New techniques are being elaborated that use clustering algorithms as a pre-process to further improve the diversity of anonymized data [16].

III. Design Choices

A. Distributed system key components

The main objective of our proposed generic solution is to enable cross-domain sharing of heterogeneous services for the driver of a vehicle in a secure and privacy-preserving way. The general idea of the framework is to derive anonymized tokens (authentication-, privacy- and service-tokens) from user and service data and manage their lifecycle (creation, storage, and transfer) among all entities. A custom public key infrastructure (PKI) forms the basis of the platform's security and ensures the identity of all communication participants. Signatures are created and verified with the Elliptic Curve Digital Signature algorithm (ECDSA). The distributed system design, partly based on the architectural proposal of [7], is depicted in Fig. 1 and explained in the following. The elaborated embedded demonstrator inside the vehicle will be referred to as "eClient" in this paper.

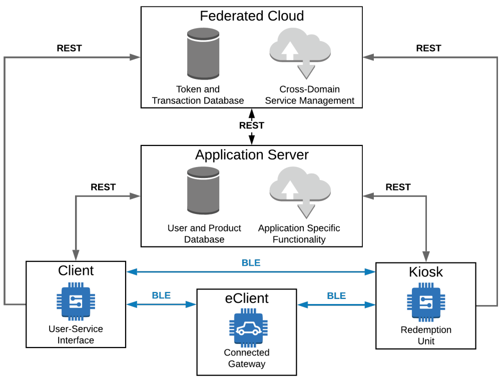


Fig. 1. Distributed system overview.

- **Client.** A mobile personal device representing a user who owns digital tokens and exchanges them for goods and services at corresponding kiosk devices via BLE. It acts as a digital service-wallet and offers an interface for obtaining services and managing the user's privacy rules.
- **Kiosk.** A mobile or stationary device that represents a terminal that can also be embedded into infrastructure (e.g. a parking gate). A kiosk device acts as a validation authority. It receives and validates tokens in order to authenticate users and authorize them to access goods and services.
- **eClient.** An embedded unit that can be placed inside a vehicle to enhance connectivity to its surroundings. It uses a client device without the need of an active internet connection to synchronize user and service data over BLE. Additionally, it is able to redeem the obtained services at kiosk devices.
- **Application server.** This server communicates with client and kiosk devices and keeps them synchronized. It offers application specific functionality and holds the real user data (e.g. email, name, etc.), as well as product and service related datasets (e.g. type, price, etc.) on application level.
- **Federated Cloud.** This works as a certification authority (CA) and token management system for issuing, signing, and monitoring tokens and maintains a transaction record database. The server's certificate is signed by an external root CA and is imported in all system entities' trust CA stores, meaning it is globally trusted. Client and kiosk devices use public key pinning for the root certificates to prevent the issuance of malicious server certificates by tampered CAs. The idea behind the Federated Cloud is to provide a common trusted layer responsible for abstracting users, products, and services from different application servers and sharing the anonymized datasets across the participating systems. In this sense, the real data always remains on application layer level, while only tokenized data is processed inside the Federated Cloud. Different application layers (incl. application server and corresponding apps) of independent service providers may join the Federated Cloud via a RESTful interface and offer their services to other service providers or users.

B. Wireless communication interfaces

The overall system provides personalized and secure access to goods and services wirelessly via BLE and a RESTful interface. Which communication interface is used by which device is also depicted in Fig. 1.

- **Representational State Transfer (REST).** REST is a software architectural style for implementing web services relying on HTTP. Client and kiosk devices as well as their corresponding application server communicate to the Federated Cloud via a RESTful API.
- **Bluetooth Low Energy (BLE).** BLE is a short-range wireless technology. It greatly benefits IoT applications due to its power saving design, the coexistence of connectionless (broadcaster and observer roles) and connection-based (peripheral and central roles) data transfer procedures, its robustness against obstacles, and compatibility with smartphones [17]. In our case BLE is used in two different ways, enabling local communication without the need of an active internet connection. On the one hand, it is utilized for the initialization process of the eClient (peripheral) by communicating to the mobile client (central), thus synchronizing user- and service-centered data between both devices. On the other hand, it is used for the actual transaction and redemption handling between a client/eClient (central) and a kiosk device (peripheral).

C. Secure authentication

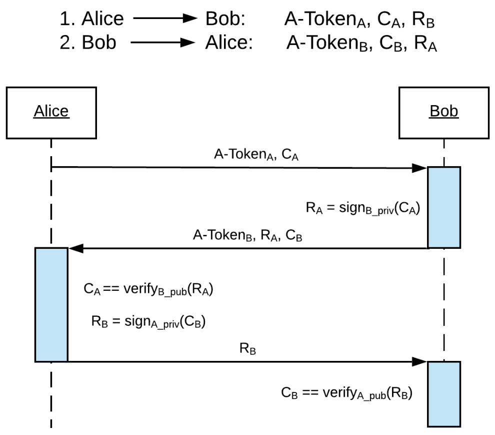
Authentication between devices as well as data-integrity are provided through **authentication tokens (A-tokens)** and a dedicated challenge-response protocol. An A-token is an extended certificate, tied to a particular device. It is server-signed and consists of the device's public key, a validity period, and token properties. A client or kiosk device's A-token is issued in the course of a distributed Kerberos-based authentication procedure involving a user-login on the application server. A one-time-ticket is issued that can be redeemed together with the device's public key during the registration procedure at the Federated Cloud. Finally, the device receives an A-token as well as an API key for accessing the Federated Cloud's REST-based interface.

In order to check the right ownership of A-tokens, a challenge-response procedure is applied when two devices communicate with each other via the BLE channel. First, A-tokens (in this example A-token$_{\mathrm{Alice}}$ and A-token$_{\mathrm{Bob}}$) are exchanged and verified with the server's public key. Alice generates a random number (challenge $\mathrm{C_A}$) and challenges Bob to sign it with his private key before it is returned to Alice. Furthermore, the received signature is verified against the random number $\mathrm{C_A}$ by using Bob's public key embedded into the previously exchanged A-Token$_{\mathrm{Bob}}$. If the verification is passed, the ownership of A-Token$_{\mathrm{Bob}}$ was proven. In case, both devices are already initialized and thus, in possession of an A-token, the same procedure is also applied for A-token$_{\mathrm{Alice}}$ before further data is exchanged. See Fig. 2 for more details regarding the challenge-response mechanism.

Since all packages passed between the devices do not contain any sensitive data but only tokenized user and service datasets, an additional encryption layer is not required. This implies benefits such as less computational efforts for all communication participants and faster data transmission.

D. Adaptive authorization of different services

We distinguish between two generic methodologies when talking about authorization. On the one hand, our concept foresees the usage of user-centric **privacy-tokens (P-tokens)**, which manage the driver's privacy level. On the other hand, service-related **service-tokens (S-tokens)** represent a digital ownership of an item or service. They are issued



1. Alice → Bob: A-Token$_A$, C$_A$, R$_B$
2. Bob → Alice: A-Token$_B$, C$_B$, R$_A$
Alice
Bob
A-Token$_A$, C$_A$
R$_A$ = sign$_{B\_priv}$(C$_A$)
A-Token$_B$, R$_A$, C$_B$
C$_A$ == verify$_{B\_pub}$(R$_A$)
R$_B$ = sign$_{A\_priv}$(C$_B$)
R$_B$
C$_B$ == verify$_{A\_pub}$(R$_B$)

Fig. 2. A-token based challenge response mechanism.

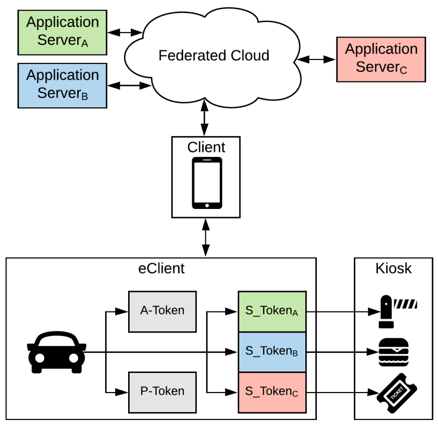


Fig. 3. Overview of application scenarios.

the moment a user obtains a product or service voucher via the application interface of the client. Both token types are always issued and signed by the Federated Cloud, bound to a specific A-Token, and only valid when presented in combination with the latter.

Furthermore, the S-Token consists of a validity period and two additional identifiers: one for the corresponding application (e.g. parking application, drive-in-restaurant application) and the other one for the actual service (e.g. right to access a specific area). They serve as indicators to specify which S-tokens are redeemable by a specific kiosk device. If these conditions are met, the appropriate S-token can be sent from a client or eClient device to a kiosk and be redeemed there. Consequently, the user is authorized to access the service for which the tokens stands. Due to the generic nature of an S-token, services of different kinds can be represented, thus enabling different use cases. An overview of a few possible application scenarios is given in the following. A pertinent schematic description can be taken from Fig. 3.

- **Car access.** As soon as an A-token and a corresponding P-token have been transferred to the eClient, thus authenticating a user and binding himself and his privacy settings to the eClient, the vehicle's unlocking signal can be triggered. Since the transfer of the tokens does not involve an active internet connection on the eClient side, and each user is able to personalize his dataflow during the trip with his personal device, this would suite a car sharing use case. Additionally, all data on the eClient could be reset after the car-sharing session.
- **Smart parking.** When a valid parking entitlement is transferred to the eClient, the vehicle is allowed to access a parking lot for the validity period of the entitlement.
- **Drive-in-restaurant.** If a food voucher is obtained, the vehicle will be able to redeem the food voucher directly at the kiosk of a drive-in-restaurant.

By contrast, a P-token contains the datasets the user wants to share as well as the user's privacy level which can be edited via the application interface of the Client. The P-token is transferred from client to eClient during the initialization process. The data that can be shared may include user-related data like gender or age, or trip-based datasets collected during the car ride, such as information about the destination, GPS coordinates, etc. depending on the underlying application. If the user decides upon a low privacy level and thus consents to the data in question being shared with other service providers, it is forwarded to the Federated Cloud and can be used in exchange to issue additional S-tokens for the driver. The uploaded data is not directly associated with privacy critical values such as his name or email-address, but just linked to his anonymized P-token. In order to cope with different privacy requirements three privacy levels were introduced:

- Level 2: Only services obtained directly by the user are accessed. No additional data is shared across the Federated Cloud.
- Level 1: Drivers may decide whether trip-based (e.g. GPS, destination) or user-related datasets (e.g. gender, obtained services) are shared.
- Level 0: User and trip-based data collected during the journey are shared with participating service providers.

This approach foresees that the application server will utilize the RESTful interface to send key/value pairs to the Federated Cloud containing the data in question. Subsequently, the values submitted are linked to the user's P-token. They become visible to other participating service providers for as long as the user does not change his privacy settings or his A-token or P-token do not expire. In further consequence, the datasets submitted may be used to create new offers and entitlements adapted to the user's needs and preferences, involving again the Federated Cloud.

In this context, the use cases mentioned above can be arbitrarily extended, like in the following example:

Generic interaction with heterogeneous service providers. In cases when data is shared, service providers may offer their users additional entitlements for accessing their services for special conditions. In concrete terms, a car wash agency (e.g. Application Server$_C$ in Fig. 3) would be able to access shared data over the Federated Cloud without knowing the real identity of the driver. For example, when the driver's destination address is shared, the car wash agency could offer him a discounted or free admission to a car wash session lying on the route to his destination. Consequently, the derived S-token$_C$ would be stored on the user's eClient and is redeemable at a dedicated kiosk device at the car wash.

IV. Implementation And Results

A. *eClient components*

The eClient consists of two complementary components, the Adafruit Feather M0 board and the Bluegiga BLE112 module. While the Feather M0 acts as control unit and state machine, the main task of the BLE112 module is to execute different BLE commands and forward the responses received back to the Feather M0 over the UART interface. The Feather M0 is based on the widely spread Arduino platform. It was picked as host device since the implementation should remain as portable as possible for similar controllers. The following software libraries were included into the project:

- **BGLib.** This library acts as a C wrapper for the event-driven BGLib protocol used to control the Bluegiga BLE112 module.
- **micro-ecc.** This library holds a lightweight ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit architectures. In our case, the elliptic curve secp256r1 (prime256v1, NIST P-256) was used. According to the 2018 ECRYPT-CSA recommendation this type of curve is recommended at least until 2028. The major advantage of ECDSA is its short cryptographic key length compared to other algorithms such as RSA. This enables faster data transfer of keys and certificates and lower memory requirements.
- **Cryptosuite.** It is a cryptographic library specialized on secure hashing and hashed message authentication. SHA-256 was used in our case.

The eClient provides possibilities to bring the connected services paradigm to vehicles, without providing a direct interface to the car. The small form factor of the Feather M0 (2in x 0.9in) combined with the possibility of attaching a rechargeable lithium polymer or lithium ion battery over the JST jack makes it even more portable.

B. *Mobile client as gateway*

The client acts as an internet-enabled gateway for the eClient. It was implemented in the form of a Java-based software library which can be integrated into Android applications. Regarding Android's BLE stack all BLE roles are supported since Android 5.0. The additional possibility to only scan for specific BLE advertisements was also introduced with version 5.0. All interactions that would require the eClient to directly communicate to the Federated Cloud are carried out by the mobile client instead. In this sense, the eClient sends out specific BLE advertisements which are noticed by the client. Subsequently a BLE connection is established and the actual request is forwarded to the mobile client and finally, carried out using the REST interface of the Federated Cloud. Authentication to the Federated Cloud is ensured by providing a secret API key for basic HTML authentication in the request, which is issued during the registration of the device. Next, the server's response is sent to the client, processed, and forwarded to the embedded device over BLE. The server response is formatted in JSON while all data transferred between the eClient and the gateway is encoded in the more compact TLV-encoding scheme, conforming to the following byte format: Type | Length | Value. This format allows the receiver to decode the information with dedicated parsing functions without requiring any pre-knowledge of the size or the semantic meaning of the data.

C. *Key and token handling*

As soon as the eClient is switched on the initialization process is started and is responsible for setting up the BLE module and the board's pins, and checking if the cryptographic keys and corresponding A-token or P-token are already stored on the device. If the device still needs to be initialized, an ECC key pair ($k_{\text{priv\_eClient}}$, $k_{\text{pub\_eClient}}$) is generated and stored. The elliptic curve secp256r1 is used for signing and verifying signatures. Therefore, the private key is 32 bytes and the public key 64 bytes long. As soon as a user approaches the vehicle with his mobile client a confirmation log will appear. If confirmed, a BLE connection between client and eClient is established and the binding procedure is triggered. First, the eClient verifies the authenticity of the mobile client:

1) The client sends its device and user-bound A-token$_{\text{client}}$ to the eClient.
2) The expiry date of A-token$_{\text{client}}$ is checked by comparing the current date and time with the timestamp value inside the token.
3) The signature of A-Token$_{\text{client}}$ is verified with $k_{\text{pub\_server}}$ before a challenge response protocol (see section III-C) is applied to verify if the A-token received really belongs to the device currently communicating to the eClient.

The next steps involve the derivation of new tokens for the eClient from the user-bound A-Token$_{\text{client}}$ and the privacy settings in form of a P-Token$_{\text{client}}$.

4) The eClient sends a data packet containing $k_{\text{pub\_eClient}}$ to the client, which will forward it together with its tokens to the Federated Cloud.
5) The server creates a derived A-token$_{\text{eClient}}$ and P-token$_{\text{eClient}}$ where $k_{\text{pub\_eClient}}$ is embedded.
6) The new tokens are sent back to the smartphone client and forwarded to the eClient through the BLE-channel.
7) Both tokens received are verified with $k_{\text{pub\_server}}$ by the eClient.

Next, a unique fingerprint (A-token$_{\text{client\_fingerprint}}$) of the mobile client is created by hashing A-Token$_{\text{client}}$ with SHA-256. For further communication this fingerprint is used together with the previously utilized challenge-response-protocol to determine if the same device responsible for the initialization mechanism is communicating to the eClient. Finally, A-Token$_{\text{client\_fingerprint}}$, A-Token$_{\text{eClient}}$, and P-Token$_{\text{eClient}}$ are stored on the eClient. From this moment on S-tokens can also be synchronized between the two devices involving again the Federated Cloud for the derivation process. Last but not least, the digital services obtained can be redeemed at corresponding kiosk entities.

D. Performance evaluation

The following three tables provide information about the timing behavior of the implementation. Average measurement values are reported.

Table I shows the time the token generation process for the eClient takes. It includes the A-Token$_{\text{eClient}}$ and P-Token$_{\text{eClient}}$ derivation involving the client and the Federated Cloud and several verification steps.

TABLE I
MEASURED TIME OF THE A-TOKEN$_{\text{ECLIENT}}$ AND P-TOKEN$_{\text{ECLIENT}}$ DERIVATION PROCEDURE

Action	Time [ms]
Retrieve and verify A-Token$_{\text{client}}$	388
Derive and receive A-Token$_{\text{eClient}}$ and P-Token$_{\text{eClient}}$	1210
A-Token$_{\text{eClient}}$ and P-Token$_{\text{eClient}}$ verification	421
eClient local data-storing procedure	73
Total	**2092**

In contrast, the information on the time required for the verification mechanism between eClient and Client as well as the creation and reception of an S-token$_{\text{eClient}}$ is given by Table II.

TABLE II
MEASURED TIME OF THE S-TOKEN$_{\text{ECLIENT}}$ DERIVATION PROCEDURE

Action	Time [ms]
Mutual verification mechanism	1288
Derive and receive S-Token$_{\text{eClient}}$	442
eClient local data-storing procedure	59
Total	**1789**

Last but not least, Table III illustrates the timing behavior of the verification and data transfer procedure between an eClient and kiosk device, and the online redemption process of an S-Token$_{\text{eClient}}$ involving the Federated Cloud.

TABLE III
MEASURED TIME OF THE REDEMPTION PROCESS BETWEEN ECLIENT AND KIOSK DEVICES

Action	Time [ms]
Mutual verification mechanism	1194
Prepare data to redeem	533
S-Token$_{\text{eClient}}$ online redemption	452
Total	**2179**

In summary, the retrieval of the tokens and the redemption process take around two seconds with the verification mechanism taking a large part of the total execution time. However, since the developed prototype does not have any realtime requirements, the overhead created by the verification mechanism is still acceptable. The values represent short waiting times for the user underlining the usability aspect of the developed prototype.

V. CONCLUSION AND FUTURE WORK

With the ongoing development and distribution of the Internet of Things, new car-to-x methodologies have also become increasingly important. In order to fuel the movement towards MaaS we designed and implemented an embedded prototype that gives cars and their drivers access to heterogeneous services (car access, smart parking, drive-in restaurant food-voucher redemption, etc.), while respecting the drivers' privacy requirements. In view of the number of not internet-connected-cars that is still not negligible [9], [12], our proof of concept takes advantage of the local wireless communication standard BLE to overcome the current non-connected transition phase in the automotive sector. In this sense, it uses the driver's mobile phone as a gateway for a coupling procedure between car and driver and the synchronization of the tokenized user- and service-related data. Furthermore, the device is able to redeem obtained services by communicating to local infrastructure (e.g. parking gates). Secure authentication and authorization as well as data integrity are enforced via cryptographic standards such as PKI and ECDSA in order to meet the high demands on security and privacy. Additionally, all data transmitted between the devices is issued, tokenized, and monitored by a central trusted server. Due to the generic nature of the tokens, services of multiple heterogeneous service providers can be represented, offering the possibility to arbitrarily extend MaaS application scenarios (e.g. drivers get a digital ticket for a car wash lying on their route). Finally, the evaluation of the transmission timings revealed low protocol execution rates – on average of approximately two seconds – thus highlighting the prototypes overall usability. Future Work will concentrate on defining a secure interface between the embedded proof of concept and the vehicle's on-board unit. In this way, additional services may be unlocked for the driver of the vehicle, due to the larger number of available data. Furthermore, we intend to increase the level of trust between participating service providers with a distributed smart-contracts-based approach.

ACKNOWLEDGEMENT

This work is supported by CISC Semiconductor and Graz University of Technology (Institute of Technical Informatics). Furthermore, parts of this work have been performed within the European Union's Horizon 2020 projects SCOTT under agreement No. 737422 and PRYSTINE under grant agreement No. 783190.

References

[1] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything," *CISCO white paper*, pp. 1–11, 2011.
[2] W. Rom, P. Priller, J. Koivusaari, M. Komi, R. Robles, L. Dominguez, J. Rivilla, and W. V. Driel, "DEWI-Wirelessly into the future," in *Proceedings - 18th Euromicro Conference on Digital System Design, DSD 2015*, 2015, pp. 730–739.
[3] T. Soultanopoulos, S. Sotiriadis, E. Petrakis, and C. Amza, "Internet of Things data management in the cloud for Bluetooth Low Energy (BLE) devices," *Proceedings of the Third International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing - ARMS-CC'16*, pp. 35–39, 2016.
[4] W. McGrath, M. Etemadi, S. Roy, and B. Hartmann, "Fabryq: Using Phones As Gateways to Prototype Internet of Things Applications Using Web Scripting," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS)*, 2015, pp. 164–173.
[5] A. Dmitrienko and C. Plappert, "Secure Free-Floating Car Sharing for Offline Cars," *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy - CODASPY '17*, pp. 349–360, 2017.
[6] The European Commission, "STEVE: Smart-taylored L-category electric vehicle demonstration in heterogeneous urban use-cases." [Online]. Available: http://www.steve-project.eu
[7] A. Rech, M. Pistauer, and C. Steger, "Increasing Interoperability Between Heterogeneous Smart City Applications," in *Lecture Notes in Computer Science*. Tokyo: Springer International Publishing, 2018, pp. 64–74.
[8] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things: An Overview - Understanding the Issues and Challenges of a More Connected World," Tech. Rep., 2015.
[9] Frost & Sullivan, "The Future of Connected & Autonomous Vehicles," *Intelligent Mobility Report*, pp. 1–12, 2015.
[10] C. Valasek and C. Miller, "Remote Exploitation of an Unaltered Passenger Vehicle," *Technical White Paper*, pp. 1–91, 2015.
[11] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," *IOActive Technical White Paper*, pp. 1–99, 2013.
[12] Fédération Internationale de l'Automobile, "What Europeans Think About Connected Cars," Tech. Rep., 2015.
[13] K. Jaisingh, K. El-Khatib, and R. Akalu, "Paving the way for Intelligent Transport Systems (ITS)," in *Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications - DIVANet '16*. Malta: ACM, 2016, pp. 25–31.
[14] P. Lawson, B. McPhail, and E. Lawton, "The Connected Car: Who is in the Driver's Seat? A study on privacy and onboard vehicle telematics technology," Tech. Rep., 2015.
[15] V. Torra, G. Navarro-Arribas, and K. Stokes, "An Overview of the Use of Clustering for Data Privacy," in *Unsupervised Learning Algorithms*, M. E. Celebi and K. Aydin, Eds. Cham: Springer International Publishing, 2016, pp. 237–251.
[16] P. Canbay and H. Sever, "The Effect of Clustering on Data Privacy," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Miami: IEEE, 2015, pp. 277–282.
[17] R. Davidson, T. Kevin, W. Chris, and C. Cuff, *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*, 1st ed., B. Sawyer and M. Loukides, Eds. O'Reilly Media, 2014.

2019 IEEE International Conference on Blockchain (Blockchain)

A Decentralized Service-Platform Towards Cross-Domain Entitlement Handling

Alexander Rech
Graz University of Technology
Institute of Technical Informatics
Graz, Austria
rech@tugraz.at

Christian Steger
Graz University of Technology
Institute of Technical Informatics
Graz, Austria
steger@tugraz.at

Markus Pistauer
CISC Semiconductor GmbH
Klagenfurt, Austria
pistauer@cisc.at

***Abstract*—As our world is becoming increasingly connected, secure and seamless cross-domain access to heterogeneous services is becoming more and more important. However, cooperation between client-server based systems in the sales domain is often difficult to achieve due to a lack of standardization as well as limited trust between service providers. Furthermore, the demand for anonymization techniques is becoming more important due to tighter regulations and increased privacy awareness of users. This is especially relevant when data needs to be shared by several parties. In this paper we describe a software service platform designed to increase collaboration between different cross-domain services. The general idea is to provide a common trusted layer for existing client-server systems that can be used for anonymizing and sharing services in the form of digital entitlements across independent service providers, their systems, and users, while ensuring secure authentication and authorization. Furthermore, we leverage the tamper-resistant properties of the blockchain and smart contracts in order to protect sensitive transactions against fraudulent manipulation and to introduce service-based agreements to the overall system.**

***Keywords*-Connected Services; Blockchain; Access Control; Security; Privacy;**

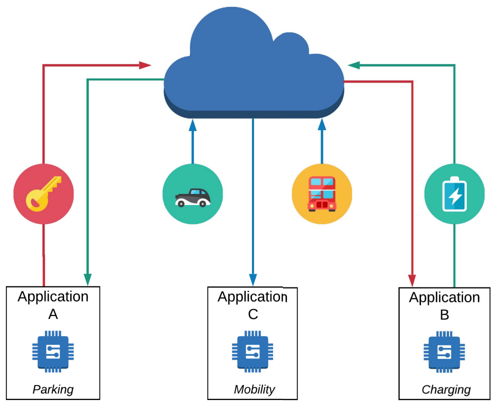


Figure 1. Digital entitlements can be exchanged between independent applications: inter-app and single-app cross-domain service exchanges are feasible.

I. Introduction

Today's service redemption systems are often designed as client server applications that are tailored to specific application scenarios. Due to rigid system boundaries, time and cost consuming integration procedures, and the lack of trust between different service providers there is often no intention of providing the possibility to share services across different domains and their systems. Additionally, public awareness of data privacy has risen significantly over the past few years and the value of data ownership has become a matter of intense public awareness and of ever greater importance to individual citizens. We want to increase the scope of businesses regarding their services and users and propose a platform for managing cross-domain services. The overall platform can be used for two scenarios: an inter-app approach, where services in form of digital entitlements are exchanged between independent platforms, and a single-app approach, where heterogeneous services are integrated into one application. Fig. 1 illustrates our idea. On the one hand, a parking application focused on issuing parking permits and a charging app that by contrast is responsible for authorizing the user to charge his vehicle may cooperate with each other. In this sense, when a costumer redeems a specific service from one service provider, additional offers from another independent service provider could be unlocked as a reward, redeemable at the corresponding redemption unit (parking gate, charging station). On the other hand, a route planer app, for instance, may suggest several possibilities to reach the user's destination by integrating and combining different mobility services (e-bikes, car-sharing, public transport, etc.). These approaches would not only benefit the customer and result in a higher user satisfaction, they would also yield benefits for participating vendors in terms of a larger advertising scope and additional sales. From a technical standpoint, the platform that is described in this paper provides secure authentication, authorization, and accounting for applications of different kinds. In order to incentivize a cooperation between different service providers, trustability shall be further enhanced by leveraging blockchain and smart contract methodologies enabling transaction security, traceability, and data protection against fraudulent changes.

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00069

A blockchain is a peer-to-peer network architecture; a distributed database fully shared across multiple entities that does not have a central authority to manage it. All database entries are synchronized across all network participants, reducing the risk of single point of failures. Data is collected into blocks which in turn are linked, starting from a predefined genesis block which defines the properties and the underlying consensus mechanism of the blockchain. Consensus on the validity of blockchain transactions is achieved with a dedicated consensus algorithm, executed by a set of network authorities that is designed in such a way that acting against the rules does not pay off. Depending on whether the majority votes in favor of accepting a new block containing multiple transactions and datasets or not, it is added to the blockchain or rejected. Blockchains can be either public, where transactions are verified by an independent group of network participants, or private, where only a set of authorized individuals can process new transactions. A mixture of both would be federated blockchains. They do not allow any person to participate in the process of reading, writing, or verifying without an explicit invitation. This concept behind federated blockchains is particularly interesting in business applications in which the participants, such as different companies, may identify, but not fully trust each other. In order to specify additional rules on how different companies, aka service providers, interact with each other, smart contracts can be used. A smart contract is a computer program running on top of a blockchain, consisting of inputs, deterministic outputs, and trigger-conditions. They are able to process and store arbitrary datastructures. When a contract is deployed, it is assigned a random address that is used in future transactions as a reference to invoke the contract's functions. It stores its results as transactions in the distributed ledger. They cannot be deleted or altered by fraudulent entities due to the tamper-resistant design of the blockchain (cryptographic primitives such as hash functions, asymmetric cryptographic signatures, etc) [1], [2].

By leveraging these blockchain and smart contract properties we are presenting a platform for enabling tamper-resistant service exchange between vendors. Due to this collaboration, services from one service provider can be made accessible to another independent service provider, who would be able to obtain and forward them to his customers on demand or in an automated way, as soon as certain prerequisites are fulfilled. Transactions including issuance, obtainment, and sharing of cross-domain services are uploaded to the blockchain via smart contracts. We believe our approach to be very general and applicable to a wide range of use cases, paving the way for a new level of interconnectivity between different cross-domain services.

II. Related Work

The following papers build their approaches on the Ethereum blockchain. Ethereum is a virtual computer, aiming to be a decentralized world computer by offering smart contract capability. The network is composed of miners, full nodes, and light nodes. Light nodes rely on full nodes for security and can validate states by downloading and verifying block headers. Full nodes comprise the whole blockchain database. A subset of these validate all blocks and execute all contracts. Users submitting transactions to the Ethereum blockchain hold a private/public keypair. The public key is used to generate a blockchain address that serves as a public user identifier as well as an identifier for deployed smart contracts. Ethereum comes with different consensus protocols [3]. The most common are Proof of Work (Etash), where miners put work into solving a puzzle, and Proof of Stake (Casper) where stakeholders bet on whether a certain block is added or not. In contrast, in Proof of Authority (Clique) a set of approved authorities decides whether a transaction is valid or not.

Independent of the actual application area or use case, many blockchain approaches focus on combining non-blockchain access-permission systems with the distributed ledger technology for increased security. One approach couples blockchain and off-blockchain storage to construct a personal data management platform in order to control access permissions to private data collected by a service. The collected data is encrypted and sent to an off-blockchain key-value store. Only a hash is retained, acting as pointer to the data on the blockchain. The data can be queried by the user or the service using the key associated with it. Subsequently, the blockchain verifies if the digital signature belongs to either the user or the service. Every time a users subscribes to a service a new transaction specifies the access permissions and another contains the hash of the data [4]. Another approach focuses on the management of electronic medical records of patients distributed across several data providers. The authors' blockchain implementation gives patients an immutable log and easy access to their medical information across providers and treatment sites. In order to guarantee that the data is not altered, hashed data pointers are stored on the blockchain with an additional query string which is intended to be executed on the corresponding server of the data providers [5]. Data assurance and resilience are also crucial security requirements in cloud-based IoT applications. The authors of this paper [6] describe a procedure for secure drone communication, focusing on the integrity of the collected data. Again, the blockchain is used together with a traditional cloud-based system. Instead of registering the drone to the blockchain, they anchor the hashed data records collected from drones to the blockchain network and generate a blockchain receipt for each record stored in the cloud. Another IoT-blockchain approach focuses on effective and trusted data distribution across several devices. All blockchain-related operations are forwarded to a gateway, which is responsible for managing access control and endpoint authentication. Devices may act as information providers and also as consumers, identified by a globally unique identifier. Every device that generates an information item may forward its hash to the blockchain. The idea of this solution is that IoT service providers store access control policies that protect sensitive data in access control providers (ACPs) and in return ACPs generate secret keys (using a hash with the device's locally available identifier as input). The path to the access control policies (e.g. URL) is known by the devices [7]. Many of today's blockchain-based prototypes, like the projects mentioned above, have one thing in common: the hybrid approach between a traditional cloud environment and blockchain layer, where hashes are pushed to the blockchain for verifying the actual data stored inside a traditional database. By doing so, the advantages of both methodologies can be combined, such as speed on the one hand, and enhanced data protection on the other hand.

Other approaches mentioned in the following paragraph discuss design choices and limitations especially for permissioned blockchain types, such as hard coded consensus algorithms, mainly utilized for industrial applications [8]. Hyperledger Fabric, for instance, is an open-source project. It is a general-purpose permissioned blockchain system which also can be seen as a distributed operating system for permissioned blockchains [9]. The authors of another paper propose an architecture based on satellite chains that are able to run different consensus protocols in parallel, making governance models more easily adaptable. Nodes join a specific satellite chain if they want to interact with other particular nodes. Each satellite chain maintains its own private ledger and prevents any node that is not part of the satellite chain from receiving or accessing transactions [10].

More and more cities are adopting smart city concepts to enhance the living quality of their citizens and optimize the resources of cities regarding healthcare, transportation, energy, and education, for instance. The paper [11] describes how service-based-middleware for cloud and fog computing may enhance smart city technologies. In this sense, applications running in specific spots in cities can utilize locally available edge-computers, end-user devices, or other nearby edge devices to access different city services and to facilitate cooperation between diverse systems. Another approach attempts to decrease the heterogeneity of preexisting systems, by concatenating their services and user pools on application level in a privacy preserving way [12]. In fact, data privacy is becoming more and more important due both to an increasing public privacy awareness and to increasingly stringent data protection rules. For example, in 2016, the European Union passed the General Data Protection Regulations (GDPR) [13]. Research is also being done on privacy preserving data management techniques. New concepts are being elaborated that use clustering algorithms as a pre-process to further improve the diversity of anonymized data [14].

III. Design

Our proposed solution, which is partly based on the design of [12], is a software platform for secure authentication, authorization and accounting of heterogeneous services in the form of digital entitlements for client-server redemption systems. It is a central place for service management, with additional decentralized control instances responsible for the integrity and availability of transaction-based data.

A. Requirements

Privacy preserving identification and authentication. When data, especially user-related data needs to be passed between different entities, privacy aspects become more important. Therefore, only tokenized user data should be exchanged and the link between real and anonymized data should be revocable. Additionally, each device needs to be authenticated when communicating to the federated platform.

Service authorisation management. A device has always to be authorized to use or redeem a specific service, which should be represented as a generic entitlement object. These entitlements should be assignable to devices, be revocable at any time, and be redeemable by dedicated control units.

Service co-modality. The interexchange of services between independent service providers shall be improved by providing a federated service layer where services and agreements between heterogeneous service providers can be defined, obtained, and subsequently forwarded to users and their devices.

Data integrity. All components interacting with each other should rely on PKI mechanisms. In this sense, certificates as well as all tokenized data should be signed by a trusted authority and be verifiable against fraudulent changes. Additionally, the integrity of agreements and transactions shall be further protected by leveraging blockchain and smart contracts capabilities.

Decentralized consortium consensus. A decentralized consensus algorithm shall be used to decrease the chance of misuse of the network by a malicious party. By relying on a consortium consensus approach additional advantages in terms of speed, network scope, and authorization handling should be achieved compared to completely public approaches.

Data availability. The risk of data loss should be drastically minimized, due to the distributed character of the blockchain. In this sense, each node shall maintain a local copy of the ledger. Furthermore, the framework should ensure that service providers control their transactions and have transparency over all agreements they are involved in.

Trusted traceability. Obtained and redeemed services as well as transaction-related data should be traceable by uploading a fingerprint to the distributed network.

B. Architectural Overview

Our platform is divided into two main parts, a business layer and a core layer. While the business layer is further composed of an application client and an application server unit, the hybrid core layer consists of a central cloud and a decentralized blockchain part. An overview of the entire system is given in Fig. 2. The business layer provides users with application-specific features, depending on the application area of the system. The application server of this layer interacts with the client devices and keeps them synchronized. It holds real user data (e.g. email, name, etc.), product and service related datasets (e.g. type, name, price, etc.), and offers application specific functionality (login, data processing, etc.), which may vary depending on the underlying business (parking provider, charging provider, etc.). The redemption unit can be part of the application server (online redemption) or outsourced to a dedicated device (parking gate). One or more business layers may communicate with the core cloud over a RESTful interface. The main tasks of the core cloud are abstraction (tokenization) and distribution of heterogeneous services across different business layers. Two types of tokens responsible for authentication and authorization are distinguished:

- **Authentication-token (A-token):** Authentication between application clients and the core layer is managed via A-tokens. They are issued during the registration process of the application client on the core layer. A-tokens are tied to a specific device, core-server-signed and consist of the devices public key, a validity period, and token properties.
- **Service-token (S-token):** The digital ownership of an item or service is represented by S-tokens, which can be compared to digital tickets or entitlements. They can be unambiguously associated with a service provider via an application-Id (a-Id) and a concrete service from the application layer via a service-Id (s-Id). The creation of an S-token can be triggered in two different ways. On the one hand, when an application client obtains a service on the application layer this triggers the token generation request on the core layer via a REST call. On the other hand, a service provider may trigger the creation of S-tokens and forward them to user devices on the application layer. After the issuing procedure of an S-token, they are bound to an A-token. If the A-token expires or is invalidated all corresponding S-tokens are invalidated too.

The core server works as a certification authority (CA) and signs all tokens. The servers certificate is in turn signed by an external root CA and is stored on all participating systems' CA stores (it is globally trusted). Application clients utilize public key pinning for the root certificates in order to prevent malicious server certificates to be issued by tampered CAs. Furthermore, the core server features a federated market-

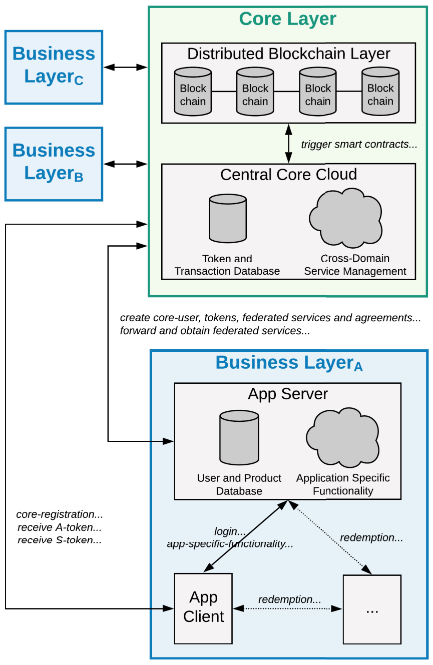

Core Layer
Distributed Blockchain Layer
Block chain
Block chain
Block chain
Block chain
trigger smart contracts...
Central Core Cloud
Token and Transaction Database
Cross-Domain Service Management
Business Layer$_C$
Business Layer$_B$
create core-user, tokens, federated services and agreements...
forward and obtain federated services...
Business Layer$_A$
App Server
User and Product Database
Application Specific Functionality
core-registration...
receive A-token...
receive S-token...
login...
app-specific-functionality...
redemption...
App Client
redemption...
...

Figure 2. Platform overview: the core layer is attached on top of business layers and responsible for issuing, sharing, and redeeming services.

place layer where services (e.g. authorization to enter a parking lot, redeem a food voucher, trigger the start of charging session) can be put, obtained by other service providers, and forwarded as anonymized S-tokens to their users. Supplementary to this, the decentralized blockchain-unit that is part of the core layer and addressable via a smart-contract based interface. It acts as a trusted, tamper-resistant storage facility for cross-domain transactions (S-token exchange). Additionally, agreements between service providers of different business layers can be defined for automating the creation and forwarding procedure of S-tokens as soon as the pre-defined conditions are met. Due to the distributed design of the blockchain, all transaction-based service data is available to multiple entities and protected against fraudulent changes. Consensus is reached by means of a Proof of Authority approach, involving participating service providers (with their corresponding application servers) as sealers.

C. Interaction Between App and Core

We shall now describe the interaction between the application and the core layer. In this we will emphasize the

.

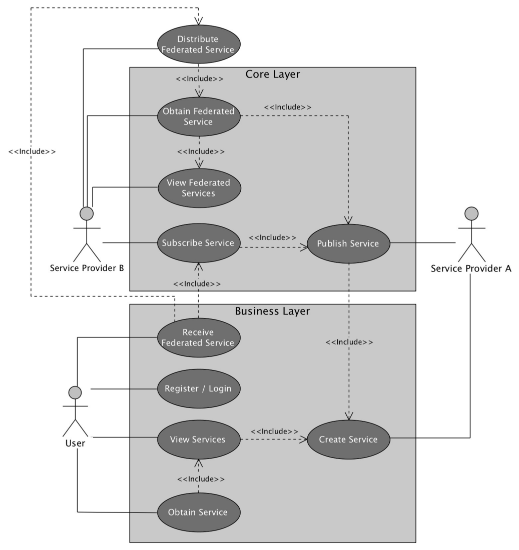


Figure 3. Interaction between the business layer and the core layer from the perspective of service providers and their users.

interplay with the blockchain layer. An overview of the most important features of the system and its actors is given by Fig. 3.

Establish a connection between business and core layer. The first step for a service provider to establish a connection from his business layer to the core foresees a registration on the core layer. This involves the submission of merchant related data and the app-identifier of the client application (e.g. Android application id) via a dedicated REST interface. Subsequently, a service provider specific application Id (a-Id) is issued as well as a dedicated API-key for further interactions from the service provider side with the core layer over the RESTful interface. Furthermore, a blockchain wallet (private-public keypair) is generated since each service provider or rather application server is involved in the consensus procedure of service-based transactions. The private key is used to sign blockchain transactions, while the public key is utilized for identification purpose. After the registration a service provider is finally authorized to publish, obtain, and forward federated services. In order to publish a federated service, descriptive fields (name, price, etc.) and a unique service-Id (s-Id) are submitted from the application layer. Subsequently, the service becomes visible for other service providers (uniquely identifiable via a-Id and s-Id).

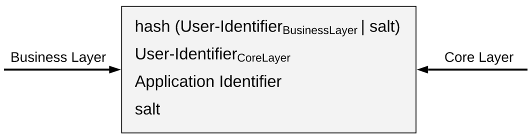

Business Layer
hash (User-Identifier$_{BusinessLayer}$ | salt)
User-Identifier$_{CoreLayer}$
Application Identifier
salt
Core Layer

Figure 4. Privacy preserving mapping of a user object between business layer and core layer – confidential data remains on the business level.

The integration of the application client into the overall framework can be done next. First, a successful login of the application client to the application server is necessary. The specific mechanism can vary depending on the underlying application. After this step, the registration between the application client and the core layer is triggered by the application server by sending a registration request. Subsequently, the core server issues a one-time registration-ticket bound to the end-user device, which is forwarded to the device through the application server. The ticket can be redeemed directly by the application client at the core cloud. Next, the creation of an anonymized core user entity is triggered, that is identifiable via a dedicated user-Id (u-Id). Finally, the device receives a user-bound A-token required for core server authentication. This distributed login mechanism assures that the core layer never receives user-related data (e.g. name, email), increasing the level of anonymity and privacy. The real user data remains on the application layer and is not distributed.

Creation of service-tokens. S-tokens can be made available to the user in two different ways: a user obtains services and entitlements on the business layer or a service is obtained by a service provider and forwarded to a user. In the first case, the obtaining of a product on the business layer side triggers the creation of an S-token on the core layer by submitting a corresponding s-Id. Consequently, the S-token is linked with the core-user. Last but not least, a new transaction entry is generated inside the core layer. The second case involves the service provider obtaining a federated service. The moment a service is obtained the creation of an S-token is triggered and the S-token can be assigned to the user via his user id (u-Id).

In order that S-tokens can be assigned to the intended user on the business layer without revealing confidential data of the user on the core layer, a privacy-preserving mapping is used. This is depicted by Fig. 4. A dedicated table is stored on the central core cloud, containing the user identifier of the application layer (e.g. email) hashed together with a salt, as well as the anonymized core user. Furthermore, the application identifier is added in order to address the right app. If a service provider intends to automate the issuing process and make it condition-dependent, agreements can be elaborated by signing a federated service. This triggers a validation flag that makes the agreement valid. An agreement consists of the federated service, additional datasets identifying the interested service provider as well as a condition that determines under which circumstances the federated service specified inside the agreement should be issued. A condition tells how many S-tokens have to be obtained or redeemed by a user before the service (reward) is issued, or which specific S-token needs to be redeemed in order to trigger a reward, for instance. If the agreement's trigger condition is met, the federated service specified inside the agreement is derived to an S-token and forwarded to the client who triggered the current transaction.

Interaction with the blockchain layer. Each transaction, which in our context includes either the creation or the redemption of S-tokens, is stored in a transaction history table on the core server in an anonymized way. Anonymized means that each transaction history entry only consists of a timestamp and different ids that identify the service provider (a-Id), the actual service (s-Id), and core user entry (u-Id), without referring to the real data that lies on business layer. All transactions that involve the obtainment of a federated service by a service provider, are additionally forwarded as key-value pairs to the blockchain through a dedicated smart contract. While the value consists of the hashed transaction entry, the key is composed of the a-Id and a timestamp. For each transaction a receipt as well as the key is generated and forwarded to the service providers of the application layer. The key provides the data management system with the ability to access the transactions for additional validation and tracking of past transactions, while ensuring the integrity of all transactions. A set of transactions is used to compose a new block, which will be confirmed by the network's sealers. If the block was accepted, it will be integrated into the existing blockchain, making it part of the tamper-resistant distributed ledger. Fig. 5 summarizes which data is stored on the blockchain via smart contracts. In short, all data that should be accessible to multiple independent parties (businesses) is saved on the blockchain via smart contracts. On the application client side, every time the transaction history is updated, the core layer queries all corresponding agreements and checks if one of the conditions is met by equalizing them with the transaction history table. If the last service that was added to the history table has resulted in a progress of the agreement's condition, this transaction is also forwarded to the blockchain and linked to the agreement. By this means the service providers can keep track of the completion state of all agreements they are involved in. The same is true for application clients if the interface of the business layer is adapted accordingly. One of the properties of the blockchain is that data once added can no longer be removed or changed. This is especially advantageous when data that should not be altered is stored, but can pose a problem for agreement handling, since often agreements should expire after a specific time or need to be canceled. We took care of this scenario to the extent that

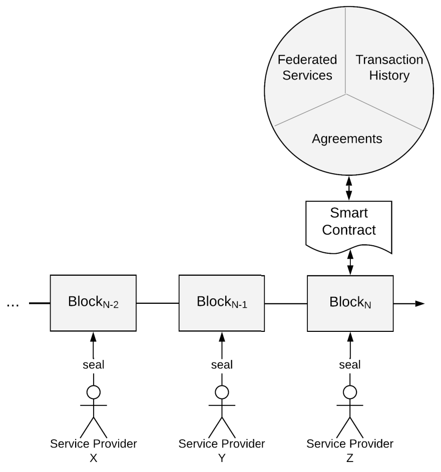


Figure 5. Smart contracts are used to store all cross-domain related service data onto the blockchain in a tamper-resistant manner.

a one-time cancellation request can be used which sets the agreement's validation flag to false. Subsequently, this makes the corresponding smart contract unusable, by blocking all further method calls.

IV. Implementation

There are two classes of entities accessing the core servers functionality: the application servers and the application clients through a RESTful interface, which is protected by basic HTTP authentication. Every call has to be authenticated using a secret API key, which is passed as a username in the basic HTTP authorization header. All communication between clients and the core server is done over HTTPS. The only exception is the device registration process, where a client certificate is not yet available. In this case, the registration-ticket is used for identifying the user. After registration, the application client uses an X.509 certificate for all subsequent TLS connections. The overall platform utilizes PKI combined with ECDSA as signature algorithm and SHA256 for hashing data.

We decided to rely on the already widespread Ethereum blockchain in the context of the blockchain part. We chose Ethereum's PoA Clique consensus algorithm, since it is suitable for achieving cooperation between several service providers in terms of speed, network scope and authorization handling. Each participating service provider becomes a sealer and is involved in the validation procedure. A local Ethereum blockchain was installed on Linux-based machines for the proof of concept we developed. To set up the network including creation of nodes, accounts and the genesis block, the Go Ethereum protocol was utilized. It is available as standalone terminal client under the name Geth. The interaction between the JAVA-based application code and the Ethereum blockchain was established using the web3js API. It is a lightweight, but highly modular Java library for working with smart contracts and interacting with nodes on the Ethereum network. Furthermore, for debugging the local blockchain network an IPC approach was chosen by accessing the local geth.ipc file with the Geth client. A bootnode was used for helping nodes to discover each other, since each node could have a dynamic IP. The bootnode itself runs on a static IP and can be used by other nodes as reference to find participating nodes.

Smart contracts were programmed in Solidity [15], a high-level programming language for the Ethereum Virtual Machine. It is statically typed, supports inheritance and user-defined types. A contract in the sense of Solidity is a collection of code (functions) and data (states) that is stored at specific addresses on the blockchain. They have to be installed in a one-time operation on all nodes by sending a transaction to the network. Once installed, modifications are no longer possible. We tested the contracts using the Remix browser application (www.remix.ethereum.org). In order to embed it into the server application the contract was compiled using the solc-compiler. During the compilation process a binary file and an application binary interface in JSON format are created. These outputs are required to generate JAVA wrapper classes with the web3j command line utility. Finally, these wrapper files can directly be integrated into an application.

V. Security Evaluation

Our platform acts as a cross-domain entitlement marketplace and integrates blockchain technology to further enhance the level of security and trust between different service providers. The business layer is composed of a client-server infrastructure, while the core is attached on top of it to provide the means for sharing different services. A security evaluation of the elaborated proof of concept is given in the following.

User and service related data from the business layer is tokenized for increased anonymity before being submitted to the core layer. Furthermore, these tokens are responsible for security goals such as authentication and authorization. On the one hand, authentication from the application client to the core layer is provided through A-tokens, which are extended certificates, tied to a particular device and issued in the course of a Kerberos-based authentication mechanism, ensuring that no real user data is forwarded to the core server. On the other hand, services are converted to S-tokens that identify the actual entitlement and the service provider responsible for the redemption. In our security model, the core layer communicates with the business layer over an encrypted channel (HTTPS). The tokens themselves (A-token, S-token), however, are not encrypted. This reduces additional computational overhead and is not security critical, since tokens are only composed of anonymized data. Furthermore, if an adversary had access to the core layer's transaction database he would not learn anything about the raw data, as it is obfuscated. Regarding transactions, only anonymized entries consisting of different ids and a timestamp are stored. Those that should be accessible by multiple parties are uploaded in a hashed format (SHA-256) to the blockchain. By doing so, participating service providers have the possibility to keep track of federated services and the completion state of agreements. If a device is lost or stolen, the possibility of spending existing tokens with that device cannot be prevented. This risk can be reduced by revoking a devices A-token in a timely manner, which subsequently invalidates all S-tokens automatically.

The overall architecture uses PKI with ECDSA for identifying communication participants and verifying the data exchanged. All tokens are core-server signed and can be verified with the server's public key. In order to prevent the issuance of interacting with a malicious core server application, clients may use public key pinning. A trusted root certificate is pre-installed on all devices. Additionally, all blockchain transactions are cryptographically signed and ensure that an attacker cannot corrupt the network. Note that while data integrity is not ensured in each node, since a single node can tamper with its local copy, we can still reduce the risk by means of a sufficient distribution of the data.

The decision making power is distributed from a central authority across multiple network participants, which strongly impedes fraudulent behavior. If hackers tamper with the block data, the network's consensus model will ensure that other nodes reject the bad node. Unlike other consensus algorithms where the most important resource is computing power (Proof of Work) or monetary resources (Proof of Stake) in the case of our Proof of Authority (Ethereum's Clique) approach, a set of pre-approved authorities is responsible for the integrity of the network. This means an adversary would need to gain control over other sealer's machines or persuade them to act maliciously, which makes an attack vector more improbable. We assume the blockchain to be tamper-resistant. This requires a sufficiently large network of trustworthy sealers. Any new service provider wishing to join the network, must first be approved by the other sealers, which gives us full control over which nodes can seal blocks. To make sure a malicious signer cannot do too much harm to the network any signer can sign at most one of a consecutive number of blocks. The same voting is applied when an authority node is removed from the network.

VI. Conclusion

Cross-domain service exchanges are often difficult to achieve due to rigid system boundaries and lack of standardization as well as a scant level of trust and privacy. In this paper we described a framework for issuing and sharing services from different application areas in the form of digital entitlements across heterogeneous applications. Our platform is subdivided into a business layer composed of a client-server system, where application specific features lie, and a core part acting as a common trusted layer for abstracting user and service data as well as for distributing different entitlements across multiple entities. The predominant trust mechanism of the platform is a public-key infrastructure (PKI) that maps an identity to a cryptographic public key using signed certificates. Authentication of devices as well as authorization of services are handled with anonymized tokens. Due to the generic nature of the tokens multiple heterogeneous entitlements can be represented, shared and seamlessly integrated in different systems and applications. Since the blockchain emerged as a tamper-resistant tool with great traceability and data integrity protection mechanisms, we extended the core layer with a decentralized blockchain part. A federated Ethereum-based blockchain with a Proof of Authority based consensus approach that involves participating services providers, is used to store and manage federated services and agreements. The communication between the centralized cloud and the decentralized blockchain part is enabled via a smart contracts based interface. Furthermore, from all transactions, including the issuing, forwarding, and redemption of cross-domain-services, data pointers are retained on the distributed ledger for verifying the data against fraudulent changes. Furthermore, by applying our framework we incentivize the use of cross-domain rewards. In this sense, a customer may receive additional offers and services coming from independent service providers as soon as pre-defined conditions between different service providers are met. Regarding future work we will look into end-to-end security strategies in order to provide means for exchanging confidential data (e.g. sensitive sensory data) across heterogeneous systems.

Acknowledgment

This work was written in tight cooperation with CISC Semiconductor GmbH and Graz University of Technology (Institute of Technical Informatics). Parts of this research have been elaborated within the research program STIP, funded by the Austrian Federal Ministry for Digital and Economic Affairs (BMDW) under the program COIN-Programmlinie "Netzwerke" (grant number 867914).

References

[1] G. Broad and H. Cambridge, "Blockchain A Beginners Guide," pp. 1–57, 2017. [Online]. Available: https://blockchainhub.net/blockchain-technology/

[2] V. Buterin, "On Public and Private Blockchains," 2015. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

[3] G. WOOD, "Ethereum: a Secure Decentralised Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, vol. 32, no. 10, pp. 1365–1367, 2018.

[4] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015*, pp. 180–184, 2015.

[5] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*, pp. 25–30, 2016.

[6] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in IoT using blockchain," *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 2017-Octob, pp. 261–266, 2017.

[7] G. C. Polyzos and N. Fotiou, "Blockchain-assisted information distribution for the internet of things," *Proceedings - 2017 IEEE International Conference on Information Reuse and Integration, IRI 2017*, pp. 75–78, 2017.

[8] M. Vukolic, "Rethinking Permissioned Blockchains," *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC '17*, pp. 3–7, 2017.

[9] "Hyperledger Fabric," 2018. [Online]. Available: https://github.com/hyperledger/fabric

[10] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, "Towards Scalable and Private Industrial Blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC '17*. Abu Dhabi: ACM, 2017, pp. 9–14.

[11] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, and S. Mahmoud, "A service-oriented middleware for cloud of things and fog computing supporting smart city applications," *2017 IEEE SmartWorld Ubiquitous Intelligence and Computing, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017 - Conference Proceedings*, pp. 1–7, 2018.

[12] A. Rech, M. Pistauer, and C. Steger, "Increasing Interoperability Between Heterogeneous Smart City Applications," in *Lecture Notes in Computer Science*. Tokyo: Springer International Publishing, 2018, pp. 64–74.

[13] European Parliament and European Council, "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016," p. 88, 2016.

[14] P. Canbay and H. Sever, "The Effect of Clustering on Data Privacy," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Miami: IEEE, 2015, pp. 277–282.

[15] "Solidity," 2016. [Online]. Available: https://solidity.readthedocs.io

Gamifying Connected Services Patterns

Alexander Rech
Graz University of Technology
Graz, Austria
rech@tugraz.at

Christian Steger
Graz University of Technology
Graz, Austria
steger@tugraz.at

Markus Pistauer
CISC Semiconductor GmbH
Klagenfurt, Austria
m.pistauer@cisc.at

ABSTRACT

Gamification, defined as the application of game design principles in non-gaming contexts, is a powerful tool for making systems lastingly attractive by exhausting rewards as soon as specific conditions are met. In the context of online or real world shops, customers are sometimes rewarded for their loyalty via additional offers or services. These offers often incentivize users to consume additional services and goods. However, most of the time they are limited to a single or a small set of vendors, which raises the question of how customers can be motivated to use services of different domains and service providers. Furthermore, the lack of standardization and increasingly stringent data protection rules aggravate the exchange of business related data with other systems. This paper introduces two patterns. First, a way for increasing joint-marketing purposes is elaborated focusing on a buying behavior based rewarding approach. Second, an overlay for existing client-server systems is described whose purpose is to anonymize and share services in form of digital tokens across different service providers, their systems, and users. In summary, said patterns establish the context for incentivizing the usage of cross domain services.

CCS CONCEPTS

• **Software and its engineering** → **Software organization and properties**; *Contextual software domains*; Software system structures; • **Information systems** → World Wide Web.

KEYWORDS

Connected Services, Data Privacy, Gamification, Cross-Domain Rewards

ACM Reference Format:
Alexander Rech, Christian Steger, and Markus Pistauer. 2019. Gamifying Connected Services Patterns. In *24th European Conference on Pattern Languages of Programs (EuroPLoP '19), July 3–7, 2019, Irsee, Germany.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3361149.3361163

1 INTRODUCTION

In the last years new emerging information and communication technologies have appeared and became part of peoples daily lives [2, 19, 20]. Smart access via mobile devices to different services available online (e.g. food order, ticket purchase) or those available locally (e.g. getting access to parking lots or specific areas) is no rarity anymore [12, 16]. From a business point of view many service providers try to arouse the customer's interest in their services using special offers or rewarding systems of different kinds without exploiting the full potential of an extensive inter-company rewarding system for themselves and their users. This paper discusses the following questions. How can service providers incentivize customers to utilize different services across independent companies? What can be done from a technical standpoint to facilitate a collaboration between different systems while focusing on privacy aspects? This is where the concept of gamification, which is based on the act of adding systemic game elements to non-game elements, comes into play. In our case gamification is strongly related to marketing techniques for encouraging the engagement between service providers, their products, services, and users. It is not competitive but focuses on cross domain rewards for participating users.

Challenge of gamifying the interconnection of different products and services. The concept of gamification in the sales domain builds upon the very common loyalty card that many brands offer, which keeps the customer rewarded inside the brand. The concept goes further by providing means to interconnect services from heterogeneous service providers, yielding benefits in terms of wider advertising scope and additional sales, due to an increased cross-domain collaboration. In this sense, a cross domain service exchange and rewarding layer could be utilized by service providers of heterogeneous domains in order to incentivize customers to utilize services of different brands.

Challenge of connecting heterogeneous services, while preserving privacy. Due to a variety of ubiquitous information processing systems, we have to deal with an increased data privacy awareness of users. Additionally, stringent data protection rules such as the European GDPR [5]) enforce protection and anonymization of user related data. From a technical point of view, we nowadays have to deal with an increasing jungle of heterogeneous systems. There is no common strategy to combine services and user pools of different domains in a privacy preserving way. Given a common trusted layer for client-server systems, strategies for anonymizing and sharing services in form of digital tokens can be applied.

Our contribution: Gamifying Connected Services Patterns. In this paper, we present two patterns: (i) the *Incentivizing Cross-Domain Service Access Pattern* and (ii) the *Privacy Preserving Service Access Pattern.* The first pattern describes the idea of a common overlayer for facilitating service and brand cooperation for joint-marketing purposes and a collective cross domain gamification experience for their customers. In contrast, the second pattern which is a proposed pattern, focuses on privacy aspects when services of diverse domains shall be exchanged between different systems and their users. Both patterns use the canonical way of pattern description. Therefore, the following sections explain the context of the patterns (sections 2.1 and 3.1), corresponding examples (sections 2.2 and 3.2), the problem and its forces (sections 2.3 and 3.3), the solution and its accompanying consequences (section 2.4 and 3.4), and resolved examples (section 2.5 and 3.5). The consequent section 4 is about related work and related patterns, while section 5 gives a concluding overview of the overall paper.

2 INCENTIVIZING CROSS-DOMAIN SERVICE ACCESS PATTERN

2.1 Context

It has been and always will be a goal for businesses to provide means for holding existing users and acquiring new ones. Despite the businesses' range of goods and services per se, making special offers is a strategy to arouse the customers' intrinsic motivation to remain loyal to a certain service or application. Gamification concepts are a way of integrating a certain level of playfulness in non-game contexts [4, 7]. They often aim towards goals of marketing and can therefore be used for special offer strategies. In todays agglomeration areas many independent information systems, such as systems responsible for providing bike or car rental services, food ordering, or smart parking, come together. Therefore, it is all the more important to provide tools for enabling higher collaboration between different service providers and motivate customers to try out new services eventually.

2.2 Motivating Example

In order to understand how a user would benefit from cross domain rewarding mechanisms let us slip into the role of a user for the following example. Take the case that you are in a place where many different services and products are available locally like in a shopping mall (see Fig. 1). You as customer can choose between several services, from parking or charging your car to eating something or going shopping in different shops, and many other services you probably do not even know about and haven't gone to yet. Some shops have rewarding programs. However, the problem is that these rewarding mechanisms often only include the products of the corresponding seller which means you get no discounts or special offers for other shops when purchasing them. You also have no incentive to enter the shops you do not yet know. Wouldn't it be convenient being rewarded with cross-domain rewards for your loyalty, maybe even in a completely automated way as soon as predefined conditions are met? For example, you could get a reward from shop A, by consuming a specific number of services from shop B. Wouldn't it be advantageous if these offers also incorporated stores that you didn't know yet, thus making them more attractive for your next visit? The described use case deals with a wide range of physically present service providers and shops. However, the same questions arise also in an online scenario where products, vouchers, etc. can be obtained via completely virtual channels.

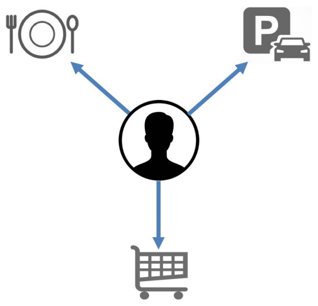


Figure 1: A customer consumes various services from different services providers (food, parking, groceries, etc.). Since there is no collaboration between different service providers, he unfortunately is not rewarded for obtaining cross-domain products.

2.3 Problem

Customers tend to be creatures of habit. Once they are used to a service and know how to obtain it, they often no longer look around for new services without an incentive, a specific reason, or an advantage.

2.3.1 Forces.

- **Lack of standardization**: Interaction/cooperation between service providers beyond their own application area boundaries is difficult to achieve due to a lack of standardization. While many custom-tailored solutions for specific application areas do exist, there is no common strategy for exchanging services of different domains, which also includes the possibility for cross-domain rewards.
- **Service provider behavior**: Since service providers do not want to lose business, they have the tendency to be concerned or at least cautious about cooperating with other service providers. Therefore their service offers often have a scope limited to a single company or a small set of vendors.
- **Effort-intensive rewarding systems**: In case rewarding systems exist the process of claiming rewards is often associated with effort for the customers, since they have to be aware of the current available services and offers as well as their conditions, and must actively claim them.
- **Unknown availability of services**: Users often simply do not know which services and corresponding offers are available, especially in the case of services that are available locally only.

2.4 Solution

A common trusted layer should provide the possibility to (i) manage services of different domains and service providers and (ii) enable cross-domain rewards according to the customers' buying behavior.

A common layer on top of existing systems should act in between different service providers (e.g. brands, merchants, etc.). This layer can be seen as a service market place, thus providing the possibility to publish services and make them visible for other service providers of different domains. We call this common layer the Connected Services Engine (CSE). The CSE should not only maintain a list of available services but also introduce the possibility to enable purchase based cross-domain rewards. It should only be accessible to participating service providers and their user pools. Its requirements are depicted in the following:

- **Authentication**: Only authenticated service providers and users shall have access to the CSE.
- **Authorization**: Each service the CSE manages stands for a specific permission (e.g. permission to get a product, redeem a service, etc.). Therefore, the common layer shall provide means to trigger the creation of these permissions and to uniquely associate them with specific customers.
- **Transaction Handling**: The obtainment and redemption of services as well as transaction-related data should always be traceable, independent from whom or from which system the transactions were initially triggered. This is necessary for applying a purchased-based rewarding mechanism.
- **Service Mapping**: The CSE should be able to map services of different domains with different customers. This is required for a cross-domain service exchange.

A *service* is the connecting link between a *service provider* and a *user*. These terms are explained in the following:

- **Service provider**: Entity which defines and provides services. Each service provider should have the possibility to list its services on the CSE and to manage their properties such as price, name, availability, etc. Additionally, a service provider decides on the scope of his services. The scope determines which additional entities (of other domains and systems) are allowed to obtain or access the service.
- **User**: Entity which obtains and consumes services from a service provider and has the possibility to get cross-domain rewards according to its purchase behavior. Due to the CSE the user may not only see services from other domains, but is also aware of cross-domain offers and their condition.
- **Service**: Product, good, voucher, etc. of a specific service provider. There are two ways of how services can be accessed. On one hand, users may obtain services directly from the CSE. On the other hand, service providers may obtain them and forward them to their users as a reward.

This pattern describes a mechanism for service providers to grant service access to other service providers while providing the possibility to trigger cross-domain rewards for different customers. Fig. 2 gives an overview on the most important communication participants and their interrelationship. While the user obtains and redeems services via interaction with a dedicated service provider, the CSE is the point of contact for different service providers. The CSE should therefore provide a communication channel for each participating service provider in order that a description of their services can be uploaded. Additionally, a global transaction registry should be maintained on CSE level. It can be used to check which specific service was obtained or redeemed by whom. Cross-domain rewards can be issued depending on conditions the service provider may specify as well as the purchase behavior of the user. For example, when a costumer obtains specific products from one shop (system A), additional offers from another independent shop (system B) could be unlocked. In this sense, the CSE is responsible of keeping track of all transactions and notifying the corresponding service providers in case conditions for rewards are fulfilled.

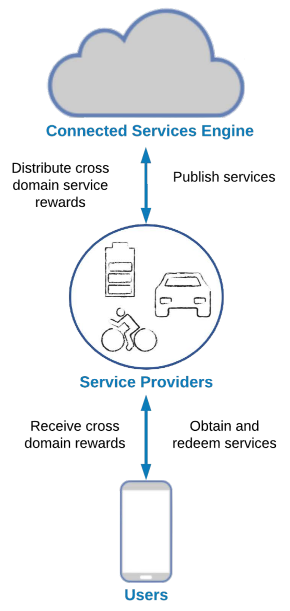

Connected Services Engine
Distribute cross domain service rewards
Publish services
Service Providers
Receive cross domain rewards
Obtain and redeem services
Users

Figure 2: Independent service providers may interact with the Connected Services Engine (CSE) to publish their services and distribute cross-domain service rewards for their users.

Different ways on how service providers and their users may interact with the CSE are possible (see also [15, 17]), which are additionally shown in Fig. 3:

- **Service Offer.** This approach provides registered service providers (and their systems) with the possibility to forward services available on the CSE to their users. In this sense, loyal customers can be rewarded with services of other companies. Via the CSE the service provider is able to get an overview about other available services, may obtain them, and forward the permissions to use them as reward to his user(s) for free or for better conditions. Alternatively, there is the possibility that a user may directly obtain the federated service. This approach could also be extended with a point-based system, meaning that for each service obtainment the user would be able to gather points, which in further consequence could be spend on new services.
- **Service Agreement.** In order to provide a way to react to specific purchase behavior of customers and establish an

automated service based rewarding system, each purchase has to be noted inside a transaction table. Additionally, a table for storing the service trigger conditions should be created. The main idea behind this is that there should be the possibility to issue condition-dependent service rewards in an automated way. The condition, for example, may specify how many services of a service provider need to be acquired before a reward is issued. The information whether the condition is met or not can be retrieved by the transaction table. Due to this transaction table all progresses can be tracked by the service provider and also their users. Finally, as soon as a condition is fulfilled, the service specified inside the agreement can be obtained by the host service provider (and his system), and forwarded to the corresponding user that fulfilled the condition. This way, the reward is issued automatically when the reward condition is met even if the completion state is not actively tracked.

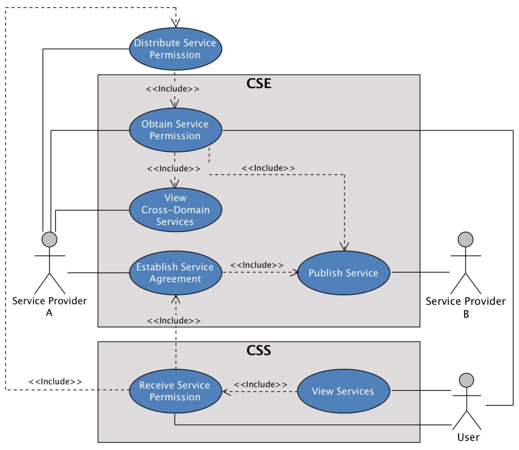


Figure 3: Overview about Service Offer and Service Agreement procedures involving a user and different service providers [17].

2.4.1 *Consequences.*

This section addresses the consequences of the *Gamifying Connected Services Patterns* pattern, subdivided into benefits and liabilities. The benefits of this pattern are:

- Due to the additional common layer put on top of existing systems and the possibility of exchanging cross-domain services, multiple services from one service provider can be obtained/used by other service providers/users, despite the custom implementations of their systems. This counteracts force **lack of standardization**.
- Through the CSE and the possibility to share cross-domain services more users can be addressed. With a higher user reach also a higher revenue can be expected by participating service providers. Additionally, by partnering up with other businesses or simply referring to their services, new business relationships can be created. Altogether, this counteracts force **service provider behavior**.
- Users can directly be rewarded with cross-domain services by their service providers, which minimizes the effort on user side and mitigates force **effort-intensive rewarding systems**. Access to these rewarding services can be triggered by service providers manually (*Service Offer*) or in an automated way (*Service Agreement*) by specifying trigger conditions.
- The scope property of a service stored on the CSE can be used by service providers to determine which entities are able to access a specific service, as long as they are able to access the CSE. In this sense, users may see available services or special offers, which counteracts the force **unknown availability of services**.

The liabilities of this pattern are discussed in the following:

- The cooperation between service providers active in the same sales segment will not increase by applying this pattern. This is because the pattern focuses on increased readiness to exchange cross-domain services (see force **service provider behavior**).
- The impact of the force **unknown availability of services** depends on how many service providers intend to participate and how they manage the scope property of their services.

2.5 Example Resolved

In order to motivate users to try out new services, different service providers may apply an inter-company-wide rewarding system based on the users' buying behavior. Let us apply the solution discussed in section 2.4 to the example of section 2.2 for the *Service Offer* approach. Assume that you (from a user point of view) want to go for a shopping spree in a shopping mall (see Fig. 4). You drive there by car and after successfully parking it in the underground garage you enter the mall. After a few hours of intensive shopping, you go to a restaurant to grab some food. Due to the CSE, both merchants (*restaurant* and *parking* service providers) know about the other party's services and are able to obtain the permission to issue corresponding vouchers. The moment you obtain the food, the employee of the restaurant may reward you with an additional parking voucher for your loyalty. The voucher can be used to exit the shopping mall for free (or for better parking conditions). Due to the global transaction registry of the CSE, it is possible to trace at any time which services were initially obtained and consumed by whom. This is also important for billing.

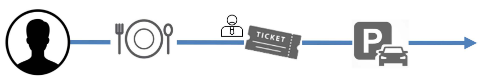


Figure 4: Service provider A (Restaurant) forwards a digital voucher to one of his customers for accessing the service (Parking) of service provider B (Garage).

Another example describing the *Service Agreement* approach is shown in Fig. 5. In this case, you purchase different goods from a certain vendor. As soon as predefined conditions are met (e.g. you have rented a car three times) you will be rewarded automatically in form of an additional service of another service provider (e.g. free e-bike ride) without actively having to claim the reward. Again the global transaction table in combination with the predefined conditions can be used for this.

Figure 5: A user automatically receives a reward (e-bike-rental voucher) by utilizing or consuming products or services from another service provider (car rental) as soon as predefined conditions are met (rented car x-times).

In summary, cross-domain offers from different businesses can be unlocked for customers, depending on their purchase/consuming behavior.

2.6 Known Uses

In the electronic marketplace area, for instance, integration methodologies between different services providers are often associated with advantages. When businesses merge into clusters they often have a decreased management overhead with less setup and maintenance costs, for instance. However, the downside is that such integration technologies and frameworks are often associated with adaption costs, especially for businesses with rigid standards [6]. Nevertheless, more and more centrally managed coalition programs are elaborated for cross-domain shopping and rewarding [11, 14, 23]. These approaches enable the customer to collect a common currency (often point-based) for obtaining specific goods and services of partnered up businesses and merchants. The more goods and products are consumed, the more points can be unlocked and credited to the user's account. The collected points can in turn be spent on various products, trigger the creation of new coupons, or be converted to real money through interaction with the rewarding platform's mobile app or website. All of the mentioned coalition programs have one central layer in common where different services can be obtained and redeemed.

3 PRIVACY PRESERVING SERVICE ACCESS PATTERN

3.1 Context

In e-commerce environments often client-server systems are used to engage customers. Among other components, such systems often consist of a mobile or a web application for the end-user as well as a corresponding server unit. While the application for the user is responsible for visualizing and obtaining services, the server-side application provides a dedicated interface for accessing different services online as well as the possibility for service providers to manage user and service related data. These systems are often tailored for specific application areas and target groups, which implies that their offers in form of products, vouchers, or other services usually have a scope limited to a single company or a small set of vendors. The context of this pattern foresees that the *Incentivizing Cross-Domain Service Access Pattern* has already been applied, which provides a generic common layer working on top of different systems. Complementary to it, this pattern addresses service providers and their client-server systems that intend to exchange service and user related data in a privacy preserving way from a technical perspective.

3.2 Motivating Example

In our example we have different client server systems (CSS) coming from heterogeneous domains as well as one common layer, the connected services engine (CSE). Let's assume that each CSS provides the functionality to register and login users as well as the possibility to let them obtain services of the system's domain. By contrast, the CSE - from an architectural perspective - is put on top of a CSS and can be utilized as service marketplace for obtaining and forwarding services between different participating systems. See Fig. 6 for an overview of the architecture. However, the necessary functionality to obfuscate privacy critical datasets has yet to be integrated.

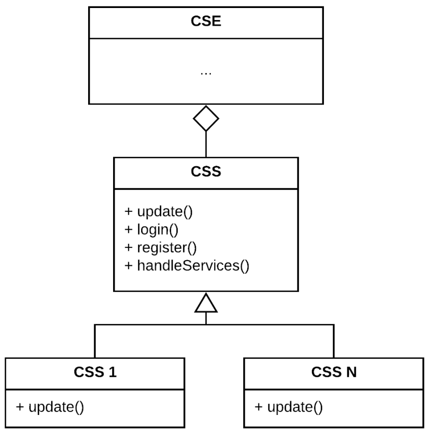


Figure 6: Interaction between different client server systems (CSSs) and the overlaying Connected Services Engine (CSE).

3.3 Problem

Due to increasingly stringent data protection rules and an increased privacy awareness on user side, privacy preserving data exchange is crucial for client server systems in the sales domain that intend to exchange user data among each other. In this sense, a lack of privacy preserving information exchange may hamper collaboration between different brands, consequently impeding a federated access to services of different service providers.

3.3.1 *Forces.*

- **Jungle of diverse data**: Diverse datasets from different client server systems are difficult to parse and compare with each other. Services cannot be shared across several systems and their users without a common basis, rule, or interface.
- **Data security**: Interfaces to other systems may lead to security risks such as disclosure of confidential data, fraudulent data manipulation, or system failures, especially when secure authentication and authorization is not provided.
- **Diverse interfaces**: For each external server to which a communication channel should be established, different interfaces have to be integrated. This complicates the integration procedure and makes the adaption of the host system's boundaries more time and cost consuming.
- **Abstraction = Information loss**: It is difficult to determine which abstraction level is required to guarantee the privacy aspect of user data , while still having enough non-anonymized information the system needs to perform its tasks.

3.4 Solution

The proposed pattern provides existing client-server systems with the possibility to share services with each other and their user pools in a privacy preserving way. This is done by introducing a cloud based layer that is responsible for anonymizing services and users from different client-server systems. Client-server systems have to register themselves at the cloud layer that keeps a list of all subscribed entities. They will be notified of state changes and also regarding the issuing procedure of service-permissions.

Our starting point is a client server based system (CSS) such as the ones found in various sales environments. It mainly consists of two components, an application client (user application) for obtaining services as well as a corresponding application server infrastructure where the user and service data is stored and managed. Companies are often reluctant to carry out severe adaptions of a systems' internal functionality and interfaces in order to connect with other systems. Therefore, an additional trusted layer (CSE) is introduced and put on top of the CSS. This external cloud-based unit may interact with different CSSs (e.g. via HTTP) and anonymizes the data transferred. The explanation of the proposed solution is divided into three steps. First, the registration procedure of a service provider is described, who is responsible for sharing services across independent parties. Second, the integration of a CSS-user is explained. This is important for the obtainment of cross-domain services. Last but not least, the focus will lie on a description of how cross-domain services can be managed.

Service Registry Phase. The CSE comprises different services and is responsible for storing them in a global service registry and providing access to it. The *Service Registry Pattern* [18] can be applied for providing means to store and handle a set of different services. For the following two reasons automated service discovery mechanisms are not required, neither on CSS nor on CSE side. First, a CSS only needs to know the location of the CSE. Second, service providers interested in accessing cross-domain services and publishing their own services on the CSE can register themselves by adding their shop, company, and service information, and are therefore already known. Each service added on the CSE gets a service identifier mapped together with a unique identifier of the service provider. Additionally, the scope of the added entries shall be set to private by default. However, it should be possible to lift their privacy settings and make them public. This way, other providers are able to find services and acquire them.

Registration user. In order to register a user on the CSE for receiving cross-domain rewards, a trusted communication channel between client and server of the CSS and the CSE has to be enabled. A Kerberos based registration procedure can be applied [8]. In this sense, the CSS is responsible for authenticating the client and retrieving a session ticket for accessing the interface of the CSE. Subsequently, an anonymous user entity is created on CSE level after submitting the session ticket and a unique CSS-user identifier. This newly created anonymized entity on CSE does not contain any specific user data like email address or name and is only used for providing the possibility to route a reward to the right user on the CSE. The only entity that knows the link between real and anonymized user objects is the CSS that triggered the registration procedure. Last but not least, a CSE-access key for the user (e.g. API-key) is issued that enables further direct communication between him and the CSE.

Cross-Domain Service Handling. The CSE is composed of several registries. Due to the distributed login procedure an anonymized *user registry* on the CSE can be maintained. Furthermore, also services are stored inside the *service registry*, which should be made accessible by the CSSs. Each time a client or a service provider wants to obtain a service, the CSE is notified by the CSS and triggers the generation of the corresponding access permission. In this context, a CSE-signed service token is created and transferred directly to the client. This uniform token is linked to the service, the corresponding service provider, and the anonymized user entity. Additionally it has a validity period, authorizing the user to get access to a specific service within the specified time frame. The corresponding CSS will be notified in case tokens are issued or state changes occur (e.g. token state switches between VALID, EXPIRED, REDEEMED, etc.).

Each activity involving the generation or redemption of a service can be stored in a *transaction registry* by the CSE. With this ever increasing transaction history database it can be determined which entity purchased or redeemed which specific service of which service provider and how many of them. A *single-app* approach where heterogeneous services are integrated within one application (same CSS user object), as well as an *inter-app* approach, where services are exchanged between different independent platforms/applications (different CSS user objects), are feasible. Latter, requires a privacy preserving mapping between the anonymized CSE-data and the CSS user entities (containing the real user data, e.g. mail, name, etc.). The mapping could be stored centrally on the CSE. In this case, a unique identifier (e.g. email) of the CSS-user object in hashed format as well as a unique service provider identifier can be put together with the anonymized CSE-user object. By calculating the hash of the CSS-user object during interaction of the CSS and the CSE and comparing it with the already stored one, the CSE can associate the CSS-user with the correct anonymized user object. In this way, even if this table is stored on the CSE, only the business in charge of the creation of the user account knows the real user data, while obfuscating the user's identity for the CSE and thus also for all other participating businesses. Alternatively, the link between real user data (CSS-user object) and anonymized user object (CSE-user object) may be stored directly on each CSS. In this case, a dedicated communication channel between CSS and CSE has to be provided to retrieve the required link between these objects.

3.4.1 Consequences.

This section is about consequences of this pattern, subdivided into benefits and liabilities. The benefits of the proposed *Privacy Preserving Service Access Pattern* are:

- By applying this pattern an increased level of trust can be achieved due to the fact that no real user information like name, email, etc. is stored on the CSE. By contrast the CSS remains in control of the real data. In order to be able to forward services from the CSE to users of a specific CSS, a mapping between real and obfuscated data is saved in a privacy preserving way. This counteracts force **abstraction = information loss**.
- From an architectural perspective, the CSE is on top of existing CSSs, thus entering a one-to-many relationship with them, which is loose by default. Since a CSS communicates directly to the CSE, without having to interacting with other systems, the number of adaptions regarding the CSS's interfaces can be reduced. This affects force **diverse interfaces**.
- The overall platform provides means for secure authentication and authorization. The tokenized service data shared between systems is signed by the CSE. Furthermore, all CSS users are registered via a secure distributed registration procedure to enable a secure link between application client, application server, and the CLE. These points affect the force **data security**.
- The force **jungle of diverse data** is counteracted due to the abstraction procedure of services and users. The anonymized tokens on the CSE are harmonized and have a uniform format.

The liabilities of this pattern are discussed in the following:

- For enabling tailor-made advertisements and service access additional datasets need to be shared across different parties (e.g. destination, GPS, gender, age). Furthermore, it is not possible to carry out a detailed data analysis on CSE layer since only obfuscated datasets are available. This affects force **abstraction = information loss**.
- Services are written using a variety of languages, frameworks, and framework versions. Further interface descriptions are required to facilitate the integration procedure between different CSSs and the CSE. This affects force **diverse interfaces**.

3.5 Example Resolved

Fig. 7 shows the interaction between a CSS and the CSE. The CSE enables the CSS to define and create services inside the service registry of the CSE. Furthermore, information about the available services can be retrieved at any time. Each transaction with the CSE is stored inside a transaction table in order to know which entities communicated to the CSE. This transaction table is also utilized in conjunction with rewarding rules/conditions. They can be created by the CSS and define under which circumstances a user should be rewarded. Due to the abstraction process and the absence of user data on the CSE the real identify of each user can be protected. The CSE is only in possession of an anonymized user entity. Only the corresponding application server of the CSE knows the real identity.

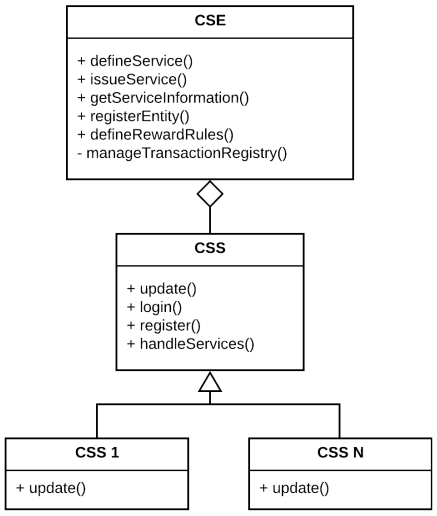


Figure 7: Interaction between different CSSs and the overlaying CSE including the handling of cross-domain services in a privacy preserving way.

4 RELATED WORK

The solution approaches described in this paper are partly based on the design of the software architectures explained in [15, 17], where different layers are responsible for abstracting and exchanging service tokens between heterogeneous client server systems. The interaction between the server-based components of the system can be realized with the *Observer Pattern*, responsible for distributed event handling and the *Whiteboard Pattern* which is derived from the former [13]. This pattern defines a subject and an observer. When a subject changes state, all registered observers are notified and updated automatically. On one hand, the subject is responsible for maintaining a list of observers and notifying them of state changes. On the other hand, it is the responsibility of observers to register themselves on a subject in order to get notified of state changes and to synchronize their state with the subject's state as soon as they are notified. Furthermore, the *Service Registry Pattern* [18] can to be taken into account, for informing participating communication participants about all available service instances.

Furthermore, a Kerberos-based protocol [8] can be used to enable distributed authentication between client, server, and the CSE. This authentication scheme follows the basic *Pattern of Brokered Authentication* [10]. The following patterns are related to the topic of creating distributed service-oriented applications. The patterns [21, 22] focus on engineering software for the cloud. By contrast, [3] elaborates a pattern language for distributed computing, while [9] focuses on resource management. Also gamification methodologies have gained popularity in the development of enterprise information systems [1].

5 CONCLUSION

On account of poor standardization and cooperation between different service providers the widespread use of cross-domain rewarding approaches is still in its inception. In this sense, today's rewarding systems often have a scope limited to a single company or a specialized application area, and therefore target only a focused set of users. Furthermore, data privacy is becoming more and more important, especially when user related data has to be exchanged between different systems. In order to increase a gamified, federated service exchange across several user-vendor systems while preserving the users' privacy, the patterns of this paper can be applied. First, the *Incentivizing Cross-Domain Service Access Pattern* introduces a layer –a federated service marketplace– that can be used by service providers to enable cross-domain service usage. Services coming from different application areas can be unlocked in a supervised or automated way for rewarding the loyalty of users, eventually resulting in the augmented usage of cross-domain services while simultaneously yielding benefits for the participating vendors in terms of increased advertising scope and additional sales. Second, the *Privacy Preserving Service Access Pattern* is based on the first pattern and elaborates a method for the privacy preserving processing of user data between multiple client-server systems. Summarized, this paper explains a purchase based gamification method in order to encourage the engagement between different brands and to incentivize customers to utilize services from different domains.

6 ACKNOWLEDGMENTS

Parts of this work have been elaborated within the H2020 project STEVE (Grant Agreement 769944). Furthermore, we would like to express our greatest gratitude to our shepherd Tiago Boldt Sousa who provided many insightful advices for this work.

REFERENCES

[1] Darius Ašeriškis and Robertas Damaševičius. 2014. Gamification Patterns for Gamification Applications. *Procedia Computer Science* 39, C (2014), 83–90. https://doi.org/10.1016/j.procs.2014.11.013

[2] Dario Bruneo, Salvatore Distefano, Francesco Longo, Giovanni Merlino, and Antonio Puliafito. 2017. IoT-cloud Authorization and Delegation Mechanisms for Ubiquitous Sensing and Actuation. *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016* (2017), 222–227. https://doi.org/10.1109/WF-IoT.2016.7845494

[3] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. 2007. *Pattern Oriented Software Architecture: A Pattern Language for Distributed Computing*. Vol. 4.

[4] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From Game Design Elements to Gamefulness: Defining Gamification. *Proceedings of the MindTrek Conference 2011* (2011). https://doi.org/10.1145/2181037.2181040 arXiv:ACM 978-1-4503-0816-8/11/09

[5] European Parliament and European Council. 2016. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016. , 88 pages.

[6] Jingzhi Guo. 2007. Business Interoperability on E-Marketplace. *Research and Practical Issues of Enterprise Information Systems II Volume 1* (2007), 257–267. https://doi.org/10.1007/978-0-387-75902-9_26

[7] Kai Huotari and Juho Hamari. 2012. Defining Gamification - A Service Marketing Perspective. *In Proceeding of the 16th International Academic MindTrek Conference* (2012). https://doi.org/10.1145/2393132.2393137 arXiv:cond-mat/0312065

[8] MIT Kerberos. 2019. Kerberos: The Network Authentication Protocol. Retrieved July 23, 2019 from https://web.mit.edu/kerberos/

[9] Michael Kircher and Prashant Jain. 2013. *Pattern-Oriented Software Architecture: Patterns for Resource Management*. Vol. 3.

[10] Microsoft Corporation. 2005. Brokered Authentication. Retrieved August 20, 2019 from http://guides.brucejmack.net/SOA-Patterns/WSSP/9.0BrokAuthKerberosDoc.htm

[11] Miles & More. 2019. Miles & More frequent flyer and awards programme. Retrieved April 6, 2019 from https://www.miles-and-more.com

[12] Nader Mohamed, Jameela Al-Jaroodi, Sanja Lazarova-Molnar, Imad Jawhar, and Sara Mahmoud. 2018. A Service-Oriented Middleware for Cloud of Things and Fog Computing Supporting Smart City Applications. *2017 IEEE SmartWorld Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation* (2018), 1–7. https://doi.org/10.1109/UIC-ATC.2017.8397564

[13] OSGi Alliance. 2004. Listeners Considered Harmful: The Whiteboard Pattern. (2004). Retrieved March 10, 2019 from https://www.osgi.org/wp-content/uploads/whiteboard1.pdf

[14] Payback GmbH. 2019. Payback. Retrieved April 28, 2019 from https://www.payback.at

[15] Alexander Rech, Markus Pistauer, and Christian Steger. 2018. Increasing Interoperability Between Heterogeneous Smart City Applications. In *Lecture Notes in Computer Science*. Springer International Publishing, Tokyo, 64–74. https://doi.org/10.1007/978-3-030-02738-4_6

[16] Alexander Rech, Markus Pistauer, and Christian Steger. 2019. A Distributed Framework Towards Local and Online Service Access. *24th IEEE Conference on Emerging Technologies and Factory Automation - Workshop on Trustable Wirelessly Connected Industrial IoT* (2019), 1715–1721.

[17] Alexander Rech, Christian Steger, and Markus Pistauer. 2019. A Decentralized Service-Platform Towards Cross-Domain Entitlement Handling. In *2nd IEEE Conference on Blockchain*. IEEE. https://doi.org/10.1109/Blockchain.2019.00069

[18] Chris Richardson. 2019. Pattern: Service Registry. Retrieved July 20, 2019 from https://microservices.io/patterns/service-registry.html

[19] Karen Rose, Scott Eldridge, and Lyman Chapin. 2015. *The Internet of Things: An Overview - Understanding the Issues and Challenges of a More Connected World.* Technical Report. 1–80 pages. https://doi.org/10.1017/CBO9781107415324.004 arXiv:arXiv:1011.1669v3

[20] Savio Sciancalepore, Giuseppe Piro, Daniele Caldarola, Gennaro Boggia, and Giuseppe Bianchi. 2017. OAuth-IoT: An access control framework for the Internet of Things based on open standards. *Proceedings - IEEE Symposium on Computers and Communications* (2017), 676–681. https://doi.org/10.1109/ISCC.2017.8024606

[21] Tiago Boldt Sousa, Ademar Aguiar, Filipe Figueiredo Correia, and Hugo Sereno Ferreira. 2016. Engineering Software for the Cloud - Patterns and Sequences. *Latin American Conference on Pattern Language* 11 (2016), 8.

[22] Tiago Boldt Sousa, Hugo Sereno Ferreira, Filipe Figueiredo Correia, and Ademar Aguiar. 2017. Engineering Software for the Cloud: Messaging Systems and Logging. *ACM International Conference Proceeding Series* Part F132091 (2017), 1–14. https://doi.org/10.1145/3147704.3147720

[23] WeeConomy. 2019. wee. Retrieved June 4, 2019 from https://www.wee.com

A Distributed Framework Towards Local and Online Service Access

Alexander Rech
Graz University of Technology
Inst. of Technical Informatics
Graz, Austria
a.rech@cisc.at

Maximilian Kammerer
Graz University of Technology
Inst. of Technical Informatics
Graz, Austria
m.kammerer@tugraz.at

Markus Pistauer
CISC Semiconductor GmbH
Klagenfurt, Austria
pistauer@cisc.at

Christian Steger
Graz University of Technology
Inst. of Technical Informatics
Graz, Austria
steger@tugraz.at

***Abstract*—Seamless integration of smart devices into our daily lives is becoming increasingly ubiquitous. In fact, different methods for accessing infrastructure and mobility services as well as connected information systems are part of today's smart cities. The problem is, however, that due to environmental and use case restrictions, no standard authentication and authorization scheme for accessing heterogeneous services (car access, charging, obtaining sensory data) can be applied. In this paper we present a distributed service management framework that provides different means for authentication and authorization to services, infrastructure, and datasets. The platform acts as a scalable trusted cloud layer that enables applications to access online and locally available smart city services. It issues and manages digital tokens for authenticating and entitling connected devices to access these services. Our developed proof of concept is a hybrid system consisting of a multi-layered cloud environment as well as mobile devices acting as clients (service-holders) and kiosks (service-redemption units) for redeeming services online (e.g. access data) or locally (e.g. access area). Even though we use a multi-layer approach, we achieve short waiting times for the user underlining the usability aspect of the developed prototype.**

***Index Terms*—Authentication, Authorization, Cross-Domain Service Access**

I. Introduction

The subject of securely connecting different, independent smart city services, spanning from topics such as smart access and Mobility as a Service (MaaS) is being worked on and makes a big part of the evolution of cities into smart cities. A service in our case is defined as entitlement for accessing a specific *resource* available online or locally, e.g. access a *vehicle*, enter a *parking lot*, be entitled to charge a vehicle at a *charging station*, or be authorized to receive specific *data*. Due to a variety of different applications, use cases, and environmental restrictions regarding connectivity no standard authentication and authorization procedure for heterogeneous service access is applicable. In order to facilitate seamless access to locally and online available services and resources in different environments and for different applications, the co-modality concept proposed in this paper combines different authentication and authorization methodologies. A cloud-based layer acts as central authentication and authorization unit, responsible for issuing and managing different digital tokens, while locally available devices are used for obtaining and redeeming these tokens.

On the one hand, tokens can directly be distributed to client devices for enabling local service redemption without the need of an active internet connection. In this case, the communication between the client and the redemption unit is established via the local wireless standard Bluetooth Low Energy. Secure authentication is achieved via a dedicated challenge response mechanism. On the other hand, our approach also provides a secure way for redeeming services online. Regarding distributed authorization, we take advantage of the open-standard protocol OAuth 2.0. It is used to let client devices access data sets and services from remote servers. This is done via a RESTful interface and a dedicated access token, which is issued online and grants the clients access to specific datasets (e.g. sensory data). This way, application related confidential information (data, encryption keys, etc.) may remain on service provider side, without the need of storing them in one central place. At the same time the risk of single point of failures is minimized.

This paper is subdivided into the following sections. Section II discusses related work while section III presents the design choices of our approach. Subsequently, section IV is about implementational aspects and section V includes a performance evaluation of our proof of concept. Last but not least, we summarize our ideas and give information on future work in section VI.

II. Related Work

Today we are dealing with an ever increasing amount of connected devices and their applications. The so-called Internet of Things (IoT) is an integral part of the technological progression we are experiencing nowadays and it interconnects heterogeneous smart devices with each other. Experts expect the market value of IoT applications to reach 14.4 trillion dollars by 2022 [1].

Especially agglomeration areas profit from the interaction between smart connected devices. This interaction facilitates the provision of smart city services (e.g. smart parking, improved traffic congestion and waste management) thus accelerating the development of ordinary cities into smart, connected cities. In this context, paper [2] defines a cross domain protocol for facilitating efficient retrieval of distributed sensory data in the context of a smart city. Paper [3] proposes a mobility management platform for connected cities. It allows city governments to formulate mobility policies that let them influence the amount of people that visit certain spots in their city. They integrate their framework into existing applications that enable the utilization of city resources. Data of mobile devices is collected and evaluated in order to adapt route planning for users, for example.

Efforts to increase the interoperability of heterogeneous systems has also led to the creation of a lightweight web service that can be used to manage sensory data [4]. The service allows users to connect to multiple data sources by making use of different data adapters and querying data from several sources. These sources include already existing sensor and IoT platforms as well as running databases and data located on local machines or clouds.

Since data transfer between different devices is at the heart of IoT applications, also tamper-resistant authentication and authorization mechanisms play an important role for trusted data exchange in both local device communication and device to cloud interaction [5], [6]. Paper [7] provides an approach for identity authentication in smart cities. An access key is created by combining different authentication types, enabling access to different services with one single device. This access key is stored on a specific computing device (e.g. a smartphone or a smart card) which subsequently authenticates itself on a backend access management system.

Regarding distributed authorization, one way to enable it across different systems is with the OAuth 2.0 protocol. It can be used to enable distributed authorization between a variety of applications. Service providers like Google and Facebook utilize it to handle authorization across different web services, for instance. The OAuth 2.0 protocol is also focus of scientific research. Examples include distributed privacy preserving authorization handling in e-Health applications where different pseudonyms are introduced for each user [8] as well as the use of OAuth to increase the security for API access in order to provide stronger protection against fraudulent users [9].

Since secure and distributed authorization is important also for the IoT, OAuth can be applied in this context. The authors of paper [10] describe how IoT systems may utilize OAuth 2.0 to authorize themselves. They point out that controlled access to constrained resources is problematic because conventional solutions for cloud and web services are difficult to utilize. This is mainly due to missing bandwidth and computationally constrained devices. To mitigate this problem, the authors propose a gateway which collects information provided by smart devices and controls the access requests to the datasets via OAuth 2.0 authorization. Similar to this, paper [11] focuses on OAuth in an IoT setting to facilitate the integration of this protocol for constrained devices. It proposes a framework for enabling IoT devices to utilize an external OAuth authorization framework based on HTTP/CoAP without having to implement the logic on each device.

III. Design

This chapter explains the design of the overall framework, starting with the architectural setup (III-A) and wireless communication interfaces (III-B). Depending on the specific services or infrastructural requirements the redemption procedure can be done online (III-C) or offline (III-D).

A. Design of platform and its key components

The high level objective of our solution is to provide different methods for accessing heterogeneous services. We distinguish between local and online redemption methodologies for authorizing systems and their users to access infrastructural and cloud-based services. Depending on the application and the use case, different sets of tokens for authentication and authorization can be created and are managed by a central trusted cloud-layer. A custom public key infrastructure (PKI) in combination with local authentication and authorization tokens form the basis of the platform's offline security-capabilities and ensure the identity of all communication participants. OAuth 2.0 is integrated into the framework for distributed online authentication and authorization. The design of our framework – partly based on the architectural proposal of [12] – is depicted in Fig. 1 and discussed in the following:

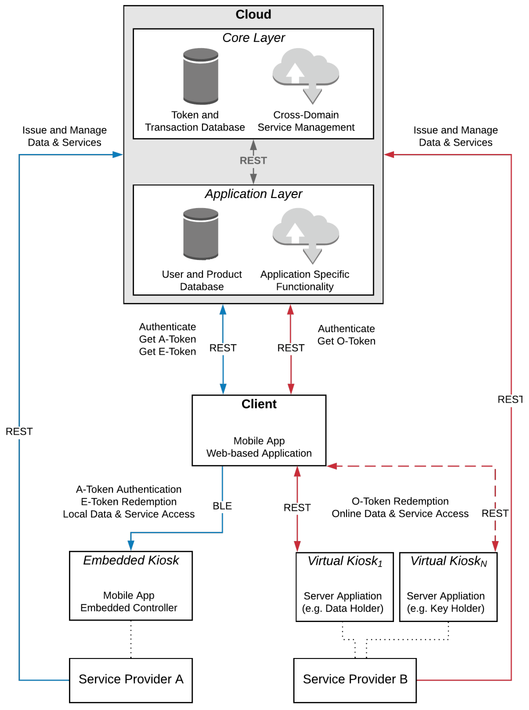

Cloud
Core Layer
Token and Transaction Database
Cross-Domain Service Management
REST
Application Layer
User and Product Database
Application Specific Functionality
Issue and Manage Data & Services
Issue and Manage Data & Services
Authenticate Get A-Token Get E-Token
REST
REST
Authenticate Get O-Token
REST
REST
Client
Mobile App Web-based Application
A-Token Authentication E-Token Redemption Local Data & Service Access
BLE
REST
O-Token Redemption Online Data & Service Access
REST
Embedded Kiosk
Mobile App Embedded Controller
Virtual Kiosk$_1$
Server Appliation (e.g. Data Holder)
Virtual Kiosk$_N$
Server Appliation (e.g. Key Holder)
Service Provider A
Service Provider B

Fig. 1. Overview of the distributed platform subdivided into a cloud part as well as client and kiosk devices interacting with each other via BLE and a RESTful interface.

- **Client.** The client represents a user application (e.g. web-application, mobile smartphone app, etc.) that obtains digital tokens which can be redeemed in different ways to get access to various services. It is possible to redeem them offline by communicating to the local environment (e.g. get access to a parking lot). Alternatively, services can also be redeemed online in order to become entitled to receive and visualize specific data. Depending on the underlying application this can be a bus ticket or the entitlement to receive sensory data stored on different servers, for example.
- **Kiosk.** A kiosk entity acts as a validation authority and service redemption unit. It receives and verifies digital tokens for authenticating and authorizing users to access different services. The services that are redeemable by kiosk entities are managed by service providers. We distinguish between two different kiosk types:
  - *Embedded Kiosk.* An embedded kiosk interacts with a client device for authenticating users, verifying their entitlements, and authorizing them to access local services via cryptographically secured tokens transmitted over Bluetooth Low Energy. The embedded kiosk is an application running on a mobile device (e.g. supervised parking use case; attendant checks parking permit) or a stationary device embedded into the infrastructure (e.g. parking gate; embedded device checks parking permit).
  - *Virtual Kiosk.* The virtual kiosk is a server-side application that holds data (e.g. sensory or mobility data) of one specific service provider. It also acts as a redemption place where the data in question can be obtained by clients by redeeming digital tokens over a RESTbased interface. The authorization procedure of a virtual kiosk is based on the OAuth 2.0 protocol [13]. An access token is sent to the client after authentication. It can be forwarded to the virtual kiosk for validation and redemption. Subsequently, the requested data is provided by the virtual kiosk. The data that is managed by a virtual kiosk is defined by the corresponding service provider via a REST interface to the cloud. In case of encrypted data where the sensitive information and the actual key is stored on different systems our concept supports the usage of multiple access tokens and virtual kiosks.
- **Cloud.** The cloud environment described in this paper is subdivided into an application layer and a core layer. The application layer is a server that remains in constant interaction with client and kiosk devices. It is adapted according to the underlying system of the corresponding service provider. In contrast, the core layer acts as trusted layer responsible for providing the necessary means – including token creation and management – for local and online authentication and authorization to different services. Which entitlements can be obtained by clients and redeemed at kiosks can be managed by the corresponding service providers through a REST-based interaction between the application layer and the core layer. These layers are described in the following passage:
  - *Application Layer.* Clients and kiosks interact with the application layer. It keeps them synchronized and is in charge of application specific functionality and data management (user, products, services). A RESTful interface enables the creation and the management of different service provider accounts and their services. Furthermore, it is the connecting element to the core layer.
  - *Core Layer.* The core layer acts as certification authority and is in charge for issuing and signing different digital tokens that can be used for authenticating users and authorizing them to access different services available locally or online. It also acts as token validator, which is needed to assure that an unauthorized user cannot access any service or data. This layer implements the OAuth 2.0 authorization protocol for entitling clients to access online resources located on virtual kiosks. This is done by utilizing the OAuth scopes definitions that describe what data clients may access. The application layer as well as devices such as clients and kiosks communicate over a REST interface with the core layer.

B. Wireless communication interfaces

The framework discussed in this paper enables and manages access to different services. Depending on the use case and the application, local wireless communication over BLE or online communication via REST can be established. The paper [14] forms the basis of these communication interfaces.

- **Bluetooth Low Energy (BLE).** BLE is a short-range wireless technology. It benefits IoT applications due to its power saving design, the coexistence of connectionless (broadcaster and observer roles) and connection-based (peripheral and central roles) data transfer procedures, its robustness against obstacles, and compatibility with smartphones [15]. It is used for the transaction and redemption handling between a client device (BLE central role) and an embedded kiosk device (BLE peripheral role) without the need of an active internet connection.
- **Representational State Transfer (REST).** REST is a software architectural style for implementing web services that rely on the HTTP protocol. Clients as well as virtual kiosk devices communicate to the cloud via a REST-based interface.

C. Online service redemption

Our approach provides a secure online redemption mechanism to access data stored on different servers. Entitlement management to access data and services is part of the core layer, while the virtual kiosks are responsible for the redemption mechanism and for providing the data in question (e.g. mobility data).

Our cloud environment includes an OAuth 2.0 part for managing the redemption procedure of online services. In this context, the cloud creates a valid **OAuth-access-token (O-token)** which purpose is to entitle a specific application to access certain datasets of another entity. The request for receiving the O-token specifies the grant type that determines which of the OAuth 2.0 authentication flows is used. In our case the authentication relies on the client credentials flow. Therefore, an additional client-ID as well as a client-secret have to be submitted. Last but not least, also an accessTokenUri (specifies where the O-token is being requested from) and the scope (defines which resources are being requested) are included in the request. If the authentication mechanism was successful, the O-token is forwarded from the core layer to the client. The token contains a validity period and the scope of the data that the client is allowed to access. The token also consists of a value-field, which functions as a token-identifier. In contrast, the token type specifies how the token will be used to access the resource. In our implementation it is a bearer type token, meaning that access to the resources is given to the bearer of the token. Fig. 2 summarizes the data fields of the O-token. Finally, the token can be redeemed by the client at a corresponding virtual kiosk in order to gain access to the data the O-token specifies. No additional authentication on client side is required. Through interaction with the core layer, the virtual kiosk is able to validate the O-token, which guarantees that the client is entitled to access the resources.

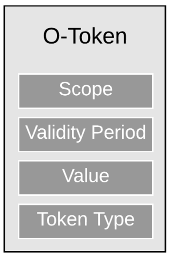


Fig. 2. Overview of an OAuth-access-token responsible for an online authorization procedure

The following example summarizes the protocol flow from above, which is also depicted in Fig. 3. The client wants to access a set of sensory data about the distribution of vehicles used in a specific city. This data is held by a service provider running a virtual kiosk. He only shares the data with whomever has the authority to do so. This authority is given by the core layer in form of an O-Token. The client authenticates himself on the core layer and receives the required authorization token (O-token). The client now sends the token to the virtual kiosk to get access to the sensory data the kiosk holds. Latter doesn't trust the client, but the core layer. Therefore the token is forwarded to the core layer in order to verify its authenticity. If the token is indeed valid, the virtual kiosk grants the client access to the resource by sending him the data requested.

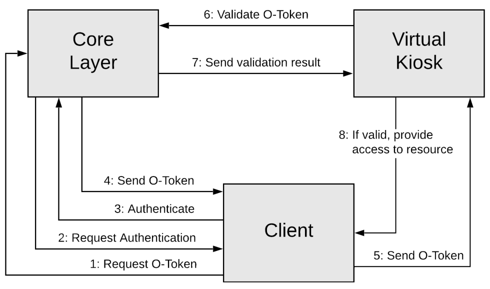


Fig. 3. Online Redemption Procedure between a client and an embedded kiosk involving the core layer

If the data that is located on the virtual kiosk (VK1) is confidential, the possibility of making use of multiple virtual kiosks (VK2) exists. This offers the possibility of encrypting the data on VK1 and storing the key for decryption on VK2. Before the client may access and use the data on VK1, he first has to get the entitlements O-token$_{VK1}$ and O-token$_{VK2}$. After authenticating himself on the core layer, O-token$_{VK2}$ can be redeemed against the decryption key on VK2 and O-token$_{VK2}$ may be utilized to receive the encrypted dataset. Finally, the data can be decryption locally with the received cryptographic key. This way, the described concept also supports end-to-end encryption methodologies.

D. Offline service redemption

Our concept does not only provide a service online-redemption approach as described in the previous subsection III-C, but foresees also the usage of secure offline authentication and authorization based on the concept of [14]. This enables the redemption of different services between mobile clients and embedded kiosks in areas without or limited network access (e.g. underground garage). Data transfer between devices is done via the local wireless standard BLE. We distinguish between two token types that are initially distributed to client devices by the core layer through interaction with the application server. An overview of the data model of the tokens, which will be discussed in the following paragraph, is given in Fig. 4.

An **entitlement-token (E-token)** represents a service and is issued and signed by the core server. It consists of a service id and an application id to uniquely identify the corresponding service and is linked to an authentication-token.

An **authentication-token (A-token)** is bound to a particular user account and device. It includes the public key of a client device, a validity period, and the core server's signature. A client or embedded kiosk device's A-token is issued in the course of a distributed registration mechanism, first involving an application-specific user-login on the application layer. Subsequently, a one-time-ticket is fetched by the application layer and passed to the client or the embedded kiosk. It can be redeemed with the device's public key at the core-layer. Finally, the device receives an A-token as well as an API key for accessing the core layer's REST-based interface; a trusted link between the local device and the cloud layers has been established.

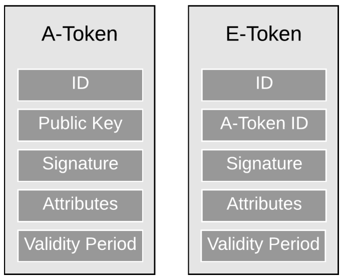


Fig. 4. Overview of an authentication-token and an entitlement token for local authentication and authorization.

The redemption of an E-token involves the interaction of an embedded kiosk. During this event the client's A-token and the E-tokens that should be redeemed are transferred to the embedded kiosk. For checking the authenticity of the two communication participants a challenge-response protocol is applied between them [14]. First both A-tokens (in this example A-token$_{\mathrm{C}}$ and A-token$_{\mathrm{EK}}$) are exchanged, checked for their validity, and verified with the server's public key. Next, the client generates a random number (challenge C_{C}) and challenges the embedded kiosk to sign it with his private key before it is sent back to the client. Furthermore, the received signature is verified against the random number C_{C}. This is done by using the kiosk's public key that is embedded into the previously exchanged A-Token$_{\mathrm{EK}}$. If this verification was successful, the ownership of A-Token$_{\mathrm{EK}}$ was proven. Before exchanging data, the same method is also applied for A-token$_{\mathrm{C}}$. Fig. 5 shows details about the mutual challenge-response procedure. If the mutual challenge response protocol is passed, E-tokens are sent from the client to the embedded kiosk. The kiosk checks their signature and validity period. Last but not least, the redemption of the E-tokens is triggered by the embedded kiosk by involving the core server via a REST interface.

IV. Implementation

Regarding our proof of concept the client app and the embedded kiosk were implemented as Android applications, while the virtual kiosk as well as the cloud layers were implemented as Java applications. Clients, kiosks and application servers access the core layers functionality through a RESTful interface, protected by HTTPS. During the distributed registration procedure an A-token and an API-key are issued. Consequently, the core layer requests are authenticated via basic HTTP authentication, utilizing the previously received API key that is passed as a username in the basic HTTP authorization header. The overall architecture relies on PKI combined with

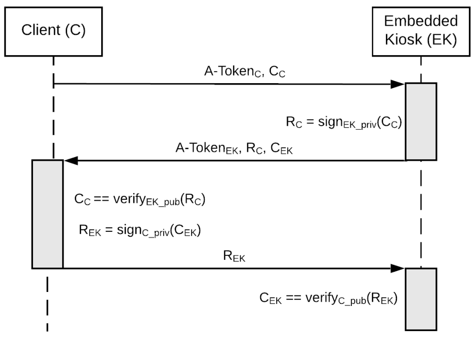

Client (C)
Embedded Kiosk (EK)
A-Token$_C$, C_C
R_C = sign$_{EK\_priv}$(C_C)
A-Token$_{EK}$, R_C, C_{EK}
C_C == verify$_{EK\_pub}$(R_C)
R_{EK} = sign$_{C\_priv}$(C_{EK})
R_{EK}
C_{EK} == verify$_{C\_pub}$(R_{EK})

Fig. 5. Mutual challenge response protocol between a client and an embedded kiosk [14].

the Elliptic Curve Digital Signature algorithm (ECDSA) and SHA256 for hashing data. Regarding ECDSA, the elliptic curve secp256r1 (prime256v1, NIST P-256) has been used. A web-based interface is used for service providers to create and manage the services that should be available for clients and redeemable at kiosk entities. It was programmed with Vaadin, a Java UI framework and library for web application developments. Regarding the usage of the OAuth 2.0 protocol, the Java library "spring-security-oauth2" was imported into the project, that provides authentication and access-control modules.

V. Performance evaluation

The following tables provide information about the timing behavior of our implementation. Each action was performed 100 times. The indicated values are the average of the measurements of each action. The performance evaluation of the offline service redemption was carried out on two Android smartphones: Xiaomi Mi Mix 3 with Android 8.0.0 (client device) and Sony Xperia Z5 with Android 7.1.1 (embedded kiosk device). Both the application layer as well as the core layer were set up as separate AWS server instances. This is also where the virtual kiosk part was integrated.

Table I shows our timing values regarding the token handling for local service redemption between a client and an embedded kiosk. First, the A-Token is created during the distributed registration procedure on the core layer. The issuing procedure of an A-token includes the ticket creation triggered by the application layer. It is redeemed by the client at the core layer, where in further consequence user and device entities are created. As long as an A-token does not exceed its validity period, this procedure does not have to be repeated. Next, there is the E-token creation. Each time a service on application layer is obtained an E-Token is issued. The client communicates to the application layer which contacts the core layer for creating the E-token. It is linked to the right user and his A-token and pushed to the corresponding client device. If the A-Token and the E-Token are on the device they can be redeemed at embedded kiosks. The mutual authentication

TABLE I
MEASURED TIME OF LOCAL SERVICE REDEMPTION BETWEEN CLIENT, CLOUD, AND EMBEDDED KIOSK

Action	Time [ms]
Creation & Obtainment A-Token	1097
Creation & Obtainment E-Token	1233
Local Authentication	1267
Redemption E-Token	244
Total	**3841**

procedure described in III-D and the redemption of the E-token on the core server take around 1.5s. Summarized, our approach – excluding the one-time operation of receiving an A-token, but including the creation of an E-token, local authentication, and the redemption procedure – takes less than three seconds.

In contrast, the timing values in Table II include the issuing process of an O-Token and the online redemption. During the online authentication the user is authenticated on application layer via a dedicated access token. Subsequently, if the verification is successful, the OAuth client credentials flow is triggered which verifies a client-ID and client-secret on core layer side. This takes approximately 0.5s. Next, the O-Token is created on the core cloud and forwarded to the client device. The redemption of the O-Token includes the sending of the O-Token from the client to the virtual kiosk, and its validation. Once the O-Token has been validated successfully it is redeemed by the core layer. Subsequently, the virtual kiosk sends the resource to the client, which in our example was a data package of 1MB size. The total time required for retrieving and redeeming the authorization token is approximately 1.5s.

TABLE II
MEASURED TIME OF ONLINE SERVICE REDEMPTION BETWEEN CLIENT, CLOUD AND VIRTUAL KIOSK

Action	Time [ms]
Online Authentication	445
Creation & Obtainment O-Token	542
Redemption O-Token	171
Obtainment resources (1MB data package)	368
Total	**1526**

Summarized, the measured execution times are in range of seconds due to a special focus on tamper-resistant data exchange. However, this is still suitable from a usability perspective.

VI. CONCLUSION AND FUTURE WORK

As the subject of providing secure access to heterogeneous services plays a major role in the further development of smart cities but no standard method can be applied because of different application requirements and environmental restrictions regarding connectivity, we introduce an approach to seamlessly access local and online services and resources (car access, smart parking, access to mobility data, etc.). Authentication and authorization management is provided by a multi-layered cloud environment that issues digital tokens for client devices (e.g. smartphones) which can be exchanged for accessing certain services or data packages at dedicated redemption units. Different token types are utilized to handle online and offline redemption approaches. The *offline redemption* process relies on a challenge response mechanism to authenticate client devices to locally available redemption units (embedded kiosks), which in turn entitle them to consume the actual service. By contrast, the *online redemption* procedure integrates the OAuth 2.0 protocol for securely authorizing clients to access data or services that are located on remote servers (virtual kiosks). While the communication between local devices is handled via Bluetooth Low Energy, the online communication is established through a RESTful interface. The performance evaluation of our framework's service access procedures fulfills the demand on low execution times regarding usability. Future work will investigate secure data transmission strategies that incorporate the embedded sector, mobile devices, and different cloud instances, while focusing on end-to-end security.

ACKNOWLEDGMENT

This work is supported by CISC Semiconductor GmbH and Graz University of Technology (Institute of Technical Informatics). Furthermore, it has received funding from the European Unions Horizon 2020 research and innovation programme SCOTT under grant agreement No. 737422.

REFERENCES

[1] Y. U. Devi and M. S. Rukmini, "IoT in connected vehicles: Challenges and issues - A review," *International Conference on Signal Processing, Communication, Power and Embedded System, SCOPES 2016 - Proceedings*, pp. 1864–1867, 2017.

[2] A. Alkhamisi, M. S. H. Nazmudeen, and S. M. Buhari, "A cross-layer framework for sensor data aggregation for IoT applications in smart cities," *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*, pp. 1–6, 2016.

[3] P. Mrazovic, I. De La Rubia, J. Urmeneta, C. Balufo, R. Tapias, M. Matskin, and J. L. Larriba-Pey, "CIGO! Mobility management platform for growing efficient and balanced smart city ecosystem," *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*, pp. 1–4, 2016.

[4] K. Chaturvedi and T. H. Kolbe, "InterSensor Service: Establishing Interoperability over Heterogeneous Sensor Observations and Platforms for Smart Cities," *2018 IEEE International Smart Cities Conference, ISC2 2018*, pp. 1–8, 2019.

[5] I. Ali and M. Asif, "Applying security patterns for authorization of users in IoT based applications," *2018 International Conference on Engineering and Emerging Technologies, ICEET 2018*, vol. 2018-Janua, pp. 1–5, 2018.

[6] D. Bruneo, S. Distefano, F. Longo, G. Merlino, and A. Puliafito, "IoT-cloud authorization and delegation mechanisms for ubiquitous sensing and actuation," *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pp. 222–227, 2017.

[7] D. E. Saputra and S. H. Supangkat, "Single access key for activities in smart city," *Proceedings - 2014 International Conference on ICT for Smart Society: "Smart System Platform Development for City and Society, GoeSmart 2014", ICISS 2014*, pp. 165–168, 2014.

[8] V. Sucasas, G. Mantas, A. Radwan, and J. Rodriguez, "An OAuth2-based protocol with strong user privacy preservation for smart city mobile e-Health apps," *2016 IEEE International Conference on Communications, ICC 2016*, no. 333020, pp. 1–6, 2016.

[9] K. Liu and K. Xu, "OAuth based authentication and authorization in open telco API," *Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012*, vol. 1, pp. 176–179, 2012.

[10] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi, "OAuth-IoT: An access control framework for the Internet of Things based on open standards," *Proceedings - IEEE Symposium on Computers and Communications*, pp. 676–681, 2017.

[11] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An oauth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1224–1234, 2015.

[12] A. Rech, M. Pistauer, and C. Steger, "Increasing Interoperability Between Heterogeneous Smart City Applications," in *Lecture Notes in Computer Science*. Tokyo: Springer International Publishing, 2018, pp. 64–74.

[13] IETF OAuth WG, "OAuth 2.0." [Online]. Available: https://oauth.net/2

[14] A. Rech, M. Pistauer, and C. Steger, "A Novel Embedded Platform for Secure and Privacy-Concerned Cross-Domain Service Access," *IEEE 30th Intelligent Vehicles Symposium, IV19 - Proceedings*, pp. 1–7, Manuscript submitted for publication.

[15] R. Davidson, T. Kevin, W. Chris, and C. Cufí, *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*, 1st ed., B. Sawyer and M. Loukides, Eds. O'Reilly Media, 2014.

Multi-Layered IoT System Design Towards End-to-End Secure Communication

Alexander Rech
CISC Semiconductor GmbH
Graz, Austria
a.rech@cisc.at

Lukas Gressl
Graz University of Technology
Institute of Technical Informatics
Graz, Austria
gressl@tugraz.at

Fikret Bašić
CISC Semiconductor GmbH
Graz, Austria
f.basic@cisc.at

Christian Seifert
Graz University of Technology
Institute of Technical Informatics
Graz, Austria
christian.seifert@tugraz.at

Christian Steger
Graz University of Technology
Institute of Technical Informatics
Graz, Austria
steger@tugraz.at

Andreas Sinnhofer
NXP Semiconductors Austria GmbH
Gratkorn, Austria
andreas.daniel.sinnhofer@nxp.com

***Abstract*—An increasing amount of sensory data, often of confidential nature, is exchanged day by day: from the sensor and actuator layers over smart gateways to the business logic and analytics level. Robust yet efficient security measures play an essential role in this interaction. However, the complexity of securely connecting different building blocks of a distributed, multi-layered systems is considerable. Security methodologies are often applied at a late stage of system development, posing problems such as inappropriate security levels, performance issues, and longer time-to-market cycles. Addressing possible security properties already in the design phase of a security-critical system helps to mitigate these problems. In this paper, we discuss a distributed, multi-layered IoT data collection system that enables data aggregation and exchange from the embedded level up to different cloud instances while supporting end-to-end secured communication. The system was designed in the course of a case study where we used a design-space-exploration tool for identifying secure processes in regard to key management and distribution. Based on our analysis results, a distributed proof of concept was developed. Subsequently, the most critical processes of the individual layers were evaluated regarding security and execution speed.**

***Index Terms*—Cyber Security; Embedded System Design; IoT Systems;**

I. Introduction

The Internet of Things (IoT) revolution is ongoing and has gained significant attention over the last few years. This revolution is based on the availability of vast amounts of information coming from the data sensed and transmitted by intelligent connected objects. While the transition to a more connected world gives rise to new possibilities, new problems such as security issues crop up. To ensure secure generation, storage, and usage of confidential data during the pre-personalization process of a connected device and during the runtime of its application, dedicated protection schemes on hardware and software levels have to be leveraged. This can be especially challenging in the case of multi-layered distributed systems due to an increasing level of complexity, the more communication participants and diverse systems are involved. However, several security problems and risks can be mitigated or even avoided by laying a particular focus on security already in the design phase of a system. This paper describes how to design and implement a multi-layered IoT data collection system for end-to-end secured communication. A case study has been conducted based on a design-space-exploration (DSE) tool that takes our system definition as input and identifies a set of potential hardware and software building blocks for secure system design. Consequently, secure communication methods are selected and implemented for establishing a trusted relationship between all devices of the distributed IoT system. Summarized, the main topics discussed in this paper are:

System definition. We define the key components of a typical IoT data collection system, including devices from a lower to the top level of an IoT system. While edge nodes act as data collectors, a dedicated gateway device enables communication to the cloud. The overall system includes the usage of local and online wireless data transmission techniques.

Design-space-exploration. Based on our initial, high-level system setup, different system design proposals with distinct security and performance properties for key management and distribution are derived by using a DSE tool. The functionality of the system is illustrated as a task graph based on the requirements of the actual application, while the architectural design is described by hardware components connected with communication buses. Consequently, possible attack scenarios are modeled to facilitate the system design decision.

Distributed proof of concept. A distributed proof of concept that fulfills the demand for efficient, yet secure data transfer is implemented according to the received DSE analysis results. Last but not least, the overall system is evaluated, taking into account execution times and the achieved security level.

II. Related Work

The ubiquitous usage of smart services can be seen through a positive trend describing an exponential increase in the number of IoT devices that are present today. In 2017, an increase of 31% of IoT systems was reported compared to the previous year, with the numbers expected to rise in the upcoming years [1]. To assess and increase the quality of such IoT-systems, design-space-exploration (DSE) tools can be applied. They are used to aid designers in considering system components related to their performance, power consumption, and security features at early development phases [2]. These tools can generally be classified through their applicability in either individual component integration [2], [3], or the overall system design [4], [5]. Novel approaches are applied during the security evaluation for the work given in this paper. These include an automated DSE with specified security constraints and formulated attack scenarios presented as Bayesian attack graphs (BAGs) [6]. Ultimately, we use these tools to present an original use case for an IoT system design that already takes into account different IoT layers and elements, individual element security, and end-to-end communication security.

IoT layers, which consider different levels of data processing, are being researched through: data collection [7] by sensor nodes, data forwarding via gateways [8], and data storage and analysis with cloud services [9]. The work of interest in this field is focused on the node network architecture, communication, and security, among other characteristics. The system that we present in this paper handles the mentioned characteristics as a one-entity platform. This platform is described through a use case which enables secure interfaces implicated on all levels.

Today, there exist several limiting factors for a successful IoT solution, such as inadequate security protection. As stated in [10], security in IoT, even to this day, presents one of the main challenges that the designers have to face. This comes from two design issues: implementing secure solutions in resource-constrained IoT devices, and the complexity increasing proportionally with the size of the network. Additionally, a well-designed system should not only be able to handle information security, but also ethical and privacy concepts [11]. A root-of-trust is required to ensure reliable communication and operation of a secured IoT product [12].

In the work illustrated by the papers [13] and [14], the focus is given in analyzing only specific aspects of security, e.g., hardware devices and security threads, respectively. Similarly, particular research interest is also given on the IoT middleware [15]. A middleware is explained as a software system that resides between an IoT device and its application. The paper goes into a survey analysis on the state-of-the-art middleware, which includes security, privacy, and trust concepts. While the work itself predicts the future development of the security analysis, it only gives insight into one characteristic (middleware) of the whole platform.

In addition to the security analyses on different layers of IoT, efforts have been made in studying the end-to-end data security-flow of such systems [16]. Each design platform can be divided into different functional units of interest, which are then evaluated on pre-defined security cases [17]. The analysis is done on real-world security attacks and applications. However, some works present a more pessimistic view of the end-to-end focused security, proposing instead an evaluation extension on the communication endpoints [18].

Further research efforts have shown that authentication and authorization procedures play a crucial role in guaranteeing trusted data processing in IoT devices [19]. As a result of many vulnerabilities usually found in IoT city deployment, a broad interest is given to secure authentication, key exchange, and their handling in smart city environments. The paper [20] deals with how the access key is stored and handled with the communication devices. Additionally, a problem in the integration of IoT products and services in smart cities is often seen through the environmental restrictions regarding connectivity. Various services might require separate secure access methodologies for accessing both local and online resources. Paper [21] provides an approach where the authentication and authorization management administers digital tokens. These tokens are used for accessing locally available resources via Bluetooth Low Energy and online resources via a dedicated RESTful HTTP interface.

While the paper presented in this section are significant in addressing different aspects of IoT security, they do not present a concluded solution for the overall end-to-end IoT security. In the further context of this paper, an end-to-end secured system is presented. The implementation and evaluation are based on the results of a DSE approach.

III. System Definition

In the following paragraphs, we define the key components of our IoT data collection system. While subsections III-A and III-B describe the layers and devices of our system, subsection III-C is about wireless data transfer methods. The described system forms the starting point for our DSE-based analysis in chapter IV.

A. Embedded Layer

The embedded layer is composed of two entities, edge nodes, and gateways (see Fig. 1).

- **Edge Node.** An edge node shall work as a secure sensor device. It collects sensory data from the local environment at constant intervals. It is one anchor of trust in our end-to-end communication scenario. The sensor node provides encryption and data authentication. Eventually, the data should be forwarded to the cloud. However, since edge node devices in IoT data collection systems are often constrained in terms of computing power and energy, more complex tasks such as big data processing or communication over the internet itself are outsourced to other devices. Instead, a power-saving local wireless protocol should be used to forward secured data packages to a nearby, less resource-constrained gateway device.

- **Gateway.** The gateway's primary purpose is to act as a connector between edge devices and the cloud layer. It should communicate with local edge nodes, but also the cloud layer over the internet. The gateway is a trusted communication participant but is not supposed to read the received encrypted data.

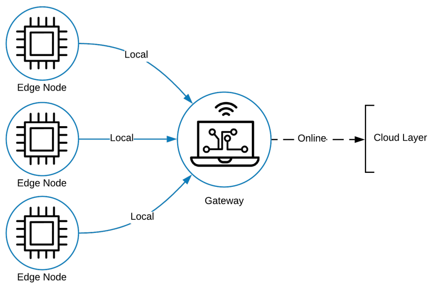


Fig. 1. Embedded layer overview: multiple edge nodes mark the start point for the end-to-end data transmission. They forward the collected data over a local wireless channel securely to a nearby gateway.

B. Cloud Layer

The cloud layer consists of two types of server instances (see Fig. 2)

- **Business Server.** This server is the direct point of contact for the gateway device and receives the secure data package. It is the second anchor of trust in our end-to-end communication, meaning that it can decrypt the received data. Furthermore, it may provide an interface for other server applications (third party cloud) to retrieve the collected data.
- **Third Party Cloud.** This cloud instance is operated by an external service provider who wants to access a specific data stream and execute some operations on it. The received raw data can be utilized for new tasks or to enhance existing ones (e.g., creating higher levels of automation) with dedicated analytic and business intelligence. Examples of different techniques include statistical modeling, data mining, exploratory analysis, predictive analytics, or diagnostic methods in general.

C. Wireless communication interfaces

Wireless connectivity is often considered as the key functionality in IoT devices, but at the same time, it is a major source of power consumption. Moreover, the plurality of wireless standards generally poses severe troubles concerning the compatibility between collocated networks. Therefore, energy efficiency (edge nodes are generally resource-constrained) and interoperability (the gateway has to support communication between edge nodes and the business server) are the major challenges in connectivity for IoT. The following communication schemes are proposed to support efficient communication between constrained embedded nodes with the cloud:

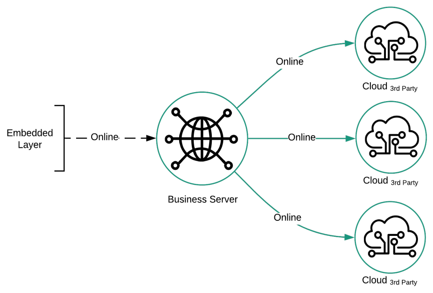


Fig. 2. Cloud layer overview: the business server receives the secured data from the embedded layer and acts as second anchor of trust in our end-to-end communication. Multiple third party cloud units may access the datasets for further analyses and evaluations.

- **Local Data Transmission.** Bluetooth Low Energy (BLE) is a short-range wireless technology. Especially due to its power-saving design, its widespread use, and also its robustness against different obstacles, it is used in many IoT applications [22]–[24]. In our scenario, it can be utilized to enable low power communication between multiple edge nodes and the more powerful gateway device.
- **Online Data Transmission.** Representational State Transfer (REST) is a software architectural style for implementing web services relying on HTTP. The gateway device communicates to the cloud layer via a RESTful interface. Furthermore, the different cloud endpoints may also use a dedicated REST API to enforce bidirectional communication.

IV. Security Measures

We used a special security-aware DSE (SaDSE) framework [25] to find the optimal design for the proposed system described in section III. We used the SaDSE framework as it was provided and utilized it to perform a prefiltering of the solution space for system designs that satisfy our requirements. We used the framework's features to model the system's functionality as a task graph, the possible hardware components, the attack scenarios as Bayesian attack graphs (BAGs), and the feasible security mitigation options. In each BAG, the designer must define certain attack goals an attacker aims to reach, adding to each goal a threshold to limit the goal's attack success probability. Each attack aims at a distinct task in the functional view. Each hardware component in the architectural description comes with certain capabilities to withstand incoming attacks. This mitigation is expressed by lowering the attacks' success probabilities for all attacks aiming at the tasks mapped to the said hardware component. This mitigation influences the goals' attack success probabilities. Figure 3 gives an overview of the SaDSE tool, showing its inputs and the generated outputs.

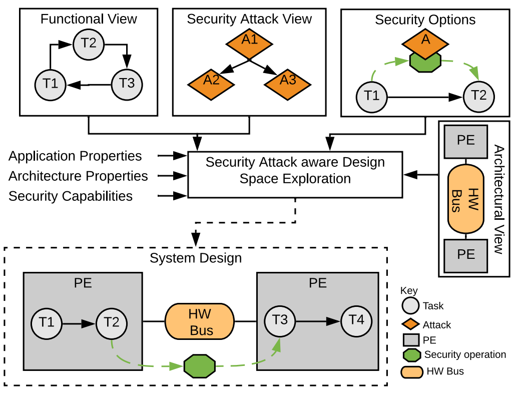


Fig. 3. SaDSE framework overview, taken from [25]. Hardware components are described as processing elements (PE). Tasks (Ts) are described in the task graphs. Attacks (As) describe the attack scenarios.

The SaDSE tool calculates the secure system partitioning and task mapping based on its inputs. It thereby determines the set of security constraints using the BAGs and the given task graph. It only allows solutions that fulfill the security constraints. The security constraints state that only solutions are considered to be secure for which the attack goals' success probabilities do not exceed their individually set thresholds. Depending on its configuration, it provides the most secure solution, or secure solutions with either optimal performance or optimal power consumption, for instance. We designed both the system's embedded and cloud layer using this SaDSE tool. The following paragraphs describe the different system designs modeled with the framework and the solutions it identified.

A. Design-Space-Exploration Phase

In the early system design, we used the SaDSE framework to determine secure solutions based on possible security attacks against the overall system, divided into an embedded and a cloud layer. Furthermore, we assessed how to provision the devices with key material. For each device in our two layers, we provided different hardware and software variants selectable by the framework. For both layers, possible components stem from the considerations described in Section III.

Embedded Layer. The hardware components for the sensor module comprise a micro-controller (MCU), a sensor module, a Bluetooth Low Energy (BLE) radio, and an optional secure element (SE). The gateway consists of an MCU, a BLE radio, a WiFi radio, and an optional SE. The MCU supports a trusted execution environment and comes with software-based cryptography (SWC) or with side-channel proof hardware-based cryptography (HWC). The SE comes with HWC, a hardware-based firewall (HWF) for task separation, and tamper-proof storage.

Cloud Layer. The business server is realized on a corresponding Hardware Security Module (HSM), a secured server platform, or on a basic server platform. The HSM supports HWC, a firewall, and tamper-proof storage. The secured server platform realization comes with an HSM support but performs security non-sensitive tasks on a less secure, but more performant platform. The basic server solution only supports SWC.

In the following, we give an overview of the system's functionality, the security attacks considered during the early system design, and the solutions proposed by the framework.

B. End-to-end Security from Edge Nodes to Cloud Services

The sensor system requires secure data transfer throughout the whole communication path from the edge nodes to the cloud service. Enforcing these requirements, we modeled potential attack scenarios on the distinct layers. Secure solutions found by the tool for the individual layers are explained.

Data Security Mechanisms:
For securing the data packets transferred between the edge nodes, the gateway, and the cloud services, we provided the framework with several security primitives. They comprise symmetric and asymmetric cryptographic algorithms, task encapsulation, and tamper-safe storage. For the cryptographic algorithms, the framework can choose from several different secret keys. These secret keys differ in their lifetime (from several minutes to years) and can derive from one another. Potential key disclosure attacks threaten each key used by the system under design. The lifetime of the secret key determines the motivation of the attacker to read it out. With its disclosure, all cryptographic actions performed with the key are rendered insecure. Hence, the location where the designed system stores the secret keys dramatically influences the overall system's security. For the security primitives used by the system to secure the transferred data, we provided the framework to select from a set of keys: asymmetric and/or symmetric keys, modeled as master or session key, which derive from the master keys.

Communication Security Mechanisms:
Using the SaDSE framework, we designed our system on both the embedded and cloud layer. We focused on the security of the communication between edge nodes and gateway, and between gateway and cloud. Furthermore, we also considered the communication between distinct cloud services. Thereby, we explicitly modeled the attack scenarios on the whole end-to-end communication. The next paragraphs describe these attack scenarios, the transferred information that must be secured, and the solutions proposed by the framework.

Embedded Layer. The edge nodes send their measured sensory data to the gateway using BLE. An adversary must not be capable of extracting the sensed data. Furthermore, each edge node must be capable of authenticating itself before forwarding the sensor information.

- **Attack scenarios.** Potential attack scenarios on the embedded layer comprise the sniffing of the BLE communication, the injection of faked BLE packets, the intrusion into the edge node's and the gateway's software stack, and the tampering with the edge node's hardware (including a potential secret key exposure). The attacker may be capable of brute-forcing cryptographic algorithms that use small encryption keys. The attacker finally aims at reading out the sensitive sensor data, exposing the encryption keys, and faking the sensor data of edge nodes.

- **Proposed solutions.** Considering the communication attacks, the framework automatically discards all plain-text communication. The framework proposes for a security optimal edge node to place all functions (except the BLE packet transmission) on the SE with the highest Common Criteria security certification.[1] Considering the performance optimal, yet secure solution, the framework suggests the implementation of the functionality on an MCU supporting HWC and HWF. It proposes placing a certificate on a SE with the highest-rated security mechanisms and deriving a session key for the symmetric encryption and authentication of the BLE packets. For the gateway, the framework proposes the usage of hardware components providing the same security mechanisms. For the gateway, it also suggests the usage of session keys derived from a certificate placed on the SE.

Cloud Layer. The gateway sends the received sensory data to the business server, which distributes it further via the cloud services using a WiFi connection. The transmitted data packets must be confidential and authentic. The communication between different cloud services must also be secured. Therefore, cloud services must perform an authentication process to establish secure communication among each other. This authentication can be realized by a direct authentication process or a central authentication service.

- **Attack scenarios.** Similar to the attacks in the embedded layer, the cloud layer attacks comprise the sniffing of the WiFi communication, the injection of faked packets, and the logical intrusion into the server software stack. Furthermore, the attacker might be able to break encryption and authentication, which rely on small cryptographic keys. Potentially, the attacker may also be capable of faking a service's identity against the server, and of intruding the central authentication service's software stack. The attack goals on the cloud layer comprise reading out the sensor data, faking sensor data of edge nodes, and hijacking the central authentication service.
- **Proposed solutions.** The framework suggests as the most secure solution for the cloud service, an HSM exclusive implementation. For the performance optimal secure solution, the framework proposes the implementation of the service's functionality on a server platform offering HWC and HWF and the storage of certificates and key material on an HSM extension. The framework suggests securing the communication between gateway and cloud service using symmetric cryptography. For the cloud service authentication process, the framework suggests as the most secure solution, the direct authentication process. The authentication process using the central authentication service comes with the drawback that a successful attack on the service renders all authentications insecure. This bottleneck enforces the use of highly secure components and redundancy mechanisms for the realization of this service.

[1]https://www.commoncriteriaportal.org/

C. Provisioning of embedded devices

The solutions described in the preceding paragraphs rely on various secret keys for performing secure communication and authentication between the different entities of the system. The secure provisioning of secret keys is crucial to guarantee the system's security throughout its whole lifetime. Disclosure of a secret key can render the whole system's security methodologies useless. Hence, we put a particular focus on also securing the provisioning process. We modeled this process using the SaDSE framework, considering three scalable approaches for the key provisioning: (i) key injection at the time of manufacturing; (ii) key fetch by the edge node from the cloud via the gateway; (iii) direct key fetch from the cloud.

- **Attack scenarios:** The potential attack scenarios consider the intrusion into the key provisioning service, the extraction of the key material, the injection of key material during the communication, and the physical compromise of the key provisioning process. The potential attacker aims at comprising and manipulating the injected keys.
- **Proposed solutions:** Based on these attack scenarios, the framework selects approach (i) to be the most secure. Approach (ii) is the least secure one, caused by potential intrusion attacks on the gateway capable of disclosing the key material during provisioning. In contrast, approach (iii) is a good candidate for achieving both security and flexibility under the premise that the edge device can connect to the internet directly.

D. Selection Phase

For selecting the appropriate system design, we focused on the evaluation of the most secure, the least secure, and the fastest secure solution. The most secure solution comes with an average success probability (asp_{avg}) of 0.0074, the least secure with an asp_{avg} of 0.0376. The fastest secure solution comes with an asp_{avg} of 0.0148. The average attack success probability is calculated as follows: $asp_{avg} = \frac{\sum_{i=0}^{N} asp_i}{N}$, where asp_i is the attack success probability of the attack goal i and N is the number of all attack goals. An attack goal is represented as a leaf in a BAG. The asp_i is calculated as the unconditional probability distribution (UPD) of attack goal i, using the Bayesian chain rule. Depending on the hardware component selection, the goal's UPD exceeds or falls below its threshold. We normalized the system performance to the performance of the fastest solution. The colorization of the solutions represents the number of exceeded attack goals. Based on the solutions proposed by the framework, we decided to base our implementation on the fastest secure solution. Hence, we made the following design choices:

- **Embedded Layer.** For the edge node, we decided to use an MCU and a SE for storing the sensitive key material and for providing HWC and task encapsulation. The gateway is also composed of the same elements. The edge nodes communicate with the gateway using BLE. In addition to our application-layer encryption, BLE offers authenticated pairing with data signing on link-layer.
- **Cloud Layer.** For the server realization, we use a server platform supporting HWC and HWF and an HSM extension for the secure storage of the server certificates and the secret keys. The server platform and the cloud services use certificates to authenticate each other. They secure their communication using symmetric cryptography. They exchange the symmetric keys using asymmetric cryptography based on the certificates.
- **Key Provisioning.** For the provisioning of the edge node devices with key / certificate material, we selected the direct injection during device manufacturing. We chose this approach as the SaDSE tool calculated it to be the most secure solution. Especially at this security-critical phase, a highly secure solution is necessary. An injected asymmetric key can be used to wrap and in further consequence, to exchange symmetric session keys for data encryption on the application layer.

V. Implementation

This section describes the most important components on the hardware and software side that have been used to realize the proposed outcome of our DSE phase discussed in chapter IV.

A. Hardware Part

The hardware part of our proof of concept consists of edge devices, a gateway module, a business server, and a third-party cloud.

- **Edge Device.** A Raspberry Pi 3 Model B device was used as an edge device. Its integrated BLE 4.2 chip was utilized for establishing local communication with the gateway. We integrated an HTS221 sensor for retrieving temperature data. Additionally, the edge device was extended with the secure element SE050. It works as a secure access module to increase the security of the host controller. The SE050 connection to the host MCU is established using the chip's I2C interface. The host MCU runs the application logic and controls all cryptographic operations (encrypting, signing, hashing, etc.).
- **Gateway.** The gateway, in our case, was a Raspberry Pi 4 Model B controller. Next to his BLE capabilities, it also has an integrated WiFi module, an Ethernet port, and the possibility to be extended with a GSM module for enabling online access. It belongs to the more powerful embedded devices and can handle multiple connections and operations simultaneously.
- **Business Server and Third Party Cloud.** These server/cloud instances were realized as Amazon EC2 instances. The AWS CloudHSM is a cloud-based hardware security module that can be enabled to create and manage cryptographic keys on the AWS cloud. The application on the server connects to the HSM using mutually authenticated TLS channels established by the HSM client software.

B. Software Part

Complementary to the hardware part, the following software libraries were included into the project:

- **SE050 Support Package.** It is a comprehensive set of resources that offers libraries on how to correctly interact with the SE050 chip for different MCUs.[2]
- **BlueZ.** A C-library that supports the core Bluetooth layers and protocols.[3]
- **Spring Boot.** This is a framework for developing standalone server applications.[4]
- **Fuel.** It is an HTTP networking library for Java/Kotlin, which supports asynchronous and blocking HTTP requests.[5]

Fig. 4 gives an overview of the most important tasks executed between the devices that participate in the end-to-end secured communication. As discussed, the key provisioning on the edge node is conducted at the time of manufacturing in a trusted environment, inducing an RSA-2048 keypair onto the device. From a temporal perspective, this is done before the actual program routine. For data encryption on the application layer, symmetric session keys (AES-256) are utilized. RSA is used for wrapping the symmetric key, before distributing it to the corresponding communication participant.

Regarding the data transfer in general from an edge node to the gateway, the edge node advertises its presence by acting as BLE advertiser. The gateway device (BLE scanner) may initiate a connection attempt. BLE Security Mode 2 / Security Level 2, including BLE-bonding, is used to enforce security by authenticated pairing with data signing. As pairing-method, the LE-Secure-Connection Passkey procedure is applied. As soon as the gateway receives the encrypted data package over BLE, the connection is closed again. Next, the communication between the gateway and the business server platform is established via HTTP and secured using the TLS protocol (incl. mutual authentication). This way, authentication of the devices and encryption of the communication channel can be provided. If the TLS handshake was successful, the data is uploaded to the server and stored securely. The services of the business server are hosted on a server platform with HSM support. The business server forms the end of our end-to-end secured communication. In respect of the sensory data collection process, edge nodes retrieve sensory data once per minute. The collected data is aggregated with an additional device identifier, the type of the collected sensory data (e.g., temperature, humidity, etc.), and a counter for reducing the

[2] https://www.nxp.com/webapp/Download?colCode=SE050-PLUG-TRUST-MW
[3] http://http://www.bluez.org
[4] https://spring.io/projects/spring-boot
[5] https://github.com/kittinunf/Fuel

risk of replay attacks before being encrypted and forwarded. In case a third-party service provider requests data from the business server, it will be decrypted by the business server after applying an application-dependent authentication scheme. For our proof of concept, an API-key based authentication scheme was used. Eventually, the data is sent to the third-party server.

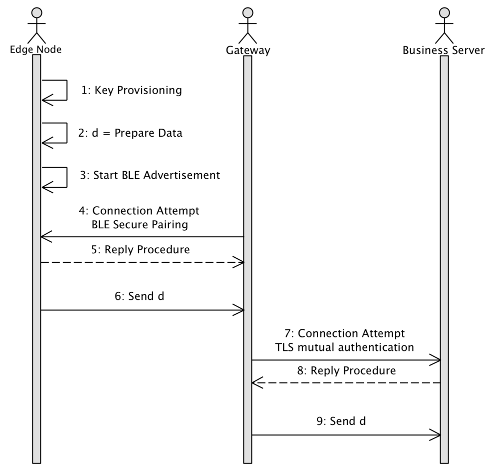


Fig. 4. General overview of the end-to-end secured data flow from edge nodes over the gateway, up to the business server.

VI. Evaluation

In this section, our results of section V are discussed regarding performance and attack success probability. Figure 5 consists of the proposed solutions returned by the DSE framework. Each solution represents the whole system, including the components on both the embedded- and the cloud-layer, and is depicted as a single point on the scatter-plot. The solutions are ordered according to their average attack success probability and system performance normalized to the performance of the fastest solution. The solutions are colored according to the number of attack goals reached by the attackers. We only considered the green solutions found by the DSE framework for the implementation of the secure sensor system. These solutions are considered to be secure. We additionally measured the execution times of two implementations, one secure and one without security means. They are also part of Figure 5. The fastest solution is depicted in the right lower corner area, while the fastest secure solution is located in the left third of the solution space. The measured performance results of our secure solution are depicted in Table I. It illustrates the timing behavior of the most important processes executed across the embedded and the cloud layer. The application logic is mainly based on the program sequence described in Fig. 4. Average timings of 50 measurement cycles are reported. Each measurement cycle involves the interaction of four edge nodes and one gateway device. The sensing-cycle-task of the sensor

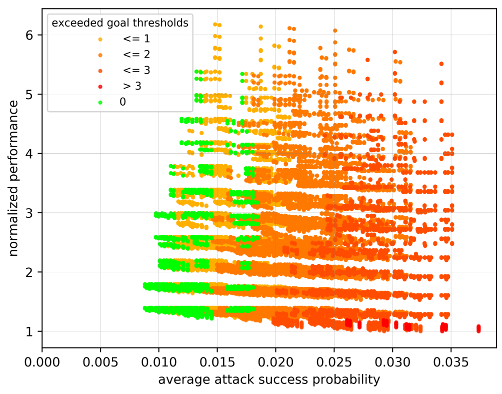


Fig. 5. Solutions found by the DSE framework for the overall system design. Each point represents a solution. The point's color indicates how many attack goal thresholds have been exceeded. The solutions are sorted by the normalized performance and the average attack success probability.

node includes the encryption of the collected data. The BLE-channel opening measurement between the edge node and the gateway describes how much time passes from the moment the sensor advertisement is captured by the gateway, including also the BLE authentication procedure until data packages are ready to be forwarded. Enabling the RESTful channel between the gateway and the business server over HTTP comprises also a TLS mutual authentication phase (total average time of 432 ms). Last but not least, the communication between business and third party servers (authentication via an API key, decryption by the business server, data transfer to the third-party server) takes 84 ms in total. The size of the encrypted payload is 24 bytes.

TABLE I
Measured Time – Secure Approach According to Chapter V

Action	Devices	Time [ms]
Sensing cycle	Edge node	342
Open Channel$_{BLE}$	Edge node $\rightleftarrows$ Gateway	1,431
Datatransfer$_{BLE}$	Edge node $\rightleftarrows$ Gateway	49
Open Channel$_{REST}$	Gateway $\rightleftarrows$ Server$_{Business}$	419
Datatransfer$_{REST}$	Gateway $\rightleftarrows$ Server$_{Business}$	13
Open Channel$_{REST}$	Server$_{Business}$ $\rightleftarrows$ Server$_{3rd\_party}$	72
Datatransfer$_{REST}$	Server$_{Business}$ $\rightleftarrows$ Server$_{3rd\_party}$	12
All actions	**All devices**	**2,338**

TABLE II
Measured Time – No Security

Action	Devices	Time [ms]
Sensing cycle	Edge node	319
Open Channel$_{BLE}$	Edge node $\rightleftarrows$ Gateway	1,173
Datatransfer$_{BLE}$	Edge node $\rightleftarrows$ Gateway	44
Open Channel$_{REST}$	Gateway $\rightleftarrows$ Server$_{Business}$	268
Datatransfer$_{REST}$	Gateway $\rightleftarrows$ Server$_{Business}$	11
Open Channel$_{REST}$	Server$_{Business}$ $\rightleftarrows$ Server$_{3rd\_party}$	27
Datatransfer$_{REST}$	Server$_{Business}$ $\rightleftarrows$ Server$_{3rd\_party}$	10
All actions	**All devices**	**1,852**

In our second evaluation step, all security means were deactivated, and new measurements were taken, e.g., no encryption, no authentication, etc. In keeping with this, Table II gives an overview of all timings without dedicated security means. Again, the average timings of 50 measurement cycles are shown. Due to the absence of additional encryption, the payload size is 15 bytes. In summary, our secure solution achieved short waiting and transmission times. Compared to the timings of the no-security variant, we have a minimum timing overhead for our setup (consisting of a total of 5 devices) despite our security-heavy implementation.

VII. Conclusion And Future Work

With the ongoing development and distribution of the Internet of Things, secure and trusted communication becomes ever more important. In this paper, we designed, implemented, and evaluated an IoT data collection system, with a special focus on secure communication through different IoT layers. While our embedded layer is composed of edge nodes (data collectors) and a gateway device (cloud connector), the cloud layer consists of a central authorization server and third-party cloud units, where data can be analyzed and monitored. Our design phase was amplified by the use of a design-space-exploration tool, identifying different secure processes at an early stage of system development. Based on the results of this exploration phase, a proof of concept was implemented and evaluated in regard to attack success probability and performance. Our results reveal short waiting times for our setup with a minimal time overhead compared to the implemented non-secure variant. Future work will concentrate on an additional scalability evaluation, which will examine the effect of an increased number of communication participants on metrics such as security and execution time.

Acknowledgement

This work was written in tight cooperation with CISC Semiconductor GmbH, Graz University of Technology (Institute of Technical Informatics), and NXP Semiconductor Austria GmbH. Parts of this research have been elaborated within the research program STIP, funded by the Austrian Federal Ministry for Digital and Economic Affairs (BMDW) under the program COIN-Programmlinie ”Netzwerke” (grant number 867914).

References

[1] A. Dean and M. Opoku Agyeman, “A study of the advances in iot security,” 09 2018, pp. 1–5.

[2] E. Kang, “Design space exploration for security,” in *2016 IEEE Cybersecurity Development (SecDev)*, Nov 2016, pp. 30–36.

[3] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, “A design-space exploration for allocating security tasks in multicore real-time systems,” in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 225–230.

[4] I. Stierand, S. Malipatlolla, S. Frschle, A. Sthring, and S. Henkler, “Integrating the security aspect into design space exploration of embedded systems,” in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, Nov 2014, pp. 371–376.

[5] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti, “Cross-layer codesign for secure cyber-physical systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 699–711, May 2016.

[6] N. Poolsappasit, R. Dewri, and I. Ray, “Dynamic security risk management using bayesian attack graphs,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, Jan 2012.

[7] A. Djedouboum, A. ARI, A. Gueroui, A. Mohamadou, and Z. Aliouat, “Big data collection in large-scale wireless sensor networks,” *Sensors*, vol. 18, 12 2018.

[8] M. Surez-Albela, T. Fernndez-Carams, P. Fraga-Lamas, and L. Castedo, “A practical evaluation of a high-security energy-efficient gateway for iot fog computing applications,” *Sensors*, vol. 19(7), 08 2017.

[9] J. Bokefode, A. Bhise, P. Satarkar, and D. Modani, “Developing a secure cloud storage system for storing iot data by applying role based encryption,” *Procedia Computer Science*, vol. 89, pp. 43–50, 12 2016.

[10] S. Dhanda, B. Singh, and P. Jindal, *IoT Security: A Comprehensive View*, 01 2020, pp. 467–494.

[11] H. Atlam and G. Wills, *IoT Security, Privacy, Safety and Ethics*, 03 2019, pp. 1–27.

[12] Ericsson - White Paper, “IoT security - protecting the networked society.” [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/iot-security-protecting-the-networked-society

[13] B. Pearson, L. Luo, Y. Zhang, R. Dey, Z. Ling, M. Bassiouni, and X. Fu, “On misconception of hardware and cost in iot security and privacy,” 05 2019, pp. 1–7.

[14] R. Roman, J. Lopez, and S. Gritzalis, “Evolution and trends in iot security,” *Computer*, vol. 51, pp. 16–25, 07 2018.

[15] A. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Sheng, “Iot middleware: A survey on issues and enabling technologies,” *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 10 2016.

[16] J. Granjal, E. Monteiro, and J. S. Silva, “On the effectiveness of end-to-end security for internet-integrated sensing applications,” in *2012 IEEE International Conference on Green Computing and Communications*, Nov 2012, pp. 87–93.

[17] Z. Ling, K. Liu, Y. Xu, Y. Jin, and X. Fu, “An end-to-end view of iot security and privacy,” *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–7, 2017.

[18] D. Clarke and S. Ali, “End to end security is not enough,” 11 2017, pp. 260–267.

[19] I. Ali and Z. Ullah, “Internet of things security, device authentication and access control: A review,” *International journal of computer science and information security*, 08 2016.

[20] D. Saputra and S. Supangkat, “Single access key for activities in smart city,” pp. 165–168, 01 2015.

[21] A. Rech, M. Kammerer, M. Pistauer, and C. Steger, “A distributed framework towards local and online service access,” in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 1715–1721.

[22] R. Davidson, T. Kevin, W. Chris, and C. Cufí, *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking*, 1st ed., B. Sawyer and M. Loukides, Eds. O’Reilly Media, 2014.

[23] M. Haus, A. Y. Ding, and J. Ott, “Managing IoT at the Edge: The Case for BLE Beacons,” *SMARTOBJECTS 2017 - Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*, pp. 41–46, 2017.

[24] T. Soultanopoulos, S. Sotiriadis, E. Petrakis, and C. Amza, “Internet of Things data management in the cloud for Bluetooth Low Energy (BLE) devices,” *Proceedings - Adaptive Resource Management and Scheduling for Cloud Computing: 3rd Internatioal Workshop, ARMS-CC 2016, PODC 2016*, pp. 35–39, 2016.

[25] L. Gressl, C. Steger, and U. Neffe, “Security driven design space exploration for embedded systems,” in *2019 Forum for Specification and Design Languages (FDL)*, Sep. 2019, pp. 1–8.

Bibliography

[1] Rahul Akolkar; Tom Chefalas; Jim Laredo; Chang Shing Perng; Anca Sailer; Frank Schaffa; Ignacio Silva-Lepe; and Tao Tao. “The Future of Service Marketplaces in the Cloud.” In: *Proceedings - 2012 IEEE 8th World Congress on Services, SERVICES 2012* (2012), pp. 262–269. DOI: 10.1109/SERVICES.2012.59.

[2] Vito Albino; Umberto Erardi; and Rosa Maria Dangelico. “Smart Cities: Definitions, Dimensions, Performance, and Initiatives.” In: *Multimedia Systems and Applications VIII* 22 (2015), pp. 3–21. DOI: 10.1080/10630732.2014.942092.

[3] Inayat Ali and Zahid Ullah. “Internet of Things Security, Device Authentication and Access Control: A Review.” In: *International journal of computer science and information security* 14.8 (2016). ISSN: 1947-5500.

[4] Ishfaq Ali and Muhammad Asif. “Applying Security Patterns for authorization of users in IoT Based Applications.” In: *2018 International Conference on Engineering and Emerging Technologies, ICEET 2018* 2018-Janua (2018), pp. 1–5. DOI: 10.1109/ICEET1.2018.8338648.

[5] Abrar Alkhamisi; Mohamed Saleem Haja Nazmudeen; and Seyed M. Buhari. “A Cross-Layer Framework for Sensor Data Aggregation for IoT Applications in Smart Cities.” In: *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings* (2016), pp. 1–6. DOI: 10.1109/ISC2.2016.7580853.

[6] Giuseppe Anastasi; Michela Antonelli; Alessio Bechini; Simone Brienza; Eleonora D’Andrea; Domenico De Guglielmo; Pietro Ducange; Beatrice Lazzerini; Francesco Marcelloni; and Armando Segatori. “Urban and Social Sensing for Sustainable Mobility in Smart Cities.” In: *2013 Sustainable Internet and ICT for Sustainability, SustainIT 2013* (2013). DOI: 10.1109/SustainIT.2013.6685198.

[7] Federico Andreini; Flavio Crisciani; Claudio Cicconetti; and Raffaella Mambrini. “A scalable architecture for geo-localized service access in Smart Cities.” In: *2011 Future Network and Mobile Summit* (2011), pp. 1–8.

[8] Leonidas Anthopoulos and Panos Fitsilis. “From Digital to Ubiquitous Cities: Defining a Common Architecture for Urban Development.” In: *Proceedings - 2010 6th*

International Conference on Intelligent Environments, IE 2010 (2010), pp. 301–306. DOI: 10.1109/IE.2010.61.

[9] Wolfgang Apolinarski; Umer Iqbal; and Josiane Xavier Parreira. "The GAMBAS Middleware and SDK for Smart City Applications." In: *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PERCOM WORKSHOPS 2014* (2014), pp. 117–122. DOI: 10.1109/PerComW.2014.6815176.

[10] Wolfgang Apolinarski; Marcus Handte; Muhammad Umer Iqbal; and Pedro José Marrón. "PIKE: Enabling Secure Interaction with PIggybacked Key-Exchange." In: *Pervasive and Mobile Computing* 10.PART A (2014), pp. 22–33. ISSN: 15741192. DOI: 10.1016/j.pmcj.2013.10.013.

[11] Hany Atlam and Gary Wills. "IoT Security, Privacy, Safety and Ethics." In: 2019, pp. 1–27. ISBN: 978-3-030-18731-6. DOI: 10.1007/978-3-030-18732-3_8.

[12] Luigi Atzori; Antonio Iera; and Giacomo Morabito. "The Internet of Things: A survey." In: *Computer Networks* 54.15 (2010), pp. 2787–2805. ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010. URL: http://dx.doi.org/10.1016/j.comnet.2010.05.010.

[13] Algirdas Avižienis; Jean Claude Laprie; Brian Randell; and Carl Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing." In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (2004), pp. 11–33. ISSN: 15455971. DOI: 10.1109/TDSC.2004.2.

[14] Asaph Azaria; Ariel Ekblaw; Thiago Vieira; and Andrew Lippman. "MedRec: Using Blockchain for Medical Data Access and Permission Management." In: *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016* (2016), pp. 25–30. ISSN: 9781509040544. DOI: 10.1109/OBD.2016.11.

[15] Tuba Bakici; Esteve Almirall; and Jonathan Wareham. "A Smart City Initiative: The Case of Barcelona." In: *Journal of the Knowledge Economy* 4.2 (2013), pp. 135–148. ISSN: 18687873. DOI: 10.1007/s13132-012-0084-9.

[16] Chitra Balakrishna. "Enabling Technologies for Smart City Services and Applications." In: *Proceedings - 6th International Conference on Next Generation Mobile Applications, Services, and Technologies, NGMAST 2012* (2012), pp. 223–227. DOI: 10.1109/NGMAST.2012.51.

[17] Subhasish Banerjee; Chukhu Chunka; Srijon Sen; and Rajat Subhra Goswami. "An Enhanced and Secure Biometric Based User Authentication Scheme in Wireless Sensor Networks Using Smart Cards." In: *Wireless Personal Communications* 107.1 (2019), pp. 243–270. ISSN: 1572834X. DOI: 10.1007/s11277-019-06252-x. URL: https://doi.org/10.1007/s11277-019-06252-x.

[18] Boualem Benatallah and Motahari Nezhad. *Interoperability in Semantic Web Services.* Ed. by Jorge Cardoso and Amit Sheth. Vol. 3387. Springer Berlin Heidelberg, 2005, pp. 22–25. ISBN: 978-3-540-30581-1. DOI: https://doi.org/10.1007/978-3-540-30581-1_3.

[19] Manuel Benedetti; Luca Ioriatti; Mauro Martinelli; and Federico Viani. "Wireless Sensor Network: A Pervasive Technology for Earth Observation." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 3.4 (2010), pp. 488–496. ISSN: 21511535. DOI: 10.1109/JSTARS.2010.2052917.

[20] Adhitya Bhawiyuga; Mahendra Data; and Andri Warda. "Architectural Design of Token based Authentication of MQTT Protocol in Constrained IoT Device." In: *Proceeding of 2017 11th International Conference on Telecommunication Systems Services and Applications, TSSA 2017* 2018-Janua (2018), pp. 1–4. DOI: 10.1109/TSSA.2017.8272933.

[21] Michael Blackstock; Nima Kaviani; Rodger Lea; and Adrian Friday. "MAGIC Broker 2: An Open and Extensible Platform for the Internet of Things." In: *2010 Internet of Things, IoT 2010* (2010), pp. 1–7. DOI: 10.1109/IOT.2010.5678443.

[22] J. Boardman and B. Sauser. "System of Systems - the meaning of of." In: *2006 IEEE/SMC International Conference on System of Systems Engineering* April (2006), pp. 118–123. ISSN: 00951137. DOI: 10.1109/sysose.2006.1652284.

[23] Jayant Bokefode; Avdhut Bhise; Prajakta Satarkar; and Dattatray Modani. "Developing A Secure Cloud Storage System for Storing IoT Data by Applying Role Based Encryption." In: *Procedia Computer Science* 89 (2016), pp. 43–50. DOI: 10.1016/j.procs.2016.06.007.

[24] Alessio Botta; Walter De Donato; Valerio Persico; and Antonio Pescapé. "Integration of Cloud computing and Internet of Things: A survey." In: *Future Generation Computer Systems* 56 (2016), pp. 684–700. ISSN: 0167739X. DOI: 10.1016/j.future.2015.09.021. URL: http://dx.doi.org/10.1016/j.future.2015.09.021.

[25] Group Broad and Harvard Cambridge. "Blockchain A Beginners Guide." In: *BlockchainHub* 1 (2017), pp. 1–57. URL: https://blockchainhub.net/blockchain-technology/.

[26] Dario Bruneo; Salvatore Distefano; Francesco Longo; Giovanni Merlino; and Antonio Puliafito. "IoT-Cloud Authorization and Delegation Mechanisms for Ubiquitous Sensing and Actuation." In: *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016* (2017), pp. 222–227. DOI: 10.1109/WF-IoT.2016.7845494.

[27] John Buford; Kshiteej Mahajan; and Venkatesh Krishnaswamy. "Federated Enterprise and Cloud-based Collaboration Services." In: *2011 IEEE 5th International Conference on Internet Multimedia Systems Architecture and Application, IMSAA 2011 - Conference Proceedings* (2011). DOI: 10.1109/IMSAA.2011.6156338.

[28] Vitalik Buterin. *On Public and Private Blockchains.* 2015. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/ (visited on 2019-01-10).

[29] Pelin Canbay and Hayri Sever. "The Effect of Clustering on Data Privacy." In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA).* IEEE, 2015, pp. 277–282. ISBN: 9781509002870. DOI: 10.1109/ICMLA.2015.198.

[30] Jorge Cardoso and Amit Sheth. "Introduction to Semantic Web Services and Web Process Composition." In: 2010. DOI: 10.1007/978-3-540-30581-1_1.

[31] Jorge Cardoso; John Miller; Jianwen Su; and Jeff Pollock. "Academic and Industrial Research: Do Their Approaches Differ in Adding Semantics to Web Services?" In: 2010. DOI: 10.1007/978-3-540-30581-1_2.

[32] Kanishk Chaturvedi and Thomas H. Kolbe. "InterSensor Service: Establishing Interoperability over Heterogeneous Sensor Observations and Platforms for Smart Cities." In: *2018 IEEE International Smart Cities Conference, ISC2 2018* (2019), pp. 1–8. DOI: 10.1109/ISC2.2018.8656984.

[33] Bin Cheng; Salvatore Longo; Flavio Cirillo; Martin Bauer; and Ernoe Kovacs. "Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander." In: *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015* (2015), pp. 592–599. DOI: 10.1109/BigDataCongress.2015.91.

[34] Simone Cirani; Marco Picone; Pietro Gonizzi; Luca Veltri; and Gianluigi Ferrari. "IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios." In: *IEEE Sensors Journal* 15.2 (2015), pp. 1224–1234. ISSN: 1530437X. DOI: 10.1109/JSEN.2014.2361406.

[35] CISC Semiconductor GmbH. *Connected Services Case Study.* URL: https://www.cisc.at/references/connected-services/ (visited on 2020-05-30).

[36] Welington M. Da Silva; Gustavo H.R.P. Tomas; Kelvin L. Dias; Alexandre Alvaro; Ricardo A. Afonso; and Vinicius C. Garcia. "Smart Cities Software Architectures: A Survey." In: *Proceedings of the ACM Symposium on Applied Computing* (2013), pp. 1722–1727. DOI: 10.1145/2480362.2480688.

[37] Maissa Dammak; Omar Rafik Merad Boudia; Mohamed Ayoub Messous; Sidi Mohammed Senouci; and Christophe Gransart. "Token-Based Lightweight Authenti-

cation to Secure IoT Networks." In: *2019 16th IEEE Annual Consumer Communications and Networking Conference, CCNC 2019* (2019), pp. 1–4. DOI: 10.1109/CCNC.2019.8651825.

[38] Robert Davidson; Townsend Kevin; Wang Chris; and Carles Cufí. *Getting Started with Bluetooth Low Energy Tools and Techniques for Low-Power Networking.* Ed. by Brian Sawyer and Mike Loukides. 1st ed. O'Reilly Media, 2014, pp. 15–66. ISBN: 9781491949511. DOI: 10.1142/S2010194512008008.

[39] Andrew Dean and Michael Opoku Agyeman. "A Study of the Advances in IoT Security." In: 2018, pp. 1–5. DOI: 10.1145/3284557.3284560.

[40] Sebastian Deterding; Dan Dixon; Rilla Khaled; and Lennart Nacke. "From Game Design Elements to Gamefulness: Defining "Gamification"." In: *Proceedings of the MindTrek Conference 2011* (2011). ISSN: 1450308163. DOI: 10.1145/2181037.2181040. arXiv: 11/09 [ACM 978-1-4503-0816-8].

[41] Sumit Dhanda; Brahmjit Singh; and Poonam Jindal. "IoT Security: A Comprehensive View." In: 2020, pp. 467–494. ISBN: 978-3-030-33595-3. DOI: 10.1007/978-3-030-33596-0_19.

[42] Asside Djedouboum; Ado ARI; Abdelhak Gueroui; Alidou Mohamadou; and Zibouda Aliouat. "Big Data Collection in Large-Scale Wireless Sensor Networks." In: *Sensors* 18 (2018). DOI: 10.3390/s18124474.

[43] Alexandra Dmitrienko and Christian Plappert. "Secure Free-Floating Car Sharing for Offline Cars." In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy - CODASPY '17* (2017), pp. 349–360. DOI: 10.1145/3029806.3029807. URL: http://dl.acm.org/citation.cfm?doid=3029806.3029807.

[44] Cynthia Dwork. *Differential Privacy.* Ed. by Michele Bugliesi; Bart Preneel; Vladimiro Sassone; ; and Ingo Wegener. Vol. 4052. Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 9783642013928.

[45] European Parliament and European Council. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.* 2016.

[46] Linna Fan; Xiaofeng Song; Weiwei Zhao; Haodan Ran; Jingzhi Li; Deyang Shi; Suining Mu; and Tao Qi. "An Anonymous Authentication Mechanism Based on Kerberos and HIBC." In: *ACM International Conference Proceeding Series* (2018), pp. 392–396. DOI: 10.1145/3290511.3290569.

[47] Paul Fremantle; Jacek Kopecký; and Benjamin Aziz. "Web API Management Meets the Internet of Things." In: *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9341 (2015), pp. 367–375. ISSN: 16113349. DOI: 10.1007/978-3-319-25639-9_49.

[48] Sylva Girtelschmid; Matthias Steinbauer; Vikash Kumar; Anna Fensel; and Gabriele Kotsis. "Big Data in Large Scale Intelligent Smart City Installations." In: *ACM International Conference Proceeding Series* (2013), pp. 428–432. DOI: 10.1145/2539150.2539224.

[49] J Granjal; E Monteiro; and J S Silva. "On the Effectiveness of End-to-End Security for Internet-Integrated Sensing Applications." In: *2012 IEEE International Conference on Green Computing and Communications*. 2012, pp. 87–93. DOI: 10.1109/GreenCom.2012.23.

[50] L Gressl; C Steger; and U Neffe. "Security Driven Design Space Exploration for Embedded Systems." In: *2019 Forum for Specification and Design Languages (FDL)*. 2019, pp. 1–8. DOI: 10.1109/FDL.2019.8876944.

[51] Lukas Gressl; Alexander Rech; Christian Steger; Andreas Sinnhofer; and Ralph Weissnegger. "A Design Exploration Framework for Secure IoT-Systems." In: *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment, Cyber SA 2020* (2020). DOI: 10.1109/CyberSA49311.2020.9139631.

[52] Lukas Gressl; Alexander Rech; Christian Steger; Andreas Sinnhofer; and Ralph Weissnegger. "Security Based Design Space Exploration for CPS." In: *Proceedings of the ACM Symposium on Applied Computing* I (2020), pp. 593–595. DOI: 10.1145/3341105.3374058.

[53] Levent Gurgen; Ozan Gunalp; Yazid Benazzouz; and Mathieu Gallissot. "Self-aware cyber-physical systems and applications in smart buildings and cities." In: *Proceedings -Design, Automation and Test in Europe, DATE* (2013), pp. 1149–1154. ISSN: 15301591. DOI: 10.7873/date.2013.240.

[54] M Hasan; S Mohan; R Pellizzoni; and R B Bobba. "A Design-Space Exploration for Allocating Security Tasks in Multicore Real-Time Systems." In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2018, pp. 225–230. DOI: 10.23919/DATE.2018.8342007.

[55] Mohammad Mehedi Hassan; Hanouf Saad Albakr; and Hmood Al-Dossari. "A Cloud-Assisted Internet of Things Framework for Pervasive Healthcare in Smart City Environment." In: *EMASC 2014 - Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities, Workshop of MM 2014* (2014), pp. 9–13. DOI: 10.1145/2661704.2661707.

[56] Kai Huotari and Juho Hamari. "Defining Gamification - A Service Marketing Perspective." In: *In Proceeding of the 16th International Academic MindTrek Con-*

ference (2012). ISSN: 9781450316378. DOI: `10.1145/2393132.2393137`. arXiv: `0312065 [cond-mat]`.

[57] IETF OAuth Working Group. *OAuth 2.0.* URL: `https://oauth.net/2` (visited on 2020-05-25).

[58] Kushal Jaisingh; Khalil El-Khatib; and Rajen Akalu. "Paving the way for Intelligent Transport Systems (ITS)." In: *Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications - DIVANet '16.* ACM, 2016, pp. 25–31. ISBN: 9781450345064. DOI: `10.1145/2989275.2989283`. URL: `http://dl.acm.org/citation.cfm?doid=2989275.2989283`.

[59] E Kang. "Design Space Exploration for Security." In: *2016 IEEE Cybersecurity Development (SecDev).* 2016, pp. 30–36. DOI: `10.1109/SecDev.2016.017`.

[60] Ajay Kumar and Mukul Panwar. "Security for IoT: An effective DTLS with public certificates." In: *2015 International Conference on Advances in Computer Engineering and Applications* (2015), pp. 163–166. DOI: `10.1109/ICACEA.2015.7164688`.

[61] Yong Woo Lee and Seungwoo Rho. "U-City Portal for Smart Ubiquitous Middleware." In: *International Conference on Advanced Communication Technology, ICACT* 1 (2010), pp. 609–613. ISSN: 17389445.

[62] Ninghui Li; Tiancheng Li; and Suresh Venkatasubramanian. "t -Closeness : Privacy Beyond k-Anonymity and l-Diversity." In: *IEEE 23rd International Conference on Data Engineering* (2007), pp. 106–115. DOI: `10.1109/ICDE.2007.367856`.

[63] Xueping Liang; Juan Zhao; Sachin Shetty; and Danyi Li. "Towards Data Assurance and Resilience in IoT Using Blockchain." In: *Proceedings - IEEE Military Communications Conference MILCOM* 2017-Octob (2017), pp. 261–266. ISSN: 2155-7578. DOI: `10.1109/MILCOM.2017.8170858`.

[64] Libelium. *50 Sensor Applications for a Smarter World.* 2020. URL: `https://www.libelium.com/libeliumworld/top-50-iot-sensor-applications-ranking/` (visited on 2020-09-25).

[65] Bingchun Lin and Guohua Liu. "The Classification of K-anonymity Data." In: *Proceedings - 2011 7th International Conference on Computational Intelligence and Security, CIS 2011* (2011), pp. 1374–1378. DOI: `10.1109/CIS.2011.306`.

[66] Zhen Ling; Kaizheng Liu; Yiling Xu; Yier Jin; and Xinwen Fu. "An End-to-End View of IoT Security and Privacy." In: *GLOBECOM 2017 - IEEE Global Communications Conference* (2017), pp. 1–7. DOI: `10.1109/GLOCOM.2017.8254011`.

[67] Ke Liu and Ke Xu. "OAuth Based Authentication and Authorization in Open Telco API." In: *Proceedings - 2012 International Conference on Computer Science*

and Electronics Engineering, ICCSEE 2012 1 (2012), pp. 176–179. DOI: `10.1109/ICCSEE.2012.275`.

[68] Ashwin Machanavajjhala; Daniel Kifer; Johannes Gehrke; and Muthuramakrishnan Venkitasubramaniam. "l-diversity: Privacy beyond k-anonymity." In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (2007). ISSN: 15564681. DOI: `10.1145/1217299.1217302`.

[69] Bao Tong Mi; Xun Liang; and Shu Sen Zhang. "A Survey on Social Internet of Things." In: *Jisuanji Xuebao/Chinese Journal of Computers* 41.7 (2018), pp. 1448–1475. ISSN: 02544164. DOI: `10.11897/SP.J.1016.2018.01448`.

[70] Microsoft Corporation. *Brokered Authentication.* 2005. URL: `http://guides.brucejmack.net/SOA-Patterns/WSSP/9.0BrokAuthKerberosDoc.htm` (visited on 2020-08-20).

[71] Charlie Miller and Chris Valasek. "Adventures in Automotive Networks and Control Units." In: *IOActive Technical White Paper* (2013), pp. 1–99. URL: `https://ioactive.com/pdfs/IOActive_Adventures_in_Automotive_Networks_and_Control_Units.pdf`.

[72] Nader Mohamed; Jameela Al-Jaroodi; Sanja Lazarova-Molnar; Imad Jawhar; and Sara Mahmoud. "A Service-Oriented Middleware for Cloud of Things and Fog Computing Supporting Smart City Applications." In: *2017 IEEE SmartWorld Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017 -* (2018), pp. 1–7. DOI: `10.1109/UIC-ATC.2017.8397564`.

[73] Rebeca Campos Motta; Káthia Marçal De Oliveira; and Guilherme Horta Travassos. "Rethinking Interoperability in Contemporary Software Systems." In: *Proceedings - 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems, JSOS 2017* (2017), pp. 9–15. DOI: `10.1109/JSOS.2017.5`.

[74] Petar Mrazovic; Ivan De La Rubia; Jordi Urmeneta; Carlos Balufo; Ricard Tapias; Mihhail Matskin; and Josep L. Larriba-Pey. "CIGO! Mobility Management Platform for Growing Efficient and Balanced Smart City Ecosystem." In: *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings* (2016), pp. 1–4. DOI: `10.1109/ISC2.2016.07580750`.

[75] Georgios Mylonas; Evangelos Theodoridis; and Luis Munoz. "Integrating Smartphones into the SmartSantander Infrastructure." In: *IEEE Internet Computing* 19.2 (2015), pp. 48–56. ISSN: 10897801. DOI: `10.1109/MIC.2015.25`.

[76] Tarak Nandy; Mohd Yamani Idna Bin Idris; Rafidah Md Noor; Miss Laiha Mat Kiah; Lau Sian Lun; Nor Badrul Annuar Juma'At; Ismail Ahmedy; Norjihan Abdul Ghani; and Sananda Bhattacharyya. "Review on Security of Internet of Things Authentication Mechanism." In: *IEEE Access* 7 (2019), pp. 151054–151089. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2947723.

[77] Ricardo Neisse; Gary Steri; and Igor Nai-Fovino. "A Blockchain-based Approach for Data Accountability and Provenance Tracking." In: *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17* (2017), pp. 1–10. ISSN: 0732-183X. DOI: 10.1145/3098954.3098958. arXiv: 1706.04507. URL: http://arxiv.org/abs/1706.04507.

[78] Mehmet Ercan Nergiz; Maurizio Atzori; and Chris Clifton. "Hiding the Presence of Individuals from Shared Databases." In: *Proceedings of the ACM SIGMOD International Conference on Management of Data* (2007), pp. 665–676. ISSN: 07308078. DOI: 10.1145/1247480.1247554.

[79] Aafaf Ouaddah; Imane Bouij-Pasquier; Anas Abou Elkalam; and Abdellah Ait Ouahman. "Security Analysis and proposal of new Access Control model in the Internet of Thing." In: *Proceedings of 2015 International Conference on Electrical and Information Technologies, ICEIT 2015* (2015), pp. 30–35. DOI: 10.1109/EITech.2015.7162936.

[80] Ahyeon Ju Park; Hye Young Kim; and Jong In Lim. "A framework of device authentication management in IoT environments." In: *2015 5th International Conference on IT Convergence and Security, ICITCS 2015 - Proceedings* (2015), pp. 8–10. DOI: 10.1109/ICITCS.2015.7292918.

[81] Bryan Pearson; Lan Luo; Yue Zhang; Rajib Dey; Zhen Ling; Mostafa Bassiouni; and Xinwen Fu. "On Misconception of Hardware and Cost in IoT Security and Privacy." In: 2019, pp. 1–7. DOI: 10.1109/ICC.2019.8761062.

[82] Pablo Punal Pereira; Jens Eliasson; and Jerker Delsing. "An Authentication and Access Control Framework for CoAP-based Internet of Things." In: *IECON Proceedings (Industrial Electronics Conference)* D (2014), pp. 5293–5299. DOI: 10.1109/IECON.2014.7049308.

[83] Charith Perera; Arkady Zaslavsky; Peter Christen; Michael Compton; and Dimitrios Georgakopoulos. "Context-aware Sensor Search, Selection and Ranking Model for Internet of Things Middleware." In: *Proceedings - IEEE International Conference on Mobile Data Management* 1 (2013), pp. 314–322. ISSN: 15516245. DOI: 10.1109/MDM.2013.46. arXiv: 1303.2447.

[84] Riccardo Petrolo; Valeria Loscrì; and Nathalie Mitton. "Towards a Smart City based on Cloud of Things." In: *WiMobCity 2014 - Proceedings of the 2014 ACM*

International Workshop on Wireless and Mobile Technologies for Smart Cities, co-located with MobiHoc 2014 (2014), pp. 61–65. DOI: 10.1145/2633661.2633667.

[85] G. Piro; I. Cianci; L. A. Grieco; G. Boggia; and P. Camarda. "Information Centric Services in Smart Cities." In: *Journal of Systems and Software* 88.1 (2014), pp. 169–188. ISSN: 01641212. DOI: 10.1016/j.jss.2013.10.029.

[86] Alexander Rech; Markus Pistauer; and Christian Steger. "A Distributed Framework Towards Local and Online Service Access." In: *24th IEEE Conference on Emerging Technologies and Factory Automation - Workshop on Trustable Wirelessly Connected Industrial IoT* (2019), pp. 1715–1721. DOI: 10.1109/ETFA.2019.8869305.

[87] Alexander Rech; Markus Pistauer; and Christian Steger. "A Novel Embedded Platform for Secure and Privacy-Concerned Cross-Domain Service Access." In: *IEEE Intelligent Vehicles Symposium, Proceedings.* Vol. 2019-June. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1961–1967. ISBN: 9781728105604. DOI: 10.1109/IVS.2019.8814123.

[88] Alexander Rech; Markus Pistauer; and Christian Steger. "Increasing Interoperability Between Heterogeneous Smart City Applications." In: *11th International Conference on Internet and Distributed Computing Systems.* Springer International Publishing, 2018, pp. 64–74. ISBN: 9783030027377. DOI: 10.1007/978-3-030-02738-4_6. URL: https://www.springerprofessional.de/en/increasing-interoperability-between-heterogeneous-smart-city-app/16204508.

[89] Alexander Rech; Christian Steger; and Markus Pistauer. "A Decentralized Service-Platform Towards Cross-Domain Entitlement Handling." In: *2nd IEEE Conference on Blockchain.* IEEE, 2019. ISBN: 9781728146935. DOI: 10.1109/Blockchain.2019.00069.

[90] Alexander Rech; Christian Steger; and Markus Pistauer. "Gamifying Connected Services Patterns." In: *ACM International Conference Proceeding Series.* Association for Computing Machinery, 2019. ISBN: 9781450362061. DOI: 10.1145/3361149.3361163.

[91] Alexander Rech; Lukas Gressl; Christian Seifert; Christian Steger; and Andreas Sinnhofer. "Multi-Layered IoT System Design Towards End-to-End Secure Communication." In: *46th Annual Conference of the IEEE Industrial Electronics Society* (2020), pp. 2213–2220. DOI: 10.1109/IECON43393.2020.9254556.

[92] Rodrigo Roman; Javier Lopez; and Stefanos Gritzalis. "Evolution and Trends in IoT Security." In: *Computer* 51 (2018), pp. 16–25. DOI: 10.1109/MC.2018.3011051.

[93] Karen Rose; Scott Eldridge; and Lyman Chapin. *The Internet of Things: An Overview - Understanding the Issues and Challenges of a More Connected World.*

Tech. rep. 2015, pp. 1–80. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv: 1011.1669v3.

[94] Eduardo Felipe Zambom Santana; Ana Paula Chaves; Marco Aurelio Gerosa; Fabio Kon; and Dejan S. Milojicic. "Software Platforms for Smart Cities." In: *ACM Computing Surveys* 50.6 (2018), pp. 1–37. ISSN: 0360-0300. DOI: 10.1145/3124391.

[95] Dany Eka Saputra and Suhono Harso Supangkat. "Single Access Key for Activities in Smart City." In: *Proceedings - 2014 International Conference on ICT for Smart Society: "Smart System Platform Development for City and Society, GoeSmart 2014", ICISS 2014* (2014), pp. 165–168. DOI: 10.1109/ICTSS.2014.7013167.

[96] Savio Sciancalepore; Giuseppe Piro; Daniele Caldarola; Gennaro Boggia; and Giuseppe Bianchi. "OAuth-IoT: An access control framework for the Internet of Things based on open standards." In: *Proceedings - IEEE Symposium on Computers and Communications* (2017), pp. 676–681. ISSN: 15301346. DOI: 10.1109/ISCC.2017.8024606.

[97] Isam Shahrour; Shyar Ali; Zahra Maaziz; and Aziz Soulhi. "Comprehensive Management Platform for Smart Cities." In: *2017 Sensors Networks Smart and Emerging Technologies, SENSET 2017* 2017-Janua (2017), pp. 1–4. DOI: 10.1109/SENSET.2017.8125019.

[98] M. Sheshikala; R. Vijaya Prakash; and D. Rajeswara Rao. "Implementation of K-Anonymity Using Android SDK." In: *Proceedings - 7th IEEE International Advanced Computing Conference, IACC 2017* (2017), pp. 866–869. DOI: 10.1109/IACC.2017.0177.

[99] Alex Snoeren; Marco Yuen; Chris Matthews; Andy Bavier; Tony Mack; Yvonne Coady; Rick McGeer; Paul Mueller; and Joe Mambretti. "GENICloud and TransCloud: Towards a Standard Interface for Cloud Federates." In: 2012. DOI: 10.1145/2378975.2378980.

[100] Statista Inc. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025.* URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (visited on 2020-08-20).

[101] I Stierand; S Malipatlolla; S Fröschle; A Stühring; and S Henkler. "Integrating the Security Aspect into Design Space Exploration of Embedded Systems." In: *2014 IEEE International Symposium on Software Reliability Engineering Workshops.* 2014, pp. 371–376. DOI: 10.1109/ISSREW.2014.29.

[102] Manuel Suárez-Albela; Tiago Fernández-Caramés; Paula Fraga-Lamas; and Luis Castedo. "A Practical Evaluation of a High-Security Energy-Efficient Gateway

for IoT Fog Computing Applications." In: *Sensors* 19(7) (2017). DOI: 10.3390/s17091978.

[103] Victor Sucasas; Georgios Mantas; Ayman Radwan; and Jonathan Rodriguez. "An OAuth2-based Protocol with Strong User Privacy Preservation for Smart City Mobile e-Health Apps." In: *2016 IEEE International Conference on Communications, ICC 2016* 333020 (2016), pp. 1–6. DOI: 10.1109/ICC.2016.7511598.

[104] Vicenç Torra; Guillermo Navarro-Arribas; and Klara Stokes. "An Overview of the Use of Clustering for Data Privacy." In: *Unsupervised Learning Algorithms*. Ed. by M. Emre Celebi and Kemal Aydin. Springer International Publishing, 2016, pp. 237–251. ISBN: 978-3-319-24211-8. DOI: 10.1007/978-3-319-24211-8_10. URL: https://doi.org/10.1007/978-3-319-24211-8_10.

[105] United Nations Department of Economic and Social Affair. *World urbanization prospects: the 2014 revision.* 2014. URL: http://esa.un.org/unpd/wup/Highlights/WUP2014-Highlights.pdf (visited on 2020-09-28).

[106] Chris Valasek and Charlie Miller. "Remote Exploitation of an Unaltered Passenger Vehicle." In: *Technical White Paper* (2015), pp. 1–91. ISSN: 2041-8205. DOI: 10.1088/2041-8205/762/2/L23. arXiv: 1211.5734. URL: http://illmatics.com/RemoteCarHacking.pdf.

[107] Ignasi Vilajosana; Jordi Llosa; Borja Martinez; Marc Domingo-Prieto; Albert Angles; and Xavier Vilajosana. "Bootstrapping Smart Cities through a Self-Sustainable Model Based on Big Data Flows." In: *IEEE Communications Magazine* 51.6 (2013), pp. 128–134. ISSN: 01636804. DOI: 10.1109/MCOM.2013.6525605.

[108] King Hang Wang; Chien Ming Chen; Weicheng Fang; and Tsu Yang Wu. "On the security of a new ultra-lightweight authentication protocol in IoT environment for RFID tags." In: *Journal of Supercomputing* 74.1 (2018), pp. 65–70. ISSN: 15730484. DOI: 10.1007/s11227-017-2105-8.

[109] Moussa Witti and Dimitri Konstantas. "A Secure and Privacy-preserving Internet of Things Framework for Smart City." In: *ACM International Conference Proceeding Series* (2018), pp. 145–150. DOI: 10.1145/3301551.3301607.

[110] Gavin Wood. "Ethereum: A Secure Decentralised Generalised Transaction Ledger." In: *Ethereum Project Yellow Paper* (2018), pp. 1–33. ISSN: 02533820. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[111] Chao Wu; David Birch; Dilshan Silva; Chun Hsiang Lee; Orestis Tsinalis; and Yike Guo. "Concinnity: A Generic Platform for Big Sensor Data Applications." In: *IEEE*

Cloud Computing 1.2 (2014), pp. 42–50. ISSN: 23256095. DOI: 10.1109/MCC.2014.33.

[112] Sai Wu; Xiaoli Wang; Sheng Wang; Zhenjie Zhang; and Anthony K.H. Tung. "K-Anonymity for Crowdsourcing Database." In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (2014), pp. 2207–2221. ISSN: 10414347. DOI: 10.1109/TKDE.2013.93.

[113] Li Da Xu; Wu He; and Shancang Li. "Internet of Things in Industries: A Survey." In: *IEEE Transactions on Industrial Informatics* 10.4 (2014), pp. 2233–2243. ISSN: 15513203. DOI: 10.1109/TII.2014.2300753.

[114] Jennifer Yick; Biswanath Mukherjee; and Dipak Ghosal. "Wireless sensor network survey." In: *Computer Networks* 52.12 (2008), pp. 2292–2330. ISSN: 13891286. DOI: 10.1016/j.comnet.2008.04.002.

[115] Chuan Tao Yin; Zhang Xiong; Hui Chen; Jing Yuan Wang; Daven Cooper; and Bertrand David. "A literature survey on smart cities." In: *Science China Information Sciences* 58.10 (2015), pp. 1–18. ISSN: 1674733X. DOI: 10.1007/s11432-015-5397-4.

[116] Andrea Zanella; Nicola Bui; Angelo Castellani; Lorenzo Vangelista; and Michele Zorzi. "Internet of Things for Smart Cities." In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 22–32. ISSN: 23274662. DOI: 10.1109/JIOT.2014.2306328.

[117] B Zheng; P Deng; R Anguluri; Q Zhu; and F Pasqualetti. "Cross-Layer Codesign for Secure Cyber-Physical Systems." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.5 (2016), pp. 699–711. ISSN: 1937-4151. DOI: 10.1109/TCAD.2016.2523937.

[118] Mengsong Zou; Javier Diaz-Montes; Ivan Rodero; Manish Parashar; Ioan Petri; Omer Rana; Xin Qi; and David J. Foran. "Collaborative Marketplaces for eScience: a Medical Imaging Use Case." In: *2014 International Conference on Collaboration Technologies and Systems, CTS 2014* (2014), pp. 500–506. DOI: 10.1109/CTS.2014.6867615.

[119] Guy Zyskind; Oz Nathan; and Alex Sandy Pentland. "Decentralizing Privacy: Using Blockchain to Protect Personal Data." In: *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015* (2015), pp. 180–184. ISSN: 0888-0395. DOI: 10.1109/SPW.2015.27.