UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA
DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

Dottorato di Ricerca in Tecnologie dell'Informazione
XXXVI Ciclo

Gabriele Penzotti

An Edge-to-Cloud Framework for Privacy-aware Management of Geospatial Data

DISSERTAZIONE PRESENTATA PER IL CONSEGUIMENTO
DEL TITOLO DI DOTTORE DI RICERCA

ACADEMIC YEARS 2020/2021 - 2022/2023

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXVI Ciclo

An Edge-to-Cloud Framework for Privacy-aware Management of Geospatial Data

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Stefano Caselli

Dottorando: *Gabriele Penzotti*

Academic Years 2020/2021 - 2022/2023

To my wife Manuela
To my family

Abstract

In today's fast-paced technological landscape, characterized by rapid innovation and transformation across sectors, a compelling demand emerges for sophisticated and adaptable systems within Smart Environments. These dynamic settings, marked by an influx of data from diverse sources and intricate distributed systems, offer remarkable opportunities alongside distinct challenges.

This Thesis responds to the need for a flexible, scalable framework that adeptly navigates the complexities of modern distributed data aggregation and processing systems, while upholding the paramount principles of data privacy and security.

At its core, the framework introduces a distributed architecture, housing a service placement algorithm that seamlessly spans from the Edge to the Cloud, meticulously crafted in accordance with the tenets of Fog Computing. This architectural approach indispensably relies on data privacy, significantly influencing applications and prioritizing reliable data management.

A second pivotal contribution is the Seamless Data Acquisition Protocol (SEAMDAP), a standard-based and modern approach designed to facilitate data collection within distributed systems. Engineered to be both user-friendly and highly customizable, SEAMDAP streamlines the intricate process of gathering data from a multitude of sources, reducing friction, and enhancing flexibility.

Lastly, the Thesis ventures deeply into the critical realms of data integrity and security, acutely acknowledging the inherent importance of georeferenced data and location verification. A robust architecture is proposed within the framework's toolkit, ensuring that data remains secure during transmission, storage, and processing, and

culminating in the exploration of advanced processing techniques such as Homomorphic Encryption and Multi-Party Computation.

Crucially, the direction taken with this framework is firmly anchored in the pursuit of standardizing the realm of smart environments while proactively addressing identified issues. Several tools presented herein have undergone rigorous testing in Smart Farming environments, each accompanied by compelling use cases. The framework holds particular promise in settings characterized by high heterogeneity, an abundance of georeferenced data, a critical need for interoperability among systems operated by diverse stakeholders, and a strong commitment to data privacy.

Summary

List of Figures

List of Tables

Introduction

In an era where technological innovation is driving a rapid transformation of productive sectors, the adoption of complex and dynamic systems, as found in environments like Smart Farming and Smart Cities, necessitates a highly adaptable and sophisticated approach. These contexts are characterized by a vast amount of information from a complex variety of heterogeneous data sources, such as sensors, IoT devices, and advanced communication networks. This wealth of data offers extraordinary opportunities to enhance the efficiency, sustainability, and quality of various "smart" processes, but it also proposes unique challenges that demand innovative solutions.

In this perspective, this Thesis responds to the need for a versatile and intelligent framework for optimal data processing and information distribution. This framework must be capable of handling the complexity of modern distributed data aggregation and processing systems, where each component is interconnected and influences the performance of the entire system.

One of the most central and overarching features of this framework is its focus on addressing data privacy and security challenges. With the growing concern for protection of personal data and the need to comply with increasingly stringent regulations, secure and reliable data management becomes essential. It is, therefore, crucial to implement security measures that not only protect sensitive data but also ensure that the data owner has full control over its storage and processing.

Another important characteristic is that the framework has been designed to be highly adaptable, capable of evolving with technological changes and integrating new data sources and emerging technologies. To this end, the tools provided by the frame-

work offer varying degrees of freedom to users, enabling them to modularly incorporate additional technologies.

Thesis Organization

This Thesis aims to explore and outline a key approach to address these challenges by introducing a framework designed to operate in such dynamic contexts. A crucial aspect of this framework is its modular nature, allowing the flexible adoption of tools and technologies to tackle the various facets of the proposed scenario.

The dissertation is organized into four chapters as follows.

- Chapter 1 introduces the context and discusses the main concepts of the Thesis, briefly commenting on the framework tools and the motivations behind the design choices.
- Chapter 2 presents a distributed architecture that can be used in systems adhering to Fog Computing.
- Chapter 3 describes SEAMDAP, an operational protocol for the integration of sensory data along the Edge-to-Cloud Continuum.
- Chapter 4 discusses the aspects related to the security of geolocated data in transmission and processing, through techniques such as Homomorphic Encryption and Multi Party Computation.

Chapter 1

Towards Smart Environments

Within increasingly connected and digitalized environments, complex distributed systems are necessary to address emerging challenges and leverage the opportunities that characterize modern life.

Currently, the vision is of a not so distant future where distributed devices and sensors will be widely deployed in urban, rural, and industrial settings. In urban contexts, these distributed systems will give rise to what we call Smart Cities. For example, they will be responsible for real-time traffic monitoring, adjusting public lighting based on human presence, optimizing building energy consumption, and managing security through intelligent video surveillance systems. Simultaneously, in rural areas, these systems will integrate into Smart Farming, supervising and coordinating agricultural operations from seeding to harvesting, with the aim of improving crop yields and reducing the use of resources like water and fertilizers. The deployment of sensors in industrial environments, under the banner of Smart Industry, will enable continuous monitoring of production processes, detecting anomalies and allowing for immediate interventions, facilitating full production planning.

The interconnection of these distributed systems, along with their ability to exchange data and information in real-time, will be crucial for addressing complex challenges such as climate change, population growth, natural resource management, and urban security. Collecting data from thousands, if not millions, of distributed

monitoring points will provide a deeper understanding of environmental and social dynamics, enabling informed decision-making and the ability to anticipate and mitigate potential issues.

Simultaneously, the intelligent use of data generated by these distributed systems will pave the way for new opportunities for innovation in sectors such as artificial intelligence and data analytics. This evolution will foster new applications in areas like personalized healthcare, autonomous mobility, water resource management, and advanced industrial production, significantly expanding the horizon of technological possibilities.

While there are various manifestations of so-called "smart" environments, some of the main common features that distinguish them can be outlined. In particular, the following three overarching themes are highlighted and briefly commented upon, as they are pervasive in the scientific debate and should be considered in the development of the distributed systems operating within them:

- **Data Management and Technologies.** Smart environments will become increasingly data and knowledge-driven, meaning that decisions will be more informed and significantly based on data collection and analysis. Themes such as *connectivity* and *interoperability* are key to ensuring that all devices and systems can efficiently collaborate, creating an interconnected network of isolated but communicating subsystems. This leads to a greater *automation* of decisions, eventually resulting in *intelligent actuation* and, in certain situations, requiring *real-time monitoring* to enable control and operation optimization with minimal latency.

- **Sustainability and Optimization.** Regarding processes, there is a shared focus on *sustainability* and *resource optimization*. The goal is to maximize system efficiency, reduce waste, and optimize the use of available resources. Therefore, sustainability is a common concern, both in *environmental* terms (for example, reducing environmental impact and responsibly utilizing natural resources) and in terms of *ethics* or morality.

- **Participation and Security.** Here, the importance of the human aspect in smart

environments is clearly emphasized, as people should be the primary beneficiaries of sustained technological efforts. One of the main aspects concerns *community engagement and involvement*. Actively involving people (such as citizens, farmers, or employees) in the planning and implementation of solutions is crucial. Fundamental aspects include raising awareness about data sharing, which could benefit the entire community, but primarily the data provider (similar to the *Open Innovation* approach). At the same time, *data security* is a critical concern because sensitive data is typically managed and must be protected from threats and breaches. These aspects are interconnected: security is essential to instill trust in data providers, providing assurances that data is used responsibly, sometimes complying with stringent *privacy* constraints.

These topics are characterized by various and complex sub-topics which, sometimes, do not only concern IT aspects.

1.1 Preliminary Definitions

In this section, the necessary elements for a comprehensive and informed understanding of the proposed framework are introduced and briefly discussed, along with the key concepts that guided its design and implementation.

Crucial concepts, such as Fog Computing and Edge Computing, will be described in the following of this Section. Then, the concepts of data privacy and georeferenced data will be commented upon, since these are fundamental elements to ensure that the proposed framework meets data security and ownership requirements while providing an optimal and personalized data processing service.

1.1.1 Paradigms

Modern distributed systems are designed with a clear focus on and orientation toward specific models of distributed computing, data collection, and service delivery. These paradigms have become established thanks to the ever-increasing adoption by both the scientific community and device manufacturers and users.

In particular, the proposed framework adheres to the paradigm of resource distribution that is increasingly gaining prominence, namely the *Internet of Things (IoT)*. The framework does not provide specific solutions for implementing IoT solutions but integrates tools capable of operating in environments characterized by a high and widespread presence of smart objects with sensory, computational, and actuation capabilities, possibly generating a vast amount of heterogeneous data.

From a computational perspective, the framework offers solutions to address issues encountered in the paradigm of *Fog Computing*, and by extension, in *Cloud* and *Edge Computing*. The efforts in this regard aim to achieve comprehensive solutions that provide full control over data and processing in the *Edge-to-Cloud Continuum*.

Next, we will further discuss the paradigms of distributed systems and how the proposed framework adopts their concepts.

Internet of Things (IoT)

The Internet of Things (IoT) is a concept that refers to the connection of physical devices to the Internet, from sophisticated industrial sensors and actuators to everyday objects, enabling them to communicate, exchange data, and interact with each other. The IoT approach is widely used in various sectors, including Smart Farming, Smart Cities, and many others, *Smart Objects* are increasingly pervasive. IoT devices generate data that can be collected, analyzed, and used to enhance efficiency, automation, and final decision-making.

This abundance of intelligent nodes brings many advantages, which from a systemic perspective can be summarized in the ability to generate large volumes of data in a short period, causing the design of distributed systems to consider the fact that they are dealing with Big Data environments.

While the proliferation of IoT nodes brings about numerous benefits, it is crucial to acknowledge that these advantages do not come without challenges and requisite technical trade-offs. In practice, the capabilities of smart objects across various implementations tend to be quite modest. These limitations manifest in several key areas: computational power, persistence, and energy resources are available in more or less restricted quantities, and external communications may not be available all the

time.

The framework, in fact, does not address specific issues in the IoT domain, such as low-power communication and the battery life of devices, as the framework solutions are intended to operate from the upper level of the network. While it does not offer specific IoT implementation solutions, the framework provides tools to manage and process IoT data efficiently, and some of them can also be adopted by resource-constrained devices.

Indeed, the tools of the framework have been designed to build distributed systems that deal with data sampling performed by IoT devices, accommodating the handling of large volumes of diverse data generated by smart objects, aiming to improve and optimize certain aspects of data management and processing in the upper layers.

In the upcoming chapters, terms like "IoT node" are used to refer to any Smart Object with typically low processing and storage capabilities, but which, in most cases, performs sensing or actuation activities. Additionally, the term "IoT Layer" is used to describe the collection of IoT nodes within the system at hand.

Cloud Computing

Cloud Computing refers to a method of delivering computing services over the Internet, allowing users to access computer resources such as servers, storage, databases, networks, software, and more without the need to own or manage the local physical infrastructure. This level offers high computing capabilities, extensive scalability in storage, and sophisticated data analysis tools. It is particularly suitable for intensive analysis tasks that require significant computing power and long-term resources, without the need to physically own or manage the hardware and software. This approach offers scalability, flexibility, and ease of management, allowing organizations to focus on their core activities without worrying about the underlying infrastructure.

In the context of this Thesis, Cloud Computing problems are not addressed. Instead, a Cloud Node considered as a high-capacity node at the extreme end of the network hierarchy, and as such, framework tools could be used by its node. From a computational perspective, any instance of Cloud Computing can be included and

considered as a network element without delving significantly into its specificities, such as particular deployment methods, vertical scalability, and more. It is regarded as an area of processing and storage that is always available and with virtually unlimited capacity.

However, when considering issues related to data privacy, ownership, efficiency, and data location, a Cloud Computing node is considered for deployment only when necessary.

Edge Computing

Edge Computing refers to a data processing model where computation and processing are moved closer to the data source, rather than being performed on remote servers in data centers or the Cloud. This approach could move all the data processing towards sensors and actuators, or only an initial part, involving filtering, aggregation, and preliminary data analysis to identify the most relevant information. In practice, Edge Computing is based on the ideas of bringing computing and data processing as close as possible to the devices or sensors that generate the data, and minimizing latency on delivery of processed results to the end-users.

This approach is particularly advantageous when a rapid, real-time response is required, as it reduces the latency compared with sending data to the Cloud for processing. In contexts with high data flows, such as IoT, Edge Computing avoids flooding the network with useless data: preliminary data analysis, filtering, and data synthesis could be performed directly on the devices or local nodes before sending only the relevant information to the Cloud for further processing.

Another important factor is geographical proximity. Performing processing closer to the source offers several advantages, such as the ability of using short-range communication protocols, limiting the capacity of potential attackers to intercept data, avoiding increased noise in the network and reducing latency. Moreover, using solutions for the spatial limitation of data can support correct data governance policies, ensuring that data are physically confined to the place (e.g., a node or a group of nodes) of interest, and thus also satisfy privacy requirements.

The concept of Edge Computing appears in various nuanced interpretations. In

this Thesis, a distinction is made between Edge nodes and IoT nodes: Edge nodes are typically considered as those closest to IoT nodes, distinguished by the fact that Edge nodes do not perform sensing or actuation. In practical examples, gateways and sensor data hubs can be classified as Edge nodes.

These nodes occupy the lowest part of the network hierarchy that the framework works with. For instance, in Chapter 2, it is discussed how certain algorithms are preferably deployed on Edge nodes, which are considered to have lower latency.

Fog Computing

One of the most advanced manifestations of distributed system models is considered to be Fog Computing. This is a data processing model that does not limit itself to considering only the nodes in the Cloud and Edge layer, but also everything lying between them. In practice, Fog Computing extends the concept of Edge Computing by moving data processing from the data source (IoT devices, sensors, etc.) to distributed servers in the "fog" of the network infrastructure. Bridging the gap between the Cloud and end devices facilitates operations like computation, storage, networking, decision-making, and data management on network nodes situated in close proximity to IoT devices.

This approach provides greater computing and storage capacity compared to edge devices, while maintaining closer proximity to the data than centralized Cloud data centers, and ensures that these critical functions occur nearest to data sources and actuators.

Fog computing, as defined by the OpenFog Consortium [1], represents a horizontal system-level architecture that distributes computing, storage, control, and networking functions closer to end-users across a continuum extending from the Cloud to IoT devices.

Fog computing allows computing functions to be distributed across various domains, fostering a collaborative environment, enabling the realization of "horizontal" platforms. An horizontal platform is a comprehensive architectural solution, obtainable from a synergistic collaboration of multi-purpose services, also belonging to different actors, which are integrated to offer results to different users for different pur-

poses. In contrast, a "vertical" platform is more specialized, offering strong support for a single type of application (a silo) but lacks the capacity for inter-platform interactions with other vertically focused platforms. Indeed, horizontal platforms have more valuable advantages in scenarios like Smart Farming, such as greater scalability and reduction of adoption times and costs, but are burdened by a high complexity and possible lack of specialization. However, Fog Computing is seen as an element capable of simplifying and making this solution effective.

In addition to promoting a horizontal architecture, Fog Computing offers a flexible platform that can adapt to the data-driven requirements of operators and users. Its primary goal is to provide robust support for the IoT, enabling efficient and responsive operations in the rapidly evolving landscape of connected devices.

An important concept associated with Fog Computing is that of geographically limited network. In applications where data and nodes are managed in territorial groups (e.g. local, regional), Fog Computing, as will be discussed in Chapter 2, is well suited to take into account these characteristics.

While various interpretations exist, in this Thesis "Fog Layer" (and its corresponding "Fog nodes") are assumed to be nodes that occupy any intermediate position between the Edge and the Cloud.

Edge to Cloud Continuum

The Edge to Cloud Continuum encompasses a perspective on data processing that emphasizes the continuity and complementarity between Edge Computing and Cloud Computing. In this context, these two approaches are not seen as exclusive alternatives but rather as extremes of a continuous spectrum of computational and storage resources.

Therefore, the concept of the Edge to Cloud Continuum underscores the importance of viewing data processing as a scalable and continuous process that adapts to the specific needs of an application. Data can be processed in various ways along this continuum, taking into account considerations such as latency, scalability, analytical complexity, and other factors. This approach allows for the optimal utilization of available resources and the optimization of computing solutions' effectiveness in

scenarios like Smart Farming and Smart Cities, where data processing requirements can vary significantly.

In this work, Fog computing is considered as the reference paradigm of an Edge to Cloud Continuum data management, capable of enabling the seamless integration and synergy between Edge Computing and Cloud Computing.

1.2 Data Management in Smart Farming

In this section a description of the data management challenges, that are encountered in Smart Farming, is reported. The insight reported below was gained in interactions and researches, especially within the POSITIVE project, that allowed the ideas exchange with many stakeholders who gravitate around modern agriculture.

POSITIVE [2] [3] has been a project of the Emilia-Romagna region for the creation of a consultancy and implementation service for irrigation and fertigation by efficiently processing data from satellites and sensors in the field. Among the main results, POSITIVE has developed open operational protocols for the interconnection of components in the irrigation ecosystem, and designed a farm information system that supports all precision irrigation activities in an integrated manner. Some of these protocols will be detailed in Section 3.4.

The monitoring and digitalisation of agricultural processes is not a trivial challenge. Farming presents intrinsic difficulties that hinder IT solutions, due to the type of activities carried out and due to legacies and common practices that are not beneficial for a smart approach.

1.2.1 Data Sources

To build an *information system* capable of bringing a solid advantage to the farm, optimizing farming practices and achieving sustainable yields, it is essential to identify the **sources of information**. In Smart Farming there is an abundance of different data sources [4], including sensors, satellite imagery, weather stations, drones, and more. These sources provide data about different characteristics of the environment, as for example soil conditions, weather patterns, and crop health.

These characteristics can be obtained by external organizations or farms, such as weather stations openly available to the public. In that case, it is necessary to relate to it, otherwise it is necessary to start a "sensorisation" process. Smart Farming is marked by an increasingly broad presence of commercial **sensors** and **actuators** [4], and thanks to the availability and affordability of IoT devices, the number of sensors and actuators deployed in agriculture is expected to grow significantly [5].

However, this leads to modern scenarios where agricultural applications require *data from diverse sources*, often provided by different manufacturers. For instance, a farmer may use soil moisture data from one manufacturer's sensors, weather data from another source, and crop health data from a completely different vendor. Integrating and making sense of data from these distinct sources is a significant challenge but is essential to create comprehensive insights.

This variety of information paths, has effects on various data management activities, starting from the simplest raw data acquisition, up to the most complex and intelligent information processing, and highlight the need for tools and structures capable of enabling *data fusion from multiple sources*.

1.2.2 Data Collection

One of the most relevant and crucial, but at the same time most limiting aspects regarding the circulation of information, concerns standardization of *data dissemination*. In fact, if at low IT levels there are protocols that have a greater adoption [6], going up to the application level there is a wider range of choices.

Confronting with the array of commercially available **sensors**, it is evident that they are *highly heterogeneous*. From an IT perspective, this variety is related to the technical characteristics of the product (e.g., environmental sampling characteristics, area of interest for data acquisition, digital or analog interfaces) and in how data is collected and communicated (e.g., data accuracy, format, collection frequency, volume), and sometimes in the presence or absence of open access modes to the product. Sometimes these characteristics are inherently tied to the product's construction, intrinsic to the developed system, while at other times, they are choices made by the manufacturer.

The deployment **sensor networks** also often exhibits a diverse approach [7]. Confronting with farmers, often ad hoc solutions are in place: it is possible to find nodes already equipped with integrated sensors capable of autonomously transmitting data externally, or sensors connected (maybe in non standard ways) to nodes with limited data storage and transmission capabilities. These nodes can implement different communication protocols and interfaces with varying data transmission intervals.

Many attempts have been made to establish ordered taxonomies and ontologies to standardize sensor **data collection** [8], but these have been often limited to certain network levels or specific use cases. In summary, what was lacking during our researches was a modern data collection approach, oriented towards the emerging architectures in distributed systems.

1.2.3 Data Orchestration

In Smart Farming it is necessary to design the system which maintains and processes data, in order to obtain an advantage from them. The heterogeneity of information to be orchestrated, the lack of systemic standardization and the dynamism of the environment are some of the factors that bring non-trivial organizational challenges in the design of the system.

To deal with all the above characteristics of Smart Farming scenarios, common approaches for building systems are to develop **ad hoc systems**, often designing a data management solution different from another. However, this approach is very limited in terms of distributed processing, often relying on sensing at the IoT level and then uploading and processing data in public Cloud, typically without an architectural organization optimized for the provided services [9, 10, 11].

A classic IoT-cloud system obviously poses challenges in terms of scalability and performance, instead highlighting the potential solutions based on modern paradigms such as *Edge* and *Fog Commuting* [12] [13]. A distributed system based on these paradigms brings many advantages and allows the creation of solutions that better respond to the problems of Smart Farming.

Indeed, one of the factors that differentiates Smart Farming from other environments is the strong presence of **interrelated phenomena**. For example, changes in

weather conditions can affect water sources, which, in turn, impact crop irrigation needs, and furthermore, tracking pollution levels and pest infestations may require information on weather patterns and soil conditions. An integrated approach to collecting and analyzing data from these interrelated phenomena can lead to more effective decision-making, but brings new challenges, for example the ability to interact with different services that travel at different speeds and affect different application domains.

This highlight the need for architectures enabling interoperability and scalability, allowing data sharing with other actors for *collaborative solutions*.

1.2.4 Data Exploitation

In the contemporary landscape of agriculture, application development operates within a unique and complex environment characterized by a confluence of distinctive attributes. This scenario offers a distinct set of challenges and opportunities, giving rise to a novel paradigm in application development.

First and foremost, applications in this context necessitate copious volumes of data, effectively positioning themselves within the realm of **Big Data**. A tendency of aggregating data from various sources into a central repository, such as a *data lake*, could be observed [14]. This data collector can store information from multiple farms, regions, and sources, allowing for comprehensive analysis. It serves as a foundation for advanced analytics and long-term trend analysis, enabling the identification of patterns and the development of predictive models.

The vast array of sensors, geographical sources, and dynamic inputs necessitates systems capable of processing and analyzing data at large scales. Indeed, a relevant approach by consortia, associations, administrations or other entities is to create and make available (free of charge or not) **expert systems** [3] [15] in order to provide agricultural insights and recommendations (e.g., Decision Support System). These systems rely on large datasets, ingested and processed from various farms and regions, and complex algorithms to generate valuable insights and offer customized guidance. For instance, they can suggest optimal planting times, irrigation schedules, and pest control strategies.

Concerning the user-side usability of applications, **direct consultation tools** are essential for farmers and other agricultural stakeholders. These tools include dashboards and applications that provide near real-time information. Farmers can use them to monitor crop conditions, track weather forecasts, and make immediate decisions regarding irrigation, pest control, and more. Data at this level should be highly responsive and available in a user-friendly format. Furthermore, applications may be capable of automating some processes, although in this case the presence of *intelligent actuators* capable of remote programming and/or activation is required [3].

In literature, there is a tendency towards the creation of farming applications with **distributed approach** [16, 17]. Unlike centralized systems, which are generally preferred for vertical applications, distributed systems can decentralize data persistence and processing. Moreover, data can be managed near to the edge, closer to the data source, suitable for real-time applications. However, nodes will come in different forms, from high-capacity servers to resource-constrained, low-power edge devices. The data management system must efficiently utilize these varied resources based on the context and needs of each application.

Collaborative services could be able to handle the **dynamic and heterogeneous scenarios** that Smart Farming is going to, with continuous additions of new actors (both public and private), sensor types, and applications. New services could be deployed and made available more easily compared to other environments: in fact, many of the phenomena affecting agriculture can be openly sampled or information could be used by everyone, for example meteorological data or satellite observations. These new entrants may bring their proprietary data formats and communication protocols, increasing the heterogeneity of data sources, unless a structured approach to the data acquisition problem is adopted.

Distributed system orchestration could lead to **integrated platforms** [18], ambitious tools for which some attempts can be found, although often vertical and created on non-open protocols. These platforms incorporate a wide range of functionalities, from monitoring and analysis to automation and decision support. They offer a holistic view of the farming operation and often feature intelligence capabilities. Indeed, *artificial intelligence* plays a critical role at various levels of data management in agri-

culture [19]. Machine learning algorithms can analyze vast datasets to identify trends and make predictions. For instance, AI can predict crop yields, disease outbreaks, and optimal harvesting times. AI-powered systems can continuously learn from new data, improving their accuracy over time.

Crucially, these applications must contend with the multifaceted nature of data. In an agricultural setting, data streams flow from diverse origins (e.g, sensors, satellite imagery, weather stations) that differ not only in their nature but also in their temporal and spatial dimensions.

Indeed, **geographical data**, forms the foundation upon which these applications operate. The spatial dimension is integral to agriculture, as it influences everything from soil composition to weather patterns. Applications must adeptly handle geodata, incorporating precise geolocation information to tailor recommendations and actions based on specific field conditions.

Privacy and **security** emerge as paramount considerations in this landscape [20]. The wealth of personal and sensitive data, from proprietary farming practices to location-specific details, necessitates robust security measures. Users entrust applications with their data, and it is incumbent upon developers to ensure the highest levels of data protection. Simultaneously, maintaining privacy is crucial: data anonymization and controlled access are key components of responsible application development.

Both the topic of geographic data and that of data privacy and security are very relevant, especially when combined together (see Chapter 4). Various efforts integrate both concepts, but what is missing is a fundamental and harmonious integration with development tools, which on the contrary in this framework is seen as a founding element.

1.3 Framework: Description and Motivations

The developed framework is a collection of tools and operational practices designed for the creation of distributed systems with a focus on Edge-to-Cloud data management. These tools are particularly effective in dynamic environments characterized by

the presence of Geospatial Big Data and a strong emphasis on privacy considerations.

The framework was developed incrementally, with elements added during the PhD journey, to address various challenges encountered.

In this section, some general features of the framework are commented. Subsequently, a brief summary of its constituent elements is provided.

1.3.1 Framework Composition

The framework is designed as a set of tools and practices to enable efficient and secure processing of georeferenced data in dynamic smart scenarios. The designed tools can be used individually or integrated together. The key elements are described below.

E2C Distributed Architecture

In literature, particularly in Smart Farming, approaches building systems that are often ad hoc, with distributed processing primarily limited to data upload and processing in the Cloud and few Edge processing approaches, as will be discussed in Chapter 2.

The first contribution of the proposed framework is about architectural aspects, identifying and addressing the needs emerging from the operational environment. In this regard, a distributed architecture that extends along the Edge-to-Cloud continuum is introduced in Chapter 2. The primary design focus has been to ensure reliable and secure data management, as well as smooth data acquisition and processing throughout their journey from the source to higher levels of analysis. The architecture defines a well-structured hierarchy adhering to the Fog Computing paradigm [1], enabling efficient management of computational and storage resources and optimized data processing distribution based on specific requirements at each level. Great attention has been paid make the architecture applicable in various contexts.

Seamless Data Acquisition Protocol

Beyond architecture, another significant challenge emerged during sensor data acquisition. Many attempts were made to establish ordered taxonomies and ontologies to standardize sensor data collection, but these were often limited to certain network levels or specific use cases, as will be described in Chapter 3. A data collection method, oriented towards the emerging architectures in distributed systems, was lacking.

Therefore, the second element of the proposed framework consist in the development of an application protocol for seamless data acquisition, offering a standard seamless approach at every level from Edge to Cloud. This protocol establishes formats and operational phases, allowing room for customization for specific use cases.

A major design goal has been to be enough generic to be independently adopted by sensor manufacturers or installers, promoting an active community participation perspective, but with several possibilities for extension and customization, in order to adapt to specific contexts. The primary innovation lies in its design for modern distributed systems, so that the protocol can be adopted with minimal effort and without substantial modifications in existing systems. The protocol's features are described in Chapter 3.

Secure Data Processing and Exchange in Untrusted Environments

Particular attention was given to issues related to data privacy and the security of processing and communications. An increasing sensitivity towards personal data can be observed among users. This sensitivity can be broken down into two aspects.

The first aspect concerns data security in the strict sense, which includes protection in terms of integrity and confidentiality during both data transmission and storage. This is achieved through the use of appropriate encryption systems, which are almost always applicable to devices from the Edge upward. While these classical techniques are not directly addressed within the framework, they are often mentioned, and none of this work proposed solutions precludes their adoption. The protection of processing in untrusted environments (where full trust cannot be placed both in servers or participants involved in the process), especially concerning data containing

geolocation, is discussed in Chapter 4. In particular, the possibilities of integrating solutions based on techniques such as *Homomorphic Encryption* (HE) and *Multi-Party Computation* (MPC) are discussed.

The second aspect is more related to the concepts of *privacy* and *data ownership*. Data privacy involves the protection of personal and sensitive information from unauthorized access and improper use. This concept is even more significant, especially for applications with high population participation. The entity responsible for aggregating such data could potentially access a large amount of personal information without the users awareness. This issue can be resolved through data collection and aggregation procedures that allow for a high level of user information anonymization, where possible. Data ownership refers to the rights, responsibilities, and value of the data itself. As data owners, individuals should have the ability to consciously and conscientiously share their data, with the capacity to decide where and how their data is stored and processed. These latter two themes are addressed within the framework as cross-cutting characteristics of the various tools.

Another relevant theme is georeferenced data, which includes spatial location information allowing data to be associated with specific points on the Earth's surface. This subject is becoming increasingly important in modern systems, which use location data for various purposes, primarily to provide effective and precise results to end-users.

Location-based services (LBS) and the concept of Proof of Location, which refers to the ability to securely and accurately verify the location of an object or device at a given moment, are often discussed in this context. This is particularly significant when working with spatial information, as errors or uncertainties in determining location can negatively impact the accuracy of analyses and decisions.

This theme is also highly relevant in the paradigms of modern distributed systems, such as Fog Computing, where, for example, territorial clustering of nodes with regional result aggregation on dedicated nodes is designed. All these aspects are discussed broadly across various tools and a particular approach is discussed in Chapter 4.

1.3.2 Research Questions

The research documented in this Thesis was conducted to address several issues discussed in general terms in the sections above. The research themes expressed are very complex and broad to satisfy entirely, but some specific aspects were more compelling and guided this Thesis work. Some of them arose after discussions with farming stakeholders, but the resulting solution was designed with a broad scope for all smart environments.

The most relevant key research questions that stood out are listed below.

RQ1 *Can a stakeholder fully trust a distributed, E2C smart system? In particular, are data safely managed under the owner's control, sharing only the desired information with the desired recipients?*

This requirement is often expressed directly by stakeholders (e.g.,farmers), who must rely on service providers that manage farm data. These data can have a high value, therefore strong and structural guarantees must be given to the owners.

RQ2 *What is the best strategy to develop a data reliable management system for Smart Farming?*

This question is more relevant for application and system developers, facing the necessities of smart farming. This question concerns the development of the entire system (rather then the individual services), the interactions between the nodes, and the methods for carrying out complex tasks.

RQ3 *Are E2C system beneficial and able to effectively satisfy the requirements of Smart Farming?*

Specifically, the response should be a reference way to create, position and effectively connect the various components, respecting domain constraints. The difficulty of this challenge is aggravated by the lack of tools applicable in generic situations and capable of managing complex data flows. The framework itself is intended to be a response to some aspects of the design process of a system.

RQ4 *How can existing and new systems work in environments with high sensor data heterogeneity?*

Referring mainly at the sensors panorama (broader and more varied than the actuators one), a problem emerges regarding heterogeneity and the so-called *technological silos*. Often, those who pay the price are stakeholders with less technical skills (e.g., farmers), since they have to deal with isolated and non communicating environments.

RQ5 *In a collaborative scenario, how to obtain and disseminate data, respecting data privacy and data ownership?*

Modern applications demand a significant amount of data to work, and those data need to flow efficiently, respecting every actor's needs. Especially in Smart Farming, events can be correlated between farms (e.g., weather) and sharing data in a collaborative way can bring advantages to everyone.

1.3.3 Main Aspects Covered

As previously commented, the framework operates in a context characterized by a wide variety of aspects. Some of these aspects could be easily found in a different smart environments.

However, the framework tools focus only on a subset of these aspects, which are most relevant in Smart Farming as well as other context. The main covered aspects are:

- **Massive Data Presence and Non-Uniform Aggregation.** At the heart of this scenario is the pervasive presence of sensor data generated by a variety of sources. These data require intelligent and non-uniform aggregation across different nodes of the system, considering the heterogeneous nature of available resources.

- **Scalable Multilevel Processing.** Data processing can occur at various levels of resources, requiring efficient planning and optimal management of information

flows between system nodes so that each node can act as a provider and/or consumer of data or services. The criteria influencing the placement of processing services include the presence of intelligent nodes capable of consuming both high-level information and low-latency data.

- **Data Privacy and Ownership.** The presence of multiple data owners with varying degrees of attention on their personal information, emphasizes the importance of ensuring data privacy and integrity, as well as enabling selective and secure information sharing.

- **Georeferenced Data and LBS.** The use of georeferenced data is essential for working in extensive geographical spaces with high positioning accuracy. The integration of LBS adds an additional layer of complexity and opportunities to enrich information but also brings additional challenges related to data privacy and security.

Chapter 2

Edge-to-Cloud Distributed Architecture

In this chapter, the reference architecture is presented and discussed. It is derived from the research paper "An N-Tier Fog Architecture for Smart Farming" [21] authored by Penzotti G., Caselli S. and Amoretti M.. The paper makes two main contributions: an N-tier fog architecture and a service placement algorithm implementation and simulation, that considers data privacy constraints.

This distributed architecture was designed considering the characteristics and needs of *Smart Farming* systems. The examples and considerations that follow will therefore be related to systems in the Smart Farming field, as well as the comparison that was made with the literature works during the first phases of the research. However, it is important to note that the design was carried out by abstracting and generalizing the problems in such a way as to make the architecture suitable for use in other smart environments as well, therefore to bring advantages and create effective systems in situations with needs similar to those described in this chapter.

This tool was designed as a response mainly to RQ1, allowing the design of a system that by design offers various guarantees to data owners. It also partially answers RQ2, promoting the use of a multi-level paradigm rather than a Cloud-IoT dualism.

The subsequent sections of this chapter are structured as follows. Section 2.1 offers an overview of related work on Fog Computing in the context of Smart Farming. Section 2.2 provides an in-depth illustration of the proposed architecture. Section 2.3 delves into the service placement scenarios, while Section 2.4 elucidates the proposed service placement strategy. Section 2.5 presents the findings from the simulation analysis.

2.1 Motivation and References

The advent of the digital agricultural revolution [22] is poised to revolutionize every facet of farming, ushering in more productivity, efficiency, sustainability, inclusivity, and resilience in agricultural systems. Achieving this integration of digitalization in agriculture hinges upon the maturation and intricacy of technologies encompassing precision agriculture, remote sensing, Big Data, analytics, cloud computing, cybersecurity, mobile devices, and intelligent systems [23].

While various solutions have emerged in the realm of Smart Farming, they remain in an early stage, offering limited intelligence. Many of these solutions are bound by automation constraints, with sensors and actuators transmitting data to private gateways isolated from the Internet [22]. Such an approach restricts the scope and scale of available data processing services. In some cases, data are routed to and processed in public clouds [9, 10, 11]. However, this practice of sending all data to the cloud proves costly and resource-intensive, especially in terms of bandwidth requirements [24]. Furthermore, the physical and virtual distances between sensors/actuators and the cloud substantially impact latency, leading to the degradation of service quality.

Yet another major challenge associated with the all-in-cloud approach pertains to data privacy. The lack of legal and regulatory frameworks governing the collection, sharing, and use of agricultural data [25] adds to the array of challenges faced by farmers contemplating the adoption of Internet-based Smart Farming technologies.

Hence, rather than transferring substantial volumes of agricultural data to the cloud, a more practical approach is to embrace Fog Computing [16, 17]. This approach entails moving computation from the cloud closer to the edge and, poten-

tially, directly to IoT devices. The constituent components of this model, encompassing computation, networking, storage, and acceleration, are denoted as "fog nodes," situated between the cloud layer and the IoT layer [1].

Intermediate Fog nodes enable a seamless integration between IoT/Edge and Cloud, presenting fundamental characteristics, such as: greater proximity of computing and storage capacities, reducing latency, network traffic, bandwidth waste, and ensuring that time-sensitive decisions can be made closer to the edge; increased load distribution between collaborative nodes, not centralizing everything in a few cloud nodes and allowing rapid offloading to nearby nodes; capacity to implementing more secure data governance policies, respecting data privacy; increased reliability of networks, reducing the points of failure.

Another aspect that emerged as very relevant concerns the management of users' data. Based on interactions with stakeholders [26], increasingly sensitivity and attention protection and controlled dissemination of personal data is emerging among users in Smart Farming domain. This has emerged in two main trends. The first concerns the will to not disclose sensitive personal data, often choosing not to provide information externally, unless in anonymous and certified way. The second aspect concerns a growing awareness of the value of data, economic and not, and therefore a desire not to make them available unless in expectation of a return (such as, for example, a better service) and a certification of a clear data management plan.

The data privacy topic is currently very relevant [20], but not broadly investigated in Smart Farming architectures literature. In order to create privacy–aware systems, the architecture proposed in this chapter includes among its design principles the concept of "data privacy", i.e., the protection of a user's sensitive data from access by third parties.

2.1.1 Reference Works

O'Grady et al. [12] provide a comprehensive survey of the current state of research employing the edge computing model in agriculture. This study highlights the potential of edge computing in various agricultural domains, but it outlines that much of the research remains in the prototype stage. While the potential benefits are evident,

there are several systemic challenges that need to be addressed to translate these concepts into meaningful impacts at the farm level. Two fundamental issues that continue to impede progress are interoperability and scalability, factors that have inspired this exploration of N-tier fog architectures.

In our perspective, the effectiveness of an architecture hinges on the applicability of service placement scenarios to real-world use cases. For instance, insights from solutions like the one presented by Angelopoulos et al. [27] have been drawn, which addresses smart strawberry irrigation in greenhouses by keeping relevant data at the network's edge. The authors concluded that their smart irrigation approach significantly outperforms traditional methods, both in terms of soil moisture control and water consumption.

Evaluating the efficiency of a fog-based solution, Ribeiro et al. [13] proposed an approach for collecting and storing data in smart agriculture environments, along with two distinct methods for data filtering within the fog layer to reduce the volume of data transmitted from the fog to the cloud. In a similar vein, Gia et al. [28] integrated artificial intelligence at the local network layer (Edge AI) to introduce a system architecture and implementation expanding the capabilities of smart agriculture and farming applications through edge and Fog Computing, coupled with LPWAN technology for large area coverage. While focusing on a different application context, Taneja et al. [29] introduced SmartHerd, a Fog Computing-assisted end-to-end IoT platform for animal behavior analysis and health monitoring in dairy farming scenarios. The platform adheres to a microservices-oriented design to support distributed computing and addresses the significant challenge of constrained Internet connectivity in remote farm locations. With fog-based computational assistance in the SmartHerd setup, the authors observed an 84% reduction in the volume of data transmitted to the cloud compared to conventional cloud-based approaches.

In a recent endeavor, Malik et al. [30] introduced a distributed toolkit facilitating users to simulate custom farming scenarios, encompassing sensor placement identification, data gathering, mobility models for mobile nodes, energy models for on-ground sensors and airborne vehicles, and backend computing support using the Fog Computing paradigm. The proposed framework also offers benchmarking capabili-

ties related to transmission delay, packet delivery ratio, energy consumption, and system resource utilization. The simulation toolkit builds upon FogNetSim++, although that platform is no longer actively maintained. In this research, a more versatile and efficient tool for simulating Fog Computing systems was required, leading us to adopt YAFS [31], whose support for dynamic network configurations offers intriguing opportunities for studying realistic service placement scenarios.

2.2 Architecture Description

In this section, the proposal for an N-tier fog architecture enabling large scale Smart Farming with performance and data privacy guarantees is presented. The architecture leverages the general concepts and philosophy of the IEEE 1934-2018 standard [1] proposed by the OpenFog consortium (merged in 2019 with the Industrial Internet Consortium).

Smart farming is characterized by many different and highly heterogeneous activities, considering both direct and support ones, as part of the global production chain. Driven by the need to reach ever higher standards in terms of quality, quantity and low environmental impact of the final product, but also by the requirement to deal with constantly changing environmental situations, these activities are increasingly computerized and automated, in the context of general greater accessibility to the most advanced technologies.

The main Smart Farming activities that are characterized by a growing adoption of IT technologies are listed below.

- Monitoring of production sites (e.g., terrain), irrigation water deposits and the crops themselves.

- Monitoring and forecasting of weather conditions (to predict high-impact events, such as drought and frost).

- Planning, automation and refinement of fertigation and irrigation activities.

- Reliable and safe traceability of products and process for compliance with market and law standards.
- Campaign diary management.

Current distributed solutions mostly rely on *IoT nodes* for sensing/actuation, and *cloud services* for storing and processing data. In the following, a summary of the main features of these two layers is reported, since the proposed fog architecture lies between them and interacts with them.

At the *IoT layer*, an huge amount of data is generated from the observation of the variables of interest, such as humidity, temperature, wind intensity and size of crops. Furthermore, at this layer it is possible to intervene and modify the surrounding environment through the actuators. IoT nodes generally have a limited and specialized processing capacity, which has an impact also on security (complex cryptographic functions cannot be executed). Some of them are provided with basic fog functions (i.e., they expose services to the upper layers, usually corresponding to data preprocessing functions). Last but not least, the IoT layer is widely characterized by different connectivity solutions, including wired and wireless protocols.

Regarding the *cloud layer*, a partial list of Smart Farming services, without distinction between public or private ones, is given below.

- Decision Support Systems for the assistance of the farmer in all phases of the crops life.
- Weather monitoring or forecasting services.
- Sensor data collection and consultation services.
- Services for control and monitoring of actuators for soil processing or irrigation.
- Remote sensing data (e.g., satellites) consultation services.

Clearly, the effectiveness of these services depends on the quality and amount of data they are fed with. Some of them are data-intensive, others are compute-intensive.

The main advantage of having them in cloud is resource elasticity. On the other hand, deadline satisfaction and data privacy cannot be fully guaranteed.[1]

To cope with these issues, an assumption is that the services that could be placed in cloud, can be horizontally scaled by creating stateless instances in the fog layer. The latter has N tiers, with different features.

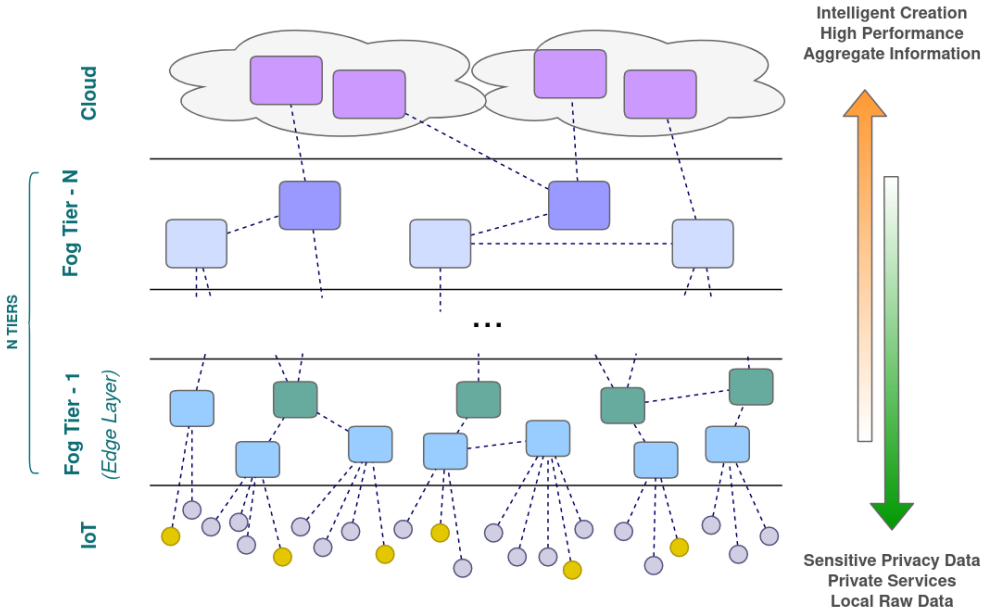

Figure 2.1: Schematic representation of the proposed N-tier fog architecture, located between the Cloud layer and the IoT layer. For simplicity, Edge layer can be transposed as the lowest Fog tier.

It is important to note that, in order to simplify the discussion of the topics, the concept and features of *Edge Computing* are not relevant. In fact, from the point of view of the service placement algorithm (Section 2.4), the characteristics of Edge nodes (such as accessibility and the low if not total absence of services externally offered) can be considered specific cases of Fog nodes, by appropriately setting the

[1] In this work, the cloud is assumed to be untrusted. In the literature, this assumption is quite common [32, 33, 34, 35].

parameters of configuration of the quantitative model (Section 2.3.1). Therefore, the Edge Layer can be transposed as the lowest Fog tiers.

2.2.1 Fog Tiers Organizations

A schematic representation of the N-tier fog architecture is depicted in Figure 2.1. Notably, the cost associated with implementing this infrastructure can be distributed between private and public entities.

The configuration of fog tiers and the allocation of fog nodes within each tier are contingent upon the particular application context [1]. A pivotal factor is the quantity of sensors and actuators within the IoT domain: a greater density of these components necessitates a higher number of fog nodes for effective management. An equally critical consideration is geographical dispersion: fog nodes situated in lower tiers must maintain close proximity to sensors to ensure minimal latency, enabling enhanced control of user data. Furthermore, the spectrum of service typology needs careful attention. As previously mentioned, aside from data aggregation, storage, and remote control of actuators, the array of services in the Smart Farming domain can be diverse and computationally intensive. Consequently, a commensurate number of nodes within the uppermost fog tiers becomes imperative.

In this conceptualization, the composition of Smart Farming-oriented fog tiers is primarily delineated by four key dimensions: data/information, services, resources, and geographic coverage. These discerning elements are detailed below.

Processed data and produced information. Processed data and produced information are characterized by granularity and sensitivity. Fine grain, highly sensitive data are treated at the lower tiers. Coarse grain, less sensitive data are handled at the higher tiers. More specifically, at the lower tiers there are countless activities of raw data generation and processing, while at the higher levels there is a considerable production of more sophisticated information. In order to be able to realize Smart Farming functions such as precision irrigation, IoT data must be associated with context information such as geolocation, type of crop, owner, etc. At the higher tiers, this information must be either not presented at all or presented to the minimum necessary degree (obfuscated, aggregated), depending on the type of high-level activities

or political/investment decisions that such information should support.

Provided services. Across the fog tiers, nodes engage in communication and data exchange to facilitate the provision and requests of application services. These services might encompass the dissemination of comprehensive information to underlying layers or actively participate in extensive data processing workflows.

Conventionally, the selection and sophistication of services increase towards the uppermost tiers. Within the lower tiers, a restricted array of service types is available, primarily concerning sensor data collection and actuator control.

Progressing up the hierarchical network layers, each node assumes an augmented awareness of the overarching network state. This heightened awareness arises from the node's ownership of a "wider" and larger amount of data, and increased processing capabilities. Consequently, this progression culminates in the emergence of "intelligent" services at the highest tiers.

Resource capacity. Fog tiers also exhibit distinct attributes based on the resource availability within their nodes, encompassing CPU, memory, and storage capacities. The processing potential afforded to a fog node augments as one ascends the tier hierarchy.

The service types that a given fog node can accommodate is intrinsically tied to its resource capacity. Fog nodes characterized by limited resources excel in tasks such as sensor data acquisition, data normalization, and the orchestration of sensor and actuator commands. Especially, these activities are often performed at the edge of the IoT layer, in edge nodes operating over a limited geographical area.

Moving to nodes with intermediate resources, spanning coverage across cities or provinces, their focus shifts toward functions like data filtering, compression, and transformation. Such nodes might also facilitate rudimentary edge data analytics to facilitate real-time or near-real-time processing.

In contrast, high-capacity fog nodes, responsible for broader regions, primarily serve functions involving data aggregation and the extraction of knowledge from the a very huge amount data.

Geographical coverage. Another feature that changes contingent on the tier is the geographical coverage of the fog nodes, ranging from small scale areas, to city-

wide, provincial, and regional extents. Geographical coverage denotes the area of interest that a given fog node encompasses. Notably, the territorial jurisdiction of a fog node, particularly within the lower tiers, is influenced by the network topology. This topology can render a node accessible solely from neighboring nodes situated within close geographical proximity. This phenomenon is especially notable within edged nodes, whose incoming communications leverage prevalent protocols tailored for limited short-range connections (e.g., IEEE 802.15.4 standard).

Implicitly related with this notion is the aspect of data privacy. Indeed, a fog node that covers a specific small area will exclusively store and process data attributed to that particular zone, avoiding interference with other data owners. Furthermore, the extent of geographical coverage also intersects with data granularity. In specific terms, a fog node targeting a very small area domain will predominantly manage raw data. Conversely, a fog node responsible for a broader regional span will engage with data that has undergone transformations, such as filtering and aggregation.

2.2.2 Fog Protocols

As previously outlined, fog nodes serve as pivotal components encompassing computational, networking, storage, and acceleration capabilities. These nodes bridge the gap between the cloud layer and the IoT layer [1]. Depending on the tier they belong and the services they host, fog nodes employ distinct protocols and message formats for communication. There is relevant difference between application-specific protocols and network protocols. While the subsequent discourse is by no means exhaustive, it centers on network protocols due to their standardization, as opposed to application-specific ones.

Within the cloud layer and higher fog tiers, the prevalent technology for implementing (micro)services is REST [36]. Consequently, the primary network protocol employed is HTTP, inclusive of its secure counterpart, HTTPS. Other well known protocols are suited for this kind of communications, as gRPC and WebSockets. Information exchange takes place through formats like JSON[2] (alongside related di-

[2]https://www.json.org/

alects such as GeoJSON[3]), XML, and CSV.

In the context of lower fog tiers and the IoT layer, the adoption of publish/-subscribe messaging transport protocols is common, with MQTT [37] standing as a prominent example. A typical MQTT system encompasses one or more publishers and subscribers, engaging in communication facilitated by a broker node responsible for message dispatching.

Given the resource limitations of IoT devices compared to fog nodes, the selection of network protocols must prioritize efficiency. Noteworthy options include 6LowPAN[4], LoRaWAN[5], and CoAP[6].

To convey and structure IoT metadata, several key initiatives have emerged: Sensor Markup Language (SenML) [38], IPSO Alliance Framework [39], and the oneM2M Base ontology [40]. For the delineation of IoT devices, the Web of Things initiative by the W3C has introduced the W3C Thing Description (TD)[7]. This format caters to processing by applications in non-constrained environments, often found within the cloud and fog tiers. It serves as the foundation for sophisticated discovery and search services.

Nevertheless, these protocols remain disjoint and exhibit limited functionality within the broader context of the Edge-to-Cloud continuum. This deficiency has prompted the development of a inclusive solution called SEAMDAP, using SenMl and W3C TD as message descriptor, which will be comprehensively examined in Chapter 3.

2.3 Service Placement Scenarios

As discussed in Section 2.2, the framework operates under the assumption that the designated application objectives are formulated as a collection of interconnected (micro)services. Within this context, certain services are accessible within the cloud

[3]https://tools.ietf.org/html/rfc7946

[4]https://tools.ietf.org/html/rfc4944

[5]https://lora-alliance.org/about-lorawan/

[6]https://tools.ietf.org/html/rfc7252

[7]https://www.w3.org/TR/wot-thing-description

environment, permitting horizontal scaling via the creation of stateless instances distributed across fog nodes. Conversely, some services remain confined to fog nodes exclusively due to data privacy imperatives.

In this analysis, a user is for simplicity an individual IoT node, denoted as US_a, being a sensor or an actuator. This assumption is substantiated by the conceptualization of a service as a set of activities geared towards either the acquisition and retention of domain-specific environmental data or the dynamic manipulation of its state. It is important to note that what presented in the subsequent sections can be easily extended even if a user's role is situated at a tier distinct from the IoT layer. Moreover, it is pertinent to underscore that the scope of all these consideration is restricted to stationary IoT node, thereby excluding mobile users from current competence.

2.3.1 Quantitative Model

To model the proposed architecture in a quantitative fashion, an extension of the notation proposed by Lera et al. [31] has been used.

A cloud or fog node is represented as D_i, with its defining attributes encapsulated within the vector AR_i, which encompasses the capacities of each physical component. These capacities are expressed as scalar values, quantified in general resource units. The CPU's capacity is measured in terms of the number of cores it possesses, while the storage memory capacity is denoted in terabytes (TB), and so forth. It is noteworthy that cloud nodes are presumed to possess virtually infinite resources. Alongside these attributes, nodes are also characterized by their processing speed denoted as IPT_i, measured in instructions per unit of time.

The network link between two interconnected nodes is symbolized as NL_{ij}, with bidirectional communication inherently assumed ($NL_{ij} = NL_{ji}$). These network links are characterized by attributes such as the propagation delay $PR_{NL_{ij}}$ and the network bandwidth $BW_{NL_{ij}}$. Hence, the network delay incurred during the transmission of a packet between connected nodes is mathematically expressed as follows:

$$ND_{NL_{ij}} = PR_{NL_{ij}} + \frac{\text{size}}{BW_{NL_{ij}}} \tag{2.1}$$

A user possesses the ability to invoke any instance of a service in the system, although the common practice is to generally opt for the closest one. In this conceptualization, each application APP_x is encapsulated as a directed graph, wherein the edges represent requests and the nodes represent services. Every application is delineated by a stipulated deadline DL_{APP_x}.

Each distinct service S_u is characterized by the resource consumption it induces upon the allocated node. Analogous to resource capacity, resource consumption is articulated as a vector CR_{S_u} featuring scalar values that quantify the consumption of individual physical components. Importantly, the execution prompted by a request is contingent not only on the service itself but also on the specifics of the request message. This message, designated as $MS_{S_uS_v}$, is denoted by the originating and target services.

Additionally, the size of the message is denoted as $SZ_{MS_{S_uS_v}}$, while the message's data privacy requisites are indicated by $DP_{MS_{S_uS_v}}$, taking values between 1 and N. The workload exerted upon the node corresponds to the number of instructions necessitated for processing a given request message, denoted as $EI_{MS_{S_uS_v}}$.

In order to describe the services placement across the entire system, we define a placement matrix denoted as P, featuring dimensions $|\mathscr{S}| \times |\mathscr{D}|$, signifying the product of the number of services and the number of nodes. Within this matrix, an entry p_{ui} holds a value of 1 if service S_u is deployed within node D_i, and it holds a value of 0 otherwise.

The first constraint necessitates that the sum of resources consumed by the allocated services must not exceed the available resources within the node:

$$\sum_u p_{ui} CR_{S_u} \leq AR_i \quad \forall D_i \tag{2.2}$$

The second constraint pertains to data privacy concerns. The placement of services must consider the data privacy requirements associated with request messages, delimiting the diffusion of sensible data:

$$p_{vi} = 1 \quad \text{iff} \quad FT_{D_i} \leq N - DP_{MS_{S_uS_v}} + 1 \quad \forall D_i, \forall v \tag{2.3}$$

Here, FT_{D_i} denotes the fog tier of node D_i.

Two optimization objectives as quantities to be maximized have been established. The first objective concern the deadline satisfaction ratio, defined as:

$$\text{deadline}(US_a, APP_x) = \frac{|RT_{RQ^n_{US_a APP_x}} \leq DL_{APP_x}|}{|RQ^n_{US_a APP_x}|} \tag{2.4}$$

In this equation, $|RQ^n_{US_a APP_x}|$ represents the count of times a request for APP_x is dispatched from user US_a, while $|RT_{RQ^n_{US_a APP_x}} \leq DL_{APP_x}|$ signifies the count of those requests that successfully meet the application's stipulated deadline. The maximization of this quantity across all users and requests within the system is required.

The second objective pertains to the service availability ratio, which is defined as:

$$\text{availability}(APP_x) = \frac{|US_a \text{ s.t. } \exists \text{ path } US_a \text{ to } APP_x|}{|US_a \text{ s.t. } US_a \text{ requests } APP_x|} \tag{2.5}$$

In this equation, the denomination quantifies the number of users that solicit APP_x, while the numeration counts the number of users capable of reaching APP_x. The optimization objective requires the maximization of this quantity for all applications within the system.

2.4 Service Placement Algorithm

Service placement serves as the mechanism for selecting suitable execution zones to place instances of services [41]. In this section, a *centralized* service placement algorithm, designed to function within a *static* instance of the previously defined N-tier fog architecture, is introduced. The algorithm fulfills two primary objectives: the prioritization and safeguarding of user data privacy, along with the timely availability of application outcomes. The proposed algorithm employs a greedy approach to allocate applications based on intrinsic, static attributes such as service graphs. Notably, this service placement algorithm can be employed in both *offline* and *online* scenarios. In either cases, the overall assumption is that a set of applications needs to be allocated across the available nodes.

From a range of potential optimization strategies [42], the chosen approach is the *first-fit decreasing greedy* approach, despite its not being the most efficient heuristic.

The rationale behind this choice is rooted in our aim to mainly underscore the advantages conferred by the N-tier fog architecture, especially when addressing diverse and multifaceted constraints.

An application is assumed to be composed by an assemblage of cooperating services, organized in accordance with a specific directed acyclic graph (DAG). For the sake of simplicity, it is assumed that a service cannot concurrently belong to multiple applications. [8] In cases where two services hosted by separate nodes necessitate direct communication, these nodes must possess a direct connection, implying mutual knowledge through name or IP address, enabling message exchange. Furthermore, a single node has the capacity to host services belonging to different applications.

As delineated in Section 2.2.1, the hierarchy of fog tiers and data privacy levels exhibits an inverse correlation. In other words, the sensitivity of processed data aligns with the tiers: higher sensitivity data necessitate lower-tier processing. The application modeling presented in Section 2.3 establishes that messages exchanged between services possess data privacy levels ($DP_{MS_{S_u S_v}}$). Consequently, it becomes possible to indirectly assign privacy levels to services (DP_{S_u}, DP_{S_v}). This is accomplished by inferring that if a message traverses from a sender service to a recipient service, both services must be allocated within tiers compliant with the data privacy level of the message. Therefore, if a service engages in the processing of data with a privacy level of p, it is restricted to placement within fog tiers spanning the range $\{1, .., p\}$.

In situations where a service engages in message exchange with multiple services, its assigned privacy level equates the highest level among those associated with the exchanged messages. This could potentially lead to a low-tier placement for the service, which might prove incompatible with the computational limitations of nodes at that tier. To mitigate this concern, a recommended strategy involves fragmenting service activities into several microservices that can be effortlessly placed.

The services hosted within a node will invariably consume its resources. For simplicity, we assume that introducing a new service to an already resource-exhausted host is not feasible.

[8] This assumption does not preclude the modeling of scenarios where distinct instances of the same service concurrently pertain to multiple applications.

Furthermore, each service is characterized by specific resource prerequisites for its execution. These requirements are adjusted in relation to the overall resource budget accessible within the host node.

The central thrust of the service placement algorithm is to allocate applications in close proximity to users (IoT layer), aiming to minimize latency and support local and distributed information management capabilities.

The order by which applications are allocated plays a pivotal role in shaping the final outcome. Thus, the initial step entails sorting applications in ascending order based on the minimum privacy level required by the services composing them. This leads to the establishment of a "privacy index" for each application, governed by the following rule:

$$\begin{gathered} index(APP_x) \leqslant index(APP_y) \\ \text{iff} \\ \min_{S_u \in APP_x} DP_{S_u} \leqslant \min_{S_v \in APP_y} DP_{S_v} \end{gathered} \tag{2.6}$$

In situations where two applications share the same privacy index, their subsequent ordering is determined by their respective deadlines (earliest deadline first).

This arrangement serves to grant delay-sensitive services a greater likelihood of placement within nodes that are geographically closer to the IoT layer, thereby minimizing latency. With the list of applications sorted, the algorithm for service placement initiates the allocation process by sequentially placing the services of the first application on the list. Services directly connected to users in the IoT layer receive priority placement, followed by services interacting with these, and so on.

In addition with respecting privacy, resource, and connectivity constraints, the service placement algorithm also strives to minimize the overall application latency. It does so by favoring the placement of services in nodes that are proximate to each other, considering the transmission time between two services within the same node as negligible. Consequently, this methodology ensures that each application is allocated within fog layers closest to IoT nodes, maximizing the proximity of processing tasks to end users.

In instances where certain services cannot be placed within fog layers due to an absence of suitable nodes, they are instead allocated within the cloud, provided they lack privacy restrictions. An application is deemed allocable only if all its constituent services can be placed. Applications that do not meet this criterion are discarded, and the algorithm proceeds to handle the next applications in subsequent iterations of the service placement algorithm.

2.5 Simulation Analysis

An evaluation of the proposed N-tier fog architecture and the accompanying service placement algorithm was conducted employing the YAFS simulator [31]. This simulator's capacity to accommodate dynamic network configurations offers valuable ways for exploring practical service placement scenarios within Fog Computing environments. The implemented software is available as open source. [9]

To ensure a comprehensive assessment, random yet realistic attributes were assigned to nodes, applications, services, requests, and users. These elements were parameterized using uniform distributions, with their respective minimum and maximum values detailed in Table 2.1, thereby facilitating a generalized evaluation.

The network topology was meticulously constructed in accordance with the architectural principles outlined in Section 2.2.1. More specifically, the arrangement features nodes called "gateways" at the lowest layer. These gateways symbolize collectives of sensors and actuators (i.e., users) engaged in interactions with the system.

The fog layer accommodates the allocation of applications, while the gateways generate requests for one or multiple instances of applications. The fog layer is partitioned into $N = 4$ tiers. As the tier number ascends, the quantity of nodes decreases, while the average processing capacity per node increases. Positioned beyond the fog layer is the cloud layer, represented by a solitary node endowed with unrestricted processing capacity. This cloud node maintains communication links with all nodes within the fog and IoT layers.

[9] `https://github.com/gPenzotti/FogComputingFarmPlacement`

Table 2.1: Values of the parameters used in the simulations

Parameter		**min-max**
Number of fog tiers	N	4
User		
Request rate (1/ms)		{1/1000,..,1/200}
Fog node		
Fog tier (number)	FT_{D_i}	{1,..,N}
Resources (res. units)	AR_i	$\{15+2t,..,19+2t\}, t \in \{0,..,N-1\}$
Speed (instr/ms)	IPT_i	{100,..,1000}
Application		
Deadline (ms)	DL_{APP_x}	{1000,..,150000}
Services (number)		{2,..,7}
Resources (res. units)	CR_{S_u}	{1,..,5}
Execution (instr/req)	$EI_{MS_{S_uS_v}}$	{20000,..,60000}
Message size (bytes)	$SZ_{MS_{S_uS_v}}$	{1500000,..,4500000}
Data privacy (number)	$DP_{MS_{S_uS_v}}$	{1,..,N}
Network		
Propagation time (ms)	$PR_{NL_{ij}}$	{5,..,30}
Bandwidth (bytes/ms)	$BW_{NL_{ij}}$	{50000,..,75000}

The node graph within each fog tier demonstrates small-world properties, characterized by a few highly connected hubs and a nearly uniformly distributed connections. The nature of the links varies across tiers to replicate the territoriality of nodes. Specifically, connections between nodes of the same tier exhibit lower latency than those across different tiers. Additionally, the average bandwidth increases as the tier number rises.

In this particular scenario, the network comprises a total of 176 nodes, encompassing 20 gateways and 1 cloud node. Excluding the cloud layer, the collective network resources sum up to 2912 units. A number of 20 distinct application classes has been defined, randomly associating them with gateways, yielding 54 application instances and 231 assigned services within the network. The cumulative resource demand of these services amounts to 562 units. The privacy index is randomly allocated to each application, with 20% of the services designated for handling sensitive data.

Two scenarios were considered: one with fog node failures and one without. In the first scenario, fog nodes were randomly chosen for failure (with uniform dis-

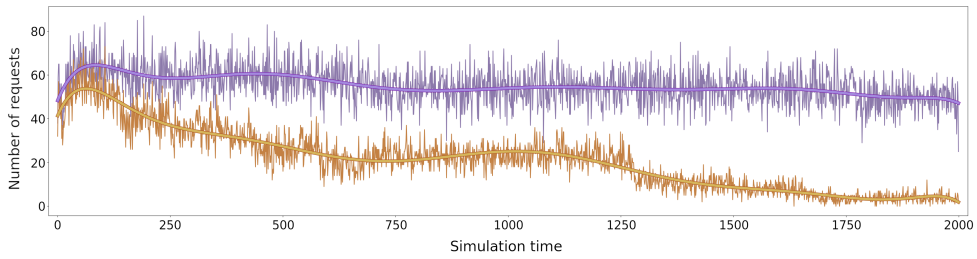

Figure 2.2: Performance of the total requests (in purple) in the system compared with the requests performed meeting the deadline during the simulation with dynamic node failure strategy (orange).

tribution) and were deactivated in a manner that resulted in 90% of the fog nodes becoming inactive. The evolution of request satisfaction is depicted in Figure 2.2. In the absence of fog node failures, all deadlines were consistently met. The purple plot indicates the total count of generated requests, aligning with the number of requests satisfied within the set deadlines. However, with the occurrence of fog node failures, the quality of service (QoS) exhibits rapid degradation. In this case, the orange plot illustrates the temporal progression of the number of requests that were successfully met within the stipulated deadlines.

In Figure 2.3, the resource distribution utilized by the placed services is represented. The resource consumption per node and the average resource consumption per fog tier corroborate that the service placement algorithm predominantly encourages allocation to the lower tiers.

2.6 Smart Farming Applications

In Smart Farming, different application scenarios can be identified in which the adoption of an advanced distributed architecture for data processing and storage would bring advantages to the individual user and the entire production.

The first application example we consider is *satellite remote sensing for decision support*. Currently, there are several services that provide data from ground multi-

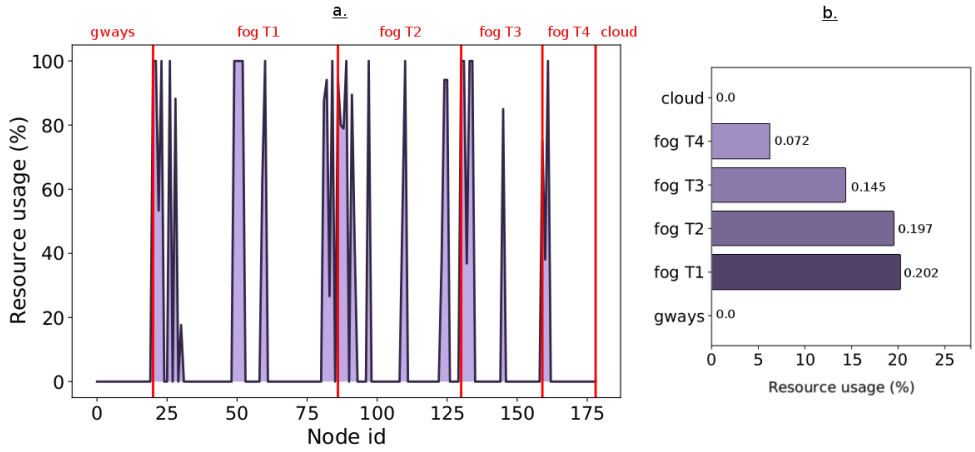

Figure 2.3: a. Resource usage for each node. Vertical red lines indicate the tier boundaries (note that services are not placed into gateways, and there is only one cloud node that is not loaded at all). b. Average resource usage per fog tier.

spectral sampling performed by satellites in continuous orbit around the planet, such as Copernicus, the Earth monitoring space program by ESA.[10] From these data, it is possible to extract vegetation indices (NDVI is one of the most popular) that allow to determine the state of the health of a specific crop. The vegetation indices can then be used by a Decision Support System (DSS) for agriculture in order to produce specialized support in some farmers' activities, such as irrigation, based on the information of each cultivated plot (crop, water balance, etc.) [3, 18].

The second application example is *weather information aggregation*. Weather monitoring is one of the most important agricultural activities, because it consists in observing fundamental parameters for the development of crops. Currently this activity is done by strategically positioning public weather stations by public bodies or small private stations installed by farms, which may or may not make such data publicly available. These data can be used to create and nurture a highly precise and specialized climate model for small areas, designed to counter adverse phenom-

[10] http://www.esa.int/Applications/Observing_the_Earth/Copernicus

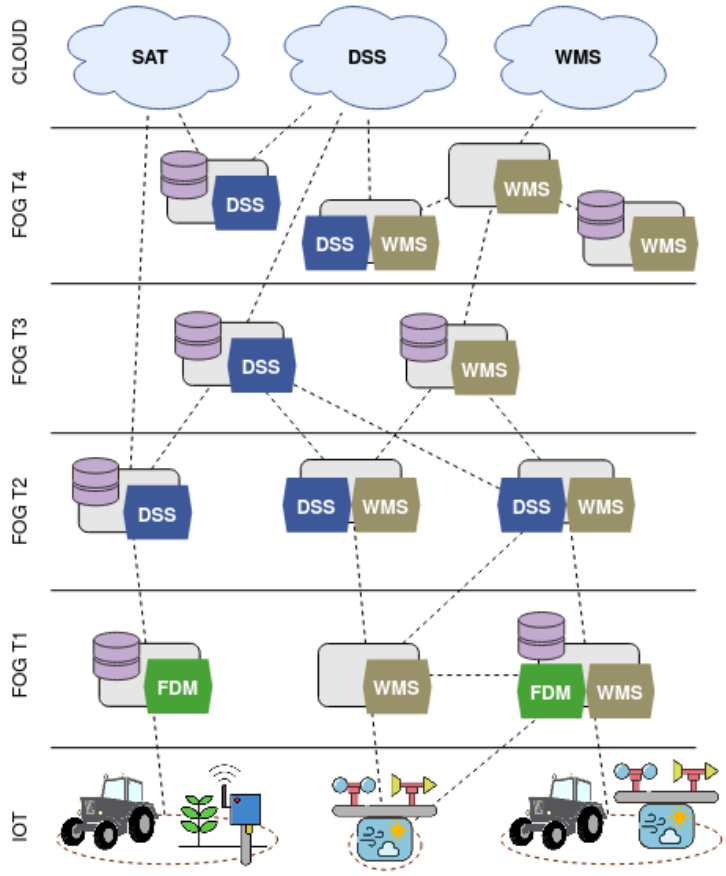

Figure 2.4: Instances of the satellite remote sensing for decision support (upper scheme) and weather information aggregation (lower scheme).

ena, such as violent rains or droughts, or harmful, such as frost and hailstorm. It is therefore possible to monitor key parameters by zones, in order to avoid the harmful effects of such events, and to intervene very quickly thanks to the locality and low latency of the nearest fog nodes. For example, if the measured values are a signal for possible frosts in a specific area, antifreeze systems connected to the network could be quickly activated. This process may involve sensitive farm data such as crops,

surfaces, sunlight exposure, etc., without exposing them to cloud services.

For these application scenarios, the following nodes and services can be identified. Examples of their composition are illustrated in Figure 2.4.

Actuators and control systems (IoT layer). Some technologically advanced farming machines (tractors, irrigation systems, etc.) are networked, thus able to receive data/commands and carry out part or all of their specific activities automatically, based on the output of the DSS and the experience of the farmer. Moreover, some advanced control systems are able to make urgent critical decisions (such as suspending an activity in case a problem is detected) based on continuous feedback.

Farm and climate sensors (IoT layer). Some sensors may provide data at the level of the single field, while other sensors (such as climatic stations) may cover larger areas and serve multiple farms.

Farm data management service (Fog layer). For one or more farms, it is possible to identify services that deal with maintaining and processing their data, such as land, crops, production cycles, processing, etc. Considering the high sensitivity of the data processed and the low geographical competence, the service will be localized to the lowest tiers of the fog network.

DSS services (Fog and Cloud layers). Modern DSSs usually leverage multiple services that belong to different stakeholders. These services implement various functionalities and communicate with each other. Some computational intensive services can be placed into higher fog tiers or in cloud. For example, let us consider services that deal with maintaining the decision-making model, or that run or train machine learning models. Other services can instead be deployed at lower fog tiers, like the services that deal with geographical information of the land (position, perimeter of the crop subdivisions, etc.) or tilling information. These data can be used by the DSS at different tiers, by properly removing sensitive information.

Climate monitoring services (Fog and Cloud layers). Raw data collection services should be placed closer to the nodes, at lower fog tiers, in order to preserve data privacy. Instead, climate models and other resource-consuming algorithms can be executed at higher fog tiers or in cloud.

Satellite data services (Cloud layer). These services are usually placed in cloud,

like the one of the Copernicus mission. Generally, satellite data are provided in an aggregate fashion, and need processing and refinement at the level of single farm, using lower fog tiers.

Chapter 3

Scalable Protocol for Sensor Data Acquisition

This chapter is derived from the research paper titled "Seamless Sensor Data Acquisition for the Edge-to-Cloud Continuum" [43] authored by Penzotti G., Tarasconi D., Caselli S. and Amoretti M.. The paper introduces an innovative application protocol known as SEAMDAP (SEAMless Data Acquisition Protocol), specifically designed for the seamless acquisition of sensor data across the Edge-to-Cloud continuum. SEAMDAP offers a range of benefits. Firstly, it is built upon existing and consolidate standards such as JSON, W3C Thing Description, and SenML, simplifying its implementation and deployment through the use of readily available software libraries. Secondly, it streamlines human intervention by primarily focusing on specifying the data collected by sensor nodes and defining how and where this data should be transmitted. Lastly, SEAMDAP enables automatic data aggregation and specialization throughout the entire Edge-to-Cloud continuum, rendering it a unique and versatile protocol.

Concerning data collection for smart farming and similar domains, SEAMDAP is designed as a solution enabling collaborative and non-collaborative data sharing, thereby outlining a vision that responds to RQ3, through standard multilevel interfaces, and to RQ4, proposing customizable and data oriented operational practices.

The sections of this chapter are structured as follows. Section 3.1 offers a brief overview of the problem and related work data acquisition along the E2C, Section 3.3 provides an overall description of main protocol characteristics of the proposed protocol, Section 3.4 delves into the implementation details, while Section 3.5 describes and comments results and insights from a simulation analysis.

3.1 Data Diffusion in the Edge-to-Cloud Continuum

The distributed and heterogeneous nature of smart environments has created significant challenges, including effectively managing data dissemination across the Edge-to-Cloud (E2C) continuum. In the context of the E2C continuum, data is generated and requested at different levels and with different frequency. This diversity is one of the key challenges in effectively processing and disseminating data in scenarios such as Smart Farming and Smart Cities. Each tier contributes data that varies in its nature, origin, and purpose, leading to its own set of challenges. At the *Edge*, the data generated often represent "raw" information coming directly from the sensors. These data are often unprocessed and contain detailed measurements, such as temperature, humidity, pressure readings, and more. The raw nature of these data makes them valuable for in-depth analysis, but also more challenging to process and transmit due to their size and to the presence of constrained devices. In *Fog* layers, the data can undergo preliminary aggregations and synthesis. This process is particularly useful for reducing the amount of data to be transmitted to the Cloud and improving overall system efficiency. Processing may involve statistical aggregation, filtering of insignificant data, or the creation of key performance indicators. These operations can help in reduce the load on the network and improve response times. In the *Cloud*, data can undergo further levels of analysis and synthesis to obtain high-level indicators. Here, the aggregated data can be subjected to machine learning models or advanced analytic techniques to extract meaningful insights. For example, in Smart Farming contexts, the analysis can lead to forecasting weather conditions, crop management or detecting anomalies. These indicators provide greater context and higher business value than raw data or lower-level aggregations.

This diversity of layers brings a number of challenges, including the need to ensure smooth and reliable communication, despite the differences in computing power and network capabilities of the various nodes. Furthermore, the absence of an established standard for data dissemination creates additional complexity in orchestrating data processing and transmission.

There is a lack of effective application-layer protocols that provide comprehensive control over sensor data flows within distributed systems (see Section 3.1.1) . Effective orchestration of data require a well-defined strategy that takes into account the complex and heterogeneous spectrum of data and applications.

3.1.1 Related Work

While the advantages of edge computing have been explored in various works (e.g., [44, 45]), limited attention has been given to its integration with IoT systems. Notably, there has been a lack of focus on the registration of unknown data sources, which is a recurring challenge.

Del Gaudio and Hirmer [46] introduced a life cycle method for device management in Industry 4.0 settings. This approach facilitates the integration of newly emerging devices into Smart Factory applications and offers solutions for dealing with failing devices. It involves the use of two non-standard metamodels: one to describe device communication in smart factory environments (the processing model) and another to specify the structure of smart factory environments (the environment model). Instances of these models are represented as JSON documents. While this method is effective in controlled environments, it faces scalability issues when applied to heterogeneous, open settings.

The work that is most comparable with SEAMDAP is that of Zanzi et al. [8], who proposed a multi-access edge computing (MEC) paradigm enabling the seamless integration of existing and future IoT platforms. Their architecture extends the multi-vendor edge environment defined by the MEC Industry Specification Group under the European Telecommunications Standards Institute (ETSI). While this solution is integrated with an industry-driven MEC architecture, it does require non-MEC adopters to add a middleware layer for bridging IoT systems with edge computing systems.

In contrast, SEAMDAP introduces a lightweight protocol that can be adopted with minimal impact on existing systems.

3.2 Preliminaries

The SEAMDAP protocol represents an effective solution towards order and efficiency in the dissemination of data within the Edge-to-Cloud continuum.

SEAMDAP is an *application level*, *client-server oriented* protocol, and can be also termed as an *"operational protocol"*. Instead of primarily focusing on defining precise interactions and establishing fixed sequences of messages, its main aim is to standardize the operational phases that must be executed between the nodes. Subsequently, within each of these phases, it delves into specifying the formats, operational methods, and use cases in detail. The overarching objective is not only to standardize communication between the involved parties but, more importantly, to foster a scalable approach to distribute data in the design of the system.

Unlike rigidly standardized approaches, SEAMDAP embraces a *semi-open philosophy*, which allows for the flexibility to adapt to the specific needs of different application environments. This flexible approach is crucial in scenarios such as Smart Farming and Smart Cities where requirements can vary greatly. In fact, it works well in environments with the presence of non-homogeneous sensors, based however on a common modeling of the entities that participate in the protocol interations.

Moreover, the chosen design allows the protocol to be configured to meet the specific needs of different levels of the continuum and various types of data. In fact, SEAMDAP has been designed to accommodate various requirements, among the most relevant:

- *transmission efficiency and performance*, to improve the speed of data collection and distribution, enabling faster responses to data requests and processing;

- *customization*, in data description, phase implementation and so on, in order to accommodate specific use case requirements;

- *ease of use and readability*, to simplify phases that require a minimum of configuration by the user (whether owner or manufacturer of the sensor) and so improve *community participation*, in order to obtain large-scale adoption and therefore wide availability of data.

3.2.1 Definitions

Definitions of common terms used for SEAMDAP description are provided below, beginning with the physical entities within the considered IoT and Edge-to-Cloud context.

The SEAMDAP protocol, as previously mentioned, serves as an operational protocol tailored for the extensive collection of sensors data. It is designed to work effectively within environments featuring a variety of non-uniform sensors, based in a shared model of the physical and abstract entities involved in the protocol's interactions.

Definition 3.2.1 (Sensor) *A sensor is a device designed to detect events or changes in its environment. The data it retrieves might be raw, often requiring minimal processing before diffusion.*

Definition 3.2.2 (Sensor Node) *A sensor node is a device housing one or more sensors, presenting a unified external interface. It accumulates sensor data, processes them if necessary, and transmits them to a gateway or an E2C node (defined below). Positioned typically at the edge of the Edge-to-Cloud continuum, it exhibits limited storage and processing capabilities.*

Definition 3.2.3 (E2C Node (Edge-to-Cloud Node)) *Within the Edge-to-Cloud continuum, an E2C node is a modular, scalable unit tasked with storing and processing data collected directly from nearby nodes. It exploits the collected data to provide services to end users or other E2C nodes. This node can belong independently to the edge, fog or cloud layer.*

Definition 3.2.4 (Gateway) *In the Edge-to-Cloud continuum, a gateway serves as a nearby element capable of collecting sensors data and forwarding them to E2C nodes, potentially applying filters.*

SEAMDAP outlines the interactions between the subsequent abstract entities.

Definition 3.2.5 (Client) *A client pertains to an end node responsible for uploading a dataset into another node called server, utilizing SEAMDAP's mechanism. The manner by which this client gathers data is irrelevant. Physically, a client can be any node, except for sensors due to performance considerations. As an example, a gateway that collects sensor data using a standard or proprietary low-power wide-area network (LPWAN) protocol, and subsequently uploads this data to a server via SEAMDAP, is a client.*

Definition 3.2.6 (Server) *A server denotes an end node that receives data from one or more clients using SEAMDAP. This type of node possesses the capability to systematically store data and often offer services. It can also transmit data to other servers, functioning as a client in relation to them.*

Definition 3.2.7 (TD Repository (Thing Description Repository)) *A TD repository denotes a node tasked with retaining static documents adhering to the W3C Thing Description protocol (as detailed in Section 3.2.2). The method through which this node acquires these descriptors is not relevant.*

An illustration of a SEAMDAP-based system is presented in Figure 3.1.

3.2.2 Adopted Standards

This section introduces the embraced standards and formats, alongside the guiding principles considered during the design of SEAMDAP. Notably, the discussion exclusively revolves around application-level standards, without involving specific protocols and mechanisms concerning the lower layers of the Internet stack.

As previously mentioned, SEAMDAP's design focused on enabling seamless integration of sensor data into the Edge-to-Cloud continuum, emphasizing high data

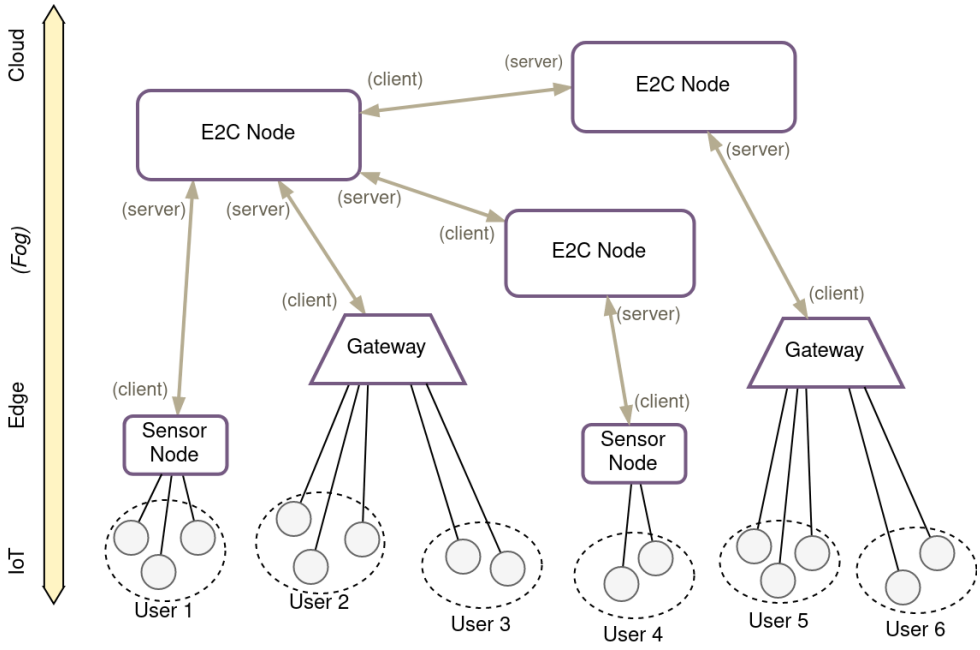

Figure 3.1: SEAMDAP-based system with Sensor Nodes, Gateways, and E2C Nodes playing the role of clients and/or servers.

control, distributed storage, and edge processing. An overarching requirement has been the creation of a protocol that is both highly scalable and straightforward, facilitating manufacturers in integrating their specifications for describing sensed data, specifying transmission details, conveying manufacturer information, and enabling the aggregation of sensor data from diverse devices.

Moreover, the potential use cases for deploying large-scale IoT systems include a wide spectrum of data types and acquisition methods. This necessitates that the proposed protocol ensures both scalability and adaptability to diverse environments, accommodating different degrees of customization.

As a result, the selected formats must exhibit extensibility and the capability to adapt to numerous contexts, even though this adaptability might render them less suitable for extremely resource-constrained devices such as LPWAN nodes. Such nodes typically handle messages with reduced memory footprints and minimal payloads,

often relying on binary protocols.

Consequently, *JSON* [47] has been chosen as the fundamental format for the protocol's messages. JSON stands out as one of the most extensively employed open standards for message formatting, enjoying implementation across major programming languages. JSON aligns seamlessly with the protocol's requisites, offering benefits like compactness, ease of parsing, human readability, and self-descriptive nature.

W3C Thing Description (TD) [48] stands as a W3C Recommendation for the abstract depiction of entities within IoT systems, encompassing sensors, actuators, sensor nodes, and smart objects in general. A TD document delineates the attributes of a given entity and outlines the actions it could potentially perform. This document can either be stored within the entity itself or within an external storage device, a particularly useful approach for resource-constrained environments. A TD document has the flexibility to describe a single entity or a collection of entities and can be formulated using various formats or markup languages (e.g., JSON).

Moreover, a TD document is constructed with four distinct *vocabularies*, which are expounded upon below:

- Data Schema: This vocabulary encompasses a metadata set designed to elucidate the attributes of entities.
- Core: The core vocabulary characterizes interactions with entities. These interactions are devoid of protocol constraints and can manifest as: *Properties*, signifying the entity's status and enabling optional status updates; *Actions*, detailing the tasks the entity can undertake; or *Events*, elucidating asynchronous interactions.
- WoT Security: This vocabulary addresses security identification and configuration, as well as data exchange mechanisms.
- Hypermedia Control: The hypermedia control vocabulary exposes web links for interacting with entities or accessing external documentation.

Leveraging these vocabularies, it becomes feasible to detail the types of data an entity deals with, as well as its pertinent information such as manufacturer name,

model name, and a short description. TD metadata further facilitates the delineation of an entity's internal status, its exposed functions (typically applicable to actuators, but also relevant for sensors with remote configuration capabilities), and the events that might be triggered through sporadic asynchronous communications.

By employing JSON-LD [49], it is feasible to devise and implement new vocabularies tailored to specific domains. JSON-LD serves as a JSON extension designed to facilitate Linked Data. This extension allows the establishment of links to vocabularies available on the web, effectively crafting standard domain-specific vocabularies. This aids in the creation of a consistent means for exchanging readable data across various applications that comprehend the content. When coupled with TD, JSON-LD enables the generation and deployment of domain-specific vocabularies intended to describe distinct types of entities.

SenML [38] is a lightweight standard designed to represent sets of measurements, resulting from one or more sensors, in a self-descriptive manner. It can be implemented in various formats depending on the most suitable representation for the employed communication protocol (e.g., JSON, XML, EXI, or CBOR). SenML is structured to enable systems with limited processing capabilities to encode and transmit data via straightforward communication protocols such as HTTP or CoAP. This standard permits servers to manage vast volumes of data effectively.

Within SEAMDAP, TD is employed to define interfaces of sensor nodes, offering general insights into their models, manufacturers, sensed data, and whether external server connectivity is required. Once the sensor nodes are characterized, SenML facilitates the acquisition and formatting of data significance. This method enables the creation of collections containing TD-described and periodically sensed data. Additionally, the protocol allows the consolidation of multiple SenML documents while retaining the identification of the specific sensor associated with a given data collection.

The proposed protocol's adaptability extends to accommodating various formats to work in different contexts. For instance, in a Smart Farming application, while TD may be used for describing sensor nodes, it might fall short. When a new sensor node is deployed, it's essential to document its position, coverage area, and ownership.

This is where the *GeoJSON* format help satisfy this requirement. A GeoJSON object is constructed using a specialized vocabulary that enables the delineation of shapes and positions for one or more geographical areas, thus providing a means of geo-locating them. This is accomplished using longitude and latitude coordinates of the vertices constituting the area (with polygons being the sole allowable shape). GeoJSON goes beyond geo-localization, allowing supplementary information about the covered area to be provided. Consequently, when registering a new sensor node with a hub, this registration message could encompass both the sensor node's TD and its corresponding geographical coverage, all encapsulated within a GeoJSON object.

3.3 Protocol Description

In this section, we delve into the details of SEAMDAP, specifically concentrating on the interactions occurring between clients responsible for providing sensor data (e.g., sensor nodes, gateways, E2C nodes) and servers (i.e., E2C nodes). Notably, the protocol does not encompass the interactions between sensor nodes and gateways, as there are already established and effective standards to address that aspect. SEAMDAP is structured around three distinct phases, which are enumerated below and subsequently elaborated upon in the following subsections:

1. Data Interface Registration (Section 3.3.1): During this phase, clients, including sensor nodes, gateways, and E2C nodes, initiate their interaction by registering their data interface. This process enables them to declare different information about data they can provide and the corresponding metadata.

2. Single-Instance Registration (Section 3.3.2):In this phase, the clients proceed to register themselves as instances of their respective data interface. This involves providing information such as location, descriptive information, and data acquisition process details.

3. Data Acquisition (Section 3.3.3): The final phase revolves around the actual acquisition of data. Clients share sensor data with servers according to the capabilities and specifications outlined during the previous phases.

For each data interface registration, multiple single-instance registrations can be made. For each single-instance registration, potentially infinite data acquisition steps can be performed.

3.3.1 Phase 1: Data Interface Registration

During the first phase of the SEAMDAP protocol, which is illustrated in Figure 3.2, the client initiates the interaction by sharing its data interface structure with the server. The data interface is essentially a description of the types of data that the client can transmit, and it is represented using a TD document. This document specifies the different properties of data that will be transmitted and associated metadata, including names, units of measurement, and validity ranges. The TD document effectively guides the subsequent data transmission process.

SEAMDAP offers two mechanisms for sharing the data interface with the server:

- *Direct Sharing*: the client sends its data interface directly to the server as part of the communication. The server receives and processes the data interface to understand the structure and capabilities of the client's data.

- *Retrieval via URL or Semantic Identifier*: Alternatively, the client can provide the server with a URL or a semantic identifier [50] that points to a public or private location where the data interface is stored. The server then retrieves the data interface from this location, allowing the decoupling between the client and the server.

Once the server obtains the data interface, it can read, validate, and interpret the incoming data correctly.

When a data interface is distributed, it essentially defines the schema for the data that will be transmitted by a client. This means that any node that comes into possession of the transmitted data can use the associated data interface to understand the meaning of the data. In this way, data interfaces play a role in identifying how and where the data should be interpreted and processed. This approach should not be

understood as a data integrity mechanism, but only as an extra layer of *data distribution* control (o privacy control), letting authorized entities have access to the full data interpretation, and make it difficult for the other nodes.

After the successful registration of the data interface, the server assigns a unique ID to it. This ID is communicated back to the client as a confirmation of the registration process. The server can decide whether to make this ID public or restrict access based on its policies. The data interface ID serves as a reference for subsequent interactions and data transmission.

This phase is crucial for enabling the server to interpret and handle incoming data accurately and efficiently.

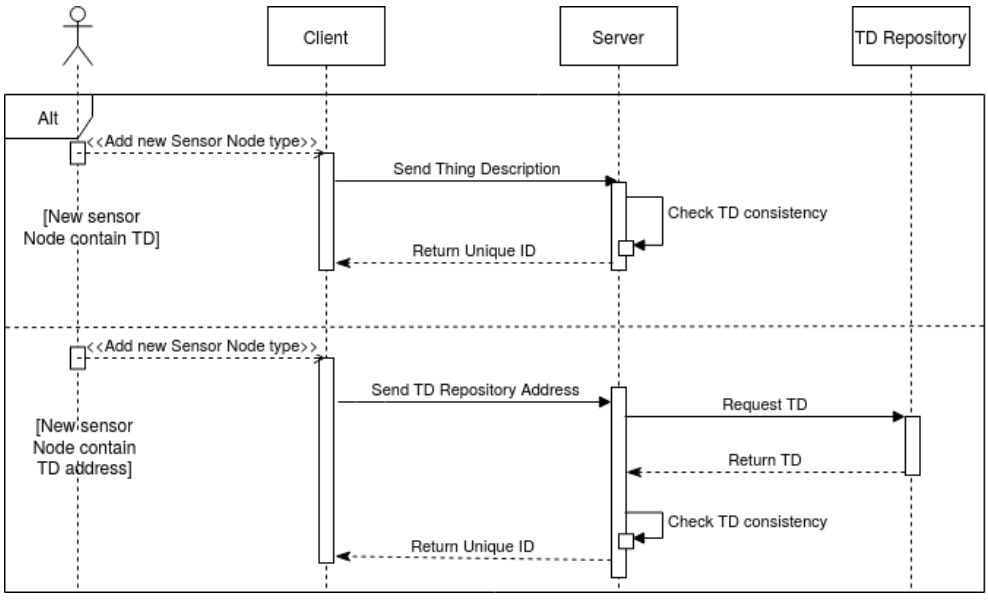

Figure 3.2: Execution of the SEAMDAP data interface registration phase. Two alternative sequences are shown; in the second one, a TD repository takes care of storing and making available the TD documents.

3.3.2 Phase 2: Single-Instance Registration

In the second phase of SEAMDAP (Fig. 3.3) , the focus is on the interaction between a sensor data provider (client) and the server. This phase involves the client notifying the server of its presence and providing meta-information about the samples it will provide.

The purpose of the *single-instance registration message* is to convey important contextual and operational information about the client and the data it will transmit. This information helps the server to better understand the nature of the incoming data, its source, and how it should be processed and interpreted. Some of the meta-information that can be included in the single-instance registration message includes:

- Geographical Position: The geographical location of the sensor node or client.
- Construction and Technological Details: Details about the sensor types, technologies used, and any specific characteristics of the client.
- Time Interval: The time interval between two data uploads, indicating the frequency of data transmission.
- Preprocessing/Aggregation Activities: Any preprocessing or aggregation activities performed on the data before transmission.

The single-instance registration message is customized for the specific use case and does not have a strict predefined format in the protocol. This flexibility allows the message to capture relevant information that is specific to the client and the application. An example is illustrated in Section 3.4.

During this phase, the client is assigned a unique ID by the server, which the client will use to identify itself in subsequent interactions. Additionally, the client must indicate the data interface it registered earlier using the interface ID assigned by the server. This linking of the data interface and the client instance is crucial for the server to correctly interpret the incoming data.

Furthermore, the method of data acquisition is established during this phase. The server and the client agree on how the data will be exchanged. There are two main approaches:

- Server Listening to Submissions: The client directly submits data to the server, which listens for incoming data submissions from the client.
- Server Requesting Data: The server requests data from the location indicated by the client when needed. The server can also specify the collection period for which it needs data.

Due the customizability of this phase, other exchange approaches are possible.

By establishing the method of data acquisition and exchanging relevant meta-information, the client and the server prepare for the actual data transmission phase, which is the next step in the SEAMDAP protocol.

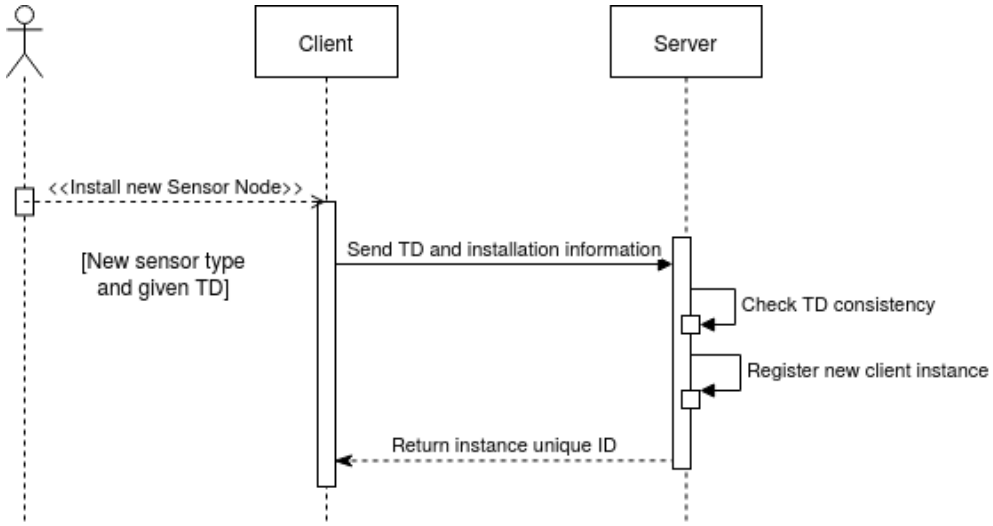

Figure 3.3: Interactions between client and server during the single-instance registration phase. The message sent by the client can be customized according to domain-specific needs.

3.3.3 Phase 3: Data acquisition

In the third and final phase of SEAMDAP (Fig. 3.4), the focus is on the transmission of sensor data from a client to a server. During this phase, the client transmits the

actual sensor data to the server. The data transmission is based on the information established in the previous two phases.

This phase is distinct from the previous two phases. Indeed, the preceding phases are characterized by single interactions. Specifically, the registration of a data interface is a one-time occurrence for each distinct interface, and the client's registration transpires as a solitary event per instance. In contrast, the third phase is the iterative interaction of the process, reoccurring over time, and thereby constituting the primary network traffic load.

In this phase, it is very common (but not necessary) that a direct data exchange takes place directly between a sensor node and a server. However, the sensor node may have stringent constraints spanning hardware resources, software capabilities, and power availability. These restrictions have directed our preference toward adopting the SenML data format [38]. SenML's salient attributes of compactness, alongside its versatile capacity for conveying comprehensive sensor data, facilitates an effective and efficient information interchange.

It is important to note that while SEAMDAP does not specifically address security aspects, it does not restrict the implementation of security mechanisms. The protocol focuses on the seamless integration of sensor data into the Edge-to-Cloud continuum and provides a framework for data exchange. Security measures can be implemented using established mechanisms for Edge-to-Cloud communication and processing. However, IoT security is a much more open research field [51]. Notable solutions have been proposed, such as IoTChain [52] and ELIB [53].

Overall, the third phase of SEAMDAP completes the data acquisition process, enabling the continuous transmission of sensor data from the client to the server, supporting a variety of use cases and application scenarios in the IoT and Edge-to-Cloud domain.

3.3.4 Analysis

The proposed protocol exhibits a high degree of efficiency with regard to communication complexity.

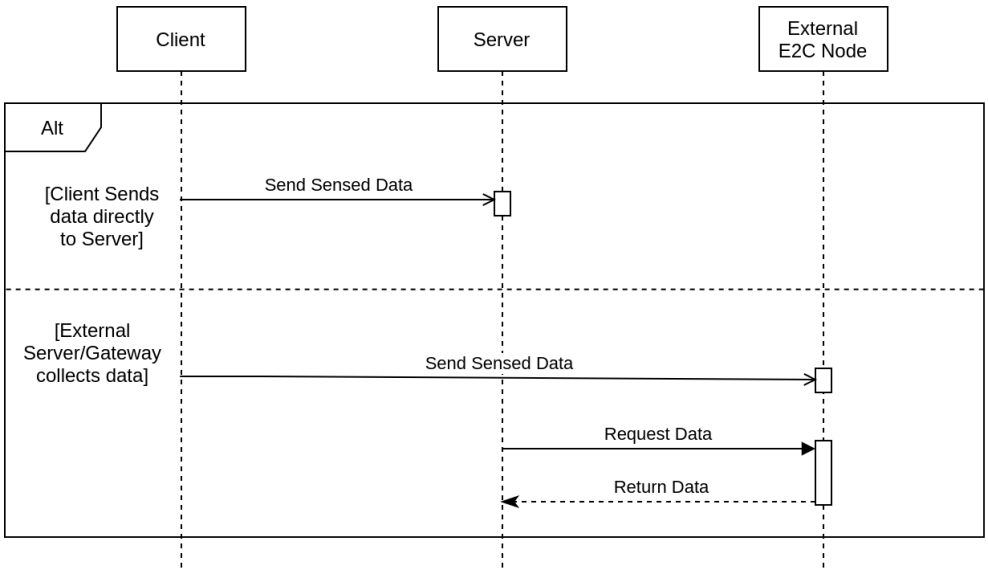

Figure 3.4: Interactions between client and server during data acquisition. This phase may be repeated sporadically or periodically, until the sensing activity ends. In most cases, the client submits sensor data to the server, but it is also possible that the sensor data are acquired by the server from an external E2C node playing the role of broker.

Claim: In scenarios involving the deployment of n sensor nodes, where there are $m \ll n$ distinct node types (distinguished by the data types they gather), the volume of automatic messages exchanged between the client and the server adheres to the complexity of O(n).

Proof. To initiate m data interface registrations, the client is required to create and forward m Thing Description (TD) documents to the server. Depending on the client's nature, these documents might either be pre-established by the sensor node manufacturer or specified by the owner of the gateway/E2C node. As illustrated in Figure 3.2, the amount of automatic messages shared between the client and the server is bounded by O(m). Following the successful registration of all data interfaces, the sensor nodes can be deployed. For every activated sensor node, the client transmits a single-data registration message to the server, subsequently receiving a response that

encompasses the instance's unique ID (as depicted in Figure 3.3). Thus, during this phase, the count of automatic messages exchanged between the client and the server is O(n). Given that $m \ll n$, it can be concluded that the overall volume of automatic messages between the client and the server aligns with O(n). □

3.4 SEAMDAP-based System Deployment

In this section, we delve into the *POSITIVE information system*, an operational framework introduced by the POSITIVE project [2], which is geared towards providing scalable sensor/satellite-driven services in the domain of Smart Farming. We specifically spotlight the incorporation of the SEAMDAP protocol within the POSITIVE information system for the collection of sensor data.

The overarching aim of the POSITIVE initiative is to establish operational protocols that render agronomically relevant information accessible to a diverse array of users, facilitating the execution of Smart Farming endeavors such as precision irrigation and crop monitoring. This objective is achieved by orchestrating a multifaceted data flow among disparate entities situated within the Edge-to-Cloud continuum.

Outlined below are the pivotal entities constituting the POSITIVE information system:

- **Sensors, Sensor Nodes, and Gateways**. Positioned at the edge of the POSITIVE information system, a variety of sensor types play a role in the system's operation. These encompass ground sensors, responsible for monitoring attributes like soil water content and chemical composition; atmospheric sensors, tasked with capturing parameters such as temperature, humidity, and solar exposure; and in-vivo sensors, which directly interact with crops by measuring variables like ion flow in sap to assess water requirements [54]. The data produced by these sensors must be conveyed to the information system. This is achieved through two primary methods: either by directly connecting sensors to sensor nodes with adequate processing capacity and power supply, enabling independent utilization of SEAMDAP, or via gateways communicating with sensor nodes using specific protocols.

- **POSITIVE Server**. The POSITIVE Server forms the backbone of the POSITIVE information system, encompassing a suite of core services. Its primary responsibilities include processing diverse data sources, managing user requests, orchestrating precision irrigation activities, and controlling remote agricultural machinery. These services can be consolidated within a singular server or distributed across a group of servers, positioned at the edge (e.g., on-farm premises), within a fog layer, or even in the cloud. Services delivered by the POSITIVE Server draw on geolocalized sensor data, such as local crop monitoring information, or harness aggregated sensor data stemming from dispersed spatial sources.

- **IRRIFRAME**: IRRIFRAME [55] is an advanced cloud-based decision support system tailored for Smart Farming. Its array of services encompasses precision irrigation guidance, derived from an amalgamation of data including satellite imagery and sensor data. The role of the POSITIVE Server revolves around supplying these data, complete with requisite preprocessing, and translating IRRIFRAME outputs into actionable insights for farmers.

Satellite data, though not gathered through SEAMDAP, are a pivotal component of the system, typically accessed via specialized standard protocols.

A visual representation of the POSITIVE Information System architecture is shown in Figure 3.5. Some entities have been omitted from the depiction for conciseness, as they are not directly pertinent to the current exposition.

Below, two illustrative scenarios involving the POSITIVE information system are considered, each showcasing how the adoption of SEAMDAP delivers distinct benefits to users. For each one, the interactions within each case and relative message examples are reported.

Case 1: Uploading Sensor Data. In this scenario, we delve into a group of farmers who possess a diverse range of sensors and seek to upload their sensor data to the POSITIVE Server. Firstly, each user determines the approach for transmitting sampled data from their sensors. Subsequently, a Thing Description (TD) document is devised for each configuration, encapsulating the specifications of the data be-

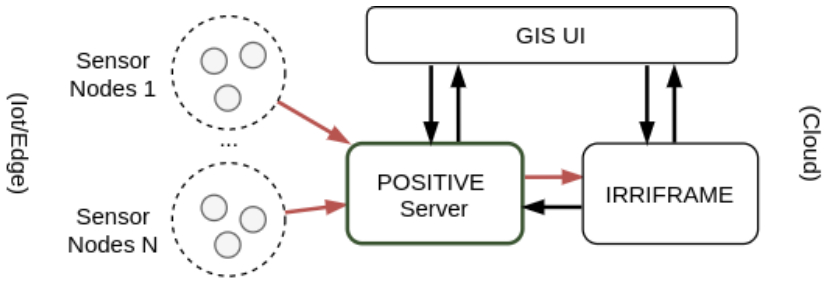

Figure 3.5: Simplified architecture of the POSITIVE Information Systems. SEAMDAP-based interactions (highlighted in red) are for sending raw sensor data to the POSITIVE Server, and for uploading processed sensor data to the IRRIFRAME service. GIS user interfaces provide farm-related visual information, including sensor node deployments.

ing collected and the intended communication methodology. The POSITIVE Server, aware of the users' data transmission configurations through the previously shared TD documents and after the single-instance registration message, initiates dynamic data acquisition based on the provided context

Two considerations must be done. Within the agricultural domain, accessing sensor data isn't always straightforward or guaranteed. Indeed, manufacturers often impose limitations on direct data access, offering proprietary channels like cloud services for data retrieval. In cases where manufacturers do not provide alternatives, users are required to develop custom applications that fetch data in a timely manner and consolidate them according to prescribed formats.

Moreover, different users opt for varying data transfer approaches. Some may transmit intricate sets of raw data, such as information pertaining to soil and atmospheric conditions. These datasets necessitate subsequent verification and processing within the POSITIVE Server. On the other hand, certain users solely convey the outcome of processing tasks performed within their own Sensor Node, streamlining the server-side processing burden.

As previously said, these TD documents are then shared with the POSITIVE Server (an illustrative example is presented in Listing 3.1). For the purpose, several

approaches can be employed to transmit the TD. Typically, TD transmission occurs through a dedicated private TD Repository, responsible for storing the document and providing it to the POSITIVE Server upon request. Alternatively, the client delivering sensor data may also provide the accompanying TD document.

The POSITIVE Server mandates that each client undertakes a single instance registration for every sensor node. Additionally, the POSITIVE Server manages spatial data, encompassing terrain geometries. Consequently, the spatial validity of sensed data and the precise location are linked to each sensor node. Notably, these data pertain to sensor nodes installed in specific locations, rather than individual sensors.

```
{
    "id": "SENSOR_NODE_ID",
    "title": "sens_ht_v1.5",
    "description":"Humidity and temperature data interface",
    "model": "MODEL_ID",
    "manufacturer": "Names",
     "properties": {
       "humidity": {
         "type": "number",
         "description": {
            "name":"hum"
            "u":"kg/m3"
            },
         "minimum": 0.0,
         "maximum": 100.0
       },
       "temperature": {
         "type": "number",
         "description": {
            "name":"temp",
            "u":"Cel"
            },
         "minimum": -30.0,
         "maximum": 60.0
       }
     }
  }
```

Listing 3.1: Simplified example of TD message for data interface recording. Two types of data are defined, with relative measurement units and validity range. Some domain-specific words have been used to describe all the characteristics of interest to the prototype.

For the depicted scenario, the single instance registration employs a customized format comprising a simple list of (key, value) pairs. This list encompasses the unique ID of the data interface coupled with its corresponding TD document. Alternatively,

the repository address from which the TD document can be retrieved can be indicated. Upon successful retrieval, the POSITIVE Server communicates the registered TD document's ID. Furthermore, a range of data pertinent to the registering user must be specified for authentication purposes. This includes the identification of the relevant cultivated plot (if applicable), geographic coordinates of the sensor node's location, and the validity area for the forthcoming sensor data provision.

Lastly, the data acquisition mode can also be specified, enabling the POSITIVE Server to request sensor data from an E2C node. This mode additionally includes parameters like the minimum time interval between two requests. In response to this registration, the POSITIVE Server returns a unique ID to the client for self-identification. Moreover, pertinent information such as the communication endpoint is provided. An exemplary representation of this process is depicted in Listing 3.2.

```
0    {
1    "TD" : "TD_UUID",
2    "UserID" : 12345,
3    "PlotID" : 56789,
4    "Name" : "name",
5    "Position" : [44.7658599, 10.3101259]
6    "Area" : <GeoJSON_Object>,
7    "Server" : {
8        "Url" : "<server_address>",
9        "Period" : 3600
10       }
11   }
```

Listing 3.2: Example of message used for single instance registration of a sensor node to the POSITIVE Server.

```
0  {"senml":[
1    {"bn":"sens_ht_v1.5", "bt":1.613561286e+09, "v":120, "u":"hum"},
2    {"v":27, "u":"temp"},
3    {"bt":1.613647686e+09, "v":35, "u":"hum"},
4    {"v":100000, "u":"temp"},
5    {"bt":1.613734086e+09, "v":64, "u":"hum"},
6    {"v":21, "u":"temp"},
7    {"bt":1.613820486e+09, "v":87, "u":"hum"},
8    {"v":22, "u":"temp"}
9    ]
10 }
```

Listing 3.3: SenML message in contract form, comprising a series of records sent by the client. The message structure complies with the TD message, indicating the type of data and a sampling time for each record.

Upon the successful registration of instances and TD documents, the POSITIVE Server is endowed with the capability to interpret the received sensor data. Each transmission can encapsulate multiple samples, each aligning with the stipulated interface structure. An illustrative instance is presented in Listing 3.3. Notably, the measurement units, as predetermined by the TD, are already established. However, for situations demanding variability, units can be explicitly specified within each SenML record.

Case 2: Uploading More Complex Information Shifting focus to the second case, it pertains to the transfer of "Soil Water Content" information from the POSITIVE Server to IRRIFRAME. For managing various Smart Farming activities, IRRIFRAME constructs and updates diverse soil and crop models. This includes utilizing insights garnered from sensor measurements, provided either directly by sensor nodes or processed by the POSITIVE Server from a multitude of sources.

In these interactions, SEAMDAP takes center stage. Here, IRRIFRAME assumes the server role, while multiple instances of the POSITIVE Server function as clients. Consequently, the number of TD documents remains limited and pre-registered, obviating the need for a single instance registration phase. Since specialized data on individual instances is superfluous, this phase is omitted. As customary, data acquisition adheres to the SenML standard, integrating tokens and IDs to accurately attribute sensor data to respective users. This process is exemplified in Listing 3.4.

```
0 {"senml":[
1   {"bn":"swc_field_id_10", "t":1.613561286e+09, "v":30, "u":"swc"},
2   {"t":1.613647686e+09, "v":35, "u":"swc"},
3   {"t":1.613734086e+09, "v":33, "u":"swc"}
4   ]
5 }
```

Listing 3.4: SenML message to be used for uploading processed data to IRRIFRAME. The Soil Water Content (SWC) value is computed by the POSITIVE Server.

In summary, SEAMDAP plays a pivotal role within the POSITIVE information system by facilitating the collection of sensor data from a diverse array of stakeholders. This process is bifurcated into two distinctive scenarios, each unfolding at distinct layers of the Edge-to-Cloud continuum.

In the first scenario, clients occupy positions within the Edge layer. These clients sometimes establish direct connections with the sensor nodes situated in the IoT layer. In contrast, servers are represented by E2C nodes located at intermediary to higher layers, characterized by medium to high processing capabilities.

Conversely, the second scenario is characterized by E2C nodes playing the role of clients. In this capacity, they transmit sensor data to a Cloud node endowed with augmented processing and storage capabilities.

3.5 Experimental Evaluation

As detailed in Section 3.4, the practical implementation and evaluation of SEAMDAP are performed within a comprehensive Smart Agriculture system. To ensure thorough and controlled assessments of SEAMDAP's functionalities, we employed a publicly available demo, whose code is accessible at `https://github.com/SEAMDAP/Demo`. This approach allowed us to meticulously analyze SEAMDAP's operational aspects, taking advantage of complete control over both client and server entities. Consequently, we could precisely configure parameters such as workload size, random delays, and error simulations.

The demo comprises a dynamic assembly of SEAMDAP clients, each representing a distinct sensor node instance to be registered. Additionally, a SEAMDAP HTTP-based RESTful server is part of the setup. Our evaluation was executed on a machine equipped with two Intel Xeon Silver 4210 CPU boasting a clock frequency of 2.20GHz and 10 physical cores. This machine also featured a maximum available RAM of 64 GB. The performance assessment was executed using the widely-utilized network protocol analyzer, Wireshark 2.6.10.

Throughout all conducted experiments, the TD file was directly registered on the server by the clients. These clients also registered varying numbers of sensor nodes with the server. Each instance communicated sampled data at regular intervals, with the inter-instance intervals varying randomly. The bounds of these random intervals, as well as other settings such as inactivity periods, were configurable.

It's noteworthy that network-level security mechanisms were omitted from these

Table 3.1: Main messages characteristics used in the simulations.

Phase	Phase 1	Phase 2	Phase 3
Message Format	TD	custom	SenML
Size Range [B]	[310 - 920]	[370 - 520]	[75, 150]
Size Range [B] (gzip lvl.6)	[220 - 340]	[270 - 340]	[75, 140]

experiments. The objective was to gauge the pure impact of SEAMDAP, excluding the potential overhead introduced by security protocols like Transport Layer Security.

To emulate genuine and coherent data flow communication, the exchanged messages were randomly generated from a pre-defined list of authentic and well-structured messages. The sizing ranges of the messages dispatched by the clients are indicated in Table 3.1. Additionally, the compressed sizes of these messages (gzip) are presented.

The demo has been evaluated under two distinct configurations, each defined by varying values for the number of registered data interfaces (TD files transmitted during phase 1) and the number of registered instances for each data interface. These numerical selections were based on the agricultural scale and dimensions of the Emilia-Romagna region. This region, based on surveys conducted by the Italian national statistical institute (Istat), encompasses approximately 60,000 agricultural enterprises.

The specifics of the two configurations are outlined in Table 3.2. Although the two configurations exhibit different ratios between interfaces and instances, they share an equivalent total instance count. As this instance count increases, so does the number of messages exchanged during the execution of the demo. Other parameters, such as inactivity intervals and the time elapsed between two uploads in phase 3, are determined randomly within predefined intervals that remain consistent across all configurations.

Regarding timing considerations, we emulated the behaviour of real users from the implemented system outlined in Section 3.4. These users typically upload one or two samples per registered sensor instance every hour. To ensure reasonable execution times for each test (capped at 120 minutes), we established the communication

Table 3.2: Parameters of the two simulation configurations.

	A	B
Data Interfaces	60000	10000
Instances per Data Interface	5	30
Data Interfaces over Instances per Data Interface	12000	333.33

Table 3.3: Simulation results. Number, total size and average latency of the messages exchanged in each experiment, grouped by phase.

	Configuration A	Configuration B
Number (Size) Ph. 1	60000 (34.44 MB)	10000 (5.74 MB)
Number (Size) Ph. 2	300000 (130.08 MB)	300000 (130.08 MB)
Number (Size) Ph. 3	8041563 (855.62 MB)	8035996 (855.03 MB)
Avg Latency Ph. 1	39.230 ms	43.833 ms
Avg Latency Ph. 2	230.612 ms	265.910 ms
Avg Latency Ph. 3	54.135 ms	52.720 ms
Avg Total Latency	60.372 ms	59.730 ms

period for each sensor instance within a random interval spanning from 75 to 150 seconds. This approach aligns the demo with the realistic usage patterns observed in the actual implementation.

The experimental results, presented in Table 3.3, confirm the property stated in Claim 4.1 of Section 4. This claim asserts that the number of automatic messages in phases 1 and 2 is proportional to the number of sensor nodes, as the number of registered data interfaces is much smaller than the number of sensor nodes.

Observing the results, it becomes evident that the most significant data flow arises from the sample uploads, directly correlated to the number of registered instances and the upload period. Phases 1 and 2 contribute minimally to the overall load. These phases are executed at most once per instance and necessitate user involvement.

Regarding latency, when accounting for random experiment fluctuations, all val-

ues resulted by simulation are consistent. Phase 1 exhibits the lowest latency due to its early execution, when network traffic and server message queues are minimal. In contrast, phase 2 displays higher latency due to traffic volume and longer processing time to verify message reliability and instance registration. Phase 3, involving small-sized messages necessitating minimal server processing, demonstrates low latency despite heavy network activity.

Concerning user manual interventions, the worst-case scenario involves generating descriptor files for phase 1 and configuring instance registration parameters for phase 2. Importantly, TD descriptor files used in phase 1 may require hardware expertise for accurate preparation, suggesting that manufacturers could facilitate integration by providing these descriptions. Additionally, phase 2, while flexible, could be executed by advanced devices or bypassed entirely with a more sophisticated server capable of autonomous content recognition and integrity evaluation for SenML messages. Thus, we may conclude that SEAMDAP minimizes user involvement, often requiring only minimal configurations.

Chapter 4

Secure Management of Georeferenced Data

The content of this chapter is derived from the paper "Enabling Location Based Services with Privacy and Integrity Protection in Untrusted Environments through Blockchain and Secure Computation", by Amoretti M, Budianu A., Caparra G., D'Agruma F., Ferrari D., Penzotti G., Veltri L., and Zanichelli F..

In this Chapter, secure data management in non-fully trusted environments is discussed, especially considering application in which *PNT* (Position, Navigation and Timing) data are involved. This topic takes on ever greater importance with the increase and proliferation of Internet-connected devices, generating an high demand of services. The request for a service from a node, whether it is at the Smart Farming level or closer to the edge of the network, is always an operation that brings significant challenges, particularly in the realms of security and data confidentiality.

The aspects that are covered concern secure communications (not at the application level), secure processing and position verification. The proposed mechanisms are therefore integrable on a system created with the tools presented in Section 2 (for the system architecture aspect), and also with those presented in Section 3 (concerning data collection from multiple sources).

The considerations and tools in this chapter mainly satisfy the requirements of

RQ5, establishing strong constraints for respecting user privacy. In this way, the requirements of RQ2 about the integrity of data in communication are also partially satisfied.

Section 4.1 introduces the problem of protecting user location data. In Section 4.2, some related work are explored and commented. In Section 4.3, a functional architecture that facilitates Location-Based Services while ensuring privacy and data integrity in untrusted environments is detailed. Section 4.4 delves into the interactions between its subsystems, accompanied by a comprehensive security analysis. Section 4.5 focuses on the location verification scheme, while Section 4.6 presents the performance evaluation of the proposed architecture.

4.1 Data Protection and Location-Based Services

In the context of modern digital services, a critical concern revolves around security and privacy of user data. As commented in Chapter 3, management of users is a complex growing theme that needs attention on the protection and dissemination of sensible and personal data, in order to preserve the privacy and the economic value of those data.

This issue becomes particularly intricate in services that hinge on geographic information, notably Location-Based Services (LBS). Typically, users share their location and contextual information with LBS providers in exchange for access to resources or facilities. The quality and utility of these services returns are intricately tied to the precision and accuracy of such data. However, the assumption of a trusted relationship between users and service providers may no longer be considered true in the some scenarios, due, for example, to the incidence of data breaches events that expose user data to attackers.

In fact, typically, service providers are external entities to a user's system, and cannot always be deemed entirely reliable, potentially making the exchange of sensitive information a precarious endeavor. This inherent vulnerability opens up the potential for data breaches that might compromise user information, thereby casting a shadow of doubt on the trustworthiness of service providers.

In addition to these security concerns, there is also the pressing matter of privacy breaches and the looming possibility of location data misuse [20]. The fear of one's location information falling into the wrong hands or being exploited without consent poses a substantial hurdle to the widespread adoption of these location-dependent services.

On the flip side, there are scenarios where users may intentionally provide false location data to gain an advantage. For example, in activities like racing competitions or transportation services, the user could be tempted to communicate forge location data, advantaging itself by falsifying the path taken. In response, service providers must possess the capability to verify the accuracy of location data and promptly identify any malicious user activities. This entails a delicate balance between maintaining data integrity and ensuring user privacy in a world where the concept of a trusted service-provider-user relationship has grown increasingly complex.

The pivotal challenge in this landscape revolves around the secure management of user data. It is imperative to ensure the authenticity, integrity, and privacy of user data while eliminating the need for unconditional trust. Various approaches to eradicating trust dependencies can be explored.

One of the main issues concerns the protection of *communications*. The adoption of appropriate communication techniques and protocols is essential to guarantee an adequate level of data integrity protection, but also to identify the correct sequence of interactions to guarantee the properties sought by the system, without disseminating unnecessary data.

Regarding *data processing*, preserving user privacy can be achieved by integrating advanced techniques and specific interaction patterns. Some of these techniques are *Private Computation*, *Zero Knowledge Proof*, *Homomorphic Encryption*, and *Multi Party Computation* [56, 57, 58, 59, 60]. Each of them brings specific advantages, but also refers to a particular scenario, bound by their requirements.

Also *data persistence* is a relevant topic in this scenario, both to resist external attacks (unwanted readings, tampering, deletions, etc.), and to maintain shared, non-alienable and non-repudiable evidence. In this case many classic security policies can be used, integrating elements of redundancy and access control, but also

adopting more recent solutions. For instance, position, navigation, and timing (PNT) information, as well as georeferenced data from one or multiple users associated with specific events, can be collected and stored in a decentralized, immutable storage system, such as a blockchain (BC) [61, 62, 63]. In cases where third-party verification is required, users may selectively disclose certain information. Attempts to tamper with the data can be detected through consistency checks between current data and previous records.

The novel aspect of this architecture lies in its simultaneous utilization of blockchain and secure computation technologies. Homomorphic Encryption and Multi Party Computation will be covered, as they are more mature solutions (especially from an IT point of view) and are better suited to the topic to be addressed. Mechanisms for sharing and processing user PNT data and georeferenced data are defined, along with a detailed exposition of the cryptographic schemes and algorithms employed. As an illustrative example, a strategy for implementing a location verification scheme based on ray casting is proposed, demonstrating how both MPC and HE can contribute to data processing in a privacy-preserving manner. The effectiveness of the proposed architecture has been validated through an emulation-based testbed.

4.2 Related Works

The design and prototypical implementation of the proposed architecture emerged from the need to address a gap in the literature concerning the safeguarding of privacy and data integrity in Location-Based Services (LBS).

PASPORT [64] is a framework tailored for secure and private location verification, employing collaborative strategies and fundamental cryptographic tools such as public key encryption and digital signatures. In PASPORT, mobile users supply a location proof (LP) to service providers to validate the accuracy of their submitted location. These LPs are generated by other mobile users who act as witnesses. PASPORT offers robust defenses against various fraudulent activities, including distance frauds, mafia frauds, terrorist frauds, LP forgery attempts, and sybil frauds. However, it's worth noting that some aspects of the collaborative procedures in PASPORT

hinge on optimistic assumptions regarding short-range communications between the prover and the witnesses. In practical tests, utilizing Bluetooth Low Energy, it was found that these assumptions do not hold.

Horton et al. [65] introduced Geoffdnet, a network of GNSS (Global Navigation Satellite System) receivers that operates based on a public blockchain and utilizes token incentives. This network comprises GNSS Miners, Validator Nodes, and Service Provider Nodes. Validator Nodes engage in a Proof of Stake (PoS) consensus protocol to validate transactions in the Geodnet blockchain, which includes observation data (e.g., space weather data) generated by GNSS Miners. This consensus protocol determines which node generates a new block at a given time point. GNSS Miners also transmit observation data to Service Provider Nodes, which are directly connected to end users. The concept is intriguing, but the absence of a comprehensive performance evaluation limits our ability to reach a conclusive judgment.

Awadallah et al. [66] proposed a scheme aimed at circumventing the ultimate authority that cloud service providers (CSPs) have over data. Their approach relies on Byzantine Fault Tolerance consensus to establish a distributed network of processing CSPs, which is constructed in accordance with client requirements. Following specific homomorphic encryption (HE)-based computations executed by all CSPs, they produce a master hash value for their database. To ensure the immutability of the produced data, these master hash values are preserved within the Bitcoin or Ethereum blockchain networks. The scheme is accompanied by an informal security analysis and an initial performance assessment.

4.3 Functional Architecture

The architecture presented in this Section is called *functional* because all the consideration about entities regards the functionality implemented, like receiving input messages and performing a transformation on the received data and serving the result to another entity. These entities are tasked with specific functions and differentiate from another based only on the role it assumes in the interaction that will be presented. For some task, a node necessitates a specific hardware and software requirements to

ensure their proper operation, but there are not considerations on the node position in the network. In fact, the positioning of these entities within the network architecture is highly flexible and not subject to strict restrictions. They can be strategically located at any layer of the network, offering adaptability to various operational needs and resource availability.

The proposed architecture is illustrated in Fig. 4.1, where *LSS* stands for Location Service System, *P* for Publisher, *LBSS* for location based service system, *Pr* for processing, *St* for storage system, *SC* for Service Consumer.

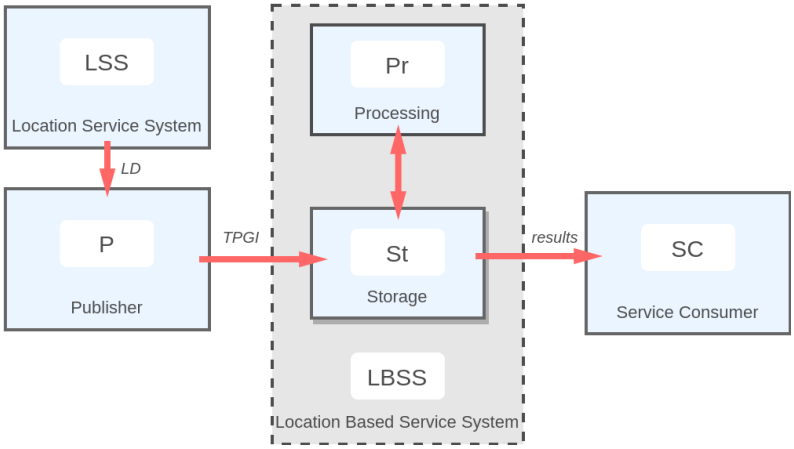

Figure 4.1: Proposed architecture.

Below the functional characteristics of the nodes that make up a system adhering to this architecture are discussed.

Location Service System (LSS)

An LSS is a system designed to implement Location Service functionalities. It serves the critical role of providing Location Data (LD) – including coordinates and, if available, error estimates – pertaining to a Traced Entity (TE). This LD is then transmitted to a Publisher (P), which, in turn, processes and conveys this data to the Location-Based Service System (LBSS). The LSS is flexible in its approach and can employ

various communication mechanisms.

Passive Communication Mechanism: With the passive communication mechanism, a TE's device acquires location information from a spatial or terrestrial location systems. Examples of such systems include GNSS-like systems, and location detection is based on physically located active devices, such as Wi-Fi Access Points (APs), PLMN or WMAN base stations, or any other radio or optical beacons.

Active Communication Mechanism: Contrastingly, the active communication mechanism operates on the basis of detecting the presence of the TE, or the TE's device, through one or more entities. These entities can be classified as follows:

1. Trusted Devices (TD): These devices are integral components of the LSS infrastructure, such as totems or base stations. They validate the location information.

2. Untrusted Devices: These are devices belonging to other TEs that come into proximity with the first TE within a specified time interval. In both cases, the location information is "certified" by the nearby device(s). In the second case, the assurance of location relies on the correctness of other TEs. To prevent potential collusive behaviors, a distributed trust and reputation management mechanism can be employed.

Depending on the chosen communication mechanism, the LD may or may not be certified, as follows:

- **Passive Communication:** LD cannot be certified (i.e., it cannot be signed by the LSS). However, if the LSS can utilize multiple devices, whether trusted or untrusted, it may attain a certain level of protection by conducting consistency checks on proximity measurements.

- **Active Communication with Trusted Device:** LD can be certified (i.e., it can be signed by the LSS).

- **Active Communication with Untrusted Device:** LD can be certified, provided that measures are in place to prevent collusive behaviors.

In cases of active communication, its possible to assume that a form of distance bounding [67] [68] is employed to prevent physical layer attacks on proximity measurements.

An LSS with passive communication continuously provides LD, allowing the Publisher (P) to periodically activate the receiver and acquire incoming LD items. In the case of active communication with trusted devices, the Publisher (P) must explicitly request LD. The response from the LSS includes digitally signed LD.

Publisher (P)

The Publisher (P) is closely associated with the Traced Entity (TE) and plays a pivotal role in the system. It receives TE-related Location Data (LD) from the Location Service System (LSS) as described in the previous subsection. From these LD, the Publisher (P) derives *Timestamped Position and Georeferenced Information* (TPGI).

The Publisher (P) then forwards these TPGI to the Location-Based Service System (LBSS). The communication pattern is straightforward, with the LBSS constantly listening for messages generated by the Publisher (P). These messages must be encrypted and signed by the Publisher (P). Upon receipt by the LBSS, the TPGI is stored in the Storage system, as detailed in the next subsection.

Location-Based Service System (LBSS)

The Location-Based Service System (LBSS) is a comprehensive system that implements Location-Based Service (LBS) functionalities. In essence, the LBSS receives TPGI messages from the Publisher (P) and performs LBS functions through processing (Pr) operations. Both input and output data can be securely recorded in a storage system (St).

In this proposed LBSS, the writing of TPGI items to the Storage system (St) is directly carried out by the Publisher (P), which employs this Storage (St) as a secure mechanism to transmit TPGI data to the LBS. The write operation is conducted in such a manner that it is always possible to verify that TPGI has been sent and to authenticate the source of the data. Collusive attacks with repudiation of the received

TPGI are only feasible if both the LBSS and the third-party storage system are malicious. However, such attacks become practically unfeasible if the Storage (St) is implemented using a highly decentralized mechanism, such as a Blockchain (BC).

The Storage (St) subsystem within the LBSS has a dual structure. It comprises a database for storing the TPGI and a Blockchain (BC) for permanently storing TPGI-derived data, like digests of the TPGI computed using a Cryptographic Hash Function. All data stored within the system are encrypted.

The Processing (Pr) subsystem of the LBSS can execute a wide range of operations. While the specific business logic varies for each LBS-based application, fundamental operations are common across all applications. A prominent example is location verification, as discussed in Section 4.5. Another shared aspect is the need to protect processed data from malicious Pr providers.

Homomorphic Encryption (HE) [69] encrypts the entire input before submitting it to a single Processing (Pr) unit. HE demands a high-performance Pr unit to process the encrypted input and produce the correct encrypted output. Notably, the input can be encrypted by the Publisher (P) before it is submitted to the Storage (St) subsystem. The encrypted output generated by the Pr unit is then returned to the Storage (St) subsystem for retrieval by authorized users possessing the decryption key, typically the Publisher (P) and the Service Consumer (SC).

Secure Multi-Party Computation (MPC) [58] divides the input into N fragments, each sent to a distinct Processing (Pr) unit owned by a different provider. These N Pr units compute specific functions on the inputs while preserving key security properties, including privacy and correctness. Privacy guarantees data confidentiality, ensuring that nothing leaks from the protocol execution except the computed output. Correctness assures the integrity of computations made by parties, with honest parties receiving the correct output. MPC computations per Pr unit are typically less computationally intensive than HE. However, MPC is viable when at least $N \geq 2$ Pr providers are involved, and at least one party is honest. A larger N increases communication overhead.

Service Consumer (SC)

The Service Consumer (SC) holds the privilege of decrypting and accessing the data provided by the Publisher (P) to the Location-Based Service System (LBSS). This access allows the SC to retrieve TPGI items from the database and TPGI-derived items from the Blockchain (BC). Furthermore, the SC has the capability to decrypt and review data produced by the Processing (Pr) subsystem of the LBSS. However, the SC is only authorized to perform a limited number of verifications to maintain the privacy of the Traced Entities (TEs).

4.4 Subsystems Interactions

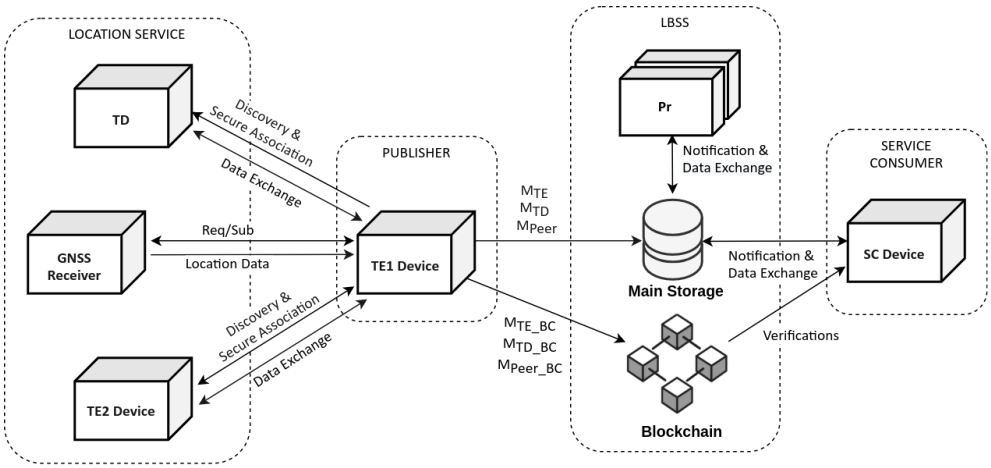

Figure 4.2: Detailed interactions between subsystems of the proposed architecture.

Figure 4.2 provides a detailed illustration of the interactions between subsystems within the proposed architecture. This configuration represents the most comprehensive and general setup, encompassing all possible Location Service System (LSS) and Storage types. It highlights the primary messages exchanged between subsystems, with these interactions and messages varying based on the LSS type (GNSS, trusted device, or peer-to-peer) and the Processing (Pr) type (Homomorphic Encryp-

tion - HE or Secure Multi-Party Computation - MPC). The Service Consumer (SC), assumed to be honest, receives application-specific notifications from the Location-Based Service System (LBSS) and is authorized to conduct verification activities on stored data.

To elucidate these interactions and messages, the most challenging scenario, which involves a peer-to-peer LSS type, is commented. The corresponding sequence diagram is delineated in Figure 4.3. In this scenario, the Publisher (P) is associated with Traced Entity 1 (TE1). The Publisher (P) generates the Timestamped Position and Georeferenced Information (TPGI) and dispatches it to the Location Service System (LSS), which is associated with another traced entity, namely TE2. Upon verifying the correctness of the TPGI, the LSS signs and returns it to the Publisher (P). Subsequently, the Publisher (P) divides the signed TPGI into two messages for the Location-Based Service System (LBSS) – one for storage in the database for future processing and the other for inclusion in the Blockchain (BC). Notably, in this bidirectional system, the LSS can interchangeably assume the role of the Publisher (P), and vice versa, certifying each other's positions reciprocally.

In the context of Homomorphic Encryption (HE), the P, to communicate data securely, sends a specific messages to the LBSS depicted as follows:

- $MPeer = \{ID_{TE1}; ID_{TE2}; \{TPGI\}_{KHeTE1}; Sig_{TE1}(\{TPGI\}_{KHeTE1})\}_{KeTE1}$
- $MPeerBC = \{ID_{TE1}; ID_{TE2}; Hash(TPGI); Sig_{TE2}(TPGI)\}_{KeTE1}$

where:

- $KeTE1$ is a symmetric key owned by the P and the LBSS used for encrypting the whole payload for preserving its confidentiality
- ID_{TE1} is TE1's identifier
- ID_{TE2} is TE2's identifier

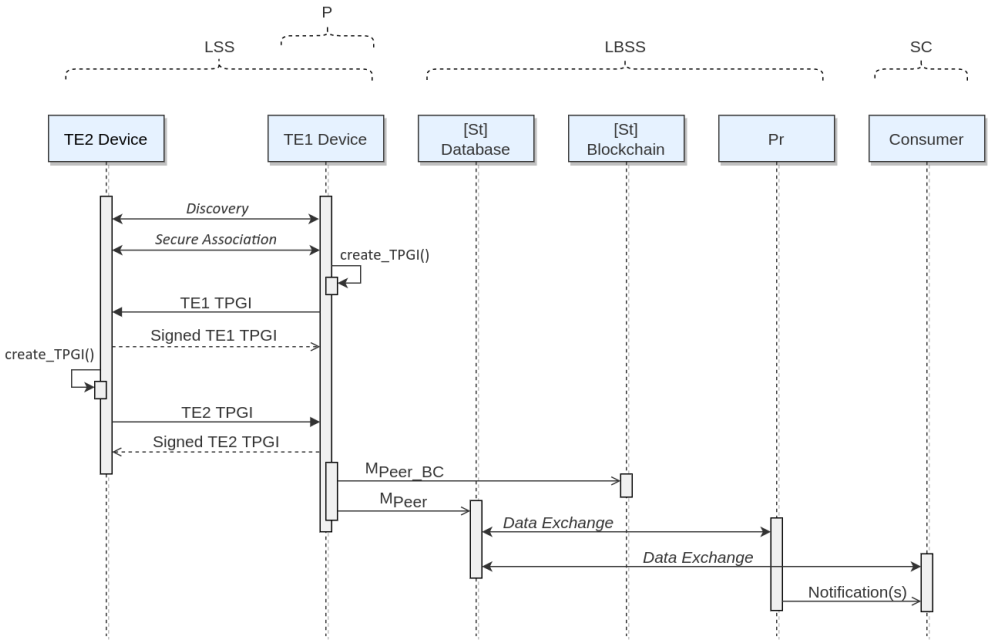

Figure 4.3: Sequence Diagram for the LSS type peer-to-peer.

- $KHeTE1$ is a symmetric key owned by the P and the SC (the encrypted TPGI is the input to the HE Pr)
- $Hash(x)$ is a Cryptographic Hash Function
- $Sig_{TE1}(x)$ is TE1's signature, meaning $\{Hash(x)\}_{KprTE1}$, where $KprTE1$ is a private key owned by the P
- $Sig_{TE2}(x)$ is TE2's signature, meaning $\{Hash(x)\}_{KprTE2}$, where $KprTE2$ is a private key owned by the LSS

When Secure Multi-Party Computation (MPC) is employed with N Processing Units (Pr), the Publisher (P) sends the following message with the LBSS:

- $MPeer = \{\{MPeer_1\}_{KPr_1}$

  ...

$\{MPeer_N\}_{KPr_N}$
$\{MPeer_1\}_{KSCeTE1}$
...
$\{MPeer_N\}_{KSCeTE1}\}_{KeTE1}$

- $MPeerBC = \{ID_{TE1}; ID_{TE2}; Hash(TPGI);$
  $SigTE2(TPGI)\}_{KBCeTE1}$

where

- $MPeer_i$ is the specific message for to the i-th Pr
- KPr_i is a symmetric key owned by the P and the i-th Pr
- $KeTE1$ is a symmetric key owned by the P and the LBSS
- $KSCeTE1$ is a symmetric key owned by the P and the SC
- $KBCeTE1$ is a symmetric key owned by the P and the blockchain
- ID_{TE1} is TE1's identifier
- ID_{TE2} is TE2's identifier
- $Hash(x)$ is a Cryptographic Hash Function
- $Sig_{TE1}(x)$ is TE1's signature, meaning $\{Hash(x)\}_{KprTE1}$, where $KprTE1$ is a private key owned by the P
- $Sig_{TE2}(x)$ is TE2's signature, meaning $\{Hash(x)\}_{KprTE2}$, where $KprTE2$ is a private key owned by the LSS

The specific message for to the i-th Pr unit has the following structure:

- $MPeer_i = \{ID_{TE1}, ID_{TE2}, TPGI_i,$
  $Sig_{TE1}(TPGI)\}$

where $TPGI_i$ is the TPGI piece assigned to the i-th Processing unit.

The symmetric keys listed above can be ephemeral/session keys derived via a standard key agreement based on public keys (which is omitted for the sake of simplicity).

4.4.1 Security Analysis

The communication between TE1 and TE2 is assumed to be secured using standard techniques, such as:

- **Secret Key Cryptography:** Also known as symmetric-key cryptography, it uses a single shared secret key for both encryption and decryption. This method is efficient for securing communication between two parties who already possess the shared key.
- **Cryptographic Hash Functions and MAC Functions:** Cryptographic hash functions are mathematical algorithms that take an input (or 'message') and produce a fixed-size string of characters, which is typically a hexadecimal number. They are used to verify data integrity. MAC (Message Authentication Code) functions are used to verify both the data integrity and the authenticity of a message.
- **Public Key Cryptography (PKC):** Also known as asymmetric-key cryptography, it involves a pair of keys: a public key for encryption and a private key for decryption. It is widely used for secure data transmission and digital signatures.
- **Digital Signatures:** Digital signatures provide a way to verify the authenticity and integrity of a message or document. They are created using the private key of the sender and can be verified using the sender's public key.
- **Collaborative Techniques: Witness-Prover Schemes:** These cryptographic schemes involve interactions between two parties, a prover and a verifier. They are used to prove the truth of a statement without revealing the actual data involved, ensuring privacy.

- **Data Storage Security Policies:** These policies define rules and procedures for securing data at rest, including encryption, access controls, and data backup strategies.

- **Access Control (AC):** Access control mechanisms determine who can access or modify data or resources in a computing environment. This includes authentication, authorization, and auditing to ensure data security.

In the case of Homomorphic Encryption (HE) usage, the Timestamped Position and Georeferenced Information (TPGI) is encrypted with a secret key ($KHeTE1$) shared between the Publisher (P) and the Service Consumer (SC), who are the end users of the system. These encrypted data are processed by the Location-Based Service System's (LBSS) Processing Unit (Pr) without revealing information to the LBSS owner. The communication between the P and the LBSS is secured through the shared secret key $KeTE1$. Storing the hash of the TPGI in the blockchain serves two purposes: i) creating an indelible, timestamped record of the TPGI, and ii) enabling the P to verify the integrity of the TPGI stored at the LBSS.

When Secure Multi-Party Computation (MPC) is employed, the TPGI is divided into N pieces, with each piece being encrypted using a secret key (KPr_i) shared between the P and the i-th Pr unit. This approach secures the communication between the P and the Pr units. The same messages are stored in the LBSS' Storage system (St), encrypted using a secret key ($KSCeTE1$) shared between the P and the Service Consumer (SC). To ensure traceability and data integrity, both the hash and the signature of the pieces are stored in the blockchain.

4.5 Location Verification

In the context of the Location-Based Service System (LBSS), one of the fundamental processing tasks is the dynamic checking of a position. This verification could be very complex, using specific interactions between P and LBSS. The position could be verified as instantaneous information, verifying boundaries requirements, or could includes spatial and kinetic constraints, considering either the current position or a

combination of current and previous positions.

For the sake of simplicity, in this work, the location verification task is considered as a simply 2D point inclusion problem. Consider a geometric shape presentable as a polygon with n edges e_1, e_2, ..., e_n. This polygon is defined by a set of vertices V_1, V_2, ..., V_n, where each edge e_i is determined by its endpoints V_i and V_{i+1}.

In this problem, the aim is to determine whether a given point lies within the interior of a geometric shape. While this simplification may not capture all real-world scenarios, it provides a foundational understanding of location verification within the LBSS.

Among various algorithms for solving point-in-polygon problems, the *Ray Casting* method stands out for its effectiveness and efficiency. For each given point p, the Ray Casting approach works by selecting a semi-infinite ray that extends from point P in an arbitrary direction. This ray's path must intersects with the polygon. As the ray encounters the polygon's edges, it alternates between entering and exiting, until the ray extends infinitely. Now, if the number of crossings between the ray and the polygon is odd, point P lies within the polygon's interior. Conversely, if the number of crossings is even, point P resides outside the polygon.

The Ray Casting method is visually represented in Figure 4.4, and a possible implementation is detailed in Algorithm 1.

In practical applications, LBSS may need to verify whether a mobile entity's current position aligns with specific spatial or kinetic constraints. This verification can be crucial in ensuring that services are delivered accurately based on the entity's location.

4.5.1 Adapting the Algorithm for Homomorphic Encryption Execution

Literature and preliminary tests have highlighted the varying performance of current Homomorphic Encryption (HE) implementations based on the types of functions they employ. Notably, when operations are limited to addition and multiplication, an algorithm working on real numbers and implemented using the CKKS scheme [70] can exhibit strong performance. However, when an algorithm relies on emulated binary

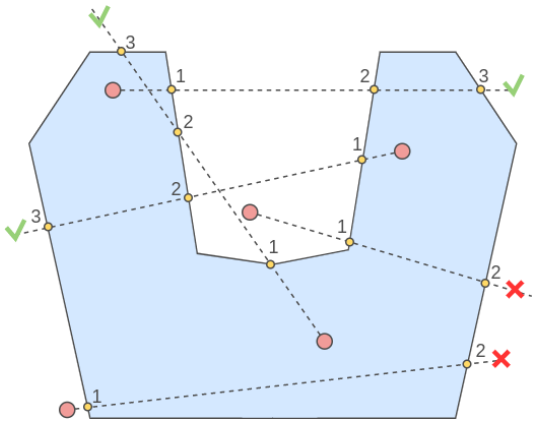

Figure 4.4: 2D Ray casting algorithm.

circuits (employing logical operators), such as the TFHE scheme [71], its performance can degrade rapidly.

For this reason, it is desirable to explore the implementation of inclusion verification using only polynomial functions. Unfortunately, the Ray Casting algorithm, like many other point inclusion algorithms, relies on operations beyond addition and multiplication, including comparisons and IF-THEN-ELSE instructions.

A potential solution involves transforming the algorithm into a sequence of algebraic operations, preferably using only additions and multiplications. This solution was explored in two steps:

- First, the point inclusion algorithm (Ray Casting) was converted into a sequence of mathematical functions, replacing IF-THEN-ELSE instructions with the step function. For instance, $IF\ (a < b)\ THEN\ f1()\ ELSE\ f2()$ becomes $step(b-a) * f1() + step(a-b) * f2()$.

- Second, the $step(x)$ function was substituted with a polynomial approximation. In this context, a truncated series of Chebyshev polynomial functions was considered as a suitable polynomial approximation within a given interval of the x axis.

Algorithm 1: Ray Casting algorithm

```
1:  P.x = the x-coordinate of the point
2:  P.y = the y-coordinate of the point
3:  rc = 0
4:  for each edge e_i=(V_1,V_2) do
5:      if P.y > V1.y1 and P.y < V2.y then
6:          Q = crossing point between the horizontal line y = P.y and the edge e_i
7:          if x < Q.x then
8:              rc++
9:          end if
10:     end if
11: end for
12: return (rc mod 2) = 1
```

In Figure 4.5, three different truncated series of Chebyshev polynomial functions, with degrees 10, 20, and 40, are illustrated. The Chebyshev coefficients were calculated to approximate the $step(x)$ function between -1 and 1. As shown in Figure 4.5 (b), the choice of the interval for computing the Chebyshev coefficients is crucial, as the resulting polynomial significantly diverges outside this interval.

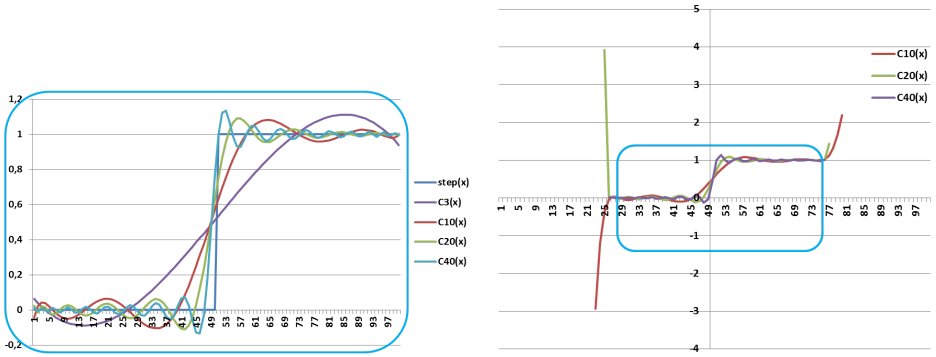

Figure 4.5: Polynomial approximation of the step function using truncated series of Chebyshev polynomial functions.

An implementation of the Ray Casting algorithm based on the use of the $step(x)$ function and its 40-degree Chebyshev polynomial approximation was tested with dif-

ferent shapes and trajectories. The results indicated accurate in-shape detection, except when the point was precisely on the shape's border.

4.6 Performance Evaluation

The proposed architecture has been implemented in an emulation-based testbed. This evaluation assesses data exchange, computational cost, communication overhead, and robustness.

4.6.1 Configuration

Two emulations have been configured and tested. The first one focuses on interactions between the LSS, the P, and the St part of the LBSS. The second emulation tests the Pr part of the LBSS. The respective testbeds are depicted in Fig. 4.6.

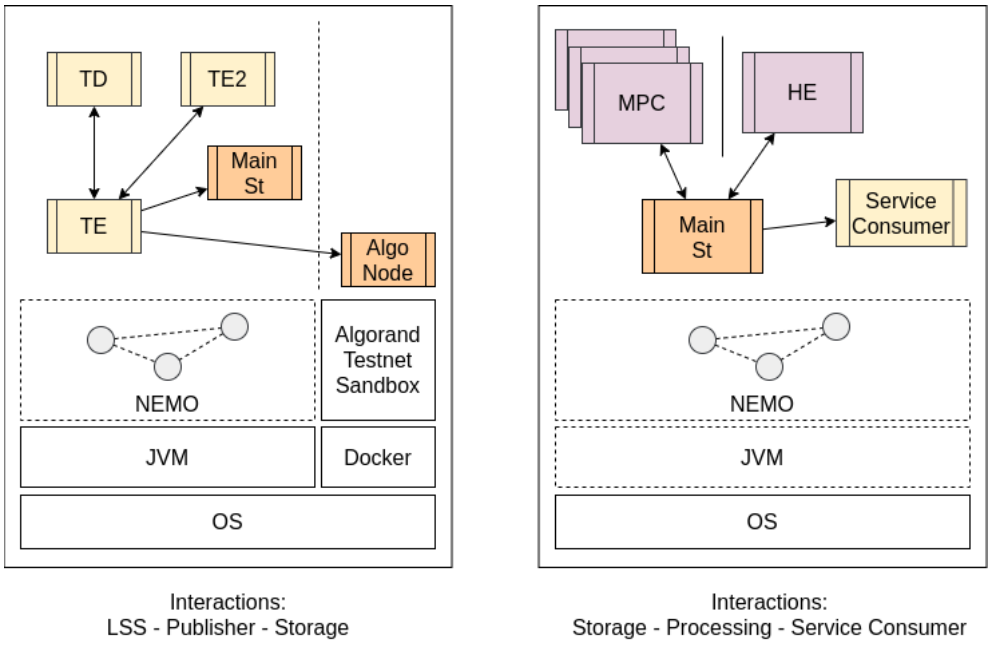

Figure 4.6: Emulation software deployment.

The first testbed, presented in Section 4.6.2, emulates interactions of a specific Traced Entity (TE) with other nodes, particularly Trusted Devices (TDs), other TEs, and the LBSS St part. This includes an emulated conventional database and the Algorand blockchain [72], known for its speed and efficiency. An Algo node runs locally (in a Docker environment) to connect to the Algorand testnet. The software for this testbed is primarily written in Java.

The second testbed executes the Ray Casting algorithm using Homomorphic Encryption (HE) and Multi-Party Computation (MPC) libraries and frameworks. Most of this software is written in C++ and Python.

Most entities are executed on Nemo[1] nodes, aiming to emulate a real network. These nodes are hosted by an Ubuntu 18.04.6 server featuring two Intel Xeon Silver 4210 CPUs with a clock frequency of 2.20GHz and 10 physical cores, and a maximum available RAM of 64 GB.

4.6.2 TPGI Creation and Storage

The interactions between one Traced Entity (TE) and several Location Service Systems (LSSs), including a GNSS system and an arbitrary number of trusted peers and P2P entities, has been emulated. Each simulation comprises a preliminary settings phase, an execution phase, and a potential results analysis. The settings phase involves configuring parameters of the simulation, including network settings, addresses, IDs, and the number of peers the TE interacts with.

In each execution of the simulation, all the processing and communications involving a single Traced Entity (TE) are performed, while other nodes are emulated as communication and processing interfaces. The TE initially acts as a Peer during the creation of Timestamped Position and Georeferenced Information (TPGI), communicating with Trusted Device (TD) nodes and other TE nodes. It then takes on the role of a Publisher when sending the results to the LBSS Storage, communicating with the main storage node and an Algorand node.

All nodes execute their routines and communicate according to a pre-calculated

[1] `https://netsec.unipr.it/project/nemo/`

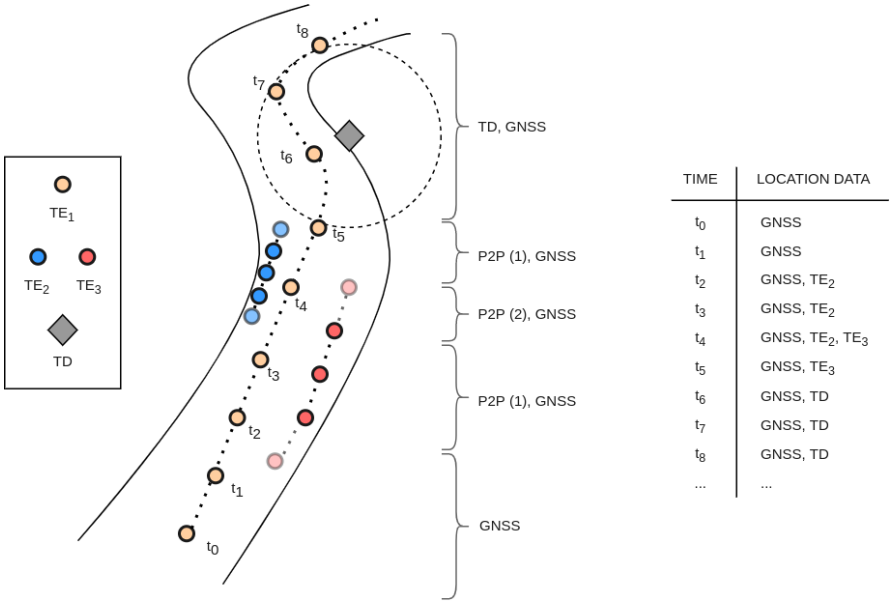

Figure 4.7: Conceptual modeling of the interactions of a TE in a sports competition. Go through its path a TE will encounter different location systems to establish its position. Depending on the type of use case, some of them may or may not be present (for example, GNSS may not be available for indoor situations), and with different visibility frequencies.

schedule, called "peer visibility model". To model interactions between peers and a TE, an algorithm was developed to generate a visibility schedules. This algorithm determines when and for how long a peer remains within the communication range of the TE. Figure 4.7 presents a graphical representation of this concept: TE follows a trajectory, encountering other peers and infrastructure devices (TD). Simultaneously, TE may or may not have GNSS signal coverage, allowing it to self-determine its position.

In order to simplify the process, the interaction scheduling is generated without modeling the underlying movement of peers: in fact, the encounters of the TE with other entities are modeled as a Poisson process, and encounter probabilities were

modeled using exponential distributions. All the process is customizable through the tuning of various configuration parameters, like simulation and sampling time, peers speed and acceleration, visibility ranges, etc.

The result of this algorithm is a schedule file, where for each time instant (determined by the sampling time) it is indicates the visibility of nodes in various location systems (GNSS, TD, and peer-to-peer).

After TE has established its position LD (Location Data) and created the TPGI items, this is sent and then further enriched by the Publisher (P). In the emulation environment, the Publisher P coincides with the TE.

Subsequent interactions occur between P and the Storage part of the Location-Based Service System (LBSS), within the emulation environment described earlier. Specifically, each TPGI creation leads to two communications: one with a conventional database and another with the blockchain (BC). The structure of these messages varies based on the Processing (Pr) type (HE or MPC).

The node hosting the conventional database receives the message, validates it, and writes it into a log file, making the data directly accessible to the Pr. For the blockchain, P contacts the Algorand node responsible for writing the message into the Algorand Testnet and awaits block consolidation.

Communications take place on Stream Sockets, which are connection-oriented and based on the TCP protocol. Messages at the application layer are constructed from scratch following the descriptions in Section 4.4, differentiating HE and MPC executions. This approach avoids constraints imposed by specific communication standards, ensuring results free of additional overhead. One exception is communications with the local Algorand node, which uses a REST API over the HTTP protocol, the official method for communicating with it.

The discrete-time dynamics of each entity has been implemented as isolated objects that operate concurrently. In particular, each entity represented in the simulation runs on different Java threads, maintaining consistency with the simulation's activity.

Random events, such as communication interruptions, were also emulated. Using the profiling tool for Java "YourKit", some performance data have been measured. In a 30-minute simulation conducted in Homomorphic Encryption (HE) mode, where

messages are tailored for HE processing and active logs are stored in files, the TE threads utilized the CPU for approximately 27316 ms. In terms of memory usage, all threads, including TE and other peers, collectively reached a peak heap memory occupancy of about 150MB and non-heap memory occupancy of 47MB, both well below the 200MB threshold.

It's essential to consider that providing highly detailed information on memory consumption and CPU usage in this case is challenging due to the inherent complexities introduced by Java, which includes various overheads and randomness stemming from the Java Virtual Machine (JVM) components, notably the Garbage Collector. Furthermore, the simulation software encompasses multiple nodes beyond the TE, making it difficult to segregate and analyze individual executions.

However, it's worth noting that despite the substantial volume of interactions in this simulation, the processing demands placed on the Publisher component are not particularly high. This suggests that the system can run efficiently even on hardware with relatively modest resources, easy to deploy this functionality in different level of an Edge-to-Cloud architecture.

4.6.3 Test of HE-based Location Verification

The objective of these tests is to evaluate the effectiveness of Homomorphic Encryption (HE) techniques in implementing the Pr system securely, ensuring the confidentiality of processed location data (LD). This is particularly crucial when LD should only be accessible to the Traced Entity (TE) and the Service Consumer (SC), excluding all other entities. So it's vital that the Location-Based Service System (LBSS), especially its Pr entities, cannot access the actual LD, using HE.

Two HE schemes have been tested: the Cheon-Kim-Kim-Song (CKKS) [70] scheme for real-number arithmetic, capable of handling floating-point operations, and the Chillotti-Gama-Georgieva-Izabachene (TFHE) [71] scheme for logical circuit-based operations, selected for its potential universality.

For these tests, the Palisade [73] library was used for the CKKS scheme, while the Google Transpiler [74] with the Palisade implementation of TFHE was employed. The choice of these libraries was based on factors like developer reputation, user

community size, ongoing source code development, quality of documentation, and available support.

Major details are reported below.

- **Palisade**

  The feasibility of implementing an HE algorithm was evaluated using the Palisade library, primarily due to its CKKS scheme support. A simplified scenario was considered, focusing on a rectangular area with limited IF-THEN-ELSE instructions to compare point coordinates with boundaries. The implemented C++ software computed resulting polynomials ($Px()$ and $Py()$) as functions of the rectangular area. These polynomials were then used to evaluate their values on a sequence of HE-encrypted input coordinates (X and Y). The resulting sequences of $Px(x_i)$ and $Py(y_i)$ are HE-decrypted, and combined through simple multiplication. The result is then compared to 0 (indicating being outside) or 1 (indicating being inside the shape). Due to computational limitations, Chebyshev polynomial functions of degrees 10 and 20 is used to approximate the step function. Table 4.1 presents the time required to compute the evaluation of 10, 100, 1000, and 10000 points when using 10-degree or 20-degree Chebyshev polynomial functions to approximate the step function.

	#Points	10	100	1000	10000
Mean Time [s]	**Degree 10**	0.716	0.724	0.744	0.739
	Degree 20	1.618	1.617	1.592	1.627

Table 4.1: Execution time of the ray casting algorithm with increasing number of points and different polynomial degrees for a square polygon, using Palisade.

- **Google Transpiler**

  This tool was assessed for compatibility with floating-point operations. However, it was found to support only basic arithmetic operations like addition,

subtraction, and multiplication. The execution time for a single operation was about 15 seconds. A ray casting algorithm was applied, requiring oprations equal to 10 times the number of vertices. Its execution time was significantly slower than the Palisade-based approach.

4.6.4 Test of MPC-based Location Verification

This test evaluates the MPC-based Processing (Pr) architecture using SCALE-MAMBA [75] and MP-SPDZ [76, 77] software libraries, selected for several reasons, including reputation of their developers and the quality of documentation and support. The ray casting algorithm was implemented for five Pr entities,configured as follows:

- Pr0 provides no input.
- Pr1 provides the x coordinates of the polygon's vertices.
- Pr2 provides the y coordinates of the polygon's vertices.
- Pr3 provides the x coordinate of the point.
- Pr4 provides the y coordinate of the point.
- The result, either 0 (indicating an outside point) or 1 (indicating an inside point), is revealed to Pr0.

The algorithm was executed using Shamir's secret sharing-based protocol for honest majority in the *actively secure with abort* security setting for malicious parties. In this setting, parties may deviate from the protocol, leading to execution failure, with notifications sent to all parties involved.

The top system monitor was used to To collect statistics on CPU load and RAM consumption. Execution time was recorded using the libraries' internal timers. The same applies to CPU load, which is cumulative over all used cores in a multi-core CPU. The values presented in the following charts have been proportionally scaled according to the number of cores available on the test machine. Traffic analysis between Pr units was done using Nemo.

Major consideration are reported below.

#Vertices	3	4	5	6	7	8	9	10
Mean Time [*s*]	5.379	5.478	5.384	5.480	5.402	5.484	5.582	5.683

Table 4.2: Execution time with increasing number of vertices using SCALE-MAMBA.

- **SCALE-MAMBA**

  The CPU load is evenly distributed across all Pr units, not exceeding 15%. RAM consumption is also uniformly distributed, with no unit consuming more than 500MB. Regardless of the number of vertices in the polygon, the execution time remains approximately 5 seconds, although there is a slight increase as the number of vertices grows (refer to Table 4.3).

  Network traffic analysis shows that the total data exchanged between the Pr units is very consistent reaching over 2GB, including all packets.

- **MP-SPDZ**

  CPU load is uniformly distributed among all processing units (Pr units) and seldom exceeds 8%. RAM consumption is uniformly spread across all Pr units and does not exceed 25 MB, significantly lower than SCALE-MAMBA's RAM usage. Regardless of the number of vertices in the polygon, execution time consistently falls within the range of 0.3 to 0.5 seconds, as shown in Table 4.3. This performance indicates that MP-SPDZ outperforms SCALE-MAMBA in terms of speed.

  The overall network traffic is significantly lower than in the case of SCALE-MAMBA, with only 160MB compared to 2GB.

#Vertices	3	4	5	6	7	8	9	10
Mean Time [*s*]	0.310	0.320	0.336	0.347	0.444	0.460	0.542	0.538

Table 4.3: Execution time with increasing number of vertices using MP-SPDZ.

4.7 Considerations

The emulations demonstrate that the proposed functional architecture could enable the secure management (transmission, store and elaboration) of georeferenced data for LBS. It is possible to conclude that the privacy and integrity protection in untrusted environments, can be effectively implemented. This includes standard encryption and authentication mechanisms, as well as more advanced strategies such as Multiparty Computation and Homomorphic Encryption.

Among the evaluated software libraries, MP-SPDZ for MPC stands out for its impressive results, driven by its reduced memory footprint and processing speed. Concerning HE, the processing speed can be unsatisfactory with current technologies, especially when employing HE schemes supporting arbitrary Boolean circuits.

In the creation of a real system, however, it is necessary to also take into account the overhead introduced by the security mechanisms, which for some nodes located at the edge of the network may be too burdensome, therefore taking into consideration other approaches more suitable for networks of low-capacity nodes . To protect processing, more complex reasoning must be done, as not all tasks can be easily converted into operations adaptable to the technologies used (HE and MPC). Furthermore, they introduce a strong overhead, which currently makes them difficult to implement for edge processing, but more suitable for large data aggregations at the fog or Smart Farming level.

Regarding the use of operational protocols, such as the one described in Section 3, the considerations outlined in this chapter concerning communication protection do not pose significant issues since they treat the payload in an agnostic manner. In the development of an actual system, it is imperative to account for the overhead introduced by security mechanisms. For certain nodes located at the network's periphery, this overhead may become excessively burdensome. In such cases, it becomes necessary to explore alternative approaches more suitable for networks with low-capacity nodes.

When it comes to protecting data elaborations, a more complex analysis is required, as not all tasks can be easily transformed into operations adaptable to the

tested technologies - Homomorphic Encryption (HE) and Multiparty Computation (MPC). Furthermore, these mechanisms introduce a substantial overhead that presently renders them challenging to implement for edge processing. Instead, they are better suited for large data aggregations at the Fog or Cloud level.

4.8 Use Cases Specification

In this section, two use cases are specified ad analyzed following the functional architecture indicate in Section 4.3, commenting how both could benefit from the adoption of a secure location verification approach.

4.8.1 Sport Race Competition

This use case involves managing a sports competition, such as a running marathon, cycling race, or motor race, using a location-based system for tracking contestant positions and ensuring fair competition.

Contestants are mandated to adhere to a predetermined race route, which may be composed of multiple laps, commencing from the beginning of the competition until they successfully cross the finish line. Throughout the event, each contestant's location is consistently tracked through the utilization of *positioning devices*, equipped with different technologies. These devices determine the exact whereabouts of the contestants and furnish the *Competition Management System* (CMS) with critical data in the form of TPGI.

In the course of the competition, as well as in its aftermath, the CMS fulfills a pivotal role by actively monitoring the contestants' positions. Should any contestant transgress the established competition rules, such as deviating from the prescribed route or employing unauthorized means, the CMS promptly notifies the competition officials. This notification is substantiated by the received location data, which serves as irrefutable evidence of the infraction. In effect, the CMS delivers a proof of rule violation to any contestant found in breach.

Importantly, the inherent design of this use case ensures that contestant privacy is not compromised, at least in the context of TPGI shared during the competition. Con-

testants willingly share this information with the CMS for the sole purpose of corroborating their adherence to the competition's regulations. This collaborative approach underlines the trust and transparency that characterizes this sport race management system.

Involved *actors* are:

- Contestants: Individuals participating in the sports competition.
- Competition Management System (CMS): An LBS (Location-Based Service) platform utilizing internal or cloud storage for managing the competition.
- Officials: Competition judges or organizers responsible for monitoring and ensuring the integrity of the competition.

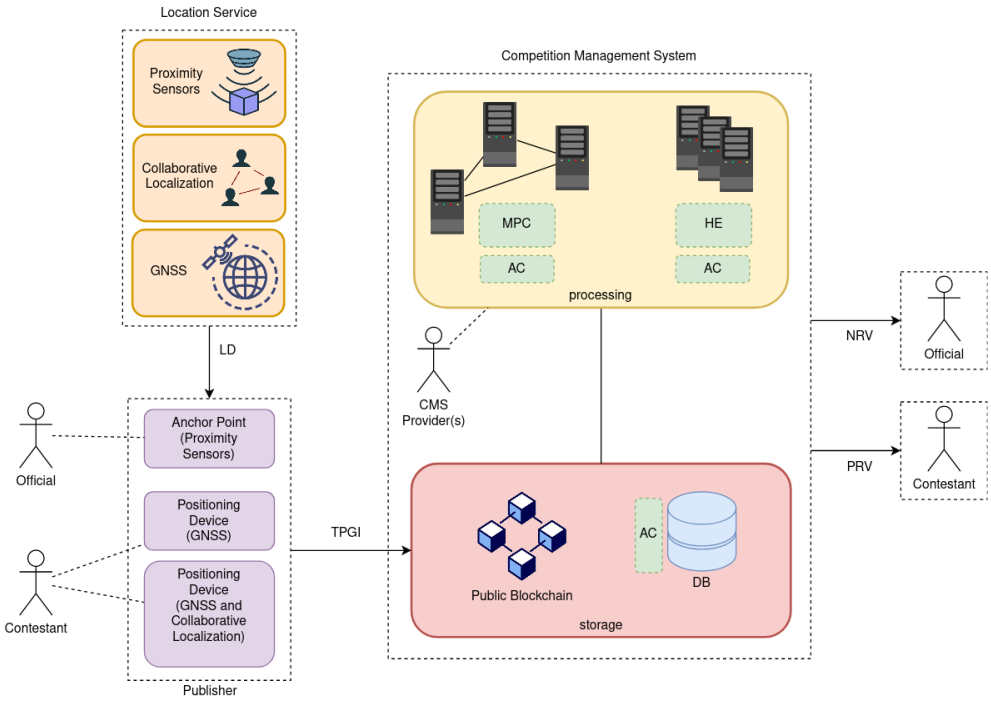

Figure 4.8: Specialized architecture for sport race tracking use case.

The specialized architecture, depicted in Figure 4.8, comprises several integral modules, each serving a distinct function:

- Location Service: Location data (LD) is gathered from an array of sources, including proximity sensors and anchor points like RFID gates and access points. Additionally, LD may be acquired from global navigation satellite systems (GNSS) such as GPS, GLONASS, and Galileo. Collaborative P2P localization may also occur among mobile devices equipped by contestants to enhance accuracy and reliability.

- Publisher: Each contestants are equipped with tracking devices, such as RFID tags, Bluetooth low-energy devices, or mobile GPS trackers. TPGI messages are periodically generated by LS sources and publish to the St.

- Storage: The Storage System employs a public blockchain, used in conjunction with a traditional database incorporating Access Control policies. The blockchain is employed to immutably and transparently store contestants' TPGI, thus ensuring indisputable evidence of their positions and rankings. In addition, it is used to record any potential misbehavior. The traditional database retains detailed TPGI, and Access Control policies are implemented to enable the auditability of the data life-cycle within the database.

- Processing: In the context of this distributed architecture, the Processing system is overseen by one or more Competition Management System Providers. Secure Multi-Party Computation (MPC) is used in scenarios where a collective of mutually untrusting entities collaboratively compute the processing functions on their inputs (TPGI fragments) without disclosing any information beyond the resultant output. Alternatively, Homomorphic Encryption (HE) may be employed, either cooperatively or competitively. In the former case, the Publisher subdivides the TPGI into multiple fragments, encrypts them, and dispatches them to the Storage system, where they are processed by multiple Pr providers. Standard Access Control policies are enforced to ensure the security of the processing.

- Service Consumer: Contestant and Official are both consumer. In the event of a rule violation, a notification is promptly transmitted to the competition official. The verification results, computed by the Pr, can be documented in the Storage system, and the competition official is granted access to them. Contestants also possess the option to request a "proof of rule violation", underpinned by the unalterable record stored within the public blockchain.

To bolster the security of data flows between these entities, a combination of message authentication and encryption mechanisms is applied, tailored to the specific communication channels in use, as depicted in Section 4.4.

4.8.2 Smart Irrigation and Fertigation

This use case regards the tracking of Smart Farming activities, in particular concerning precise irrigation or fertigation.

Farmers can utilize a Decision Support System (DSS), which functions as a public or commercial service. DSSs provide scientifically accurate irrigation and fertigation advice, acting as an LBS. The advice is personalized to the farmers' fields based on biophysical parameters and vegetation indices derived from satellite and/or ground sensor data (TPGI). Typically, a specialized *agronomic* DSS is employed for this purpose [3].

The DSS generates its advice based on publicly available agronomic research and is subject to verification by the scientific community. This open approach allows other entities to validate the DSS's outputs. In some cases, farmers may be required to furnish report data regarding their irrigation activities (*Consumed Resources, RC*) to the *Authority*, such as a government agency. This reporting serves as proof that they have adhered to the irrigation prescription and is often necessary to access economic incentives.

For instances where farmers deviate from the prescriptions, the Authority should receive notifications (NRV). These data could be verified using a *Prescription Verifier System* (PVS), a service that operates in parallel with the DSS. The PVS is responsible for ensuring that users have correctly applied the specified quantities of

resources, adhering to the spatial and quantitative parameters essential for gaining economic concessions. It's important to note that the PVS does not need access to the proprietary algorithms or "industrial secrets" of the DSS but requires capabilities to verify adherence to constraints. The PVS may be provided by the Authority or another designated PVS provider.

Data provided by farmers typically includes georeferenced information about the plot (e.g., position and perimeter) and other relevant characteristics (soil composition, slope, etc.). Information about cultivation and farm activities in relation to the plot is also supplied. Sampling activities, facilitated by ground sensors, encompass the collection of atmospheric, soil, and crop parameters.

In addition to ground sensor data, information from remote sensing sources is incorporated. Remote sensing data is primarily acquired through satellite-based monitoring, either from a public Earth monitoring service like the Copernicus mission or a commercial service. This information provides a fundamental overview of a plot, which can be further enhanced by ground sensor data.

Before sharing, user-provided data might undergo obfuscation or aggregation, depending on the system's requirements. Such processing can be carried out by the farmers themselves or by other stakeholders involved in the process.

The *actors* involved are:

- Farmer: a user requiring irrigation and fertigation advice for his crops. These activities need actuators for precise resource distribution and may have sensors for ground sampling.
- DSS Provider: entity providing Resource Prescription (RP) advice.
- Prescription Verifier System (PVS) Provider: Service that runs in parallel with the DSS. It is tasked with verifying that users have applied resources in accordance with RP advice. PVS does not require access to the proprietary algorithms of the DSS but needs functions to verify compliance with constraints. This service may be provided by the same entity as the Authority.
- Authority: An actor responsible for offering economic concessions, such as a

Regional Government, after a demonstrate compliance with the RP.

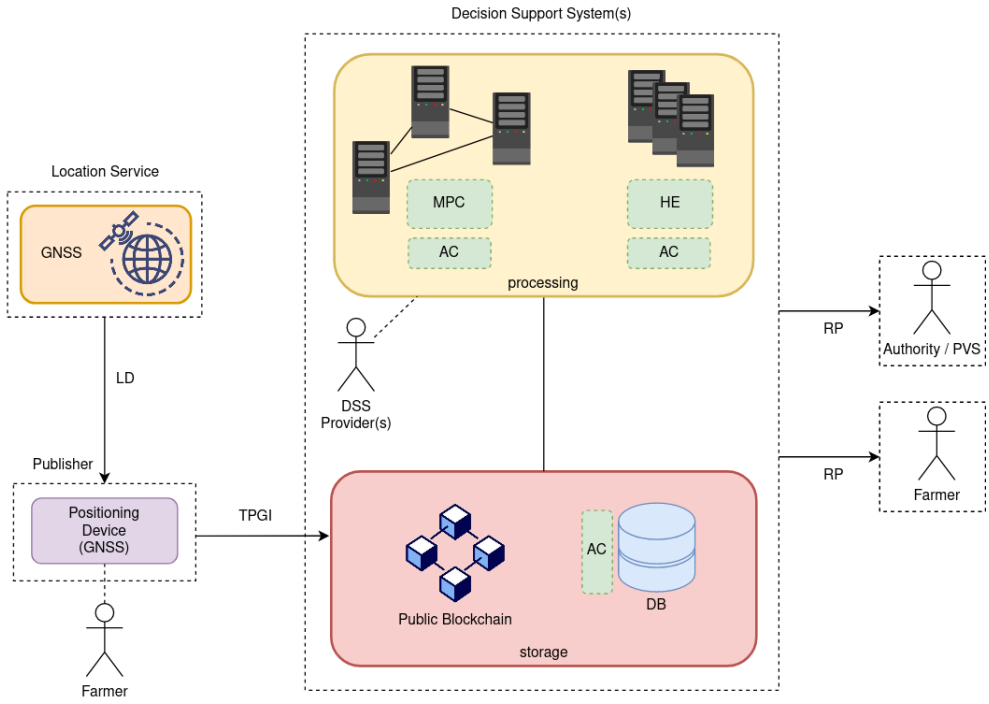

Figure 4.9: Specialized architecture for precise irrigation or fertigation use case.

The specialized architecture depicted in Figure 4.9 is dedicated to the DSS and is composed of the following key modules:

- Location Service: Location Data (LD) is primarily sourced from a Global Navigation Satellite System (GNSS), which includes satellite systems like GPS, GLONASS, and Galileo. These location data are collected directly from ground sensors or, in some cases, are hard-coded during the sensor installation.
- Publisher: Farmer's intelligent actuators generate TPGI as a combination of LD, augmented with biophysical parameters and vegetation indices obtained through ground sensors and remote sensing samples. These TPGI messages are periodically published to the DSS storage.

- Storage: The Storage System encompasses a blockchain, used in tandem with a conventional database, incorporating Access Control policies. Blockchain component serves to store control data related to the TPGI sent to the DSS, such as hash values, ensuring an immutable and distributed control entity that safeguards against malicious data manipulation. Meanwhile, the traditional database handles daily changes within the storage system and implements Access Control policies to facilitate the audit trail of the data's lifecycle.

- Processing: Within this distributed architecture, the Processing system is under the ownership of one or more DSS Providers. In scenarios where secure Multi-Party Computation (MPC) is employed, a group of mutually distrusting entities collaborates to jointly compute the processing function on their respective inputs (TPGI fragments), with a strict focus on not revealing any information beyond the output. Due to the complexity of the models used for computing Resource Prescriptions (RP), Homomorphic Encryption (HE) may not be a feasible option. However, alternative mechanisms, such as multiple redundant DSS systems or validation against historical data, are employed to ensure the accuracy of the provided RPs. Access Control policies are standard practice to further bolster the security of the processing.

- Service Consumer: Ultimately, the Resource Prescription (RP) is furnished to the farmer by the DSS. The DSS results can be recorded in the Storage System, granting the farmer and other authorized entities access to this data.

As a *critical issue*, it is highlighted that the use of a passive positioning system such as GNSS is often the only available choice, which makes the positioning data unreliable. Dedicated hardware, advanced GNSS systems (for example using RTK) or active positioning systems are to be considered in the creation of a physical system, in order to obtain reliable positioning data and correspondent applied resource.

Conclusion

In the ever-evolving landscape of technology, where innovation continues to reshape various sectors, the demand for sophisticated and adaptable systems for Smart Environment is both pressing and evident. These settings, characterized by a profusion of data from diverse sources and complex distributed systems, propose remarkable opportunities as well as unique challenges. In response to these demands, this Thesis has sought to address the need for a flexible and effective *framework* that can deal with some aspects of the intricate panorama of modern distributed data aggregation and processing systems, while safeguarding data privacy and security.

The proposed framework has been designed to thrive in dynamic and interconnected contexts, and revolves around the core idea of being able to adapt to an ever-changing technological landscape. It embraces modularity, enabling the integration of tools and technologies that can efficiently address the multifaceted aspects of the explored scenarios.

An essential facet of our framework is its unique *E2C Distributed Architecture*, extending service placement from the edge to the cloud, designed with the principles of Fog Computing. It strongly prioritizes reliable data management, enabling the respect of data privacy as a paramount requirement, and offering scalability and adaptability to diverse contexts.

Another significant contribution is the *Seamless Data Acquisition Protocol (SEAM-DAP)*, a standardized approach designed for modern distributed systems. It allows for seamless data collection from a variety of sources while being user-friendly and customizable.

Eventually, the work has also delved into the critical subjects of *data privacy and security*. Acknowledging the importance of georeferenced data and *Location-Based Services* (LBS), the framework provides a foundation for addressing the challenges of secure data management for the topic of location verification. These features open up new possibilities for enriched data analysis, yet simultaneously introduce additional complexities. In this context, the significance of safeguarding data during transmission, memorization and processing is addressed. The potential integration of advanced processing techniques has been explored. *Homomorphic Encryption* and *Multi-Party Computation* could be used to address privacy and security concerns, especially in untrusted environments.

The direction taken with this framework is to bring standardization to the field of smart environments, and to address some issues identified. Some of the presented tools were tested in Smart Farming environments, and for each one, a use case has been proposed. In fact, the framework could bring advantages in environments characterized by high heterogeneity (e.g., in data, interfaces, applications), using a lot of geo-referenced data, demanding interoperability between systems belonging to different actors, and in a strong sensitivity to respect for privacy.

The tools and techniques presented lay the groundwork for future research to build upon. By further exploring these areas, it is possible to advance efficiency, security, and sustainability of smart environments, enhancing the quality of service and resource management.

In terms of future research, the Thesis sets the stage for several avenues:

- Interoperability and Standardization. There are several scenarios that would benefit from introducing a standard approach, such as for data collection in this framework. For example, it is possible to introduce protocols for the management of intelligent actuators, such as irrigation machines in the Smart Farming field.

- Privacy-Preserving Techniques. It is possible to expand the scenario of solutions that can be adopted for data protection. For example, architectural solutions that involve the use of techniques such as Private Computation and

Zero Knowledge Proof can be studied and implemented. Federated Learning techniques could be explored to enable secure and privacy-preserving Machine Learning algorithm.

- Scalability and Resource Management. Exploring techniques that allow to improve the scalability of systems, seeing microservices as the currently most relevant element for the implementation of services. For example, online service placement algorithms with fault tolerance and load balancing features could be explored.

- Environmental and Ethical Considerations. Exploring tools and practices that promote environmental sustainability and ethical practices, aligning with the current focus on sustainability. The service placement could be done taking into account the energy consumption of a node and its environmental impact, conditioned by the availability of electricity produced from renewable sources.

- AI-based solutions. Implement solutions based on Machine Learning and AI in general for the management of various processes, such as application deployment or general data exploitation.

Bibliography

[1] COM/EdgeCloud-SC. IEEE 1934-2018 - IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing, 2018.

[2] POSITIVE Project Team. POSITIVE project homepage. http://www.progettopositive.it/, 2022.

[3] Michele Amoretti, Dario Lodi Rizzini, Gabriele Penzotti, and Stefano Caselli. A scalable distributed system for precision irrigation. In *Proc. IEEE International Conference on Smart Computing (SMARTCOMP 2020)*, pages 338–343, Bologna, Italy, September 2020.

[4] Sandya De Alwis, Ziwei Hou, Yishuo Zhang, Myung Hwan Na, Bahadorreza Ofoghi, and Atul Sajjanhar. A survey on smart farming data, applications and techniques. *Computers in Industry*, 138:103624, 2022.

[5] Muhammad Shoaib Farooq, Shamyla Riaz, Adnan Abid, Kamran Abid, and Muhammad Azhar Naeem. A survey on the role of iot in agriculture for the implementation of smart farming. *IEEE Access*, 7:156237–156271, 2019.

[6] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai. A survey of iot security based on a layered architecture of sensing and data analysis. *Sensors*, 20(13), 2020.

[7] Vasileios Moysiadis, Panagiotis Sarigiannidis, Vasileios Vitsas, and Adel Khelifi. Smart farming in europe. *Computer Science Review*, 39:100345, 2021.

[8] Lanfranco Zanzi, Flavio Cirillo, Vincenzo Sciancalepore, Fabio Giust, Xavier Costa-Perez, Simone Mangiante, and Guenter Klas. Evolving Multi-Access Edge Computing to Support Enhanced IoT Deployments. *IEEE Communications Standards Magazine*, 3(2):26–34, 2019.

[9] Foodie Consortium. Foodie project. https://www.foodie-project.eu/, 2017.

[10] Mehdi Roopaei, Paul Rad, and Kim-Kwang Raymond Choo. Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging. *IEEE Cloud Computing*, 4(1):10–15, 2017.

[11] Miguel A. Zamora-Izquierdo, José Santa, Juan A. Martínez, Vicente Martínez, and Antonio F. Skarmeta. Smart farming iot platform based on edge and cloud computing. *Biosystems Engineering*, 177:4–17, 2019.

[12] M.J. O'Grady, D. Langton, and G.M.P. O'Hare. Edge computing: A tractable model for smart agriculture? *Artificial Intelligence in Agriculture*, 3:42–51, 2019.

[13] Franklin Magalhães Ribeiro, Ronaldo Prati, Reinaldo Bianchi, and Carlos Kamienski. A nearest neighbors based data filter for fog computing in iot smart agriculture. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 63–67, 2020.

[14] Olivier Debauche, Mahmoudi Saïd, Pierre Manneback, and Frederic Lebeau. Cloud and distributed architectures for data management in agriculture 4.0 : Review and future trends. *Journal of King Saud University - Computer and Information Sciences*, 33, 10 2021.

[15] Konstantinos Perakis, Fenareti Lampathaki, Konstantinos Nikas, Yiannis Georgiou, Oskar Marko, and Jarissa Maselyne. Cybele–fostering precision agriculture & livestock farming through secure access to large-scale hpc enabled virtual industrial experimentation environments fostering scalable big data analytics. *Computer Networks*, 168:107035, 2020.

[16] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, 2012.

[17] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[18] Zhaoyu Zhai, José Fernán Martínez, Victoria Beltran, and Néstor Lucas Martínez. Decision support systems for agriculture 4.0: Survey and challenges. *Computers and Electronics in Agriculture*, 170:105256, 2020.

[19] Yaganteeswarudu Akkem, Saroj Kumar Biswas, and Aruna Varanasi. Smart farming using artificial intelligence: A review. *Engineering Applications of Artificial Intelligence*, 120:105899, 2023.

[20] Susanne Barth and Menno D.T. de Jong. The privacy paradox – investigating discrepancies between expressed privacy concerns and actual online behavior – a systematic literature review. *Telematics and Informatics*, 34(7):1038–1058, 2017.

[21] Gabriele Penzotti, Stefano Caselli, and Michele Amoretti. An n-tier fog architecture for smart farming. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2021.

[22] N.M. Trendov, S. Varas, and M. Zeng. Digital technologies in agriculture and rural areas - status report. Technical report, Nations Food and Agriculture Organization of the United, 2019.

[23] Angelita Rettore de Araujo Zanella, Eduardo da Silva, and Luiz Carlos Pessoa Albini. Security challenges to smart agriculture: Current state, key issues, and future directions. *Array*, 8:100048, 2020.

[24] Mohamed Baghrous, Abdellatif Ezzouhairi, and Nabil Benamar. Smart farming system based on fog computing and lora technology. In *Embedded Systems and Artificial Intelligence*, pages 217–225, Singapore, 2020. Springer Singapore.

[25] Leanne Wiseman, Jay Sanderson, Airong Zhang, and Emma Jakku. Farmers and their data: An examination of farmers' reluctance to share their data through the lens of the laws impacting smart farming. *NJAS - Wageningen Journal of Life Sciences*, 90-91:100301, 2019.

[26] Emma Jakku, Bruce Taylor, Aysha Fleming, Claire Mason, Simon Fielke, Chris Sounness, and Peter Thorburn. "if they don't tell us what they do with it, why would we trust them?" trust, transparency and benefit-sharing in smart farming. *NJAS - Wageningen Journal of Life Sciences*, 90-91:100285, 2019.

[27] Constantinos Marios Angelopoulos, Gabriel Filios, Sotiris Nikoletseas, and Theofanis P. Raptis. Keeping data at the edge of smart irrigation networks: A case study in strawberry greenhouses. *Computer Networks*, 167:107039, 2020.

[28] T. Nguyen Gia, L. Qingqing, J. Peña Queralta, Z. Zou, H. Tenhunen, and T. Westerlund. Edge ai in smart farming iot: Cnns at the edge and fog computing with lora. In *2019 IEEE AFRICON*, pages 1–6, 2019.

[29] Mohit Taneja, Nikita Jalodia, John Byabazaire, Alan Davy, and Cristian Olariu. Smartherd management: A microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming. *Software: Practice and Experience*, 49(7):1055–1078, 2019.

[30] Asad Waqar Malik, Anis Ur Rahman, Tariq Qayyum, and Sri Devi Ravana. Leveraging fog computing for sustainable smart farming using distributed simulation. *IEEE IoT Journal*, 7(4):3300–3309, 2020.

[31] I. Lera, C. Guerrero, and C. Juiz. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019.

[32] Dan Boneh, Divya Gupta, Ilya Mironov, and Amit Sahai. Hosting services on an untrusted cloud. In *Advances in Cryptology - EUROCRYPT 2015*, pages 404–436, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[33] Florian Kelbert, Franz Gregor, Rafael Pires, Stefan Köpsell, Marcelo Pasin, Aurélien Havet, Valerio Schiavoni, Pascal Felber, Christof Fetzer, and Peter Pietzuch. Securecloud: Secure big data processing in untrusted clouds. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 282–285, 2017.

[34] Lei Xu, JongHyuk Lee, Seung Hun Kim, Qingji Zheng, Shouhuai Xu, Taeweon Suh, Won Woo Ro, and Weidong Shi. Architectural protection of application privacy against software and physical attacks in untrusted cloud environment. *IEEE Transactions on Cloud Computing*, 6(2):478–491, 2018.

[35] Jingxian Cheng, Saiyu Qi, Wenqing Wang, Yuchen Yang, and Yong Qi. Fast consistency auditing for massive industrial data in untrusted cloud services. In *Proc. of the 2020 on Great Lakes Symposium on VLSI*, GLSVLSI '20, page 381–386, New York, NY, USA, 2020. ACM.

[36] Roy Thomas Fielding and Richard N. Taylor. *Architectural Styles and the Design of Network-Based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[37] A. Banks, E. Briggs, K. Borgendale, and R. Gupta. MQTT Version 5.0 - OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, 2019.

[38] Internet Engineering Task Force (IETF). Sensor Measurement Lists (SenML). RFC 8428, 2018.

[39] IPSO. IPSO Alliance Framework. http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[40] Mahdi Ben Alaya, Samir Medjiah, Thierry Monteil, and Khalil Drira. Toward Semantic Interoperability in oneM2M Architecture. *IEEE Commun. Mag.*, 53(12):35–41, 2015.

[41] Farah Aït Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys*, 53(3), 2020.

[42] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.*, 47(4), 2015.

[43] Gabriele Penzotti, Davide Tarasconi, Stefano Caselli, and Michele Amoretti. Seamless sensor data acquisition for the edge-to-cloud continuum. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 1–8, 2022.

[44] Hesham El-Sayed, Sharmi Sankar, Mukesh Prasad, Deepak Puthal, Akshansh Gupta, Manoranjan Mohanty, and Chin-Teng Lin. Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment. *IEEE Access*, 6:1706–1717, 2018.

[45] Daniel Balouek-Thomert, Eduard Gibert Renart, Ali Reza Zamani, Anthony Simonet, and Manish Parashar. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. *The Int.l J. of High Performance Computing Applications*, 33(6):1159–1174, November 2019.

[46] Daniel Del Gaudio and Pascal Hirmer. Seamless integration of devices in industry 4.0 environments. *Internet of Things*, 12:100321, 2020.

[47] ECMA Int.l. The JSON data interchange syntax. ECMA-404, 2017.

[48] W3C WoT Working Group. Web of Things (WoT) Thing Description. https://www.w3.org/TR/wot-thing-description/, 2020.

[49] W3C JSON-LD Working Group. Json-LD. https://json-ld.org/, 2022.

[50] Simon Fernandez, Michele Amoretti, Fabrizio Restori, Maciej Korczyński, and Andrzej Duda. Semantic Identifiers and DNS Names for IoT. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, Piscataway, USA, 2021. IEEE.

[51] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access*, 7:82721–82743, 2019.

[52] Olivier Alphand, Michele Amoretti, Timothy Claeys, Simone Dall'Asta, Andrzej Duda, Gianluigi Ferrari, Franck Rousseau, Bernard Tourancheau, Luca Veltri, and Francesco Zanichelli. IoTChain: A blockchain security architecture for the Internet of Things. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, Piscataway, USA, 2018. IEEE.

[53] Sachi Nandan Mohanty, K.C. Ramya, S. Sheeba Rani, Deepak Gupta, K. Shankar, S.K. Lakshmanaprabu, and Ashish Khanna. An efficient Lightweight integrated Blockchain (ELIB) model for IoT security and privacy. *Future Generation Computer Systems*, 102:1027–1037, 2020.

[54] Filippo Vurro, Michela Janni, Nicola Coppedè, Francesco Gentile, Riccardo Manfredi, Manuele Bettelli, and Andrea Zappettini. Development of an In Vivo Sensor to Monitor the Effects of Vapour Pressure Deficit (VPD) Changes to Improve Water Productivity in Agriculture. *Sensors*, 19:46–67, 2019.

[55] IRRIFRAME Team. IRRIFRAME. https://www.irriframe.it, 2022.

[56] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[57] Hua Sun and Syed Ali Jafar. The capacity of private computation. *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[58] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-Scale Secure Multiparty Computation. In *ACM SIGSAC Conf. on Computer and Communications Security*, 2017.

[59] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li. An efficient privacy-preserving location-based services query scheme in outsourced cloud. *IEEE Transactions on Vehicular Technology*, 65(9):7729–7739, 2016.

[60] H. Shen, M. Zhang, H. Wang, F. Guo, and W. Susilo. A lightweight privacy-preserving fair meeting location determination scheme. *IEEE Internet of Things Journal*, 7(4):3083–3093, 2020.

[61] Michele Amoretti, Giacomo Brambilla, Francesco Medioli, and Francesco Zanichelli. Blockchain-based proof of location. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 146–153, 2018.

[62] B. Li, R. Liang, D. Zhu, W. Chen, and Q. Lin. Blockchain-based trust management model for location privacy preserving in vanet. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3765–3775, 2021.

[63] Y. Xudong, G. Ling, L. Yan, Z. Hairong, G. Quanli, Z. Jie, and W. Hai. A blockchain-based location privacy-preserving scheme in location-based service. *Mobile Information Systems*, 2022, 2022.

[64] Mohammad Reza Nosouhi, Keshav Sood, Shui Yu, Marthie Grobler, and Jingwen Zhang. PASPORT: A Secure and Private Location Proof Generation and Verification Framework. *IEEE Transactions on Computational Social Systems*, 7(2):293–307, 2020.

[65] M. Horton, D. Chen, Y. Yi, and W. Xiaohua. Global Earth Observation Decentralized Network. Technical report, GeoDAO, 2021.

[66] Ruba Awadallah, Azman Samsudin, Je Sen Teh, and Mishal Almazrooie. An Integrated Architecture for Maintaining Security in Cloud Computing Based on Blockchain. *IEEE Access*, 9:69513–69526, 2021.

[67] Stefan Brands and David Chaum. Distance-Bounding Protocols. In *EUROCRYPT '93*. 1994.

[68] Mridula Singh, Patrick Leu, and Srdjan Capkun. UWB with Pulse Reordering: Securing Ranging against Relay and Physical-Layer Attacks. In *Network and Distributed System Security Symposium*, 2019.

[69] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, 2009.

[70] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[71] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[72] Algorand Team. Algorand. https://www.algorand.com/, 2022.

[73] PALISADE Teams. PALISADE. https://palisade-crypto.org/software-library/, 2022.

[74] Google. FHE C++ Transpiler. https://github.com/google/fully-homomorphic-encryption, 2022.

[75] KU Leuven. SCALE MAMBA. https://homes.esat.kuleuven.be, 2022.

[76] Marcel Keller. *MP-SPDZ: A Versatile Framework for Multi-Party Computation*, page 1575–1590. Association for Computing Machinery, New York, NY, USA, 2020.

[77] Marcel Keller. MP-SPDZ. https://github.com/data61/MP-SPDZ, 2022.

Acknowledgments

This thesis represents the conclusion of a complex journey, full of challenges that I was able to overcome thanks to many people who, fortunately, surround me.

I want to express my gratitude, first and foremost, to my supervisor, Professor Stefano Caselli. What I have learned from his valuable advice, although it represents only a small part of his vast experience, has been of immense help to me. His humanity has made the journey even smoother.

Similarly, I want to thank with affection Professor Michele Amoretti, who played a key role in my research, guiding and assisting me. I have always admired his professionalism and efficiency in his work, and I hope to one day reach his standards.

In the workplace, we spend most of our time, and it would be extremely more challenging without colleagues and friends capable of creating an environment in which we feel comfortable. I want to thank those who have contributed to creating this atmosphere.

No one is useless in this world if they are capable of lightening the burden of another person. I want to thank all the friends from Lunigiana and Parma (and beyond) for the time and joy they have given me, even though I feel less affection for half of them half as well as them deserve.

One of the things I have learned from living far from home is a real and persistent desire to return. Without a doubt, this is partly due to the charm that the Apennines and its woods have on me. However, it is mainly because my family resides there. I want to thank everyone for their support and unconditional love.

Finally, I want to express my gratitude to a person who, every time I think of

them, makes me smile because I feel that they have completed my life. To my wife Manuela, to whom I owe my desire to breathe: I would prefer to share a single life with you rather than face all the ages of this world alone.

* * *

Questa Tesi rappresenta la conclusione di un percorso complesso, pieno di difficoltà che ho potuto sormontare grazie a molte persone che per fortuna mi circondano.

Desidero esprimere la mia gratitudine, in primo luogo, al mio tutor, il Professor Stefano Caselli. Ciò che ho imparato dai suoi preziosi consigli, anche se rappresenta solo una piccola parte della sua vasta esperienza, è stato di immenso aiuto per me. La sua umanità ha reso il percorso ancora più agevole.

Allo stesso modo, desidero ringraziare con affetto il Professor Michele Amoretti, che ha svolto un ruolo chiave nella mia ricerca, guidandomi e aiutandomi. Ho sempre ammirato la sua professionalità ed efficienza nel lavoro, e spero un giorno di poter raggiungere i suoi standard.

Nel luogo del lavoro, trascorriamo la maggior parte del nostro tempo, e sarebbe estremamente più difficile senza colleghi ed amici in grado di creare un ambiente in cui ci sentiamo a nostro agio. Desidero ringraziare coloro che hanno contribuito a creare e sempre ravvivano questa atmosfera.

Nessuno è inutile in questo mondo se è capace di alleviare il peso di un'altra persona. Voglio ringraziare tutti gli amici della Lunigiana e di Parma (e oltre) per il tempo e le gioie che mi hanno donato, nonostante io nutra per meno della metà di loro metà dell'affetto che meritano.

Una delle cose che ho imparato vivendo lontano da casa è una reale e persistente voglia di tornare. Senza dubbio, questo è dovuto in minor parte al fascino che l'appennino e i suoi boschi hanno su di me. Però, per la maggior parte è dovuto al fatto lì c'è la mia famiglia. Desidero ringraziare tutti loro per il sostegno e l'amore incondizionato.

Infine, voglio esprimere la mia gratitudine a una persona che, ogni volta che ci penso, mi fa sorridere, perché sento che ha completato la mia vita. A mia moglie Manuela, a cui devo la mia voglia di respirare: preferirei condividere una sola vita con te piuttosto che affrontare tutte le ere di questo mondo da solo.