GDPR-Conform Machine Learning in Cross-Enterprise Environments

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurswissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Timon Sachweh

Dortmund

2026

Dekan:	Prof. Dr. Jens Teubner
Gutachter*innen:	Jun.-Prof. Dr. Thomas Liebig
	Prof. Dr. Thorsten Jungeblut

Abstract

Cross-enterprise Machine Learning (ML) promises substantial utility gains by combining complementary data held by different organizations, yet it faces persistent barriers from data sovereignty requirements, heterogeneous IT landscapes and regulatory constraints such as the EU General Data Protection Regulation (GDPR). Although Federated Learning (FL) mitigates centralized data collection by keeping raw data local, practical deployments remain exposed to indirect leakage via shared model updates (e.g., membership inference or gradient-based reconstruction) and to governance frictions when coordinating training across organizational boundaries. In addition, GDPR obligations go beyond confidentiality and include lifecycle requirements, most prominently the Right to Erasure (Art. 17), which makes post-hoc deletion requests technically challenging once models have been trained on personal data.

This thesis addresses these challenges by designing an integrated, privacy-preserving and GDPR-conform FL stack for execution in federated Data Spaces. First, it contributes *FedDScon*, an engineering framework that operationalizes FL-capable Data Space infrastructures on top of modern connector technology (with an emphasis on Eclipse Dataspace Connector concepts) and automates essential processes such as asset provisioning, contract negotiation and governed data transfer setup. Second, it develops a layered privacy-by-design toolkit for distributed learning in Data Spaces. Approaches such as *DP-LLP* and *DP-LSTM* are proposed, which restrict cross-party communication to differentially private aggregates for time-series learning and the *HEX-FL* framework, which protects model updates during federated aggregation using CKKS-based (multiparty) homomorphic encryption. Third, it introduces *CeMUFl*, a certified federated unlearning approach that integrates an unlearning procedure into the FL pipeline and adds a hash-based certification mechanism to make compliance-relevant model versioning verifiable across heterogeneous participants.

Finally, the thesis synthesizes these artifacts into *Feder*, a modular end-to-end framework that composes the Data Space governance layer (FedDScon), the privacy-preserving training layer (HEX-FL and DP-based methods) and the compliance/lifecycle layer (CeMUFl) into one coherent architecture for regulated, cross-enterprise analytics.

Acknowledgments

The past years of my doctoral journey have been deeply transformative – both professionally and personally. They have given me the chance to explore research that excites me, collaborate with brilliant minds and grow through countless discussions, challenges and shared accomplishments. I am sincerely grateful for all the experiences and the support I have received along the way.

First and foremost, I would like to express my deepest gratitude to my supervisor, Thomas Liebig, for his continuous guidance, encouragement and trust. Your mentorship has profoundly shaped my academic path and allowed me the freedom to pursue ideas that inspired me. The opportunities to engage in exciting research directions and to participate in conferences and collaborations have been invaluable for my professional and personal development.

I would also like to thank my colleagues and friends — Andreas Roth, Helen Kuhlmann, Jan Henning Büns, Jonas Werhahn, Pierre Haritz and Siba Mohnsen — for the stimulating discussions, their constant support and the great atmosphere in our working group. Each of you contributed in your own way to making this time not only productive but also truly enjoyable.

To my co-authors — Christoph Heinbach, Daniel Boiar, Dennis Maecker, Felix Harenbrock, Helen Kuhlmann, Henning Gösling, Kolja Berger, Lucas Pohling, Marco Kremer, Oliver Thomas, Sonika Gogineni and Tom Pieper — thank you for the inspiring collaboration, constructive feedback and the shared enthusiasm that enriched every project we worked on together. Your ideas and commitment have significantly enhanced the quality and depth of our research outcomes.

I am profoundly thankful to my family — my parents, Sabine and Stephan and my brother, Leon — for their boundless love, patience and support over all these years. Your confidence in me has been a constant source of strength and I am truly thankful for your unwavering encouragement.

Finally, my deepest thanks go to Victoria (Vicky), the love of my life. Your support, understanding and boundless motivation have accompanied me every step of the way. You remind me daily of what truly matters and I look forward to all the adventures that lie ahead with you by my side.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement and Research Goals . . . 2
- 1.3 Contribution . . . 4
- 1.4 Outline . . . 6

2 Related Work **9**
- 2.1 Federated Data Spaces . . . 9
  - 2.1.1 Industrial Data Spaces . . . 10
  - 2.1.2 Gaia-X . . . 11
  - 2.1.3 IDSA and Gaia-X comparison . . . 12
- 2.2 Connector Technology . . . 12
  - 2.2.1 Industrial Data Spaces Connector . . . 13
  - 2.2.2 Eclipse Data Space Connector . . . 14
- 2.3 General Data Protection Regulation . . . 15
  - 2.3.1 Core Privacy Principles . . . 15
  - 2.3.2 The Right to Be Forgotten . . . 16
  - 2.3.3 Sensitive Data and the Protection of Special Categories . . . 16
  - 2.3.4 Implications for Machine Learning and Cross-Enterprise Environments . . . 17
- 2.4 Data Privacy . . . 18
  - 2.4.1 Differential Privacy . . . 18
  - 2.4.2 Homomorphic Encryption . . . 19
- 2.5 Federated Learning . . . 20

3 Research Methodology **23**
- 3.1 Design Science Research . . . 23
  - 3.1.1 Three-cycle View of DSR . . . 24
  - 3.1.2 Applying DSR Methodology: Iterative Artifact Design and Evaluation for Federated Data Spaces . . . 25
- 3.2 Research Project as Case Study: GAIA-X 4 ROMS . . . 26
- 3.3 Constructed Use-Case Based on Standardized Datasets . . . 28

4 Operationalizing Data Spaces for Federated Learning Workflows **31**
- 4.1 Introduction . . . 31

4.2 Related Work . . . 33
4.2.1 Federated Data Spaces . . . 33
4.2.2 Industrial Data Spaces and IDS-RAM . . . 34
4.2.3 Eclipse Data Space Connector . . . 35
4.3 Systematic Comparison between IDS and EDC . . . 36
4.3.1 Evaluation Method: ATAM . . . 36
4.3.2 Derivation of ATAM Screening Questions . . . 37
4.3.3 ATAM-Based Evaluation of EDC and IDS . . . 38
4.3.4 Implications of the ATAM Evaluation . . . 39
4.4 Scenario-based Connector Evaluation . . . 41
4.4.1 Multi-Agent System based on top of IDS services . . . 41
4.4.2 Intelligent Arrival Time Prediction in Data Ecosystems . . . 45
4.5 FedDScon Framework . . . 50
4.5.1 FedDScon Features and Capabilities . . . 51
4.5.2 Automated Process for Data Space Communication Establishment 52
4.5.3 Provided Services and Templates of FedDScon . . . 57
4.5.4 FedDScon Usage . . . 59
4.6 Summary . . . 60
4.6.1 Systematic Comparison between IDS and EDC . . . 60
4.6.2 Scenario-based Connector Evaluation . . . 61
4.6.3 FedDScon Framework . . . 61
4.7 Discussion . . . 62

5 Privacy-Preserved Federated Learning in Cross Enterprises 65
5.1 Introduction . . . 65
5.2 Related Work . . . 68
5.2.1 Differential Privacy . . . 68
5.2.2 Homomorphic Encryption and CKKS . . . 70
5.2.3 Federated Learning . . . 74
5.3 Differentially Private Learning from Label Proportions (DP-LLP) . . . 75
5.3.1 Problem Setting and Motivation . . . 76
5.3.2 Differential Privacy for Label Proportions . . . 76
5.3.3 LLP in a Fully Distributed Time Series Architecture . . . 77
5.3.4 Differentially Private Computation of Label Proportions . . . 79
5.3.5 Experimental Evaluation of Distributed Traffic Data . . . 80
5.3.6 Advantages in the Context of Federated Data Spaces . . . 82
5.4 Differential Privacy Encoded in LSTM Model Architectures (DP-LSTM) . 83
5.4.1 Positioning with respect to Previous Sections and Related Work . 83
5.4.2 Cross-Enterprise Setting and Network Model . . . 84
5.4.3 Time-Series Label Proportions and Differential Privacy . . . 85
5.4.4 Model Architecture: Local LSTM with Spatial Aggregates . . . 86
5.4.5 Distributed Data Exchange in Federated Data Spaces . . . 90
5.4.6 Experimental Evaluation . . . 90
5.4.7 Advantages in the Context of Federated Data Spaces . . . 93

5.5 Homomorphic Encryption in Federated Learning environments (HEX-FL) 94
5.5.1 Survey of CKKS Libraries . . . 94
5.5.2 HEX-FL Architecture and Multiparty Protocol . . . 97
5.5.3 Evaluation . . . 103
5.5.4 Advantages in the Context of Federated Data Spaces . . . 109
5.6 Summary . . . 110
5.6.1 Differentially Private Learning from Label Proportions (DP-LLP) 110
5.6.2 Differential Privacy Encoded in LSTM Model Architectures (DP-LSTM) . . . 111
5.6.3 Homomorphic Encryption-based Federated Learning (HEX-FL) . 112
5.7 Discussion . . . 112
5.7.1 Comparison of DP-LLP and DP-LSTM . . . 113
5.7.2 Performance Differences between Homomorphic Encryption Libraries . . . 115
5.7.3 Combining DP-LLP, DP-LSTM and Homomorphic Encryption for Multi-Level Privacy . . . 117

6 Federated Unlearning with Data Spaces 119
6.1 Introduction . . . 119
6.2 Related Work . . . 120
6.2.1 Privacy-Preserved Federated Learning . . . 120
6.2.2 FedSSU . . . 121
6.3 FedSSU Integration in HEX-FL . . . 122
6.4 Certification of the Unlearning Model . . . 123
6.4.1 Attack Scenarios . . . 124
6.4.2 Interaction Between Certification Components . . . 125
6.4.3 Hash-Based Verification Process . . . 126
6.5 Evaluation of Unlearning Capabilities . . . 130
6.5.1 Time-Related Performance Comparison . . . 131
6.5.2 Anamnesis Index . . . 132
6.5.3 Unlearning Performance corresponding to λ . . . 133
6.5.4 Jensen–Shannon Distance per λ . . . 134
6.6 Summary . . . 136
6.7 Discussion . . . 137

7 Integrated Framework for GDPR-Conform Federated Learning 139
7.1 Introduction . . . 139
7.2 Architecture Concept . . . 140
7.3 FedDScon . . . 141
7.4 HEX-FL . . . 142
7.5 Certified Machine Unlearning in Federated Infrastructures (CeMUFl) . . 142
7.6 Feder: Integrated Framework . . . 142
7.7 Feder Application . . . 143
7.7.1 Use Case Description . . . 143

7.7.2 Use Case Participants . . . 144
7.7.3 Feder Usage Evaluation . . . 144
7.8 Summary . . . 148
7.8.1 Feder: Integrated Framework . . . 148
7.8.2 Evaluation . . . 149
7.9 Discussion . . . 149

8 Summary and Outlook **151**
8.1 Answers to Research Questions . . . 151
8.1.1 Research Question 1 . . . 151
8.1.2 Research Question 2 . . . 152
8.1.3 Research Question 3 . . . 152
8.1.4 Research Question 4 . . . 153
8.2 Summary of Contributions . . . 154
8.3 Discussion . . . 154
8.4 Outlook . . . 155

List of Figures **157**

List of Tables **159**

List of Algorithms **161**

List of Acronyms **163**

Bibliography **167**

1 Introduction

This chapter outlines the context, objectives and structure of the dissertation within the broader landscape of privacy-preserving and General Data Protection Regulation (GDPR)-conform Federated Learning (FL) in cross-enterprise Data Spaces. It first introduces the Motivation*, highlighting the increasing relevance of data-driven collaboration, regulatory constraints and emerging Privacy-Preserving Machine Learning (PPML) technologies. Building on this foundation, the subsequent section* Problem Statement and Research Goals *derives the central research questions that guide this work. The* Contribution *section then summarizes the key conceptual, architectural and algorithmic contributions, linking each to the corresponding Research Questions and positioning them within modern federated and privacy-aware system architectures. Finally, the* Outline *section provides a concise overview of the remaining chapters, clarifying how the theoretical foundations, methodological choices and implemented frameworks jointly realize a holistic approach to secure and regulation-compliant FL across enterprises.*

1.1 Motivation

Over the past decade, digital transformation has profoundly changed the way organizations operate, make decisions and create value. Increasingly, business processes are powered by data-driven methods and intelligent systems that enable automation, optimization and prediction. In this context, data has become one of the most critical assets for enterprises. It serves as the foundation for new business models, algorithmic decision-support systems and cross-enterprise value creation. This growing reliance on data analytics and Machine Learning (ML) has created a paradigm shift towards data-centric architectures and processes (Zha et al., 2025).

Modern enterprises rarely operate in isolation. Instead, they form complex and dynamic ecosystems with numerous stakeholders, including suppliers, service providers, customers and partners, who jointly contribute to value creation along the entire process chains. These collaborative ecosystems, often called *data ecosystems* or *federated Data Spaces*, are highly dependent on secure and standardized data exchange mechanisms. They enable participants to combine distributed data sources to generate collective intelligence and support data-driven innovation (Otto and Jarke, 2019). However, such close interconnections between organizations also introduce substantial technical, legal, and governance-related challenges.

A central requirement in these distributed networks is the ability to integrate algorithms and analytic models across organizational boundaries. This integration allows the application of advanced ML methods to shared or complementary data distributed across different enterprises. However, the ability to apply algorithmic intelligence in cross-enterprise networks is strongly limited by data protection and information security constraints. Especially in the European context, the GDPR, while essential to guaranty individual privacy and strengthening user trust, imposes strict conditions on the processing and exchange of personal data (Timan and Mann, 2021; Voigt and Von dem Bussche, 2017). These regulations restrict the free flow of sensitive or person-related data, which many ML models depend on for accurate prediction and training.

The tension between privacy-enhancing regulation and data-driven business optimization creates a fundamental challenge for data-intensive methods. On the one hand, regulations like the GDPR promote data sovereignty and individual control, fostering trust and accountability within digital ecosystems. On the other hand, they limit the potential of modern ML algorithms that rely on large, heterogeneous, and representative datasets. Consequently, research and industry are striving to develop novel techniques that balance privacy and data utility, enabling ML applications in compliant ways.

One particularly promising direction is the concept of FL (Kairouz et al., 2021; Yang et al., 2019). FL allows multiple parties to collaboratively train shared ML models without transferring raw data, thus maintaining local data protection and confidentiality. Complementary research in PPML enhances FL with cryptographic techniques such as Homomorphic Encryption (HE), secure multi-party computation, or Differential Privacy (DP) (C. Chen et al., 2025; Truex et al., 2019). Meanwhile, the rise of cross-enterprise architectures, such as the Eclipse Dataspace Connector (EDC) and frameworks like *Gaia-X* or the International Data Spaces (IDS) introduces standardized governance and interoperability principles for secure data exchange on a conceptual level.

Despite this progress, these developments remain fragmented. FL frameworks, privacy-preserving mechanisms and Data Space architectures often emerge as isolated solutions that solve distinct parts of the overall challenge. Currently, there exists no integrated framework that seamlessly combines cross-enterprise data management, advanced privacy-by-design techniques, regulatory compliance and scalable ML capabilities in one coherent environment. Addressing this gap forms the motivation for this research and sets the foundation for the following problem statement.

1.2 Problem Statement and Research Goals

Building on the motivation outlined above, this work addresses several interrelated challenges that stem from the growing need for collaborative and privacy-preserving ML in distributed business environments. Although there are multiple standalone approaches, their integration into a unified, privacy-compliant and technologically practical architecture remains unresolved.

First, there is a lack of comprehensive overviews and systematic evaluations of existing tools and methods that address federated, privacy-preserving, and cross-enterprise ML. Available solutions are often highly specialized, focus on individual aspects, such as model aggregation, encryption, or secure communication, and lack integration or standardization across domains and platforms. Consequently, they are difficult to combine and customize for diverse business use cases. These limitations form the foundation for the first research question:

Research Question 1 *How can modern software technologies and architectures be used to create a standardized and sovereign environment for cross-enterprise FL tasks?*

Second, privacy protection and regulatory compliance are essential elements for the adoption of cross-enterprise ML services. However, traditional privacy mechanisms often reduce the accuracy of the model or restrict scalability. Moreover, privacy-by-design principles must be embedded directly into the architecture of such systems to ensure sustainable compliance rather than post-hoc enforcement. This motivates the second research question:

Research Question 2 *How can privacy-by-design be guaranteed for FL tasks in those standardized and sovereign environments?*

Third, legal regulations such as the GDPR not only impose requirements for individual data protection, but also influence the underlying system requirements, documentation and data governance mechanisms. An open research problem is the transparent translation of GDPR principles into technical system design and implementation patterns for cross-enterprise environments. These principles include identifying and documenting a clear lawful basis for data processing and implementing required rights, such as the right to be forgotten. In addition, they require enforcing data minimization in all data collection, storage and processing activities. They also demand accountability through mechanisms that enable auditing, monitoring and tracking of responsibility. Finally, they mandate purpose limitation so that personal data is used strictly in line with specified, explicit and legitimate purposes. This leads to the third research question:

Research Question 3 *What influence can GDPR have on such modern frameworks? Is it possible to make the provided framework GDPR compliant and what is required to ensure such compliance?*

Finally, there is no holistic framework that simultaneously addresses all mentioned aspects: distributed data environment, privacy and sovereignty support, regulatory compliance as well as integrated ML capabilities. Bridging these dimensions requires a cross-disciplinary solution that combines technical design, organizational governance and legal abstraction. Therefore, the central objective of this dissertation is to design and implement an integrated framework for federated, GDPR-compliant, and privacy-preserving ML across enterprises, which will be addressed in the last research question for this work:

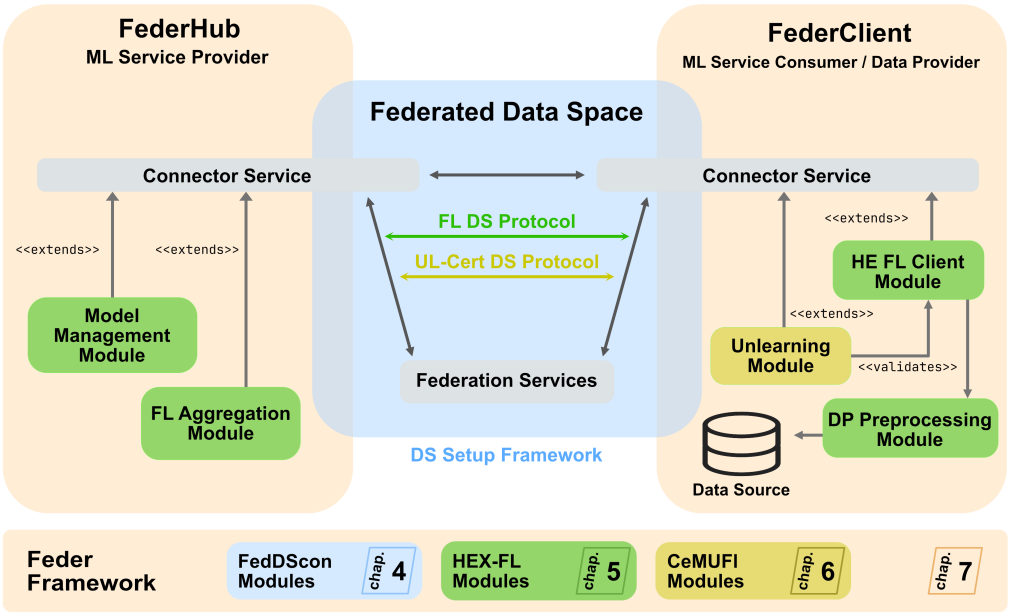


Figure 1.1: Overview of the different developments for GDPR-conform privacy-preserving machine learning in Federated Data Spaces. Each contribution associated with the research questions is illustrated in different colors and referenced in the respective chapters (see chap.).

Research Question 4 *How can a holistic framework be developed that integrates all parts of cross-enterprise business process chains, GDPR regulations and advanced ML methodologies?*

Through these research questions, this dissertation aims to contribute to the ongoing transformation of digital business ecosystems. The results intend to provide conceptual, architectural and technological foundations for trustworthy, interoperable, and regulation-compliant Federated Learning environments, ultimately enabling sovereign and privacy-enhanced algorithmic collaboration in modern data ecosystems.

1.3 Contribution

This dissertation presents several key contributions that collectively address the four research questions formulated in the previous section. An overview of these contributions and their inter-dependencies is illustrated in Figure 1.1, which depicts the relationship between individual contributions and their integration into the overall research framework for GDPR-conform, PPML in federated data environments.

The first major contribution of this dissertation is an extensive evaluation of existing *Data Space* technologies with respect to their feature sets, usability and production readi-

ness. Based on a comprehensive analysis and several prototypical deployments, a practical framework named Federated Data Space Connector (FedDScon) was developed by Sachweh et al., 2025b. FedDScon serves as an easy-to-use infrastructure framework that enables the rapid setup and configuration of federated Data Spaces in modern environments. It provides the essential building blocks for secure data exchange and acts as the foundational layer upon which all subsequent developments of this thesis are built. A detailed description of this contribution can be found in Chapter 4, which directly addresses **Research Question 1**. The corresponding modules are depicted blue in Figure 1.1.

The second and most substantial contribution of this work relates to the design and implementation of two algorithmic stacks for privacy-preserving distributed ML. These address **Research Question 2** and form the technical core of the privacy-preserving mechanisms. The first algorithm integrates DP into the local training process by injecting controlled noise into the data prior to model optimization. This task is handled by the *DP Preprocessing Module* shown in green. The approach ensures individual-level privacy within decentralized learning without significantly compromising model accuracy. The second algorithm introduces a *Homomorphic Encryption*-based extension to the standard Federated Averaging procedure. It enables the encryption of model parameters such that weight updates remain inaccessible to intermediaries and can only be decrypted after secure aggregation across multiple participating clients. Both algorithms are implemented within the Homomorphic Encryption Extended Federated Learning (HEX-FL) framework, highlighted with green in Figure 1.1. Together, they constitute the core PPML component that enables confidential model training in cross-enterprise settings, which is detailed in Chapter 5.

While ensuring privacy through cryptographic and DP techniques provides strong protection guaranties, full GDPR conformity additionally requires mechanisms to enforce the *right to be forgotten*. To meet this regulatory necessity, this dissertation introduces a certified unlearning protocol integrated into the federated environment. The corresponding *UL-Cert DS Protocol* and the Unlearning Module shown in Figure 1.1 (highlighted in yellow) constitute the third main contribution. Both combined provide a certified approach to enforce the deletion of private data from all participating clients within the network, addressing **Research Question 3**. By combining distributed unlearning with hash-based verification, *UL-Cert DS Protocol* and the *Unlearning Module* ensure transparency, regulatory compliance and practical applicability. The contributions resulting in the Certified Machine Unlearning for Federated Learning (CeMUFl) framework are detailed in Chapter 6.

Building on the three main contributions, the final contribution of this dissertation integrates all developed components into a unified framework that addresses **Research Question 4**. This comprehensive framework, called *Feder*, combines the infrastructural capabilities of FedDScon, the privacy-preserving mechanisms of HEX-FL and the certified module CeMUFl. Therefore, *Feder* contains all components depicted in Figure 1.1 and provides a modular and easy-to-deploy architecture for GDPR-compliant FL in cross-enterprise environments, which is detailed in Chapter 7.

Furthermore, *Feder* has been used and tested in the Gaia-X 4 ROMS project (see Section 3.2), a real-world initiative focused on developing Gaia-X-based ecosystems for mobility services and automated vehicle fleets. This production integration confirms its usability in running Data Spaces, fulfilling the requirements outlined in FedDScon for standardized, sovereign and federated environments. The resulting framework represents the complete synthesis of the research outcomes presented in this thesis, enabling secure, adaptable and legally compliant federated ML across organizational boundaries.

1.4 Outline

The following section provides a structured overview of the dissertation and introduces the logical composition of the research work. The dissertation is organized into eight chapters, each addressing specific aspects of the overarching research objective to enable privacy-preserving and GDPR-conform Federated Learning in cross-enterprise environments. The subsequent outline briefly summarizes the content and objectives of each chapter, clarifying how the individual components collectively contribute to the comprehensive research framework developed in this work.

Chapter 2: Related Work

This chapter introduces the theoretical foundations for the subsequent chapters. It reviews the state of the art in *Federated Data Spaces*, connector technologies and relevant GDPR requirements. Furthermore, it outlines privacy-enhancing methods such as DP and cryptographic protection in the context of FL.

Chapter 3: Research Methodology

Chapter 3 outlines the research design and methodological framework employed throughout the dissertation. It details the systematic procedure adopted to ensure scientific rigor and reproducibility, integrating conceptual modeling, prototype implementation and empirical evaluation. The practical relevance of the research is demonstrated through a representative case study illustrating real-world cross-enterprise collaboration scenarios. In addition, several smaller sample datasets are described, which constitute the experimental foundation for the evaluation work presented in subsequent chapters. Together, these methodological elements provide the structural basis for the empirical validation of the frameworks and algorithms developed.

Chapter 4: Federated Data Spaces

This chapter introduces the development of the FedDScon framework, the main contribution of this section. FedDScon provides essential building blocks for the establishment of Data Spaces in modern infrastructures in an efficient and user-friendly manner. To design

this framework, a foundational analysis of existing *Data Space* technologies is conducted alongside prototypical setups of various architectures. Based on these findings, FedDScon is implemented to enable straightforward creation and management of federated Data Space infrastructures. This chapter is based on the following publications:

- Kremer, M., Pohling, L., Gösling, H., Heinbach, C., **Sachweh, T.**, Gogineni, S., & Berger, K. (2023). An Intelligent Arrival Time Prediction Service in a Federated Data Ecosystem: The Minimum Viable Demonstrator of the GAIA-X 4 ROMS Research Project. SSRN Electronic Journal.
- Maecker, D., Gösling, H., & **Sachweh, T.** (2024). Setting up a ROS2-based Multi-Agent System implementing the Contract net Protocol and IDS Connectors. In Engineering Multi-Agent Systems. Springer Nature Switzerland.
- Maecker, D., Harenbrock, F., Gösling, H., **Sachweh, T.**, & Thomas, O. (2024). Synergizing Trust and Autonomy: Gaia-X Enabled Multi-Agent Ecosystems for Advanced Freight Fleet Management. In Engineering Multi-Agent Systems. Springer Nature Switzerland.
- **Sachweh, T.**, Kuhlmann, H., Gösling, H., & Liebig, T. (2025b). Federated Data Spaces as an Enabler for Smart City Services - Logistics Use-Case Example. 2025 IEEE European Technology and Engineering Management Summit (E-TEMS). **(best paper award)**.

Chapter 5: Privacy-Preserved Federated Learning in Cross Enterprises

Chapter 5 focuses on the conceptualization and implementation of advanced techniques for privacy-preserving Federated Learning in cross-enterprise settings. Two methodological directions are investigated: Distributed Learning from Label Proportions (LLP), extended through *DP* to ensure protection of sensitive information at the model level and a HEX-FL, which enables secure model exchange in untrusted environments. The combination of these techniques results in a comprehensive privacy-preserving FL architecture that safeguards data confidentiality throughout all stages of model training and aggregation. This chapter is based on the following publications:

- **Sachweh, T.**, Boiar, D., & Liebig, T. (2021). Differentially Private Learning from Label Proportions. In Communications in Computer and Information Science. Springer International Publishing.
- **Sachweh, T.**, Boiar, D., & Liebig, T. (2022). Distributed LSTM-Learning from Differentially Private Label Proportions. 2022 Institute of Electrical and Electronics Engineers (IEEE) International Conference on Data Mining Workshops (ICDMW).
- **Sachweh, T.**, Kuhlmann, H., & Liebig, T. (2023). Empowering Data Owners with Homomorphic Encrypted Federated Learning in Decentralized Data Spaces. International Symposium on Location-Based Big Data and GeoAI 2023 (LocBigDataAI 2023).

Chapter 6: Federated Unlearning with Data Spaces

In accordance with the GDPR's provisions on the right to be forgotten, this chapter extends the privacy protection mechanisms introduced earlier by introducing Federated Unlearning capabilities within Data Space environments. It presents a formal unlearning algorithm and a certification protocol that together enable the verifiable removal of data from distributed learning models. The proposed hash-based certification approach ensures that each participating client can cryptographically attest to compliance, thereby providing an auditable and transparent mechanism that confirms the permanent deletion of data across all clients within the cross-enterprise network.

Chapter 7: GDPR-Conform Federated Learning in Cross Enterprises

Building on the contributions of the preceding chapters, Chapter 7 integrates the developed mechanisms into a holistic and GDPR-conform FL framework, as conceptually illustrated in Figure 1.1. It specifies the modular interfaces and interdependencies between components, including the Federated Data Space foundation, the HEX-FL framework and the certified unlearning mechanism. The resulting system constitutes a comprehensive, privacy-preserving and compliant FL architecture tailored to collaborative data ecosystems in industrial and cross-organizational contexts. This chapter is based on the following publication:

- **Sachweh, T.**, Kuhlmann, H., & Liebig, T. (2025). Feder: Privacy-Preserving Federated Learning Across Enterprises. Trustworthy Artificial Intelligence (AI) Summit.

Chapter 8: Summary and Outlook

The concluding chapter summarizes the main findings and contributions of this dissertation and reflects on how they address the formulated research objectives and challenges. It discusses the implications of the proposed frameworks for PPML in cross-enterprise environments. Finally, it outlines promising directions for future research, including methodological extensions and large-scale practical deployments.

Additional Publications

The author additionally contributed to other peer-reviewed publications that do not directly contribute to findings of this work.

- Gösling, H., Maecker, D., Pieper, T., **Sachweh, T.**, & Heinbach, C. (2024). A Procedure for Conceptualizing and Implementing Spade Agents. In Engineering Multi-Agent Systems. Springer Nature Switzerland.
- **Sachweh, T.**, Haritz, P., & Liebig, T. (2024). Using Petri Nets as an Integrated Constraint Mechanism for Reinforcement Learning Tasks. 2024 IEEE Intelligent Vehicles Symposium (IV).

2 Related Work

This chapter provides the foundational background for this dissertation, which investigates GDPR-compliant, PPML in cross-enterprise environments. It surveys the current state of the art key concepts, technologies and regulatory frameworks that support secure, sovereign and interoperable data exchange across organizational boundaries. Besides the data exchange, it also details current privacy and ML methodologies for distributed ML applications, which will be used in the following chapters as foundation for novel technologies.

The chapter begins with an overview of Federated Data Spaces, examining their principles, architectures and ecosystem initiatives that enable controlled and trustworthy data sharing. It then considers connector technologies, such as the Industrial Data Space Connector (IDS) and the EDC, which operationalize standardized interfaces and policy-driven mechanisms for secure data exchange. Next, it outlines the GDPR, detailing the legal requirements and compliance obligations that shape both the design of Federated Data Spaces and the deployment of privacy-preserving analytics. Finally, this chapter addresses privacy-preserving technologies, including DP and HE, which are central to enabling ML on distributed data while protecting sensitive information which is required by governmental regulations. Together, these sections provide the necessary background knowledge and context for the research contributions presented in the following chapters.

2.1 Federated Data Spaces

Federated Data Spaces are emerging as a foundational paradigm for enabling secure, sovereign and interoperable data exchange across organizational boundaries. Unlike traditional centralized data-sharing models, Federated Data Spaces prioritize the autonomy of data owners, allowing them to retain control over their data while participating in collaborative data ecosystems. This approach is particularly pertinent in sectors such as healthcare, finance as well as manufacturing, where data sensitivity and regulatory compliance are crucial. This control can be guaranteed by design of the Data Spaces, which specifically feature the following typical criteria.

Data Sovereignty and Control This is a defining feature of Federated Data Spaces, where data owners retain control over their data at all times, including the ability to define usage policies, restrict access and enforce conditions even after sharing. Traditional data exchange methods, such as centralized databases, Application Programming Interfaces (APIs), or ETL-based pipelines, typically require transferring full copies of data to a central

repository, at which point the original owner loses granular control over how the data are used. In contrast, Federated Data Spaces leverage connector-based architectures that allow data to remain at the source while enabling controlled access for processing or analysis.

Policy-Driven and Trust-Based Sharing Federated Data Spaces handle policy-driven and trust-based sharing through formalized usage policies and trust frameworks integrated directly into the exchange process. Standards like the Dataspace Protocol and policy languages such as Open Digital Rights Language (ODRL) allow data owners to specify permissions, prohibitions and obligations programmatically. Typical data exchanges often rely on informal contracts or coarse-grained access control, which cannot dynamically enforce data usage conditions across organizational boundaries. Federated Data Spaces therefore provide automated policy enforcement, enhancing accountability and compliance.

Decentralization and Federation Unlike centralized exchange platforms or cloud-based data lakes, Federated Data Spaces adopt a federated architecture: data is distributed across independent organizations and connectors mediate access rather than aggregating data centrally. This reduces data duplication, mitigates privacy risks and supports compliance with regulations such as GDPR by limiting unnecessary data movement.

Compliance-Ready Frameworks Federated Data Spaces explicitly integrate regulatory compliance into their design. GDPR-aligned mechanisms, audit trails and consent management are considered from the very beginning. Typical data exchanges often implement compliance as an afterthought, leading to higher legal and operational risk.

Federated Data Spaces as such are primarily driven by two initiatives, which both address this criteria, but from different perspectives. First, the Industrial Data Spaces Association and second, all research projects around Gaia-X.

2.1.1 Industrial Data Spaces

The International Data Spaces Association (IDSA) is a non-profit organization that develops standards, reference architectures and certification mechanisms for secure and sovereign data exchange. The association was created through research projects majorly leaded by Fraunhofer institutes. To this day they have more than 180 company members and built up a big partner network in over 30 countries. IDSA's core mission is to operationalize the principles of data sovereignty, ensuring that data owners maintain control over their assets even when sharing them across organizational boundaries. IDSA provides the IDS-Reference Architecture Model (RAM), which defines the structural components of a Federated Data Space from IDSA's perspective (Otto et al., 2019). IDS-RAM

includes the concept of connectors, metadata registries and usage policies. It also establishes the Dataspace Protocol, which specifies standardized communication patterns, trust frameworks and policy enforcement mechanisms for data exchange. By combining technical specifications with governance guidelines, IDSA enables organizations to implement interoperable and auditable Data Spaces that comply with legal and regulatory requirements. Several research studies and projects have analyzed the effectiveness of the IDS framework in industrial and cross-domain settings. For instance, Jung and Dörr, 2022 highlight the role of IDS in enforcing data usage policies, while Usländer and Teuscher, 2022 discuss the applicability of IDS principles for distributed industrial use cases in scenarios like Industry 4.0. IDSA also maintains a certification program to ensure that connectors and services conform to its standards, fostering trust among participants. Based on the IDSA developments and the IDS-RAM technology stack, several Data Spaces were created, which are used in different domains to exchange data across company borders. One example is the Mobility Dataspace, which uses the developments from IDS to exchange data in the mobility domain between stakeholders in this sector (Pretzsch et al., 2022).

In general one can say that the IDSA is an association that developed a well defined software stack, which addresses a lot of european regulatory requirements. The IDS-RAM serves as software stack for simple adoption to use-case specific implementations. Important to notice is, that IDS-RAM always focuses on a domain driven Federated Data Space. That means, that setting up a Data Space with IDS-RAM creates a Space, where only domain related data will be exchanged between parties, that act in this domain. That is the reason, why each domain creates its own Federated Data Space: One for mobility data (Mobility Dataspace), one for the logistics sector (DASLOGIS) or one for the energy domain.

2.1.2 Gaia-X

The Gaia-X Association for Data and Cloud (fran.: association internationale sans but lucratif (AISBL)) is a European initiative aimed at establishing a federated, secure and transparent data infrastructure across cloud and IT service providers. Unlike IDSA, which focuses primarily on industrial data-sharing ecosystems, Gaia-X targets broader infrastructure-level interoperability, enabling organizations to exchange data across sectors while adhering to European values of transparency, trust and sovereignty. This initiative was proposed in 2019 and since then, various research projects have been carried out in different European countries with the aim of first designing concepts and standards for Gaia-X and then developing central services for it. In a further phase, starting in 2021, there were follow-up projects that dealt with setting up use-case-specific Data Spaces. One of these projects is GAIA-X 4 ROMS, which dealt with intermodal logistics chains and public transport.

Gaia-X defines a federation services architecture, including components for identity and trust management, service catalogs and data governance. It emphasizes interoperability

standards, compliance with legal frameworks such as GDPR and the creation of a federated data ecosystem that integrates heterogeneous cloud services and infrastructure providers. Gaia-X also promotes data governance rules for federated services, ensuring that providers and consumers can operate under transparent and auditable conditions.

2.1.3 IDSA and Gaia-X comparison

While both IDSA and Gaia-X aim to foster secure, interoperable and sovereign data sharing, there are key differences in their scope, focus and implementation strategies.

IDSA sets their scope to create industrial ecosystems with focus on domain-specific Data Spaces where Gaia-X wants to address a wider range. Gaia-X wants to enable a cross-sectoral federated cloud and data infrastructure, where all participants can collaborate in joined manner. Therefore, Gaia-X has a strong focus on the federated infrastructure, interoperability and compliance to be able to address the broader collaboration setting.

In terms of software components and architecture, both provide central services for interaction and the core functionalities. Gaia-X also includes the infrastructure, as partners in the Gaia-X community are cloud infrastructure providers or the self hostable Sovereign Cloud Stack can be used for deployment infrastructures of business applications (Rone, 2024). The IDSA on the other hand does not cover infrastructure, but they provide a connector approach to adapt business applications to the Data Space (see Section 2.2). The developments, variations and general concept of the connector is described in the following section.

In terms of governance, compliance and sovereignty both initiatives provide standards and solutions to enable these features in the Data Spaces.

In summary, IDSA provides the technical and governance building blocks for domain-specific Data Spaces with a strong focus on data sovereignty, while Gaia-X establishes a federated infrastructure layer that enables broader interoperability and compliance across sectors. By combining IDSA concepts with Gaia-X principles, organizations can leverage both technologies to build a scalable, interoperable infrastructure that ensures GDPR-compliant data handling. IDSA technology components can operate within Gaia-X's federated ecosystem to enforce data usage policies and sovereignty, enabling secure and sovereign data exchange even between parties that may not fully trust each other.

2.2 Connector Technology

The IDS-RAM model, as described in Section 2.1.1, provides the foundation for all modern Data Space technologies. Building on this model, the concept of establishing connector technologies for data exchange emerged. Connector technologies form the technological

backbone of modern Federated Data Spaces, enabling secure, policy-driven and interoperable data exchange between independent participants. This is made possible by deploying a connector at each party. The connector acts as an interface between the business applications and the Data Space services. Therefore, the connector abstracts away the complexity of the Data Space for the company and allows more easy interaction with Data Spaces. They also serve as the trusted interface between a data provider and a data consumer, ensuring that data transfers comply with technical, organizational and legal constraints defined by the data owner. Within the European context, two major connector frameworks have evolved over time: the Industrial Data Space Connector (IDS Connector), developed by the IDSA, and its modern successor, the EDC. Together, they embody the shift from conceptual architectures toward scalable, open-source implementations of data sovereignty.

2.2.1 Industrial Data Spaces Connector

The Industrial Data Space Connector (IDS Connector) originated as a key component of the IDS initiative, later institutionalized under the IDSA. Its core function is to facilitate secure and sovereign data exchange between organizations without requiring centralized storage or transfer intermediaries (Otto et al., 2018; Wrona et al., 2020). The IDS Connector acts as a trusted gateway within a federated data ecosystem. It provides three essential capabilities:

- **Secure Communication:** By employing standardized protocols (e.g. HTTPS, OAuth2 and Dataspace Message Protocol), it ensures confidentiality and integrity during data transfer.
- **Usage Control and Policy Enforcement:** The connector embeds usage policies defined using formal policy languages such as ODRL (De Vos et al., 2019). These policies defined by ODRL govern how shared data can be processed, stored or redistributed.
- **Trust and Certification:** The connector operates within a certified trust framework, verifying the identity, security posture and compliance of participants before enabling data exchange.

Through these mechanisms, the IDS Connector established a sovereign data exchange model in which data remain under the control of its owner while being accessible to authorized partners under explicit contractual and technical guarantees. Early implementations, such as the Fraunhofer IDSA Reference Connector, demonstrated the applicability of this approach in industrial supply chains, predictive maintenance and cross-company analytics (Otto and Jarke, 2019; Sang et al., 2021). However, the IDS Connector does not support any extension of new features. Consequently, all required capabilities for every Data Space scenario must be embedded directly within the connector itself, leading to a complex, monolithic service that restricts scalability and hinders seamless integration into modern cloud or edge infrastructures. This challenge motivated the evolution

toward a more modular, open and cloud-native architecture called Eclipse Data Space Connector.

2.2.2 Eclipse Data Space Connector

The EDC emerged as an open-source evolution of the IDS Connector, designed to align more closely with modern distributed computing paradigms and the GAIA-X federated data infrastructure initiative. The EDC project is hosted under the Eclipse Foundation and jointly developed by industry and research partners such as Microsoft, Fraunhofer ISST, BMW Group and Deutsche Telekom.

The EDC reinterprets the IDS concepts of trust, sovereignty and interoperability within a cloud-native and modular architecture. Its primary objectives are to simplify deployment, improve extensibility and provide interoperability across heterogeneous platforms and Data Spaces. Compared to the IDS Connector, which focused heavily on the enforcement of static usage control policies, the EDC emphasizes flexible, pluggable components and protocol abstraction, allowing it to operate across multiple governance models and technology stacks.

The EDC implements the Dataspace Protocol, a standard jointly developed by the IDSA, the AISBL and the Eclipse Foundation (Koen et al., 2025). The Dataspace Protocol (DSP) defines the technical interaction model for negotiation, agreement and execution of data exchange between connectors in different federations. This enables interoperability across independent data spaces while maintaining the principles of data sovereignty.

The architecture of the EDC consists of the following layers:

- **Control Plane:** Handles contract negotiation, authentication and policy validation between participants.
- **Data Plane:** Executes the actual data transfer using secure communication channels, separated from control logic to enhance performance and compliance.
- **Extensions Layer:** Provides hooks for integrating external identity providers, catalog services and trust frameworks.

As the EDC was built from the ground up based on the IDS Connector experience, it addresses a lot of the issues that the IDS Connector had. Thus the EDC is much more lightweight and allows more simple extensibility through the extensions layer which allows integration of custom libraries to expand the capabilities of the connector to custom requirements. One drawback is the lack of an app delivery feature, that the IDS Connector had. It follows that the code-to-data principle can only be implemented via the extension of the EDC libraries which is currently not the case. Through its adherence to open standards and extensibility, the EDC provides a technical foundation for GDPR-compliant Federated Data Spaces capable of supporting advanced analytical applications.

2.3 General Data Protection Regulation

Federated Data Spaces as described previously, provide an architectural paradigm enabling secure and sovereign data exchange across organizational boundaries. Frameworks such as the EDC exemplify the technological foundation for trusted interoperability, where participants retain control over their data while facilitating cross-domain collaboration. These infrastructures are particularly relevant for data-driven innovation, such as ML and analytics across enterprises or sectors. Although Federated Data Spaces provide the technical mechanisms for controlled data exchange, their design and operation must be guided by legal and ethical frameworks that define what constitutes responsible and lawful data handling.

The GDPR, established by the European Union, serves as the primary regulatory foundation for ensuring that data sovereignty, individual privacy and accountability are preserved throughout these exchanges. Understanding the GDPR is therefore essential for aligning federated data infrastructures and ML applications with European data protection requirements. This section discusses the core privacy principles of the GDPR, with a particular focus on the right to be forgotten and explores their implications for ML, especially in federated and cross-enterprise settings.

2.3.1 Core Privacy Principles

The GDPR defines a set of principles that govern all personal data processing activities (European Parliament and Council of the European Union, 2016). These principles outlined in Article 5 ensure that data processing respects the rights and freedoms of individuals while promoting accountability among data controllers and processors.

- **Lawfulness, Fairness and Transparency:** Data must be processed lawfully and transparently. Individuals must be informed of how their data is collected and used, ensuring accountability of data controllers (Tikkinen-Piri et al., 2018).
- **Purpose Limitation:** Data should only be collected for specified, explicit and legitimate purposes as well as not further processed in a manner incompatible with those purposes.
- **Data Minimization:** Only data necessary for the specified purposes should be collected and retained.
- **Accuracy:** Data must be accurate and kept up to date, while inaccurate data must be erased or rectified without delay.
- **Storage Limitation:** Personal data should only be stored as long as necessary for the intended purpose.
- **Integrity and Confidentiality**: Data must be processed in a manner that ensures appropriate security, including protection against unauthorized or unlawful processing.

- **Accountability:** Data controllers are responsible for demonstrating compliance with the above principles (Voigt and Von dem Bussche, 2017).

These principles operationalize the idea of privacy by design and by default (Cavoukian et al., 2009), urging organizations to integrate privacy protection mechanisms into the architecture of their systems from the outset. In federated ML contexts, these requirements translate into designing systems that minimize central data aggregation and enhance transparency through auditable data flows.

2.3.2 The Right to Be Forgotten

One of the most prominent rights introduced by the GDPR is the right to erasure, commonly known as the right to be forgotten (Article 17). This right empowers individuals to request the deletion of their personal data once it is no longer needed, consent has been withdrawn, or when the data has been unlawfully processed (Kamarinou et al., 2016).

While conceptually straightforward, this right poses significant technical challenges in ML. Once a model has been trained on personal data, removing that data's influence may not be feasible without retraining from scratch. This challenge has led to the emergence of research fields such as machine unlearning, which aims to remove the effects of specific training samples from learned models (Bourtoule et al., 2021).

In distributed or federated contexts, where model updates rather than raw data are shared, ensuring the ability to "forget" contributions becomes even more complex. Techniques such as federated unlearning and privacy-preserving model aggregation have been proposed to reconcile these rights with the operational realities of federated systems (Wu et al., 2022).

2.3.3 Sensitive Data and the Protection of Special Categories

The GDPR designates special categories of personal data, including information about health, ethnicity, political beliefs, or biometric identifiers as sensitive data (Article 9). Processing such data is generally prohibited unless explicit consent is obtained, the data is required for reasons of public interest, or it is used for scientific research under strict safequards.

From a ML perspective, handling sensitive data introduces heightened risk of privacy violations, discrimination, or re-identification (Shokri et al., 2017a). Even in federated systems, where raw data remains local, model updates can inadvertently leak information about sensitive attributes through inference attacks (Melis et al., 2019). Consequently, compliance with GDPR requires not only organizational controls but also technical mechanisms such as DP, secure multiparty computation and HE to ensure robust protection (Truex et al., 2021).

2.3.4 Implications for Machine Learning and Cross-Enterprise Environments

The GDPR's emphasis on privacy and data sovereignty significantly influences the design of ML pipelines. Traditional centralized learning paradigms that rely on large-scale data collection and aggregation are increasingly viewed as incompatible with the data minimization and purpose limitation principles. FL has emerged as a promising paradigm that aligns more closely with GDPR requirements (Kairouz et al., 2021). By enabling model training across distributed nodes without transferring raw data, FL promotes local data control, reduced data exposure and enhanced compliance with privacy regulations. Nevertheless, challenges remain:

- **Data traceability and consent management:** Determining whether model updates contain personally identifiable information remains difficult.
- **Right to be forgotten in federated ML settings:** Once a global model is trained using distributed contributions, removing an individual's data retroactively is nontrivial.
- **Cross-enterprise communication:** Collaborative learning across organizational boundaries must respect differing privacy policies, data retention periods and jurisdictional constraints.

Addressing these issues demands privacy-preserving ML techniques including DP, encryption and secure aggregation to provide measurable guarantees of compliance and trust in cross-enterprise environments. Those techniques are integrated into the contributions of this thesis in Chapter 5, where the privacy-preserved FL framework HEX-FL is described.

Beyond its technical implications, the GDPR serves as a global benchmark for data protection and AI governance (Taddeo and Floridi, 2018). It has inspired similar regulatory efforts worldwide, such as the California Consumer Privacy Act (CCPA) and Brazil's Lei Geral de Proteção de Dados (LGPD), contributing to the harmonization of digital ethics standards. For FL and cross-enterprise collaboration, GDPR compliance fosters interorganizational trust, ensuring that participants can engage in data exchange without compromising individual privacy. This regulatory alignment is particularly critical in sensitive sectors such as healthcare, finance and public administration, where data-driven innovation must coexist with stringent privacy protections.

The GDPR establishes a comprehensive framework for lawful, fair and transparent data processing, emphasizing accountability, data minimization and user empowerment through rights such as data portability and erasure. These principles present both constraints and opportunities for the design of ML systems.

In federated and cross-enterprise contexts, compliance with GDPR not only ensures legal conformity, but also drives technological innovation in privacy-enhancing computation. The next section explores such privacy-preserving technologies, including DP and HE which operationalize GDPR principles in data-intensive ML environments.

2.4 Data Privacy

The regulations mentioned in the previous section have especially encouraged the adoption of innovative privacy-preserving techniques in the field of ML, where large volumes of data are processed and analyzed. When focussing on ML applications there are currently two noteworthy approaches present that try to secure personal data with different approaches. One method is called DP and the other HE. These methods provide robust solutions for addressing privacy concerns while enabling the continued advancement of ML applications. Both methods will be used in this work for enhancing privacy in a joined manner and are therefore introduced in the following.

2.4.1 Differential Privacy

DP is more a formal mathematical framework than a specific library or implementation. This framework was introduced by Dwork, 2006 and gives a metric to quantify and limit the privacy risk, which arises from statistical or ML outputs. The main idea is to ensure that the inclusion of any single individual's data has a negligible effect on the outcome of any computation, thereby providing measurable privacy guarantees. Formally DP is defined as follows:

An algorithm A satisfies (ϵ, δ)-DP if, for any two neighboring datasets D_1 and D_2 differing by one record and for any possible output S the following inequality holds:

$$Pr[A(D_1) \in S] \leq e^{\epsilon} * Pr[A(D_2) \in S] + \delta \tag{2.1}$$

In this formula, ϵ represents the privacy loss parameter and δ defines the probability of privacy leakage beyond the ϵ limitation. Smaller values of ϵ and δ indicate stronger privacy protection since the probability distributions over the output set S must be more similar in order to satisfy the inequality.

This is the formal definition of DP as invented by Dwork, 2006, which defines the privacy criteria. However, there exist different approaches to accomplish DP by definition. One is Local Differential Privacy (LDP), where noise is added directly on the client side before data is shared, ensuring that raw data are never exposed to the public. In contrast to LDP, where data is secure whenever it leaves the company boundaries, there exists the approach of Global Differential Privacy. In this setup, a trusted aggregator adds noise to aggregated outputs, such as model gradients or query results. This approach requires a trusted aggregator as the DP guarantee can only be hold after adding the noise from the trusted aggregator. Besides these solutions, which are loosely coupled to ML algorithms, there are also approaches available, that combine the DP guaranty directly into the ML models by, for example, clipping gradients or adding noise during backpropagation. This method is very well shown in the work from Song et al., 2013 where typical backpropagation steps are modified to include DP guaranties. Detailed implementations and how to adapt them to the framework proposed in this work will be given in the corresponding chapters.

While DP offers strong theoretical guaranties, its privacy trade-off remains challenging in practice due to the noise-budget. Adding excessive noise can significantly degrade model accuracy, while insufficient noise weakens privacy protection. Moreover, determining an appropriate privacy budget for complex ML tasks is non-trivial and domain-dependent (Jagielski et al., 2019).

Besides these difficulties, it is, however, a simple solution which gets recognized as regulatory compliance enabler, providing measurable assurance aligned with GDPR's requirement for data protection. With the noise-budget it is also configurable such that for high demanding data protection applications, the privacy factor ϵ can be set to increased privacy, so that the data are more secure.

2.4.2 Homomorphic Encryption

Besides DP, there are also standard security methods like encryption available. The main problem with encryption is that usually only data transfer is encrypted, where the data provider has to encrypt the data and the receiver needs to decrypt them before doing any operation on it. This raises the issue that the receiver needs to be a trusted party and has the right to process the original data.

HE is part of these typical encryption methods. However, it addresses the problem as it uses a cryptographic technique that allows computations to be performed directly on encrypted data without decrypting them first (Gentry, 2009). This property enables collaborative analytics and ML across untrusted environments while preserving data confidentiality. In HE schemes, given a plaintext message m, encryption $E(m)$ allows computation such that:

$$E(m_1) \oplus E(m_2) = E(m_1 \oplus m_2) \tag{2.2}$$

In this equation $\oplus$ denotes an arithmetic or logical operation (e.g. addition or multiplication). The result, once decrypted, matches the computation as if it was performed on the plaintexts directly.

There exist three stages of HE that deliver different features. First, there are the Partially Homomorphic Encryption (PHE) (Ryu et al., 2023), which support addition or multiplication operations. Typical encryption algorithms that fall into this category are known schemes such as Paillier (Fazio et al., 2017; Paillier, 1999) or RSA (Rivest et al., 1978). These schemes are very limited, as they often only allow for a single type of operation. As second stage, there exist the so called Somewhat Homomorphic Encryption (SHE), which allow multiple operations on encrypted data. However, they are limited in the amount of operations that they support. After a limited number of operations before decryption, the amount of noise overwhelms the ciphertext. This issue is addressed by the last stage of HE algorithms, which is called Fully Homomorphic Encryption (FHE) (Yousuf et al., 2021). Those allow for an arbitrary number of computations on ciphertexts, while remaining computationally expensive. Modern FHE schemes, such as Cheon-Kim-Kim-Song (CKKS) (Cheon et al., 2017) or the schemes of Brakerski et al. (Brakerski, 2012;

Brakerski and Vaikuntanathan, 2011, 2014; Brakerski et al., 2014), use a method called Ring Learning With Errors (RLWE) (Lyubashevsky et al., 2010) to compute ciphertexts and execute operations on them.

The development from PHE to SHE to FHE is due to developments over time, resulting in more and more possibilities being developed for operations on ciphertexts. However, Federated Data Spaces can only benefit from FHE and its privacy guarantees if the schemes can be used in a decentralized setting. Since it is obvious that FHE is used in distributed scenarios, secure multiparty computation (Secure Multi-Party Computation (MPC)) (Damgård et al., 2012) was also integrated into common FHE schemes relatively quickly, so that client-wide operations can also be performed on encrypted data.

For FL tasks in Federated Data Spaces, FHE in combination with MPC can be a solution to maintain sovereignty over data, ML models, or model updates. However, these encryption schemes are known to be computationally demanding, often increasing computation time and memory consumption by several orders of magnitude.

2.5 Federated Learning

Besides current advancements in cross-enterprise network infrastructures like Federated Data Spaces and requirements from GDPR, which result in a focus of data privacy with technologies like DP or HE, the development of ML models also changes. More and more companies are changing from collecting all data in a data lake with central ML training to more distributed ML training. The focus changes to built software, which uses the paradigm of code-to-data, where code is executed on the edge of the network. FL (Silva et al., 2023; Yurdem et al., 2024; Zhang et al., 2021) is a concept that is well suited to support this development.

FL is a distributed machine learning approach that enables multiple entities, often called clients, to collaboratively train a shared model while keeping their local data private. Unlike traditional centralized models, where data is pooled in one location, FL ensures that raw data remain on client devices. Instead, clients only share model updates or parameters with a coordinating entity to iteratively improve a global model. This approach helps to preserve data privacy and reduces risks related to data breaches, making it particularly suitable for sensitive domains.

There are several common FL architectures and strategies, each tailored to different application scenarios and system constraints. Centralized FL is the most conventional approach in which a central server orchestrates the training process. Clients train local models on their private data and periodically send updates of model parameters to the server. The server aggregates these updates to refine the global model and redistributes it to clients for further local training. Although this approach simplifies coordination and aggregation, the central server can become a bottleneck and single point of failure.

In contrast, with Decentralized FL, there is no central server that coordinates the updates of the model (Q. Li et al., 2025). Instead, clients communicate directly with each other in a peer-to-peer approach, sharing updates and collectively forming the global model. This method enhances fault tolerance and avoids central bottlenecks, but its effectiveness can depend heavily on the network topology of participating clients.

A combination of Centralized and Decentralized FL is called Heterogeneous Federated Learning (Ye et al., 2023). With this approach, clients with different computational capabilities, data distributions and model architectures can train various local models that differ in structure and complexity. Methods such as HeteroFL (Diao et al., 2020) facilitate training under these conditions while still producing a unified global inference model, which is critical for real-world deployments on a variety of devices such as smartphones, IoT sensors or edge devices.

The Cross-Silo Federated Learning (Bao et al., 2023; Huang et al., 2021; K. Liu et al., 2022) method can be seen as the fourth architecture for FL tasks. This approach involves collaboration between a limited number of reliable organizations or data silos, where each participant typically has a large dataset and a stable network connection. Such a setup is common in inter-organizational collaborations such as multi-hospital research or financial fraud detection where data privacy, regulatory compliance and organizational agreements are paramount.

All these FL methodologies commonly claim to have higher privacy guarantees, than sending data to a central instance, which trains on the whole dataset. However, there exists a lot of research which shows how training data can be examined from FL models that are trained in a distributed manner (Dimitrov et al., 2022; Jin et al., 2021; Ren et al., 2022).

These studies show that, even in FL, sensitive information can still be extracted from model updates. As a result, the exchanged data remains vulnerable to exposure, and FL by itself is insufficient to guarantee GDPR-compliant data sharing in complex cross-enterprise networks. Additional privacy-enhancing techniques are therefore required to achieve truly privacy-preserving, legally compliant data exchange.

3 Research Methodology

The previous chapter provided insights into the current state of the art methods for complex software architectures, governmental regulations, ML tasks as well as data privacy and will be used as foundation for the research in this work.

To provide a systematic research approach, this chapter introduces the research methodology that is used to address the research questions of this work. A structured method is required that can simultaneously incorporate practical requirements from real-world application scenarios and draw on expert knowledge, as well as current state-of-the-art research results. Design Science Research (DSR) provides such a methodology as it explicitly links the construction and evaluation of innovative artifacts with both a research base and a concrete application environment.

In the following, DSR is first outlined as the overarching methodological framework in Section 3.1, including the three-cycle model and its specific application to this thesis contributions in Table 3.1. This demonstrates how DSR leverages existing knowledge, integrates stakeholder expertise from the Gaia-X 4 ROMS project and iteratively turns theoretical research into usable artifacts within real-world application domains. Thus, DSR does not provide its own contribution, but the methodological approach for the research and development of all other contributions in this thesis. Subsequently, the use-cases that instantiate the DSR environment are introduced in Sections 3.2 and 3.3, since DSR relies on such contexts to derive relevant requirements and validate artifact utility.

Two complementary use-cases are employed in this work. The first is a real-world scenario derived from the GAIA-X 4 ROMS project, which provides realistic cross-enterprise requirements and constraints from the mobility domain (Section 3.2). The second is a constructed use-case based on synthetic benchmark data, which enables faster and more controlled research iterations by removing dependencies on external partners and operational infrastructures (Section 3.3). Together, these use-cases supply the application contexts required by DSR while also supporting reproducible and efficient evaluation of the developed solutions.

3.1 Design Science Research

DSR is a research methodology that focuses on the creation and evaluation of artifacts designed to solve identified organizational or societal problems. According to Hevner et al., 2004 and Hevner, 2007, the DSR aims to bridge the gap between theoretical research and practical application by producing knowledge that is rigorously grounded in

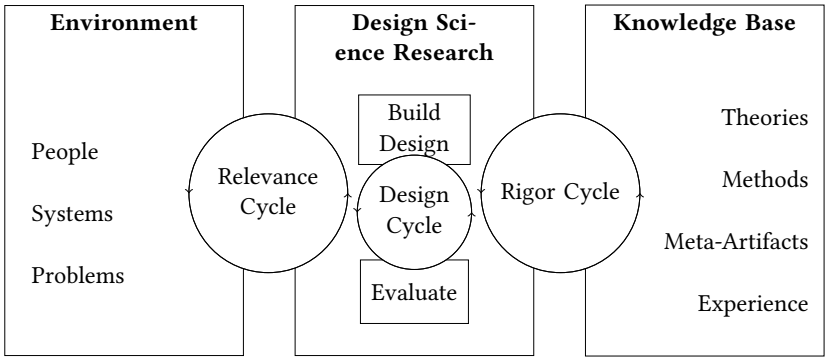


Figure 3.1: Three Cycle View of Design Science in Information Systems Research (derived from Hevner, 2007).

theory and relevant to real-world contexts. The approach emphasizes iterative processes of building, evaluating and refining artifacts, such as models, methods, or systems within their application environments. Therefore, DSR is a powerful tool for investigating complex problems and deriving solutions to those problems, which makes it particularly well suited for the problem statement and the research questions derived in this work. With the iterative process, DSR allows for multiple rounds, where necessary stakeholders always get involved to define new and relevant problems more in detail, which can then be addressed in a new iteration.

3.1.1 Three-cycle View of DSR

Hevner, 2007 formalized this perspective through the three-cycle view of DSR. This view consists of three areas, called Environment, Design Science Research and Knowledge Base, which are shown as rectangles in Figure 3.1. In this formalized definition, the Environment represents people, systems, problems and other aspects, which can be derived from real application environments. Therefore, the Environment area is the representative of the real world for the DSR. The Knowledge Base on the other hand presents the theoretical view. It wraps theoretical ideas, knowledge about typical methods or algorithms and experience from various specialists. The third and final view is the Design Science Research block, which contains everything related to the research development process. This contains the creation of artifacts as well as the evaluation of such artifacts.

In addition to the different views, Hevner, 2007 introduced an iterative research approach by integrating the activities of DSR into three interrelated cycles, which combine the different areas: the Relevance Cycle, the Rigor Cycle and the Design Cycle which are also shown in Figure 3.1.

The Relevance Cycle connects the research to the application environment. It ensures that the research addresses real-world problems and that the artifacts produced are applicable and useful in their intended context. Inputs from the environment, such as requirements,

opportunities and constraints guide the design process while outputs are evaluated within the same context for utility and effectiveness.

The Rigor Cycle links the research to the existing knowledge base, comprising theories, frameworks and empirical findings from prior research. It ensures that the design process is informed by relevant theoretical foundations and that the results contribute back to the scientific body of knowledge. This cycle thus guarantees the scientific validity and generalization of the research outcomes.

Finally, the Design Cycle represents the core of the DSR process, encompassing the iterative activities of building and evaluating artifacts. Insights from both the Relevance and Rigor Cycles continuously inform these iterations, allowing for refinement based on empirical testing and theoretical grounding.

Together, these three cycles establish a comprehensive framework that balances practical relevance, scientific rigor and innovative design. This cyclical interplay ensures that DSR not only produces functional artifacts, but also contributes to the advancement of knowledge within the field. With these features, DSR is a well suited methodology to develop innovative artifacts with state-of-the-art methods that still address all requirements from the application environment. Therefore, this methodology is well suited for research projects that have a higher Technology Readiness Level (TRL) as they have to deliver artifacts fitted to specific environments.

3.1.2 Applying DSR Methodology: Iterative Artifact Design and Evaluation for Federated Data Spaces

The DSR methodology is applied iteratively across all following contributions, with Relevance and Rigor Cycles adapting specifically to each research question's focus. For **Research Question 1 (RQ1)** with FedDScon as contribution, the Relevance Cycle extracts mobility Data Space requirements from Gaia-X 4 ROMS, while the Rigor Cycle draws from state-of-the-art architectures (EDC, IDS, Gaia-X). HEX-FL addressing **Research Question 2 (RQ2)** shifts Relevance to GDPR privacy regulations and distributed ML constraints, with Rigor sourcing DP/HE/FL research. **Research Question 3 (RQ3)** (CeMUFl) centers Relevance on the regulatory context within the European Union (EU) and grounds Rigor in ML unlearning and hash verification research. **Research Question 4 (RQ4)** (*Feder*) focuses the Relevance Cycle on end-user usability requirements from project partners, with the Rigor Cycle integrating the prior three artifacts.

Iterative Design Cycles synthesize these tailored Relevance and Rigor inputs through build, evaluate and refine processes, producing deployable artifacts validated through prototypes and Gaia-X production deployment. This RQ-specific adaptation ensures both contextual relevance and theoretical rigor throughout the federated Data Space research trajectory.

Only through this structured DSR methodology is the final contribution *Feder* achievable as a proven production-ready system for GDPR-compliant Federated Learning in

Contribution (Research Question (RQ))	Relevance Cycle (Environment)	Rigor Cycle (Knowledge Base)
FedDScon (RQ1)	Gaia-X 4 ROMS mobility Data Space requirements	EDC, IDS, Gaia-X architectures
HEX-FL (RQ2)	GDPR regulations, distributed ML constraints	DP/HE/FL research literature
CeMUFl (RQ3)	"Right to be forgotten", stakeholder privacy	ML unlearning (Federated Secret Sharing Unlearning (FedSSU)), hash verification research
Feder (RQ4)	End-user usability from project partners	Integration of prior artifacts

Table 3.1: DSR Relevance and Rigor Cycles per Contribution along with the Research Questions (RQ).

cross-enterprise environments. The systematic interplay of Relevance, Rigor and Design Cycles across all four research questions enables the progression from research theory to a comprehensive, validated framework usable in production that addresses real-world deployment requirements while maintaining scientific rigor.

3.2 Research Project as Case Study: GAIA-X 4 ROMS

As described, the Three Cycle View of Design Science Research provides a structured research based methodology to develop large scale software architectures with a lot of dependencies and can therefore be suited to develop novel frameworks as proposed in this work.

Around the year of 2020 there happened a lot of developments regarding modern data exchange infrastructures. Some running projects specifically focused on the Data Spaces approach and the European Union funded a lot of projects, which should provide federation services for GAIA-X, a new Data Space ecosystem. In 2021 new projects were funded, which were planned to be based on the federation services and had the research goal to evaluate how suited GAIA-X Data Spaces are for different economic domains.

Some of these funded projects were grouped into the Future Mobility Domain project family, called GAIA-X 4 Future Mobility, which can be seen in Figure 3.2. All five projects in this project family had to develop demonstrators in the mobility sector, but with different focuses. They all use the same base Infrastructure Ecosystem (red) and the GAIA-X service stack (green) from previous research projects. However, the Data and Service Ecosystem layer changes for each projects alignment. This work will use the GAIA-X 4 ROMS project as a case study, because it has accompanied the project throughout its duration and used the projects results for requirements to modern cross-enterprise infrastructures.

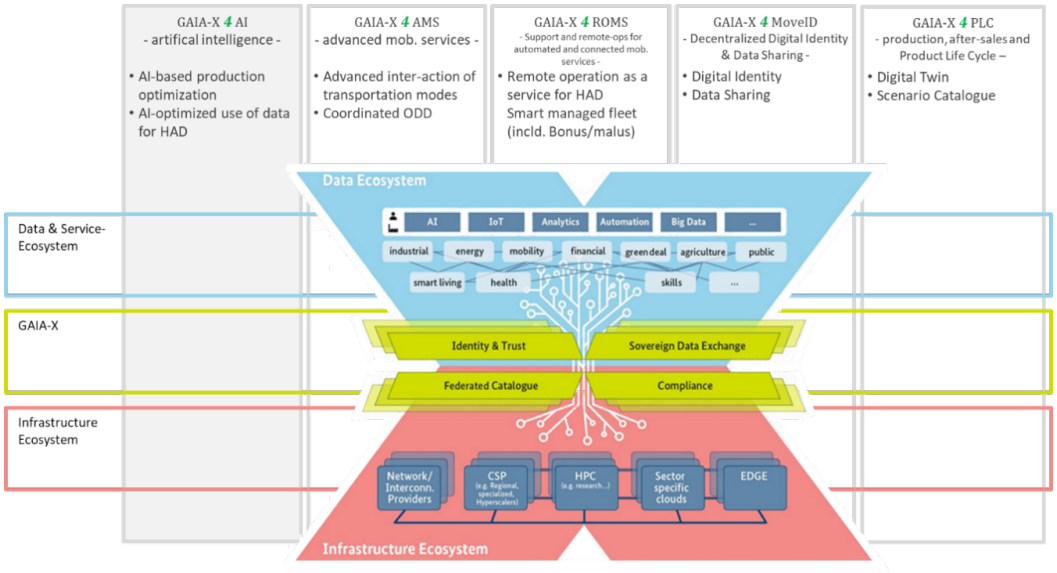


Figure 3.2: Overview of the Gaia-X ecosystem and the research project family in the area of future mobility. Figure used from the research project.

The research project GAIA-X 4 ROMS, which stands for "GAIA-X - Support and Remote-Operation of automated and connected Mobility Services" was funded by the Federal Ministry for Economic Affairs and Climate Action from 2021 to 2025 with a target TRL of 5 to 6. So, the project needed to reach a state in which technology was validated and demonstrated in relevant environments.

GAIA-X 4 ROMS had two use-case objectives. First, public transportation was to be simplified with the help of Data Space technology. Here, customers should be able to select any public transportation options and combine them as they wish. It was important at all times that customers knew transparently where their data was being sent. The Data Space should then dynamically provide the best offer.

As a further use-case, the project set the goal of establishing a multimodal transport chain in the logistics sector, with different companies covering different areas of the transport chain. First- and last-mile delivery was covered by autonomous parcel robots, and the main run was carried out by the project partner Krone.

The research questions for this project were, to test the current possibilities of GAIA-X technology, where other technologies need to be built upon and what limitations currently exist. In addition, innovative solutions to these problems were designed and implemented as part of the project.

With this, GAIA-X 4 ROMS provides a great example use-case for cross-enterprise communication in real world scenarios. It also shows that FL tasks, which, for example, can do estimated time-of-arrival prediction or parcel optimization across company boundaries, are relevant topics in modern architectures that need to be addressed. Additionally,

the project serves as a platform for relevant practitioners and researchers to develop efficiently with the DSR methodology.

Therefore, this research project is used as a case-study for this work, where we can present which component is required for an efficient FL setup in novel cross-enterprise environments with real world relevance. However, this setup is quite challenging for research evaluation due to lack of data for research and difficulties in reproducibility for others.

3.3 Constructed Use-Case Based on Standardized Datasets

Although the real-world case study presented in this work primarily serves to illustrate the industrial and organizational relevance of novel FL frameworks in Data Spaces, it is not suitable for extensive experimental evaluation. To ensure reproducibility, eliminate dependencies on external data providers, and enable controlled experimental setups, this work utilizes standardized benchmark datasets as a constructed use-case for evaluation.

The constructed use-case is based on publicly available datasets such as Canadian Institute for Advanced Research (CIFAR)-10, CIFAR-100, Modified National Institute of Standards and Technology (MNIST) and a traffic-oriented Sydney Coordinated Adaptive Traffic System (SCATS)-related dataset. CIFAR-10 and CIFAR-100 were originally introduced by Krizhevsky, Hinton, et al., 2009, MNIST by LeCun and Cortes, 2010, while SCATS-based data originate from adaptive traffic signal control environments (McCann, 2014). These datasets are widely used and highly relevant in the communities of ML and intelligent transportation systems and facilitate rapid research progress, reproducible experimental design and simplified comparison of algorithmic performance in different FL configurations.

The *CIFAR-100* dataset is a curated subset of the 80 Million Tiny Images dataset created by Torralba et al., 2008. It comprises 60,000 RGB images of size 32×32 pixels, categorized into 100 fine-grained classes organized under 20 coarse-grained superclasses (Krizhevsky, Hinton, et al., 2009). Each class contains 600 samples, with 500 assigned for training and 100 for testing. An extract of the class hierarchy is presented below:

- **people**: baby, boy, girl, man, woman
- **reptiles**: crocodile, dinosaur, lizard, snake, turtle
- **small mammals**: hamster, mouse, rabbit, shrew, squirrel
- **trees**: maple, oak, palm, pine, willow
- **vehicles 1**: bicycle, bus, motorcycle, pickup truck, train
- **vehicles 2**: lawn-mower, rocket, streetcar, tank, tractor

This hierarchical label structure allows for flexible experimentation across different granularities of classification complexity and supports heterogeneous data partitions in FL experiments.

The *CIFAR-10* dataset serves as a simplified variant of CIFAR-100, containing only ten object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck (Krizhevsky, Hinton, et al., 2009). It includes 60,000 RGB images of the same 32×32 pixel resolution, with each class consisting of 6,000 samples equally divided between training and test partitions. Its lower complexity and smaller number of classes make it particularly suitable for baseline experiments and model calibration studies.

For experiments requiring an even simpler classification task, the MNIST dataset is employed. MNIST consists of 70,000 grayscale images of handwritten digits (0–9), each of size 28×28 pixels (LeCun and Cortes, 2010). Despite its simplicity, MNIST remains one of the most widely used datasets for evaluating fundamental aspects of neural network architectures and for assessing privacy-preserving FL scenarios with low computational requirements.

In addition to image-based benchmarks, a *SCATS-related* traffic signal dataset is considered for constructed use-cases focusing on intelligent transportation and adaptive signal control. SCATS generates rich operational data on traffic signal states and detector measurements in urban road networks, which can be leveraged to model decentralized learning across intersections or control centers in a Data Space (McCann, 2014). Such a dataset enables the evaluation of FL methods in a domain that closely reflects real-time, safety-relevant control applications while still providing a structured and well-defined experimental setting.

In summary, the use of standardized datasets such as CIFAR-10, CIFAR-100, MNIST and a SCATS-based traffic dataset enables reproducible and computationally efficient evaluation setups across both generic computer vision and domain-specific control tasks. Depending on the experimental objective, domain and model complexity, one or more of these datasets are adopted as the constructed use-case to assess the proposed FL framework in a controlled and comparable manner.

All findings and evaluations presented in the following chapters are grounded in the principles of the DSR methodology. They are derived from the application of this methodology to the relevant use-cases identified in this work—specifically the *GAIA-X 4 ROMS* research project and the constructed data set-based use-cases described above. This dual-use-case approach ensures that both practical relevance and methodological rigor are systematically addressed within the overall research framework.

4 Operationalizing Data Spaces for Federated Learning Workflows

The previous chapter detailed the DSR methodology, including its iterative process of problem identification, artifact design, demonstration, evaluation and communication, as well as the practical Gaia-X–related use cases that contextualize all research questions addressed in this work. This chapter applies that DSR framework by systematically ***evaluating*** *Data Space connector technologies (Industrial Data Space Connector (IDS) and EDC) around the Gaia-X use case and developing* ***FedDScon****, a reusable, EDC-based framework that simplifies the setup of Gaia-X–compatible Federated Data Spaces for FL workflows, as the main contribution addressing* ***Research Question 1****. Section 4.1 motivates Federated Data Spaces and their natural alignment with FL's privacy-preserving, decentralized principles, while explaining how Gaia-X–compatible connectors define the core problem space. Related Work (Section 4.2) positions Federated Data Spaces, IDS and EDC within existing architectures, standards like the DSP, as well as governance models, providing essential conceptual and technical background. Building on this, Section 4.3 conducts a rigorous Architecture Trade-off Analysis Method (ATAM)-based comparison of IDS and EDC architectures, communication patterns and suitability for FL-centric workflows, highlighting EDC's advantages in extensibility, performance efficiency and flexibility. The subsequent Section 4.4 operationalizes these insights through Gaia-X–aligned Minimum Viable Demonstrators (Minimum Viable Demonstrators (MVDs)) that implement representative business processes with both connectors, empirically validating and refining the evaluation criteria via real-world data exchange patterns in multi-agent systems and arrival prediction scenarios. Finally, Section 4.5 synthesizes the recurring patterns, design decisions and empirical lessons from prior sections into the* ***FedDScon*** *framework, offering generalized integration templates, configuration utilities and extension mechanisms for scalable FL tasks in sovereign Data Spaces, thereby closing the DSR cycle with a deployable framework as contribution for* ***Research Question 1****.*

4.1 Introduction

The increasing digitalization of industry and the rise of data-driven business models have made collaborative data ecosystems an essential component of modern value chains. In such ecosystems, multiple parties, often from different organizational or even jurisdictional backgrounds, need to exchange and process data securely, efficiently and in compliance with legal and contractual constraints. Traditional centralized data exchange approaches, where all partners upload and process data in a single controlled environment,

conflict with the growing demand for data sovereignty, privacy and regulatory compliance (Otto et al., 2022). This tension has motivated the development of Federated Data Spaces as decentralized infrastructures that enable organizations to share and utilize data in a standardized yet sovereign manner.

Federated Data Spaces are based on the principle that data remain under the control of its originator and are only shared according to enforceable policies. This concept has rapidly gained importance with the increasing interconnection of industrial ecosystems, from supply chain networks and smart manufacturing to the health and mobility domains. By enabling trusted, cross-organizational data exchange, Federated Data Spaces form the backbone of modern digital infrastructures, supporting the European vision of open, interoperable and sovereign data ecosystems exemplified by Gaia-X and Catena-X. As industries evolve toward these interconnected environments, the ability to flexibly and securely integrate analytical workflows, such as distributed optimization, AI, or ML becomes a critical success factor.

In this context, the relationship between Data Spaces and FL is particularly relevant. FL provides a paradigm where multiple participants collaboratively train ML models without centralizing raw data. This not only protects sensitive information, but also aligns seamlessly with the decentralized, sovereign and policy-driven principles of Data Spaces. Integrating FL into Federated Data Spaces would enable advanced, privacy-preserving AI workflows to operate within regulated environments, making it possible to harness data distributed across company boundaries while respecting governance requirements. Therefore, evaluating how FL can be integrated into such infrastructures is a central question for modern data-driven collaboration frameworks and a key contribution of this chapter.

To realize these concepts, standardized architectures and connector technologies such as the IDS framework and the EDC have emerged. These implementations form the technical foundation that enforces interoperability, trust and policy compliance across heterogeneous systems. Within these environments, connectors serve as autonomous entities representing data providers and consumers, implementing semantic interoperability and secure contract management. Such standardized components make Federated Data Spaces a natural platform for extending industrial data sharing toward Federated AI ecosystems.

Building on this foundation, the following chapter provides an in-depth analysis of connector technologies and their implications for FL integration. It begins with a theoretical evaluation of Data Space architectures based on IDS and EDC, followed by practical demonstrations through Minimum Viable Demonstrators (MVDs) that showcase their functionality in distributed learning contexts. The chapter concludes with the introduction of FedDScon, a framework designed to simplify the setup and configuration of Federated Data Spaces tailored for Federated Learning tasks, enabling scalable, interoperable and privacy-preserving data collaboration across organizations. Therefore, FedDScon provides a solution to **Research Question 1** and serves as the main contribution in this chapter.

4.2 Related Work

A definition of what Federated Data Spaces are and what primary targets they address is already described in Chapter 2. This section will focus on presenting the state-of-the-art implementation for Data Space architectures like Gaia-X and Industrial Data Spaces as well as for EDC and IDS. It will also provide details on which components serve what features and interfaces.

4.2.1 Federated Data Spaces

Federated Data Spaces represent an architectural and governance paradigm for secure, trust-based and sovereign data exchange across organizational boundaries, without requiring centralized data aggregation (Otto et al., 2019, 2022). In contrast to traditional data hubs or data lakes, Data Spaces follow a *data stays at the source* principle and enable *data usage instead of data ownership* through standardized communication and policy mechanisms.

From a structural perspective, Federated Data Spaces combine technical interoperability, trust infrastructures and governance frameworks into a cohesive ecosystem. Typical roles and components include:

- **Data Providers and Data Consumers**, who publish and consume data services and data products.
- **Metadata Brokers**, which enable discovery of available data offerings and services in the space.
- **Identity and Trust Services**, which issue credentials, verify identities and maintain trust relationships.
- **Clearing Houses**, which provide non-repudiation, logging and audit trails for data transactions.
- **Connector Endpoints**, deployed at each participant, that enforce security, policies and technical interoperability.

Federation services provide identity management, participant onboarding, cataloging and governance oversight, while connector-based exchange ensures that each organization keeps data under local control (Otto et al., 2019). A key enabler of interoperability in modern Data Spaces is the DSP, a set of specifications designed to facilitate interoperable data sharing governed by usage control and based on Web technologies. The DSP standardizes message semantics and API interactions for:

- **Negotiation** of data usage contracts between connectors.
- **Agreement** and management of the contract lifecycle.
- **Data Transfer** orchestration between heterogeneous infrastructures.

Research has increasingly focused on formalizing machine-readable data usage policies, for example, based on the ODRL and aligning them with technical enforcement mechanisms and distributed ledger-based audit trails to strengthen accountability in Data Spaces (Jung and Dörr, 2022; Usländer and Teuscher, 2022). These approaches help bridge the gap between legal requirements for data governance and machine-executable enforcement in Federated Data Spaces, which is particularly relevant for GDPR-aligned analytics and FL scenarios.

4.2.2 Industrial Data Spaces and IDS-RAM

The IDSA (see Section 2.1.1) provides the IDS-RAM, which defines the blueprint for interoperable and sovereign data sharing in industrial and cross-domain ecosystems (Otto et al., 2019). The IDS-RAM describes an ecosystem consisting of the following core components:

- **IDS Connector**: Endpoint at each participant, enforcing security, policy and interoperability.
- **Metadata Broker**: Catalog for discovering and querying data offerings and services.
- **Identity Provider and Dynamic Attribute Provisioning Service (DAPS)**: Services for issuing, managing and validating identity credentials and attribute-based tokens.
- **IDS App Store**: Repository for certified data applications that can be deployed into connectors.
- **Clearing House**: Service for logging, monitoring and auditing data transactions.

The IDS Connector is the central software component that links an organization to the Data Space. It provides secure communication, metadata handling and policy enforcement between data providers and consumers, as well as serves as the endpoint for the exchange of data and services (Otto et al., 2019). Beyond the connectors, the metadata broker, identity provider, clearing house and app store jointly establish the operational backbone of an IDS-based Data Space.

Data sovereignty in IDS is operationalized through usage control and policy enforcement mechanisms. Policies, often modeled with ODRL, are evaluated by a Policy Decision Endpoint and enforced by a Policy Enforcement Endpoint within the connector, allowing obligations such as:

- **Retention limits**, e.g., deletion of data after a specified time period.
- **Purpose and usage restrictions**, such as non-commercial use only.
- **Prohibitions on forwarding**, restricting onward sharing to third parties.

The IDSA-defined certification and conformity assessment procedures ensure that the connectors, core services and participants meet security, interoperability and governance requirements, creating a trusted ecosystem for data sharing between organizations (Otto et al., 2019).

Typically a Data Space gets deployed for a specific business domain, like logistics or for forests. Therefore, to this point, there are multiple domain-specific Data Spaces deployed, which all have implemented IDS concepts and components. For example, the Mobility Data Space or the Catena-X Automotive Network are running instances, that incorporate the Data Space technologies (Otto and Jarke, 2019; Pretzsch et al., 2022). These implementations confirm the applicability of IDS-RAM as a technical and governance framework for domain-driven Federated Data Spaces, in which each ecosystem establishes its own specialized data-sharing environment while following common architectural principles.

4.2.3 Eclipse Data Space Connector

The EDC represents the next-generation, open-source implementation of the IDS Connector principles, adapted for cloud-native, modular deployment and aligned with the Gaia-X and IDSA joint efforts. It was developed under the Eclipse Foundation (as part of the Tractus-X and Eclipse Data Space projects) to address the limitations of the original IDS Connector, notably its complexity, monolithic structure and lack of cloud interoperability.

Architecturally, the EDC separates the responsibilities into a *control plane* and a *data plane*, complemented by an *extensions layer*.

- **Control Plane**: Manages contract negotiation, policy validation, identity and authentication, and catalog management through management APIs.
- **Data Plane**: Executes the actual data transfer over secure communication channels, implemented using technology-specific adapters (e.g., Hypertext Transfer Protocol (HTTP) endpoints, object storage, message queues).
- **Extensions Layer**: Provides integration points for external identity providers (OAuth2, DAPS, decentralized identities), catalog services, monitoring systems and custom policy engines.

The EDC is designed to be compliant with the Dataspace Protocol and related specifications developed jointly by IDSA, Gaia-X and the Eclipse Data Space Working Group (see Section 2.1). By implementing DSP-based negotiation, contract agreement and transfer orchestration, EDC connectors from different federations can interoperate, enabling cross Data Space data exchange and thus supporting Gaia-X's goal of trans-sectoral federation.

In contrast to the original IDS Connector, which includes an explicit app execution environment for data apps, the EDC focuses primarily on acting as a policy-enforced data

exchange gateway. Code-to-data scenarios are typically implemented via integration with container orchestration platforms or function-as-a-service environments rather than through a connector-internal app framework. However, EDC-based architectures have been adopted in several applied projects (e.g. Catena-X and Gaia-X lighthouse projects), demonstrating their suitability as a cloud-native connector for GDPR-aware data sharing and as a technical backbone for federated learning workflows in Data Spaces (Neidhardt, 2024; Reiberg et al., 2023).

4.3 Systematic Comparison between IDS and EDC

In this section, the focus shifts from related work to a theoretical analysis of existing Data Space Connector technologies, building on and extending the results of a previous Bachelor's thesis on the evaluation of EDC and IDS for FL scenarios (Rambau, 2026). The goal of this analysis is to evaluate both connector technologies with respect to their fundamental feature sets and to derive well-founded insights into which connector is best suited to address **Research Question 1**. This section first introduces the applied evaluation method, the Architecture Tradeoff Analysis Method (ATAM) and motivates its suitability for comparing connector architectures in FL settings. Subsequently, the derivation of the ATAM screening questions is presented, followed by a systematic, ATAM-based comparison of EDC and IDS and a brief discussion of how the theoretical findings were validated in a practical prototype setup.

4.3.1 Evaluation Method: ATAM

The comparison between EDC and IDS is structured using the Architecture Tradeoff Analysis Method (ATAM), a scenario–based approach to evaluate software architectures with respect to multiple, potentially conflicting quality attributes (Kazman et al., 1998). ATAM proceeds in four phases:

1. elicitation of business drivers and quality attributes,
2. description of the architecture and construction of concrete scenarios,
3. qualitative analysis of how each architecture responds to these scenarios and
4. consolidation of tradeoffs and sensitivity points.

For the FL context, FL itself is modeled as a *growth scenario* that imposes additional requirements on existing Data Space architectures, such as bi-directional communication, scalable participant management and flexible integration of privacy-preserving mechanisms. ATAM is well suited here because it explicitly targets forward-looking scenarios and focuses on architectural qualities like flexibility, security and performance. Additionally, it avoids an overreliance on low-level performance metrics that would be difficult to generalize across heterogeneous industrial deployments.

4.3.2 Derivation of ATAM Screening Questions

The concrete screening questions used for the evaluation are derived by combining the concept of a scenario of ATAM with the quality characteristics of International Organization for Standardization (ISO)/IEC 25010:2023 for the quality of software products. In a first step, relevant FL-specific requirements are identified, such as the need for active model update push, passive data provision, high availability during training rounds and straightforward onboarding of new FL participants. These requirements are then mapped to ISO 25010 characteristics such as *Functional Suitability*, *Performance Efficiency*, *Compatibility*, *Interaction Capability*, *Reliability*, *Security*, *Maintainability* and *Flexibility*.

For each quality attribute, at least one question is formulated that expresses a concrete FL scenario in which the connector architecture is stressed. The set of questions resulting from this study is shown in Table 4.1 and forms the qualitative evaluation grid for both connectors. For example, question **1.1** reflects the need for both passive data provisioning and active update dissemination, while question **2.1** captures the impact of simultaneous client requests during a global model aggregation step.

Variable	Question	Goal
Functional Suitability	**1.1**: Can the connector both send data and make it available passively?	The FL scenario requires a global model that must be sent to the participants. The participants must also be able to provide their data as easily as possible and to access it easily.
Performance Efficiency	**2.1**: How does the connector scale when a large number of clients simultaneously send requests to the connector?	When model updates are performed, many requests are sent simultaneously, so a stable connection to the participants should be ensured in order for the data exchange to run smoothly.
Compatibility	**3.1**: How well can the connectors be established on different hardware platforms?	In FL, there is a heterogeneous field of participants, so it is important that the connectors can be established as easily as possible on different infrastructures.
Interaction Capability	**4.1**: How easily can FL participants use data and are special policies required for this?	Simple provision of the global model is essential to guarantee smooth operation. In addition, privacy and data governance require that the data does not get misused.
Reliability	**5.1**: Are the connectors always available?	Participants should always be able to send their updates or receive updates to the global model.

Table 4.1: ATAM evaluation questions for Connector suitability in FL tasks (Rambau, 2026).

Variable	Question	Goal
Reliability	**5.2**: Do the connectors also work with poisoning?	The connector functionality should not be affected by model poisoning.
Security	**6.1**: How is the connector secured along the sharing path?	The data should be well secured along the sharing process to prevent misuse or falsification.
Security	**6.2**: How well is the logging and securing of data exchange and use?	Data owners should be able to determine how their data is used.
Maintainability	**7.1**: How easily can new functional blocks be integrated?	New FL functionality, such as aggregation logic, should be easy to integrate into the existing connector setup.
Maintainability	**7.2**: How easily can the systems be understood and tested?	The connectors and FL components should be designed so that they are easy to understand and to test.
Flexibility	**8.1**: How easily can FL participants be integrated or removed?	It should be possible to add or remove participants quickly so that the heterogeneous participant field can be covered well.
Flexibility	**8.2**: How well can the connectors be adapted to new standards?	In Federated Learning, new standards should be established and implemented easily through the connectors.

Table 4.2: ATAM evaluation questions for Connector suitability in FL tasks (Rambau, 2026). (continued)

4.3.3 ATAM-Based Evaluation of EDC and IDS

Based on the screening questions in Table 4.1, both connectors are qualitatively evaluated, producing the evaluation summary shown in Table 4.3. This table references the question number (**Q-Nr.**), type of connector (**EDC** and **IDS**) and **Reason**, why the categorization is evaluated like this. The qualitative ratings are encoded using the following symbols:

- + indicates that the connector satisfies the corresponding criterion well in the FL scenario considered.
- +/- indicates that the connector only partially fulfills the criterion, for example due to architectural limitations, missing tooling, or open implementation questions.
- - indicates that the connector does not adequately support the criterion in its current form or only with disproportionately high effort.

- **?** indicates that the assessment is inconclusive, for instance because of lacking empirical evidence (e.g., penetration tests) or insufficient documentation.

The *Functional Suitability* is fulfilled by both connectors, as each supports passive data provision and active data transfer once a bidirectional connection is established between the consumer and the provider. For *Performance Efficiency*, both architectures offload scalability concerns to the underlying communication protocol, but EDC offers additional protocol flexibility by supporting alternatives such as Kafka, which can buffer data streams and thus better accommodate bursty FL update patterns.

In terms of *Compatibility* and *Interaction Capability*, the shared Java code base allows deployment on a wide range of hardware platforms, which is beneficial in heterogeneous FL environments and both connectors support policy-based usage control via ODRL. *Reliability* is comparable, as both connectors can be operated continuously. The EDC can further leverage Kafka to mitigate transient connectivity issues. Security along the data sharing path remains difficult to assess conclusively, because there are no publicly available penetration tests of the underlying Data Space protocol, so both systems are rated as undetermined for the question **6.1**. *Maintainability* and *Flexibility* slightly favor EDC, as its open-source extensibility simplifies the integration of new functional blocks (such as aggregation logic or poisoning detection) and the adoption of new Data Space–conform standards compared to the more centrally managed extension model of IDS.

4.3.4 Implications of the ATAM Evaluation

The results summarized in Table 4.3 indicate that both connector technologies can, in principle, be adapted to support FL scenarios, but with different levels of effort and architectural alignment. Overall, the EDC shows advantages in several qualities that are particularly relevant for FL, notably *Performance Efficiency*, *Maintainability* and *Flexibility*, whereas IDS tends to be stronger in more centralized governance-heavy settings.

Regarding individual criteria, both connectors achieve a positive rating for *Functional Suitability* (**1.1**), *Interaction Capability* (**4.1**) and *Reliability* (**5.1**, **5.2**), which implies that either solution can meet the fundamental communication and availability requirements of FL. However, for *Performance Efficiency* (**2.1**) and the dynamic integration of new functional blocks (**7.1**), EDC receives more favorable ratings (**+** and **+/-** respectively), due to its extensible, component-based architecture and protocol independence, while IDS is evaluated more cautiously (**+/-** and **-**) due to its more monolithic design and centrally managed extension mechanisms.

From these theoretical observations, it can be deduced that EDC is generally better suited as a basis for FL-oriented connector architectures, particularly when extensibility and the integration of advanced FL features (e.g., custom aggregation logic, poisoning detection, or privacy-preserving components) are required. At the same time, the evaluation also shows that IDS is not fundamentally unsuitable for FL. Rather, additional engineering effort would be necessary to compensate for its architectural rigidity and to integrate

Q-Nr.	EDC	IDS	Reason
1.1	+	+	Both connectors allow for passive provisioning and active sending without any problems. The prerequisite is the establishment of a bidirectional connection.
2.1	+	+/-	Both connectors outsource performance to the communication protocol for large queries. The EDC also offers protocol independence.
3.1	+/-	+	Thanks to the Java code base, the respective connector can be easily integrated on almost all devices. Another advantage of the IDS is the existing Docker image.
4.1	+	+	Once the bidirectional connection has been established, data can be used easily and without any problems. Policies can be specified as desired via ODRL.
5.1	+	+	The connectors are always available. Since this is a European project, time differences are not an issue.
5.2	+	+	Poisoning has no direct impact on how the connectors work. However, the EDC's open source strategy makes it easier to establish mechanisms that provide security.
6.1	?	?	Currently, there are no penetration tests available that verify the actual security of the connection.
6.2	+	+	The clearing house makes it very easy to log data usage in detail.
7.1	+/-	-	Planned changes must be Data Space compliant. The IDS can only integrate blocks centrally via the host, which is better solved in EDC through extensibility via open source libraries.
7.2	+/-	+/-	Both systems offer documented step-by-step instructions. Problems include the lack of a Docker image for EDC and the limited functionality of the GUI for IDS.
8.1	+	+	Adding participants works perfectly with both connectors. It is important to set up a bidirectional connection so that each new participant can both use and provide data.
8.2	+	+/-	New standards must be Data Space compliant if they are to be adopted. The open source approach at EDC makes implementation easier in terms of compliance.

Table 4.3: ATAM evaluation questions for Connector suitability in FL tasks (Rambau, 2026).

the required FL-specific mechanisms within the existing governance and usage-control framework.

The qualitative results of this ATAM-based analysis therefore provide a conceptual foundation for subsequent practical demonstrators, in which the compatibility of connector feature sets with concrete FL workloads and deployment environments will be evaluated. In particular, the identified strengths of EDC in terms of extensibility and protocol flexibility, as well as the stronger governance focus of IDS, will guide the design of prototype implementations and the selection of integration patterns in real-world federated Data Space scenarios.

4.4 Scenario-based Connector Evaluation

With the systematic and theoretical comparison of EDC and IDS connectors in mind, it is crucial to implement some test scenarios to evaluate those systems in real world demonstrators as DSR requires. To do so, the author of this work developed multiple demonstrators with different use cases and technologies to prove the findings resulting from the previous comparison for complex data exchange through connectors. Each demonstrator focuses on the features and complexity of these connectors (IDS or EDC), as they are the touch point for possible customers and abstract the Data Space complexity. With this, every demonstrator aims to create an innovative Gaia-X conform Data Space with all Gaia-X related concepts (Section 2.1). As IDS and EDC are not directly compatible with Gaia-X central services, the connector's features needed to be extended to be able to communicate with those services. The demonstrators were developed in the timeline of the GAIA-X 4 ROMS research project (see Section 3.2) and showcase parts of the total use case of the project. This work will first present works which use the IDS as connector technology in two demonstrators that cover different aspects of the GAIA-X 4 ROMS use case. One shows possible usages in Multi-Agent System (MAS) and the second focuses on a more data centric approach, where one party is the data source and the other one will be the destination. Both demonstrators address different types of data exchange. In the MAS demonstrator, the data is pushed asynchronously (see Section 4.4.1), whereas the second demonstrator actively pulls the data (see Section 4.4.2). The second demonstrator was implemented in two versions, one with IDS and one with EDC, to allow for a better comparison between both connector technologies. Insights from these demonstrators are subsequently used in the next section to develop FedDScon, which finally addresses **Research Question 1**.

4.4.1 Multi-Agent System based on top of IDS services

The demonstrator showcasing the MAS interaction with Federated Data Spaces is published in Maecker, Harenbrock, et al., 2024 and used the MAS development approach from Gösling et al., 2024. The author of this work specifically contributed to Maecker, Harenbrock, et al., 2024 by providing the necessary IDS connectors and Data Space ser-

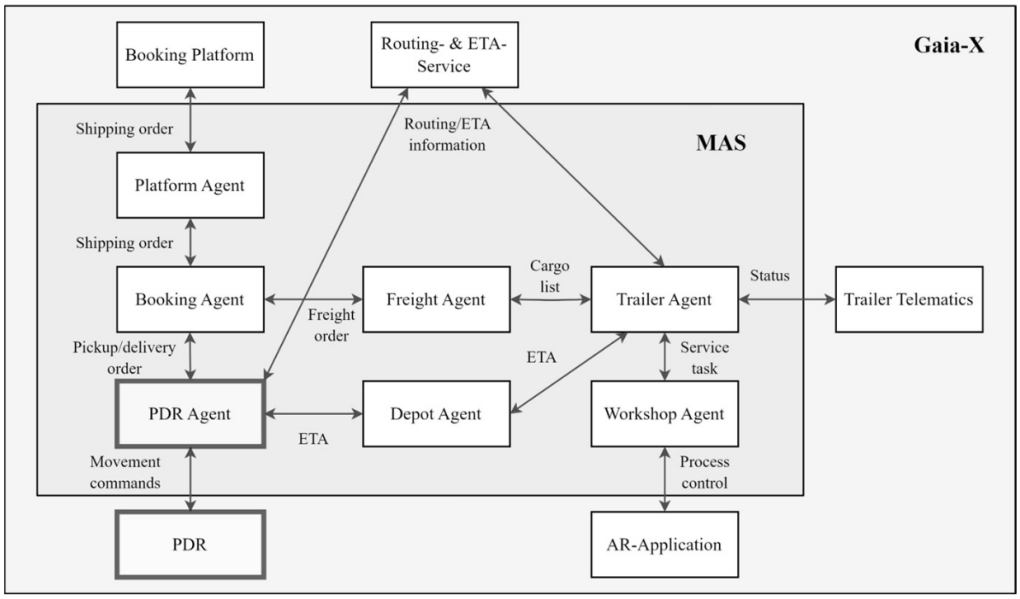


Figure 4.1: Multi-Agent System overview derived from Maecker, Harenbrock, et al., 2024.

vices, as well as configuring them jointly to run the MAS in a Gaia-X equivalent Data Space. Figure 4.1 shows the entire MAS system with all the necessary agents presented in Maecker, Harenbrock, et al., 2024. This system is designed to develop autonomous decisions for an innovative parcel management and delivery system. Thus, there are multiple agents interacting with each other like the Platform Agent, Booking Agent and PDR Agent, which are all addressing different stages for parcel delivery. One controls trailers (Trailer Agent), one controls parcel delivery robots (PDR Agent) and another manages new incoming bookings (Booking Agent). All of these agents interact with each other within the MAS asynchronously via a specialized event system called Spade (Palanca et al., 2020). This library is written in Python and allows complex agent interactions in heterogeneous systems with a propriety communication protocol. Besides the agent system, this figure shows various services around the MAS, like the Booking Platform, Routing- and Estimated Time of Arrival (ETA) or the Parcel Delivery Robot (PDR), which all need to communicate to the agent system.

Organizational Credential Manager integration for Gaia-X conform auth-services

Therefore, the paper not only showed a sample implementation of a MAS, but also presented a solution for communicating across company borders with a Data Space architecture. For this part, the Industrial Data Spaces services were used, as during development, the degree of maturity for the EDC was not comparable. The usage is as follows:

Each company deploys one customized connector, which is capable of communicating with the Spade protocol and handles everything related to the Data Space. This paper especially focused on the integration of the Gaia-X Organizational Credential Manager (OCM), which handles authentication and authorization for all user accounts in a Data Space. For this demonstrator, the adoption of OCM is necessary, as tasks of the PDR Agent include validating users and checking if they are in possession of a valid ID after which the corresponding compartment of the robot can be opened. Therefore, the OCM validation feature via connectors was mandatory to verify the Gaia-X issued credential for this use case.

Propriety data exchange protocols in IDS connectors

After the issue of authenticating and authorizing users was addressed by implementing OCM into the connectors, there was a second issue regarding deployment of those agents in the Data Space ecosystem. General communication is handled via the Spade library that uses an underlying Contract Net Protocol (CNP) for message and task transfer between agents. This protocol is propriety and therefore is not simple to include into Data Space communication.

However, Maecker, Gösling, and Sachweh, 2024 show a solution in which multiple agents can communicate with CNP and IDS connectors as the gateway between participants and the Data Space. The author of this work contributed greatly to this paper by implementing and configuring the IDS connectors, as well as in the extension of IDS connectors to conform to the CNP protocol matching the Data Space requirements defined by the Gaia-X standard. This IDS extension and deployment is mandatory to be capable of running this use case and evaluating the capabilities of IDS in the Gaia-X context. Figure 4.2 shows a sample data flow with CNP without OCM authentication in the demonstrator context of the previously described MAS environment. The sequence diagram presented here is an abstract definition of the data flow for two parties in the Data Space. Agent A in this case could be the PDR agent and Agent B could be the Booking Agent or other external services like the PDR itself or the ETA service, all shown in the overview of Figure 4.1. This demonstrator validates the possible use of IDS connectors in inter company communication with proprierty protocols, which are crucial for complex business processes. For the scenario described in the paper and presented here, the PDR Agent (Agent A in the following) is developed by one company called German Research Center for Artificial Intelligence and the PDR (Agent B in the following) is an autonomous robot from some external company. Both need to communicate via CNP, which is used by the Spade library in the MAS system for exchanging tasks between the service components.

Figure 4.2 shows the necessary communication flow until Agent B can receive tasks from Agent A. The preparation requires the creation of two local networks, one for Agent A and one for Agent B. During the demonstration, those networks are created by virtual Docker networks overlayed on the host machine. Each network consists of an Agent A/B service representing the application layer and an extended IDS connector.

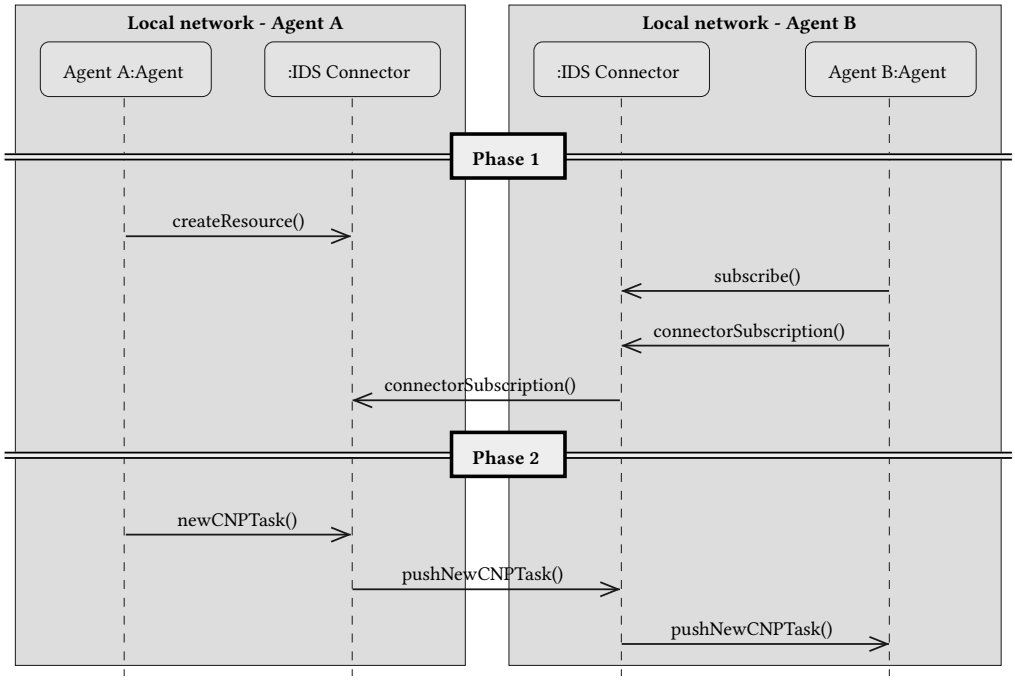


Figure 4.2: Simplified sequence diagram for CNP communication between two agents. Figure adopted from Maecker, Gösling, and Sachweh, 2024.

IDS data exchange flow

In general, data flow can be clustered in two phases. The first phase indicated by a horizontal line labeled `Phase 1` in Figure 4.2 is mandatory for the establishment of data exchange connections. The second is the data flow in general. As presented in this figure, the first phase requires configuration and setup overhead, which is unnecessary without Data Spaces. In parallel, it provides all the advantages of Data Space communication as described in Section 4.2.

For the data flow initiation process, the data provider, in this case Agent A, first needs to create their data offering. This offering is indicated by *create resource* and is further forwarded to Data Space services with the use of the IDS connector.

After resource creation, the data consumer, here Agent B, can find the resource in the catalog of the Data Space and subscribe with a subscription pattern to new data provided by the resource of Agent A. For this, the catalog search is indicated by *subscribe* and the specific subscription to the resource of IDS connector of Agent A is presented in the sequence diagram by *connector subscription*. As soon as both IDS connectors succeed in establishing a one directional subscription, the expected data flow can begin.

This data flow can be seen in the second phase of the sequence diagram. It starts with a new data package (CNP task) for Agent A, which is pushed to the IDS connector. The connector then knows which other connectors are subscribed to this resource and pushes

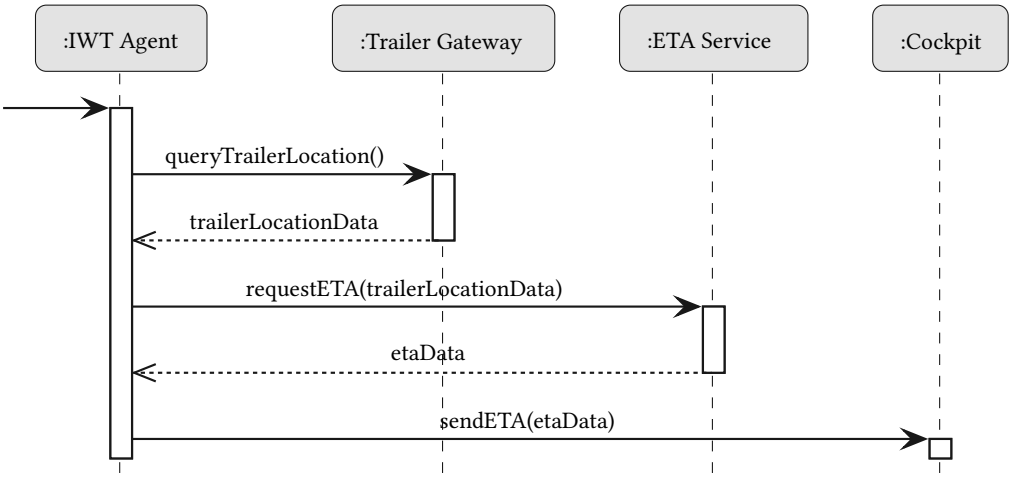


Figure 4.3: Dataflow for the synchronous communication with IDS. Figure adopted from Kremer et al., 2023.

the data package further to the connector of Agent B. During the subscription initiation, Agent B defined a callback address to which new data should be provided. This address is now used to push the data to one endpoint of Agent B where they are finally consumed.

4.4.2 Intelligent Arrival Time Prediction in Data Ecosystems

In addition to the agent based demonstrator that focuses mainly on asynchronous data exchange between two parties, the author of this work also contributed to Kremer et al., 2023 and Sachweh et al., 2025b, where a larger scale scenario from the research project is focused with more partners involved.

Kremer et al., 2023 demonstrates this business use case based on the same IDS connector technology as in the previous section, while Sachweh et al., 2025b shows a solution with the more modern EDC connector, which is limited to asynchronous communication. Reason for this is that the EDC strictly splits data providers and data consumers and therefore only allows uni-directional communication for a single asset. Both connector technologies have specific advantages and disadvantages that can be derived along this Long Haul Freight Use Case.

Long Haul Freight Use Case

With this demonstrator, the entire long haul transport business process of the research project should be addressed. Therefore, this demonstrator includes all services and partners which have to interact with trailer data for the main run. Figure 4.3 presents the various participants and the required data flows in an abstract sequence diagram. It shows that this demonstrator again contains a MAS component, called IWT Agent. This agent

is created for each physical trailer to create a one-to-one relationship between the IWT agent and the trailers. Receiving positional and freight data from the trailers is possible through the Trailer Gateway. In addition to trailer data, the IWT agent requires routing and time estimation data for its internal decision making process. This part is handled by an external service provider, which can provide the data. In Figure 4.3, this service is denoted as ETA-Service. The final service of this demonstrator is the Cockpit, an external provider used by workers for cargo loading or end customers for information on package location.

However, a smooth flow of data between all services is essential for the successful implementation of this business use case. Therefore, it is mandatory to ensure communication as shown in Figure 4.3. The IWT Agent needs to be able to collect trailer related data, especially geo-positional coordinates. Those location data are afterwards used to request an ETA prediction from the ETA-Service which provides a result to the IWT Agent. The agent then reacts on the information and can check if the trailer is on time or is delayed. All information, location, as well as freight data and time schedule, is sent to the Cockpit for possible end users.

In this data flow use case all communication of the core services are developed with Representational State Transfer (REST) interfaces based on PUT and POST requests in synchronous manner. This design allows to demonstrate data exchange, which is not push based messaging but needs synchronous data exchange. For Data Spaces this is a special case, as they are more designed to conform to the data sharing principle, without directly integrating business processes. The sharing concept only contains data flows that are unidirectional with one data provider and one or multiple data consumers. For complex business processes, this limitation is usually not suitable and therefore solutions for synchronous or nearly synchronous data flows need to be tested.

Synchronous Communication with IDS

As previously denoted, the long haul business use case was developed in two Data Space technologies. The first version is implemented with IDS connectors similar to the MAS demonstrator and published in Kremer et al., 2023.

This work describes the relevance for the usage of Data Spaces, especially for Gaia-X Data Spaces, and details the implementation of the IDS connector. It also includes some arguments, why the EDC is not suitable for demonstrator usage at this time, but that it could be a good alternative in the future. The full paper focuses on the demonstrator with particular emphasis on communication with the ETA service as a central AI service component.

In Kremer et al., 2023 the author of this work strongly contributed to the demonstrator implemented with the ETA service as a core component. He created the necessary services for the demonstrator, which include the ETA service as well as a demo iWT agent

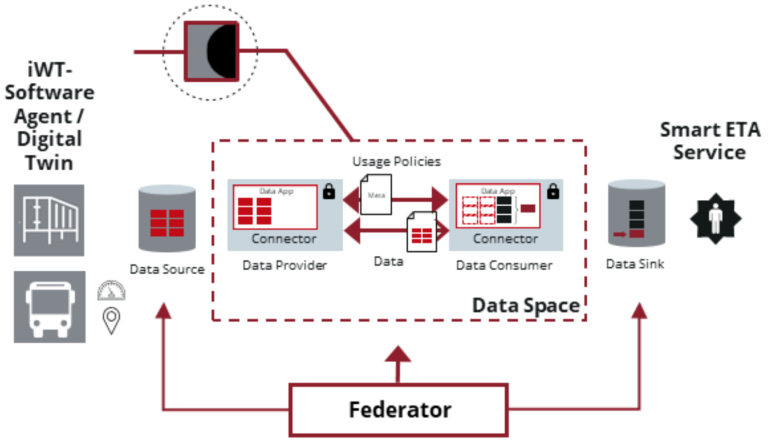


Figure 4.4: Components of the federated data ecosystem with IDS. Figure extracted from Kremer et al., 2023.

and gateway service to receive telemetry data via REST. In addition, he configured the connectors according to the requirements defined in the data flow description.

Figure 4.4 shows a part of the demonstrator proposed by Kremer et al., 2023, where the agent shares location data from trailers to the ETA service. It also shows the concept of the Data Spaces with unidirectional communication flows from Data Source and Data Sink. In this case, the iWT agent, which administers a specific trailer from an external company, sends (provides) location data as well as target locations to the ETA service. This service then uses the location data provided to calculate the ETA predictions, which are sent to an agent. As data needs to be sent back to the agent, both services would need to act as Data Source and Data Sink.

In this scenario, the authors of the paper used a trick in the IDS technology stack, which allows an HTTP Pull request to send metadata as path parameters with a simple data pull request. Thus, only the Smart ETA service needs to create a data offering with Pull configuration. As soon as the iWT agent concludes the necessary contract with this offering, the data flow between both services can begin.

With this trick in the architecture, the iWT agent can pull ETA service data based on the concluded contract. It attaches the relevant location data to the pull request, which is forwarded between both connectors inside the Data Space. The IDS connector finally requests new data with attached metadata from a specific endpoint of the ETA service, which calculates requested information and sends them back afterwards. This example shows how to extend the functionality of the Data Space protocol to allow synchronous data exchange between two services based on IDS technology.

That said, it is not recommended to implement long term services like this, as it is not the desired method for Data Space communication and could be removed at any time.

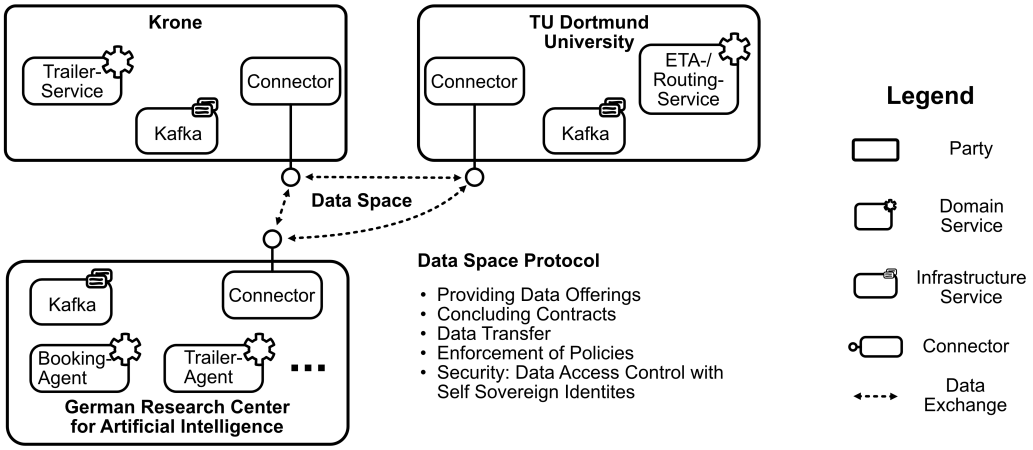


Figure 4.5: Overview of the demonstrator containing the Kafka message broker. Figure extracted from Sachweh et al., 2025b.

Here, only one communication direction is shown for the demonstrator. However, the complete demonstrator was showcased in Kremer et al., 2023 and fully functional with IDS connector technology. All synchronous communications of the long haul business use case were implemented with the metadata workaround.

Asynchronous Communication with EDC

An alternative to the IDS technology trick is presented in Sachweh et al., 2025b. This paper shows how the long haul business use case can be created with standardized Data Space protocol format. It uses the EDC connector as it is the more modern version of the IDS technology and was built on the experiences received from the IDS. Based on the evaluation in Section 4.3 it is also the connector that is better suited for FL tasks in Data Spaces. As shown in Figure 4.5 this demonstrator once again contains the same services provided by the author of this work. It has the Trailer-Service, which represents the trailer gateway, contains the Trailer-Agent formerly known as iWT agent as well as the ETA-/Routing-Service, which provides ETA predictions.

Along with the services, which are distributed across multiple companies denoted with Krone, German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)) and TU Dortmund university, each company deploys one EDC connector, which communicates to the Data Space. In addition, they all deploy a Kafka message broker, which is used to create the required data flow similar to a synchronous data exchange.

In this demonstrator, data from the services and agents do not directly get pushed to the connector, but first sent it to the local Kafka instance, which is a message based service similar to the MAS demonstrator. Because of this, all services and agents needed a minor

update to be able to communicate to Kafka. The connector was also enhanced by an extension that allows Push communication via Kafka services. This extension was developed during the research project and published in the official EDC extensions library.

As denoted in the figure, this demonstrator is based on receiving location data from trailers, publishing them in the Data Space and forwarding a request for ETA prediction to the corresponding service. To allow this type of communication, the demonstrator has a two step communication protocol similar to the MAS demonstrator. First, all (bi-)directonal data exchanges need to conclude contracts via the EDC to allow defined data access. For this setup, it requires participants to conclude contracts between the following parties:

- *Krone* → *DFKI*: Data push, as soon as new location data from related trailers is available.
- *DFKI* → *TU Dortmund University*: Data push to create an eta prediction request.
- *TU Dortmund University* → *DFKI*: Data push after calculating an eta prediction.

As one can see, for a typical synchronous HTTP Rest call, where parameters like location data is sent with the request, the Data Space requires significant overhead. The reason for this is that the HTTP call needs to be split into two data flows between the data providers and the data sink. One is the prediction request, which would usually be the payload of the HTTP Rest call and the other one is the response. Upon signing the contracts, each participant defines the source and destination endpoints of the data. In this scenario, those are all different Kafka services with a unique topic, such that it is explicit which topic represents a data source or data sink and is related to a concluded contract. Based on this registration process, the full data flow of the business use case is defined and the services can communicate as intended.

The Kafka message-based push communication is mandatory to simulate a synchronous HTTP call, as the service needs to push a HTTP request body to the other party (in this case the ETA service). If this were the standard data collection protocol, as is usually used for pulling data from a data source, the authors could not guaranty nearly synchronous communication. Therefore, some sort of push communication needed to be implemented in the demonstrator to simulate a synchronous HTTP Rest call. For this, the authors used the message-based approach, but could also have used an HTTP push communication similar to the IDS connector. Figure 4.6 shows an abstract data flow with the Kafka service. It shows a sequence diagram with two participants indicated by gray backgrounds. The one participant provides new data to the Data Space and the `Consuming` side receives the data from the provider.

To illustrate the data flow using Figure 4.6, we will take a closer look at the contract concluded between DFKI and TU Dortmund University, i.e. the request for ETA prediction. To provide the request to the Data Space, it is first sent from the Backend-Service (Trailer-Agent) to the local Kafka service. The Backend service is required to send the data to the topic, specified during the registration process. The reason for this is that the

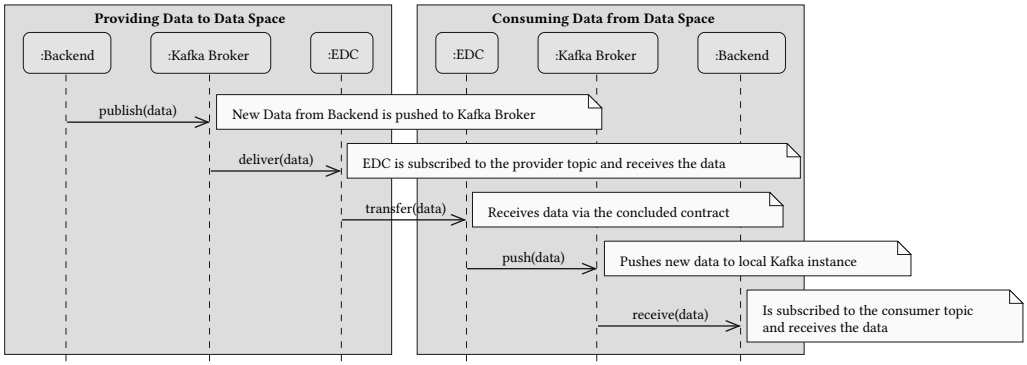


Figure 4.6: Data flow with Kafka service between services and Data Space as described in Sachweh et al., 2025b.

connector subscribes to this topic and is automatically notified when new data are sent. Based on the topic, the connector decides to which of the contracting parties it sends its data.

The receiving data process shown in Figure 4.6 is initiated with the `transfer(data)` call. As this demonstrator uses the Push methodology via Kafka, the new data is pushed to the consuming connector. Thus, it does not request new data from the data sources. As soon as new data are available for the connector, it pushes them directly to the local Kafka service on the consumer network. For the described scenario, this is the network from the TU Dortmund University, which uses the request for ETA prediction to calculate new ETA responses. During the registration and contract concluding phase, everything is configured such that the topic is specified, where new data from the ETA prediction request need to be forwarded. Consequently, the Backend-Service knows which topic to listen for new incoming requests.

At this point, only one way of the EDC data flow communication is described. However, the other ones are similar, except that they listen and push data to different Kafka topics. Thus, it is not necessary to describe the rest of the data flow. The important part to mention is that contexts get lost due to the asynchronous Data Space architecture. This means that direct matching cannot be made between an ETA prediction request and the corresponding response. To address this, the request and response data payloads are extended with a message specific Universally Unique Identifier (UUID) to allow for the matching.

4.5 FedDScon Framework

The previous sections presented different demonstrators that were mostly defined and configured by hand. Every service was manually started and the contracts between connectors were closed manually via Postman or Insomnia REST requests. This methodology is not suitable for larger scale set ups and does not allow good reproducibility.

To allow reproducible results, the paper from Sachweh et al., 2025b not only provided a sample demonstrator with EDC connectors for message based communication, but also offered a framework called FedDScon. The code for the framework is published in Sachweh et al., 2025a and can be used to set up a minimal Data Space with EDC connectors.

The framework allows to configure all necessary service components for a minimal Data Space configuration between two participants. With the framework a data asset can be provided, which the other connector can consume. This process is automatically managed after a necessary framework call is made.

As FedDScon was proposed with the EDC demonstrator, it uses the EDC connector in version 0.6.4 with HTTP and Kafka communication protocols already included. The EDC connector also emerged from the evaluation as the most suitable connector, as it is the best documented, divides the data flow strictly according to the distributed and asynchronous Data Space architecture and can be expanded as desired using provided or custom implemented libraries.

In the following, FedDScon with all features gets described in detail. To do so, first, the application scenarios of FedDScon and how the framework can be used will be described. The working method for the automated part of the framework is then presented along with a sequence diagram. To conclude the description, the repository structure and provided services are detailed along with the tasks each service performs. Finally, this work outlines the usage of FedDScon with the necessary configurations and commands.

4.5.1 FedDScon Features and Capabilities

FedDScon allows a Data Space to be created between two parties as MVD. This MVD Data Space, once created, consists of at least two connectors and if applicable, an authentication service for the MVD scenario. Additionally, FedDScon can start Kafka services on each participant's network to provide asynchronous messaging via a Kafka message broker. The framework uses the integrated catalog feature of the EDC to provide assets to the other connector. It provides the base configuration with the Kafka extension of the EDC in version 0.6.4 and can be modified to custom needs. With the base setup, an MVD can be created where data can be pushed or pulled between connectors via Kafka or HTTP REST. For the service side, FedDScon provides sample connectors, the Kafka service and possible interfaces to federation services like DAPS or the Data Space Catalog.

Besides the services itself, FedDScon provides sample configurations for deployment of the services in Kubernetes via Helm, Docker-Compose or directly on host systems, which allows one to deploy the MVD in nearly every modern environment. Some configurations need to be made to change the configuration to match the new environment, but this can be handled via a configuration process provided by FedDScon.

The configuration process changes the templates of assets and policies provided with the framework. Those templates are also provided to use for demonstration or to be able

to adapt to custom needs for specific data exchange protocols or multi-directional data exchange. Templates for assets and starting a transfer are especially relevant as they can be configured to use different data sources and targets behind the connectors.

With the configured templates, the framework also allows for fully autonomous creation of a contract between the two demonstrator connectors. To reach this target, the autonomous process first creates multiple assets on the provider side, after which the consumer requests the resource.

Thanks to these versatile features and the resources provided, FedDScon can be considered as a comprehensive framework for the development of Data Spaces. The complexity can be significantly reduced by the many examples and automations, allowing a quick introduction to Data Space technology based on the EDC.

4.5.2 Automated Process for Data Space Communication Establishment

One major feature of FedDScon is the automated establishment of a data transfer pipeline, which is based on a contractual agreement. To achieve this goal, multiple steps need to be executed. FedDScon allows one to run these steps autonomously until the data transfer pipeline is created. The steps can be divided into two parts. First, the provider needs to create assets, which are published in the Data Space by pushing the asset to the federated catalog. In the second phase, the consumer needs to search the catalog and request a data access to transfer the data. In the following sections, those two autonomous Data Space flows are described along a sequence diagram, which shows the DFKI as providing party and TU Dortmund University as consuming party.

Autonomous Asset Provisioning into the Data Space

This part describes the automatic asset creation, that is only done on the provider side. Therefore, Figure 4.7 details only services from the provider side, such as the Kafka instance and the IDS connector. The figure shows a sequence diagram of the steps usually required by an administrator of the company to provide an asset to the Data Space. FedDScon fully replaces the administrator and handles all required requests on its own.

Asset Creation Asset provisioning begins first by defining the asset itself. To do this, FedDScon needs to do a POST request to the connector with the defined asset as a JavaScript Object Notation (JSON) payload. A sample representation with data related to the use case is shown in Algorithm 4.1.

This JSON structure consists of four main sections. The first is the definition of the context. Each message related to the EDC connector has such a section, which allows to define the available fields in the rest of the message. For example, Algorithm 4.1 references the w3id.org/edc/v0.0.1/ns/ standard on line 3 and therefore, this message needs to conform to the edc namespace version of v0.0.1.

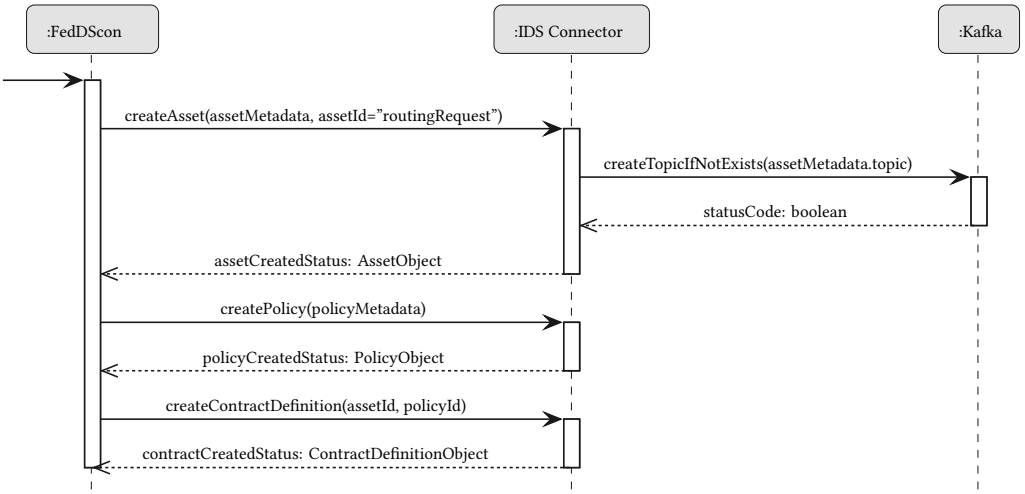


Figure 4.7: Create Request Asset (when executed by an administrator). FedDScon replaces a typical administrator by autonomously executing required requests.

The block from line 5 to line 10 is specifically related to the creation of an EDC asset. For such an asset, it is mandatory to provide some specific data, like an id, in this case the `routing-request-asset`. Also, the name (line 8) and content type (line 9) of the asset, which get provided to the Data Space are useful to specify for others to find the asset in the catalog.

The most relevant information is contained in the block named `dataAddress`, which contains a reference to the provided data. For a single file, this could be the path to the file or, like in this case, a dynamic data source. A dynamic data source could be an HTTP endpoint, or as configured here a Kafka service, which provides new data. The definition of the data source needs to define a type first (line 13) and then all mandatory configurations to consume the data source. For Kafka, these are the `kafka.bootstrap.servers` and `topic` attributes that refer to the running Kafka service instance. FedDScon will automatically fill the right url and topic in the JSON structure to ensure the right configuration for the related Data Space.

In the last section on line 21, each EDC request body contains a `properties` section that can provide additional information but is not mandatory. The following requests keep the same shape of request body, but with some other attributes added. For this reason, only relevant attributes that the framework sets dynamically are mentioned for further requests. Regarding the request structure, please refer to either the FedDScon templates or the official EDC documentation.

While the post request with a request body similar to Algorithm 4.1 is executed, the DFKI connector will check, if the topic specified in line 18 exists. If not, the connector will create the topic. Once everything was executed successfully, FedDScon receives feedback from the connector with a success message and an id which the created asset got.

Algorithm 4.1 EDC request body for creating an asset in the Data Space.

```
{
  "@context": {
    "edc": "https://w3id.org/edc/v0.0.1/ns/"
  },
  "asset": {
    "@id": "routing-request-asset",
    "properties": {
      "name": "Asset for creating routing requests",
      "contenttype": "application/json"
    }
  },
  "dataAddress": {
    "type": "Kafka",
    "properties": {
      "name": "Routing request data address",
      "kafka.bootstrap.servers": "<<kafka-host>>",
      "type": "Kafka",
      "topic": "routing-request"
    }
  },
  "properties": {}
}
```

Policy Creation Similarly to asset creation, a policy needs to be defined as well. This is done again by executing an automatic POST request to the connector with relevant policy information. Part of the mandatory information are a unique id specified with `@id` as well as the policy itself. As policies can be dynamic and arbitrarily complex, the EDC uses an existing framework called the ODRL, which can express a variety of policies. For testing purposes, FedDScon provides a simple policy to allow access to an asset as soon as a contract is concluded. However, the provided policy can be adapted freely to define custom policies along the ODRL language.

For this request, the connector evaluates the conformity of the ODRL policy specification and tests the uniqueness of this entity. When successful, FedDScon receives the response of the connector with the id of the created policy.

Contract Definition Creation After assets and policies are automatically created, the connector has no knowledge of the relationship between each other. Therefore, it is mandatory for the catalog publishing to create a contract definition. This contract definition is, once again, an EDC POST request similar to the one in Figure 4.7 from FedDScon to the connector. There, a contract id is specified, as well as the ids of an `accessPolicyId`, `contractPolicyId` and the asset id for the `assetsSelector`. By adding these ids, the

access to the asset is checked by the policy for each request after the contract is concluded.

With the successful response of the connector the contract definition gets created and the connector adds the defined asset with all related information, like policy, data access operations, etc. At this point, FedDScon finished the automated asset creation and provisioning to the Data Space on the provider side and now has to automate the contracting part on the consumer side.

Autonomous Contract Concluding of Assets

The process of concluding a contract inside the Data Space is initiated on the consumer side. For the sample demonstration of an eta request asset, the TU Dortmund University wants to consume this asset from the DFKI. This process is shown again in a sequence diagram in Figure 4.8, where the consumer side of services is represented primarily. The consumer related connector, Kafka and FedDScon are depicted, as well as the provider connector, which is required for the contract and data transfer. In this case, the FedDScon replaces a person that usually executes the process manually, by the automated and preconfigured framework feature to establish a data transfer between two connectors.

Fetch Catalog The first step to create this type of communication is to request the current state of the catalog. This is shown in Figure 4.8 by calling the right interface on the TU Connector such that this connector requests the Data Space catalog. For the MVD scenario, this requires querying the decentralized catalog located inside the DFKI Connector, which contains the previously created asset.

The response of the DFKI Connector contains a list of all available assets with referenced policies, names and data exchange protocols. In general, all information, which was previously defined, can be seen in this catalog, which is forwarded to FedDScon.

Contract Negotiation and Agreement The framework uses the information it has about the asset, such as its name, to retrieve the corresponding asset id from the catalog. With these data, the data consumer can start the contract negotiation process, which is handled autonomously in an asynchronous manner between both connectors.

As a final result of this contract negotiation, FedDScon receives a response from the TU Connector, which contains a `contractAgreementId`. This id is used by the framework to get the current contract agreement itself, which contains another id that is required to start the final data transfer.

Data Transfer The transfer can be initiated after all previous steps were executed successfully. FedDScon uses the returned id from the contract agreement to request a

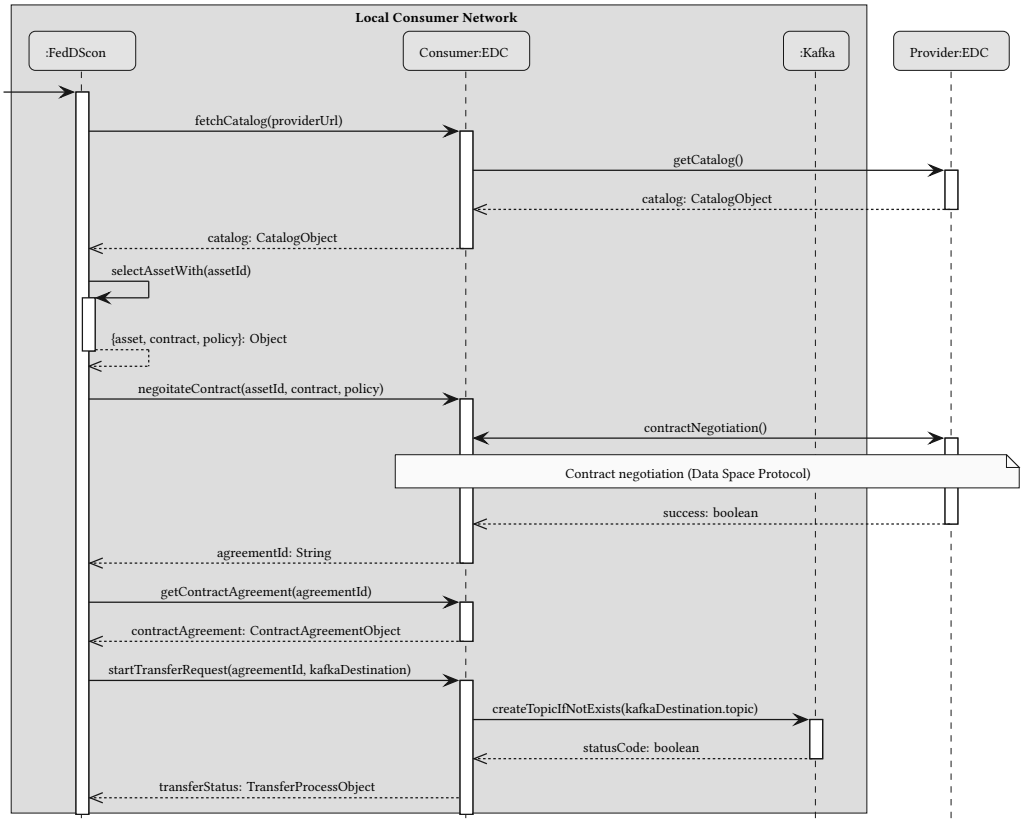


Figure 4.8: Concluding Contract between two participants with FedDScon.

`startTransfer`. Similarly to the asset definition, `startTransfer` requires some more configuration to create the data flow as intended. First, it needs to define which underlying contract is used for the data transfer. Second, FedDScon must specify a possible destination for the data received. This could be HTTP interfaces or, in this scenario, a topic on the Kafka service. FedDScon, once configured correctly, will automatically set the right endpoints for a contract and create the topic on the Kafka service, while the transfer between the connectors is started.

The `startTransfer` request as shown in Figure 4.8 only returns an id of the transfer, while the transfer itself runs automatically in the background if no errors have occurred.

However, sometimes one would like to know if the transfer still runs or if errors occurred in the contract-closing phases. Therefore, the framework enables administrators to automatically request the status of the transfer. The framework uses the last established contract and data flow for `transferStatus` evaluation.

Advantage over typical Data Space Interfaces

Besides FedDScon there exist several other options to configure Data Spaces and assets itself. Especially EDC connectors have User Interface (UI) services, which can be connected. However, these UI services take over a large part of the process, so users no longer understand which steps are necessary to complete contracts. FedDScon addresses this issue by displaying each step individually, including the responses from the connectors.

Alternatively, a connector could be configured manually. This allows a person to learn how it works very well, but for many assets in particular, this is a redundant task in the long term that can be automated easily. In this respect, FedDScon also offers the solution that, although the process is explained in fine detail, partial automation on both sides (provider and consumer) enables a significant increase in productivity.

4.5.3 Provided Services and Templates of FedDScon

FedDScon not only provides a system for simple configuration of data assets, contracts and EDC connectors in general, but can also be used for the deployment of MVD Data Space services. For the services and templates provided, the repository contains multiple structured directories, which as well as the provided features, are detailed in the following sections.

Connector

The demonstrators described previously already indicate that EDC connectors are highly relevant for an MVD Data Space. For this reason, FedDScon not only provides a Docker container that implements an exemplary connector, but also a basic template as a Java project for the connectors. This basic template is located in the `connector` directory and uses Gradle for dependency management. It already contains a federated catalog extension to communicate with Gaia-X catalogs and also includes extensions for HTTP and Kafka communication. For additional features, one would need to modify the `build.gradle.kts` file to add other edc libraries.

With the provided template, FedDScon shows how to embed dependencies from official sources and how to include custom developed libraries, like the Gaia-X catalog extension, which is also part of this repository and can be found in the `roms` directory.

Messages

In addition to connectors and the automated process for creating assets, it is also important to understand how communication with the EDC connector works. Therefore, FedDScon offers the option to view every message sent to the connector. Each of the requests described in the process flow is located in the `messages` folder. There, one can see which request bodies are sent to the connector, for the process flow to work.

These messages can also be modified to accommodate custom needs. If one then uses FedDScon's automated process flow, the modified messages get used to configure assets and policies accordingly. Some of the fields in the messages are automatically set by the framework and are therefore required to be kept as is, while others can be changed like the data source, for example, which can be modified to use mqtt as service if the connector supports this type of data exchange.

Docker Related Configuration

FedDScon not only provides service templates and messages, but can also deploy the services to a production ready state. To do this, FedDScon uses a container based approach, which uses Docker as build pipeline. Therefore, the repository contains a `Dockerfile` that contains all the necessary definitions and configurations to build an Open Container Image (Open Container Initiative (OCI)) of the EDC connector. This image can then be used for running the connectors so that they do not need to run directly on hosts hardware.

Deployment Options

To this point, the described Framework provides a lot of tools to configure as well as build services and the resulting Data Space. However, the execution of the configured services is also a big part of FedDScon, as without running service instances the full MVD Data Space could not function. Therefore, FedDScon provides different deployment options for testing and production deployments.

First, there is the straightforward deployment option to simply run the services (EDC connectors) directly on the operating system. This option uses the connector directory to build the connector with Gradle and afterwards runs this instance with an `application.properties` file which configures the EDC accordingly.

An alternative is a deployment option based on the OCI container image. This image can be used to run the EDC connectors with Docker or Docker-Compose as virtualized services. For Docker-Compose FedDScon provides a sample deployment configuration file, which contains two connectors, as well as a Kafka service for MVD deployment. All required configurations of the services are also present in this sample file.

Similarly to the Docker-Compose deployment, FedDScon provides one last deployment feature. The EDC can be configured to run inside a Kubernetes cluster. For this, the framework provides Helm charts, which define the full EDC deployment with default configurations. Possible override values are defined in the `helm` directory in the `connector-override-values.yaml` file. With this template, the EDC connectors can be deployed directly with all mandatory configurations to a cluster with the following command:

```
helm install -f ./helm-charts/connector-override-values.yaml provider
./helm-charts/connector
```

This kind of deployment is best suited for production deployments, as one gains all benefits from using a cluster, like fault tolerance, as well as re-deployments on errors or scalability.

4.5.4 FedDScon Usage

All previous explanations of FedDScon's demonstrators and features reveal that the framework provides very extensive functionalities and offers all necessary features for MVD Data Spaces.

However, the usability of templates and services is only sufficient as long as the complexity can be kept low for beginners. Therefore, the framework provides an easy introduction to the use of all these features by primarily offering two entry points for MVD Data Spaces with FedDScon.

One entry point is a script that allows the automatic configuration of all message templates and deployment configurations. The usage of this entry point script is shown in Algorithm 4.2, where `./setup.sh` calls the entry point bash script.

Algorithm 4.2 FedDSGcon tool calling for configuring endpoints of the demonstrator Data Space.

```
./setup.sh  --kafka kafka:29092 \
  --provider-management <url>/management \
  --provider-control <url>/control \
  --provider-public <url>/api \
  --provider-protocol <url>/protocol \
  --consumer-management <url>/management \
  --consumer-control <url>/control \
  --consumer-public <url>/api
```

The entry point allows to configure deployments and message templates according to url or asset name changes. Algorithm 4.2 for example changes the kafka service urls as well as all provider and consumer endpoints of the connectors. With the execution of this script, the deployment configurations in host, Docker or Helm based scenarios get changed, as well as the request bodies for the automatic asset creation and contract concluding process flow. This is required if someone changes deployment scenarios, as in Docker, the connector url changes from `localhost` to the name of the container (e.g. `provider`). In the sample provided, the `<url>` represents a placeholder for each custom url of the provider and consumer connector to reference the right targets.

With this entry point script, the complete MVD Data Space scenario can be quickly configured for every user, regardless of whether they are experienced or inexperienced. It is also possible to use the configuration option for existing running connectors to enable asset and contract completion for them.

In addition to the `setup.sh` entry point, the `run.sh` script serves as the second entry point for FedDScon beginners. It can be used to build and run the configured services. Therefore, the run script has the following options while executing:

- `build`: Builds the connector container that can afterwards be used to deploy in Docker or Kubernetes.
- `kafka`: Runs a preconfigured Kafka service based on the `apache/kafka-native` implementation.
- `docker-containers`: Starts the custom EDC Connectors as Docker containers in watch mode, such that the logs can be seen for further testing.
- `provider`: Runs a single Connector as provider with configured parameters from a `properties` file inside the FedDScon in an active shell.
- `consumer`: Works analogously to the provider command; however, starts a consumer with the consumer `properties` configuration.
- `messages`: Starts executing all messages to establish a bi-directional message-based communication between a provider and consumer Connector. It uses the parameters defined in the messages for the creation of assets and policies.

In a way, `run.sh` can be seen as the initial entry point. With `run.sh`, a MVD Data Space can be created, while automated asset creation and contract negotiation can be carried out. Therefore, both entry points are mandatory for beginners to get started with an MVD Data Space based on EDC connectors.

4.6 Summary

This chapter provided a systematic comparison between IDS and EDC connector technologies, a set of scenario-based demonstrators for empirical validation and the introduction of the FedDScon framework as a practical basis for creating Federated Data Spaces. Together, these elements established EDC-based MVDs as the central foundation for subsequent Federated Learning architectures and experiments.

4.6.1 Systematic Comparison between IDS and EDC

The structured evaluation of IDS and EDC focused on their suitability for FL tasks in Gaia-X compliant Data Spaces. The IDS connector offers a comparatively high level of abstraction and thus simplifies usage, but suffers from limited documentation quality, software maturity issues and reduced adaptability to emerging Data Space technologies. In contrast, the EDC follows the current Data Space Protocol and Gaia-X specifications more closely, exposes a modular and extensible architecture and is explicitly designed to integrate with upcoming Data Space and federation services. Although its configuration

and operation are more complex, this complexity is largely a result of its broader feature set and higher degree of flexibility.

The systematic comparison further revealed that IDS and EDC differ significantly in terms of extensibility, protocol alignment and suitability for evolving FL scenarios in federated Data Spaces. While IDS-based solutions can, in principle, be extended to support additional interaction patterns, their lack of extensibility limit their maintainability and reuse in heterogeneous ecosystems. EDC, on the other hand, is built around a clear separation of concerns between control and data planes, provides a well-defined extension model for integrating additional data plane technologies (such as HTTP and Kafka) and is continuously evolving with emerging Gaia-X and Data Space Protocol specifications. From this systematic evaluation, EDC therefore appears to be the more suitable connector technology for modern Data Space environments, particularly when long-term interoperability, extensibility and alignment with federated governance models are required.

4.6.2 Scenario-based Connector Evaluation

The scenario-based evaluation operationalized the theoretical comparison by implementing multiple cross-organizational demonstrators for logistics and FL oriented use cases. In the first group of demonstrators, IDS connectors were used to integrate ROS2-based multi-agent systems for autonomous parcel logistics with Gaia-X-equivalent services such as organizational credential management, identity, authentication and authorization across company boundaries. These implementations showed that IDS connectors can successfully bridge proprietary interaction protocols and synchronous request–response interactions into an originally unidirectional data sharing model, albeit at the cost of custom metadata extensions and protocol mappings.

A subsequent demonstrator implemented the long-haul freight scenario using IDS and EDC connectors in combination with Kafka-based messaging. Here, services and agents published and consumed data via local Kafka brokers, while EDC instances managed contracts, policies and data plane integration. This setup preserved data sovereignty and Gaia-X conformity and it demonstrated that EDC's extensible architecture can support complex, FL oriented topologies through modular data plane extensions. In sum, the demonstrators empirically validated that EDC provides a more sustainable and adaptable foundation for federated Data Spaces than IDS, particularly when realistic, asynchronous and cross-organizational data exchange patterns are required.

4.6.3 FedDScon Framework

Building on these insights, the FedDScon framework was introduced to systematically reduce the operational complexity of working with the EDC and to provide a reproducible basis for setting up federated Data Spaces. FedDScon replaces error-prone, manually configured demonstrators by offering a reusable framework that automates asset provisioning, contract negotiation and data transfer setup between two EDC participants. Its

core purpose is to enable researchers and developers to concentrate on higher-level Data Space and FL logic, instead of dealing with low-level REST calls, connector orchestration and detailed configuration.

From a functional perspective, FedDScon creates a minimal Data Space between two parties that includes at least two EDC connectors, optional authentication and federation services, while supporting both HTTP and Kafka-based communication in the evaluated EDC version. It automatically provisions data assets, publishes them to a federated catalog, discovers them on the consumer side, negotiates contracts and initiates transfers, thus covering the complete lifecycle from asset definition to active data flow. This is realized through an automated two-phase procedure in which the provider side first creates assets, policies and contract definitions, and the consumer side then fetches the catalog, negotiates a contract and triggers data transfers, programmatically executing all required EDC API calls while keeping the interaction steps transparent.

FedDScon further offers templates, services and deployment options that facilitate reuse and portability. It includes example EDC connector templates (Java/Gradle) with extensions for federated catalogs, HTTP and Kafka, as well as sample deployments of Kafka brokers and optional Gaia-X federation services such as catalog extensions. Multiple deployment targets are supported, ranging from localhost setups over Docker/Docker-Compose to Kubernetes via Helm charts, so that the same Data Space scenario can be executed on developer laptops or production-grade clusters with only minor configuration changes. Configuration and entry points are handled through JSON templates for assets, policies and transfer requests, which keep all mandatory EDC fields preconfigured while allowing adaptation to different data sources, topics and protocols. Two main scripts act as entry points for beginners: one script rewrites all configuration and message templates for a specific environment and the other builds and runs connectors, Kafka and the automated message flow, enabling complete Data Spaces to be set up and executed with only a few commands. By combining EDC's extensible architecture with automation, Kafka integration, catalog usage and flexible deployment, FedDScon provides a practical starting point for prototyping FL-oriented Data Spaces.

4.7 Discussion

The systematic comparison between IDS and EDC, complemented by the scenario-based demonstrators, highlights a clear shift from conceptually mature but rigid connector technologies towards a more extensible and ecosystem-aligned architecture. IDS connectors contribute a useful abstraction layer and have proven capable of supporting cross-organizational data exchange in controlled scenarios, but their limited extensibility, fragmented documentation and weaker integration with emerging Gaia-X and Data Space Protocol specifications restrict their long-term applicability. EDC, in contrast, embodies many of the lessons learned from IDS by separating control and data planes, implementing the Data Space Protocol natively and offering a modular extension model that can integrate diverse data plane technologies such as HTTP and Kafka. This positions EDC

as a central building block for future Federated Data Spaces, especially when interoperability, data sovereignty and alignment with Gaia-X compliant federation services are required.

Within this context, the contributions of this thesis extend beyond a systematic comparison of connector technologies and can be summarized more explicitly as follows:

- **Extensive connector evaluation:** A systematic evaluation of IDS and EDC was conducted, combining theoretical analysis with multiple practical demonstrators. This dual perspective correlates architectural properties with observable behavior in realistic use cases (e.g., ROS2-based parcel logistics and long-haul freight scenarios), thus revealing concrete strengths and weaknesses of each connector technology.
- **Design and implementation of FedDScon:** Building on the evaluation results, the thesis introduces **FedDScon** as a comprehensive framework that addresses all requirements from **Research Question 1**. **FedDScon** automates asset provisioning, catalog publication, contract negotiation and data transfer setup between EDC participants, thereby encapsulating complex connector interactions in a well designed framework.
- **Operationalization of extensible Data Spaces:** By supporting both HTTP and Kafka-based communication, integrating federated catalogs and optional Gaia-X federation services and offering multiple deployment options (localhost, Docker/-Docker-Compose, Kubernetes/Helm) along with configurable JSON templates and entry scripts, **FedDScon** turns EDC's theoretical extensibility into a practically usable foundation for federated Data Spaces.
- **Foundation for subsequent PPML methods: FedDScon** is adopted as the baseline execution environment for all subsequent PPML methods developed in this thesis. The PPML architectures, protocols and algorithms in the following chapters are instantiated, orchestrated and evaluated on top of **FedDScon**-based MVD Data Spaces, ensuring reproducibility and direct applicability to Gaia-X oriented Data Space ecosystems.

In addition, **FedDScon** serves as both a didactic and exploratory tool. On the one hand, it hides low-level configuration and REST interaction details from users primarily interested in experimenting with FL and PPML algorithms. On the other hand, it exposes the underlying interaction steps and responses so that advanced users can inspect and understand EDC-based Data Space behavior. In doing so, it enables rapid prototyping of Minimal Viable Data Spaces while maintaining transparency over data sovereignty, policies and protocol interactions. As a result, **FedDScon** is the main contribution that addresses **Research Question 1** and builds the foundation for the following chapters, in which the proposed FL and PPML methods are designed, instantiated and evaluated.

5 Privacy-Preserved Federated Learning in Cross Enterprises

*The previous chapter introduced the FedDScon framework as a comprehensive solution to **Research Question 1**, demonstrating how federated coordination, governance and lifecycle management for cross-enterprise analytics can be implemented in practice. However, while FedDScon provides the architectural and organizational foundation for such workflows, it does not guaranty by itself that the underlying ML procedures satisfy formal privacy requirements for individual records or sensitive model updates.*

*This gap motivates **Research Question 2**, which asks how expressive ML models can be trained collaboratively in federated and cross-organizational settings while preserving strong privacy guarantees. In contrast to the architecture-centric perspective of the previous chapter, the focus here is on the algorithmic design of privacy-preserving learning schemes that integrate DP and HE directly into the training process.*

*To this end, the chapter researches three complementary algorithms that jointly address **Research Question 2** as main contribution: **Differentially Private Learning from Label Proportions (DP-LLP)**, a differentially private variant of Learning from Label Proportions for fully distributed time series prediction introduced in Section 5.3; **Differentially Private Long Short-Term Memory (DP-LSTM)**, which extends DP-LLP with deep Long Short-Term Memory (LSTM)-based temporal encoders to increase model capacity under DP, presented in Section 5.4; and **HEX-FL**, a homomorphically encrypted Federated Learning approach that protects model updates during aggregation, described in Section 5.5. Together, these methods form a coherent privacy-preserving ML toolkit that combines advanced model expressiveness with rigorous statistical and cryptographic protection in federated environments. Their overall impact is summarized in Section 5.6 and discussed in Section 5.7. The discussion also includes the main contributions of this chapter, especially aimed at **Research Question 2**.*

5.1 Introduction

The increasing interconnection and interoperability of data ecosystems across enterprises, as enabled by emerging Data Space architectures, have created new opportunities for collaborative ML and advanced analytics in cross-organizational settings (Kairouz et al., 2021). In the previous chapter, Data Spaces were introduced as a novel interaction paradigm that enables organizations to share as well as access data along with services

in a governed, interoperable and sovereign manner. This paradigm provides the foundation for new forms of cross-enterprise collaboration, where data-driven value chains and joint innovation processes can emerge without forfeiting autonomy or control. The key to realizing this potential is the development of mechanisms that allow organizations not only to exchange static data but to dynamically deploy, train and serve ML models within such ecosystems while adhering to regulatory and contractual obligations. The shift from viewing Data Spaces merely as an integration and exchange layer toward considering them as a secure execution environment for privacy-aware, model-centric services marks a pivotal evolution in the design and utility of federated data ecosystems (Kairouz et al., 2021).

However, cross-enterprise collaboration in data analytics and ML faces substantial challenges due to stringent legal, ethical and organizational constraints on data usage. The European GDPR provides a particularly influential framework, imposing explicit restrictions on the processing of personal data and mandating the implementation of technical and organizational safeguards to ensure privacy, accountability and transparency (Voigt and Von dem Bussche, 2017). Beyond personal data, organizations frequently handle sensitive operational, financial, or industrial information that is subject to confidentiality, intellectual property, or competition law constraints. Consequently, even when raw data remain within local infrastructures, the joint model training or prediction sharing process can still expose private information through indirect leakage channels such as gradients, model parameters or inference outputs (Shokri et al., 2017b; Zhu et al., 2019). These risks have been empirically demonstrated to enable reconstruction or membership inference attacks, in which adversaries can deduce properties of individual training samples or detect their presence in the training dataset.

In regulated or strategic environments, such vulnerabilities pose significant barriers to the establishment of trust between participants. They threaten compliance not only with legal frameworks such as the GDPR but also with the self-governed principles of Data Spaces, which emphasize transparency, accountability and sovereignty as preconditions for collaboration. Hence, a central challenge for the realization of Data Space based analytics is to enable distributed learning workflows that guarantee both compliance and confidentiality while retaining the benefit of collective intelligence across organizational boundaries. Achieving this balance requires a paradigm of privacy-aware ML, which integrates techniques that mitigate information leakage at both the algorithmic and system levels.

To address these challenges, this chapter focuses on federated and distributed learning paradigms specifically adapted to the architectural and governance characteristics of Data Spaces. FL provides a natural foundation, as it enables multiple parties to collaboratively train a global model by exchanging intermediate model updates rather than sharing raw datasets. This approach aligns strongly with the principles of locality, control and interoperability of the Data Space, since the data remain under direct control of their originators throughout the lifecycle of the model (Kairouz et al., 2021). Nevertheless, FL itself does

not inherently guarantee privacy, as model updates can still encode latent information about local data distributions or individual samples.

Therefore, this chapter extends FL with advanced privacy-preserving mechanisms that operate at both the statistical and cryptographic levels to strengthen protection against information leakage and adversarial inference and therefore provides a solution to **Research Question 2**. These mechanisms ensure that even during the collaborative training or aggregation phases, participants can contribute to the improvement of the joint model without exposing proprietary or personal data. From a governance perspective, this integration of algorithmic privacy-enhancing techniques with the federated data governance model of Data Spaces paves the way for trustworthy and regulation-aligned analytics infrastructures.

The proposed perspective situates Data Spaces not only as facilitators of data exchange but as secure computational infrastructures capable of supporting analytics and artificial intelligence processes under jointly controlled privacy and compliance constraints. By combining statistical noise injection strategies with encrypted computation techniques within a federated coordination framework, strong privacy guarantees can be achieved without prohibitive degradation in model performance. In this way, PPML emerges as a foundational building block for the next generation of Data Spaces, enabling GDPR-aligned, transparent and secure data-driven innovation across enterprise boundaries.

This chapter presents two main contributions that jointly advance privacy preservation in cross-enterprise FL within Data Spaces. The first contribution addresses privacy at the model level by integrating DP into the training process. Two customized DP algorithms are proposed that specifically adapt to the distributed and heterogeneous nature of Data Space environments, complemented by a generalizable approach designed for Deep Learning models. Together, these algorithms provide a flexible and principled mechanism to bound information leakage from model updates while maintaining high utility and compliance with regulatory privacy requirements.

The second contribution introduces the HEX-FL framework, which extends FL with HE to ensure confidentiality during model aggregation. By encrypting model weights before transmission and enabling secure aggregation without direct exposure, the framework provides strong cryptographic protection against inference and reconstruction attacks in collaborative settings. When combined, DP-based model protection and the HEX-FL aggregation scheme establish a multi-layered privacy architecture, that protects both data and model information throughout the FL workflow. This integrated approach directly addresses **Research Question 2**, providing a foundation for trustworthy and regulation-aligned ML within cross-enterprise Data Spaces.

5.2 Related Work

ML in federated Data Spaces is subject to particularly high requirements in terms of privacy, security and data sovereignty. For privacy-preserving ML, current approaches predominantly follow two main strategies: DP, which perturbs computations with calibrated noise to provide formal individual-level guarantees and HE, which enables mathematical operations directly on encrypted data without revealing plaintexts. In addition, distributed training in cross-organizational environments such as federated Data Spaces requires a coordination framework, where FL with Federated Averaging represents the state-of-the-art method for jointly training models without sharing raw data. The following subsections present these concepts and the corresponding state-of-the-art methods, implementations and frameworks for DP, HE and FL.

5.2.1 Differential Privacy

DP has emerged as the formal gold standard for quantifying and guaranteeing individual privacy during data analysis and ML (Dwork, 2006; Dwork and Roth, 2014). It provides a mathematically rigorous framework ensuring that the outcome of any computation is statistically indistinguishable whether or not an individual's data is included in the input dataset. This fundamental property makes DP robust to auxiliary information attacks that undermine traditional anonymization techniques, such as k-anonymity and l-diversity (Machanavajjhala et al., 2007; SWEENEY, 2002).

Foundations of Differential Privacy

The concept of DP was introduced by Dwork, 2006, defining privacy in terms of measurable limits on output distributions. A randomized mechanism M satisfies (ε, δ)-Differential Privacy if, for any pair of neighboring datasets D' and D'' differing in one record and for any subset S of possible outputs, the following inequality holds:

$$Pr[M(D') \in S] \leq e^{\varepsilon} Pr[M(D'') \in S] + \delta. \tag{5.1}$$

Here, ε represents the *privacy loss* or *privacy budget*, controlling how much an output distribution may differ when one individual's data changes. A smaller ε denotes stronger privacy, while δ is a slack parameter that allows for slight probability deviations. This definition ensures that the inclusion or exclusion of any individual has only a bounded effect on the output distribution.

The *global sensitivity* Δf of a query function f quantifies the maximum impact that a single record can have on the output:

$$\Delta f = \max_{\substack{D', D'' \in D, \\ ||D' - D''||_1 = 1}} ||M(D') - M(D'')||_1. \tag{5.2}$$

The amount of noise added to the results of f is calibrated proportionally to this sensitivity.

Mechanisms for Achieving Differential Privacy

Mechanisms for enforcing DP add random noise drawn from well-defined statistical distributions whose parameters depend on Δf and ε, where *Laplace* and *Gaussian* functions are the most common ones.

The Laplace mechanism is based on the Laplace distribution with probability density function (PDF):

$$lap(x|\sigma,\mu) = \frac{1}{2\sigma} e^{-\frac{|x-\mu|}{\sigma}}, \tag{5.3}$$

where μ is the mean and σ is the scale parameter. In DP, it is typically scaled as:

$$lap(x|\frac{\Delta f}{\varepsilon},0) = \frac{\varepsilon}{2\Delta f} e^{-\frac{|x|\varepsilon}{\Delta f}}. \tag{5.4}$$

The Laplace mechanism proposed by Dwork, 2006 satisfies pure ε-DP, while the Gaussian mechanism proposed in Dwork and Roth, 2014 provides (ε,δ)-DP, using Gaussian noise with a variance determined by the desired privacy budget and sensitivity.

Alternative formulations have been developed to improve composability and privacy accounting over multiple queries. Concentrated Differential Privacy (CDP) proposed in Bun and Steinke, 2016 and Rényi Differential Privacy (RDP) by Mironov, 2017 generalize DP using divergence measures that simplify analysis in iterative or multi-step algorithms. These frameworks have become essential in deep learning and large-scale data analyzes, where repeated computations would otherwise cause excessive consumption of privacy budgets.

Composition and Privacy Accounting

One of DP's core strengths lies in its composability: Privacy loss in multiple analyzes can be quantified and bounded. The sequential composition theorem (Dwork and Roth, 2014) states that applying k mechanisms each satisfying (ε_i,δ_i)-DP results in a general loss of privacy of $(\sum_i \varepsilon_i, \sum_i \delta_i)$. This property enables DP to be used in complex pipelines while controlling cumulative information leakage.

To refine utility in multi-round settings, the advanced composition and privacy amplification by subsampling have been introduced, proving that partial participation or random subsampling of records reduces the effective privacy loss. These results have strongly motivated the integration of DP into iterative algorithms such as gradient-based optimization.

Practical Applications of Differential Privacy

DP has found widespread application across domains involving sensitive data. Prominent examples include:

- **Official statistics and census:** The U.S. Census Bureau applied DP to the 2020 Decennial Census (Abowd, 2018), marking the first large-scale national deployment. Similar approaches are explored by Eurostat and other statistical agencies.
- **Search and recommender systems:** Companies such as Google and Apple have implemented local DP for telemetry collection and personalized suggestions without storing identifiable user data (Ding et al., 2017; Erlingsson et al., 2014).
- **Machine learning and AI:** DP is increasingly applied in model training and evaluation pipelines to mitigate information leakage from models (Abadi et al., 2016; Papernot et al., 2021). Here, noise is injected into gradients or objective functions, preserving privacy while allowing the model utility.
- **Healthcare and biomedical:** DP techniques enable sharing and analysis of privacy-preserving medical datasets (Ficek et al., 2021), where strict compliance with data protection regulations (e.g. GDPR, HIPAA) is required.

Current Challenges and Research Directions

Although DP provides formal guarantees, its practical adoption remains challenging, particularly when applied to distributed and federated settings. Selecting an appropriate ε is highly context-dependent and remains one of the most debated aspects of DP, as no universally accepted threshold defines a satisfactory balance between privacy and utility. Furthermore, applying DP in high-dimensional real-world datasets often leads to degraded model accuracy due to sensitivity-driven noise calibration. The integration of DP into FL scenarios introduces additional complexity: privacy guarantees must be maintained across multiple decentralized participants who perform local computations and exchange aggregated model parameters. Ensuring consistent and interpretable privacy budgets across heterogeneous clients is difficult, as each party may have distinct data distributions and sensitivity levels. In fully decentralized or cross-organizational settings, coordinating global privacy guarantees while honoring local data sovereignty remains an unresolved challenge. As a result, establishing governance frameworks, privacy accounting methods and interoperability standards for DP in these environments continues to be an open problem.

In general, DP has established itself as a foundational pillar in the study of data protection, uniting rigorous theoretical principles with practical mechanisms for privacy-preserving computation. Its broad applicability enables it to function as a link between privacy theory and a wide range of practical fields, spanning statistical analysis, artificial intelligence and more.

5.2.2 Homomorphic Encryption and CKKS

HE enables computations to be performed directly on encrypted data, allowing untrusted parties to process sensitive information without learning anything about the underly-

ing plaintexts (Gentry, 2009). In an HE scheme, ciphertexts support at least addition and multiplication operations that correspond to addition and multiplication on the underlying plaintexts, thus enabling the evaluation of arithmetic circuits in the encrypted domain (Brakerski and Vaikuntanathan, 2014). Modern HE constructions are typically based on lattice problems such as Learning With Errors (LWE) and Ring-LWE, which provide post-quantum security and efficient implementations suitable for data-intensive workloads (Lyubashevsky et al., 2010).

Classical FHE schemes, starting from Gentry's seminal construction based on ideal lattices, demonstrated the theoretical feasibility of arbitrary computation on encrypted data, but accompanied by prohibitive computational overhead (Gentry, 2009). Subsequent schemes such as the Brakerski-Gentry-Vaikuntanathan (BGV), Brakerski-Fan-Vercauteren (BFV) and leveled FHE variants improved efficiency by better managing ciphertext noise and omitting frequent bootstrapping in many practical settings (Brakerski and Vaikuntanathan, 2014; Fan and Vercauteren, 2012). However, these schemes are mainly tailored to exact integer arithmetic, which complicates their direct use in ML scenarios where real-valued computations and floating point semantics are predominant (Lee et al., 2022).

CKKS for Approximate Arithmetic

The CKKS scheme has been proposed as a leveled FHE scheme specifically designed for approximate arithmetic on real and complex numbers (Cheon et al., 2017). Instead of encrypting integers, CKKS encodes vectors of complex (or real) values as fixed-point numbers with a scaling factor, enabling approximate addition and multiplication with controllable approximation error. Formally, CKKS operates on a polynomial ring $R_q = \mathbb{Z}_q[\bar{X}]/(X^N + 1)$, where q is the modulus of ciphertext and N is the degree of the polynomial. A ciphertext consists of a pair $(c_0, c_1) \in R_q^2$ and decryption with secret key $s \in R_q$ approximately recovers the plaintext m through:

$$m \approx \mathrm{Dec}_s(c_0, c_1) = \left\lfloor \frac{c_0 + c_1 \cdot s}{\Delta} \right\rceil \bmod t \tag{5.5}$$

, where Δ is the scaling factor and t is the plaintext modulus controlling precision (Agrawal and Joshi, 2023; Cheon et al., 2017). Homomorphic operations correspond to polynomial operations on ciphertexts followed by rescaling and modulus switching steps, which keep the encoded values within the desired precision and control the growth of noise.

For ML, CKKS provides two essential advantages. First, it supports approximate real-valued arithmetic, matching the floating point computations used in linear layers, convolutions and polynomial approximations of activation functions (Lee et al., 2022). Second, it enables vectorized (Single Instruction, Multiple Data (SIMD)-style) packing, where multiple values are embedded into a single ciphertext, allowing efficient homomorphic evaluation of vector–matrix and matrix–matrix operations that dominate the runtime of

many ML models (Lee et al., 2022; Xie et al., 2024). Since most ML models are robust to small numerical perturbations, the approximation error introduced by CKKS is typically acceptable, making CKKS a natural choice for privacy-preserving ML over encrypted data.

Multiparty Homomorphic Encryption

Multiparty Homomorphic Encryption (MHE) extends single-key HE schemes to a multiparty setting where several participants jointly hold a distributed secret key while sharing a collective public key (C. Mouchet et al., 2021). Each party generates a secret key share and a collaborative key generation protocol computes a public key that can be used to encrypt data. Decryption then requires an interactive protocol in which parties compute and exchange partial decryption shares that are combined to recover the plaintext without revealing individual key shares (C. Mouchet et al., 2021).

Let $s = s_1 + \cdots + s_n$ be the secret key decomposed into shares $s_i \in R_q$ held by n parties. Given a ciphertext (c_0, c_1), decryption is conceptually based on the expression:

$$c_0 + c_1 \cdot s = \sum_{i=1}^{n} \left(\frac{c_0}{n} + c_1 \cdot s_i \right) \tag{5.6}$$

,where each party locally computes its partial term $c_0/n + c_1 \cdot s_i$ and contributes it to a collective decryption step (C. Mouchet et al., 2021). When instantiated with CKKS, MHE allows multiple organizations to jointly perform encrypted computations on distributed data while ensuring that no single party can decrypt intermediate results alone. This is particularly relevant in cross-organizational ML scenarios, where data holders wish to collaboratively evaluate models on combined datasets without giving up full control over decryption capabilities.

Rationale for CKKS in Machine Learning

ML models, especially deep neural networks, rely heavily on real-valued operations such as inner products, convolutions and non-linear activation functions that are most naturally implemented in floating point arithmetic (Lee et al., 2022). Although integer-based HE schemes can emulate these operations using fixed-point encodings, this generally requires careful parameter engineering, large ciphertext moduli and deeper circuits, which can significantly increase computational overhead (Fan and Vercauteren, 2012).

CKKS directly supports approximate real-number arithmetic, enabling more compact circuit representations and often shallower depth for typical ML workloads (Cheon et al., 2017). In addition, its packing and rotation primitives allow for efficient implementation of linear algebra kernels, which are essential for scalable encrypted inference and, in some scenarios, encrypted training (Lee et al., 2022; Xie et al., 2024). Consequently, CKKS has become a de facto standard in many privacy-preserving ML frameworks and studies that

target encrypted inference on neural networks or other models requiring floating point operations (Lee et al., 2022).

State-of-the-Art CKKS Libraries

Several mature open-source libraries implement CKKS and provide the tools necessary to deploy HE in practical ML analytics scenarios. These libraries differ in programming language, performance characteristics and support for multiparty extensions but share a common focus on efficient CKKS-based computation.

Microsoft SEAL and related implementations Microsoft Simple Encrypted Arithmetic Library (SEAL) is a widely used C++ library that implements BFV and CKKS, providing high-level APIs for parameter selection, encryption, homomorphic evaluation and decryption (Fawaz et al., 2021). SEAL has become a reference implementation for academic and industrial prototypes in privacy-preserving ML and is frequently used in empirical studies that evaluate HE performance and accuracy trade-offs (Lee et al., 2022). Closely related implementations and research prototypes extend SEAL or use its parameter sets to benchmark different HE-based ML pipelines (Tsuji and Oguchi, 2024).

Lattigo Lattigo is a Go-based lattice cryptography library that implements full-Residue Number System (RNS) variants of the BFV, BGV and CKKS schemes, providing optimized support for integer and approximate real-number arithmetic over encrypted data (EPFL-LDS, Tune Insight SA, 2024; C. Mouchet et al., 2021). Its modular architecture offers dedicated packages for each scheme (e.g. BFV, BGV, CKKS) together with shared low-level Ring-LWE primitives and configurable key-switching and bootstrapping procedures, enabling developers to tailor homomorphic computations to specific performance and precision requirements (C. V. Mouchet et al., 2020).

OpenFHE and other frameworks OpenFHE is a modern C++17 HE library that supports several schemes, including BGV, BFV and CKKS. It offers advanced features such as bootstrapping, scheme switching and hardware abstraction for acceleration (Badawi et al., 2022). It integrates CKKS as a first-class scheme for approximate arithmetic and provides comprehensive tooling for configuring parameters, managing noise growth and executing complex encrypted workflows (Badawi et al., 2022). Comparative studies and benchmarks indicate that SEAL, Lattigo, OpenFHE and related frameworks currently represent the state of the art in terms of functionality, performance and community support for CKKS-based applications (Tsuji and Oguchi, 2024).

Overall, HE and in particular the CKKS scheme with its support for approximate floating point arithmetic provides a practical cryptographic foundation for evaluating ML models on encrypted data. Together with multiparty variants and mature open-source libraries such as SEAL, Lattigo and OpenFHE, CKKS enables privacy-preserving analytics and ML

across organizational boundaries while keeping data in encrypted form throughout computation (Badawi et al., 2022; Lee et al., 2022; Xie et al., 2024).

5.2.3 Federated Learning

DP and HE provide strong mechanisms for protecting individual records and enabling computation on encrypted data, but they do not, on their own, define how ML models are collaboratively trained across multiple organizations. A framework is needed that coordinates decentralized training on local datasets while avoiding raw data sharing. FL addresses this, as it is a collaborative ML paradigm in which multiple clients jointly train a global model without sharing their raw data (Kairouz et al., 2021). Instead, each client performs local training on its private dataset and only exchanges model parameters or parameter updates with a coordinating server. FL therefore embodies data minimization principles and is particularly suitable for cross-enterprise and cross-silo scenarios where regulatory, organizational, or technical constraints prevent central data aggregation (Rieke et al., 2020).

Basic Federated Optimization Setting

Let $\mathcal{K} = \{1, \ldots, K\}$ denote the set of participating clients. Each client $k \in \mathcal{K}$ holds a local dataset $\mathcal{D}_k$ with $n_k = |\mathcal{D}_k|$ samples and the global dataset size is $n = \sum_{k=1}^{K} n_k$. The goal is to minimize a global objective function of the form

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \tag{5.7}$$

where w denotes the model parameters and $F_k(w)$ is the local loss function on the client k, typically defined as the empirical risk over $\mathcal{D}_k$ (McMahan et al., 2017). This formulation reflects the contribution of each client proportional to its dataset size and naturally captures heterogeneous data distributions across clients.

Federated Averaging

The Federated Averaging (FedAvg) algorithm (McMahan et al., 2017) is the de facto standard optimization method in FL. Training proceeds in communication rounds $t = 0, 1, \ldots, n$ coordinated by a central server. At the beginning of round t, the server holds the current global model w_t and selects a subset $\mathcal{S}_t \subseteq \mathcal{K}$ of clients to participate. The protocol can be described as follows:

1. **Broadcast:** The server sends the current global model w_t to all selected clients $k \in \mathcal{S}_t$.
2. **Local update:** Each client k initializes its local model with w_t and performs E Stochastic Gradient Descent (SGD) epochs or a similar optimizer on its local dataset $\mathcal{D}_k$, obtaining an updated model w_{t+1}^k.

3. **Aggregation:** The server collects the local models w_{t+1}^k from all participating clients and computes a weighted average to obtain the new global model:

$$w_{t+1} = \sum_{k \in \mathcal{S}_t} \frac{n_k}{\sum_{j \in \mathcal{S}_t} n_j} w_{t+1}^k. \tag{5.8}$$

This iterative model averaging can be interpreted as an approximate distributed gradient method that takes advantage of substantial local computation between communication rounds, thus reducing communication cost compared to fully synchronous distributed SGD (Kairouz et al., 2021; McMahan et al., 2017). The weighting by n_k ensures that clients with larger datasets have a proportionally higher influence on the global update.

Federated Learning in Cross-Enterprise Scenarios

In cross-enterprise and cross-silo environments, such as healthcare networks or industrial data ecosystems, data is typically stored in separate organizational silos and cannot be merged due to legal, competitive or technical constraints (Rieke et al., 2020). FL offers a framework for collaborative model training in these settings, enabling organizations to benefit from a shared global model while retaining control over their local data. Applications include medical image analysis, outcome prediction and other data-driven services where multi-institutional training datasets are critical to model robustness and generalization (Rieke et al., 2020).

However, by itself, FL does not provide formal privacy guarantees against inference attacks on exchanged model updates (Nasr et al., 2019). Therefore, practical deployments in sensitive cross-enterprise environments need to integrate other privacy methods, to overcome this issue.

5.3 Differentially Private Learning from Label Proportions (DP-LLP)

In Data Spaces and federated data ecosystems, time series data from Internet-of-Things (IoT) devices are often collected in a geographically distributed and organizationally fragmented manner. At the same time, regulatory constraints such as the GDPR require technical safeguards that prevent inference about individual behavior from shared analytical artifacts. In this section, a differentially private extension of the LLP paradigm is presented, which enables fully distributed prediction on time series data by exchanging only noisy label proportions between neighboring nodes. The proposed algorithm builds upon the LLP framework and extends it to a differentially private LLP (DP-LLP) version introduced in Sachweh et al., 2021, to which the author of this work has contributed fully. This DP-LLP algorithm can be interpreted as an instantiation of privacy-preserving distributed learning tailored to resource-constrained IoT deployments and Data Space environments.

5.3.1 Problem Setting and Motivation

Consider a set of m distributed sensor nodes $n_1, \ldots, n_m$ deployed in a road traffic network, each continuously measuring local traffic flow, e.g. vehicles per time interval. The objective is to jointly learn predictive models that forecast future traffic states, such as congestion levels at each node, without transferring raw time series data to a central server. Instead, each node stores its own measurements locally and only exchanges aggregated information with a limited set of neighbors, thereby reducing communication overhead and enhancing privacy. This setup is representative for data-driven services in federated Data Spaces, where data sovereignty and communication efficiency are crucial constraints.

The central challenge in this setting is to avoid privacy leakage through the exchanged aggregates. Although LLP already operates on aggregated label proportions rather than individual examples, the proportions can still reveal sensitive information about all individuals in a batch, especially in highly skewed regimes (Stolpe and Morik, 2011; Stolpe et al., 2015). For example, if all vehicles in a time window are classified into the highest speed category, then the corresponding label proportion vector reveals that everyone drove very fast during that period. To address this, the algorithm proposed in Sachweh et al., 2021 augments LLP with a rigorous notion of individual-level privacy using the DP framework (Dwork and Roth, 2014). The resulting method, denoted DP-LLP, perturbs the proportions of the exchanged labels with Laplace noise calibrated to the sensitivity of the batch-level counting function, thus providing formal guarantees that the contribution of any individual sensor measurement cannot be inferred from the communicated data.

5.3.2 Differential Privacy for Label Proportions

The formal foundations of DP, including neighboring datasets, (ε, δ)-DP, global sensitivity and the Laplace and Gaussian mechanisms, have been previously introduced in Section 5.2.1; in particular, this part refers to the definitions and discussion in Equations (5.1) to (5.3) and the accompanying exposition (Dwork, 2006; Dwork and Roth, 2014). Building on this background, the present section focuses on how these concepts are instantiated for the batch-wise label statistics used in LLP.

In the LLP setting, the randomized mechanism M is applied locally at each node and takes a batch B as input of labeled time series samples, outputting aggregate label counts or proportions that are then shared with neighboring nodes. The underlying query function f maps a batch B to the vector of label counts in a finite label set $\mathcal{Y}$ and its global ℓ_1-sensitivity:

$$\Delta f = \max_{B, B' : \|B - B'\|_1 \leq 1} \|f(B) - f(B')\|_1 \tag{5.9}$$

measures the maximal change of these counts when a single record in the batch is modified. For simple counting queries, changing one individual can increase one label count

and decrease another by at most one, which yields $\Delta f = 1$ and makes f an ideal candidate for the Laplace mechanism discussed in the related work section.

To obtain differentially private label proportions, a Laplace mechanism with privacy parameter ε is applied component-wise to $f(B)$, i.e.:

$$M(B) = f(B) + (Z_1, \ldots, Z_{|\mathcal{Y}|}), \tag{5.10}$$

where Z_i are independent Laplace random variables with mean 0 and scale $1/\varepsilon$, as introduced in the related works section. The resulting noisy label counts are subsequently clipped to a feasible interval and normalized to sum to one, yielding a probability vector over labels for each batch. In contrast to the abstract treatment of DP in the related work, this instantiation directly targets the batch-level label statistics exchanged in LLP and thus ensures that, for any fixed ε, the proportions of the communicated label reveal only a bounded amount of information about the presence or absence of any individual time series in the underlying batch. For repeated use in many batches and communication rounds, the composition results for DP can be used to account for the cumulative loss of privacy of the general DP-LLP training procedure.

5.3.3 LLP in a Fully Distributed Time Series Architecture

The starting point for DP-LLP is the distributed LLP algorithm for IoT sensor networks described in Stolpe et al., 2015 and Sachweh et al., 2021. In this setting, m wireless sensor nodes $n_1, \ldots, n_m$ are deployed in a spatial network and continuously collect local time series measurements such as traffic flow. Each node n_j maintains a local dataset:

$$D(j) = \{(x_{j,t}, y_{j,t}) \mid t \in \mathcal{T}_j\} \tag{5.11}$$

, where $x_{j,t} \in \mathbb{R}^p$ denotes a feature vector derived from a sliding window of raw sensor values and $y_{j,t} \in \mathcal{Y}$ is a discretized label representing the traffic condition at a future prediction horizon. Concretely, a window size w is chosen and for each time point t the feature vector is constructed from measurements in the interval $[t - w + 1, t]$, while the label $y_{j,t}$ corresponds to the traffic flow at time $t + r$ for some prediction horizon r (Sachweh et al., 2021). The label space $\mathcal{Y}$ is defined by partitioning the continuous traffic flow range into a finite set of intervals (e.g., low, medium, high), which yields a multi-class classification problem.

As illustrated in Figure 5.1, the local dataset $D(j)$ is organized as a matrix whose rows correspond to sliding windows of measurements and whose last column contains the associated labels $y(j)$. To apply LLP, each dataset $D(j)$ is partitioned into h non-overlapping batches $B_1, \ldots, B_h$ of equal size b. For each batch B_i, node n_j computes the empirical label histogram

$$c_i = (c_{i,1}, \ldots, c_{i,|\mathcal{Y}|}), \tag{5.12}$$

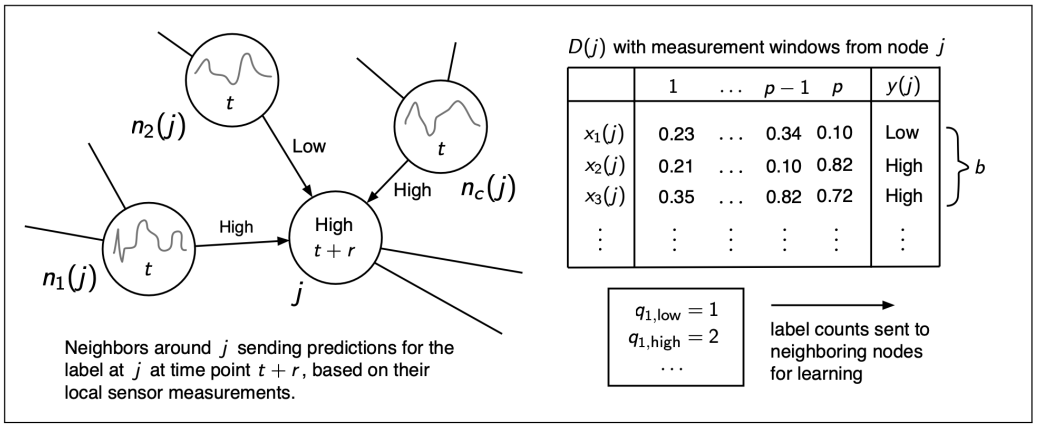


Figure 5.1: Illustration of the distributed LLP setting at node j: neighbors $n_1(j), \ldots, n_c(j)$ send predictions for the label at node j at time $t + r$ based on their local sensor measurements. The table on the right shows the local measurement windows $D(j)$ and the corresponding labels, which are aggregated into label counts (e.g., $q_{1,\text{low}}, q_{1,\text{high}}$) and sent to neighboring nodes for learning. Adapted from Sachweh et al., 2021; Stolpe et al., 2015.

where $c_{i,\ell}$ counts the occurrences of label ℓ in B_i. Normalizing c_i by the batch size yields the label proportion vector:

$$q_i = \frac{1}{b} c_i \tag{5.13}$$

and the resulting batch-wise label proportions are sent from node n_j to its c nearest neighbors $n_1(j), \ldots, n_c(j)$, determined, for example, by geographical distance.

Each receiving node n_k uses its own local feature vectors and the label proportions obtained from neighbors to train local LLP models via a clustering-based loss minimization procedure (Sachweh et al., 2021). More precisely, node n_k performs a k-means clustering on its local features $D(k)$ to obtain k clusters, assigns random initial labels to clusters and then iteratively refines these assignments so that the induced label proportions on local batches match as closely as possible the proportions received from each neighbor. This refinement can be implemented using a local search strategy with multiple random restarts. The outcome is a set of local classifiers $f_{k,1}, \ldots, f_{k,c+1}$, each trained with respect to one neighbor's label proportions, whose predictions for a given time point are finally combined by majority vote. This architecture is fully distributed in the sense that no central coordinator is required and only batch-level label proportions are exchanged between nodes, making it well suited for Data Space scenarios with strict locality and sovereignty constraints.

5.3.4 Differentially Private Computation of Label Proportions

The key modification of the original LLP algorithm consists in rendering the label information exchanged between neighboring nodes differentially private. In the distributed setting considered here, each node n_j partitions its local dataset D_j into h disjoint batches $B_1, \ldots, B_h$ of size b and for each batch computes empirical label counts and proportions over a finite label set $\mathcal{Y}$. Without additional protection, these batch-level label proportions can reveal sensitive information about individual trajectories, for example, if a batch happens to contain almost exclusively a single label. To mitigate this risk, the proposed method replaces exact label proportions with ε-differentially private approximations obtained by perturbing the underlying label counts with Laplace noise calibrated to their global sensitivity.

Formally, for a batch B_i and a set of labels $\mathcal{Y}$, define the counting function $f(B_i) \in \mathbb{N}^{|\mathcal{Y}|}$ that maps B_i to its vector of labels counts. Since changing one individual in B_i can affect at most one label count by one, the ℓ_1-sensitivity of f is equal to 1. Following the Laplace mechanism for pure ε-DP, independent noise drawn from $\mathsf{Lap}(0, 1/\varepsilon)$ is added to each component of $f(B_i)$, where $\varepsilon > 0$ denotes the local privacy parameter. The noisy counts are then clipped to a feasible interval and normalized to obtain a probability vector over labels, which is used as a differentially private estimate of the empirical label proportions for B_i. The matrix $Q(j) \in \mathbb{R}^{h \times |\mathcal{Y}|}$ collecting these privatized proportions over all batches at node n_j is subsequently shared with neighboring nodes and replaces the exact proportion matrix in the LLP training procedure.

Algorithm 5.1 summarizes the complete procedure executed at a single node n_j. For each batch B_i, the algorithm first computes the raw label counts $Q(j)_{i,y}$ for all $y \in \mathcal{Y}$ and the corresponding batch size $m = \sum_{y \in \mathcal{Y}} Q(j)_{i,y}$. It then adds Laplace noise with scale $1/\varepsilon$ to each count, clips the perturbed values to the interval $[0.001, m]$ to avoid negative or zero entries and to bound numerical outliers, as well as finally normalizes the resulting vector so that it sums up to one. The output matrix $Q(j)$ is thus a batch-wise collection of privatized label distributions that satisfies ε-DP at the level of individual time series records under the standard neighboring-dataset notion for counting queries.

In the DP-LLP training pipeline in general, this mechanism is invoked after the local datasets have been partitioned into batches and before any information is exchanged between nodes. At each node, the privatized matrix $Q(j)$ produced by Algorithm 5.1 replaces the non-private label proportion matrix used in the original LLP algorithm, while the subsequent clustering-based model learning and neighbor communication steps remain unchanged. In this way, the proposed modification introduces a tunable privacy layer on top of the existing LLP framework, with ε controlling the trade-off between formal privacy guarantees and the fidelity of the label information available for distributed learning.

Algorithm 5.1 Computation of ε-differentially private label proportions at node n_j

Require: Batches $B_1, \ldots, B_h$, label set Y, privacy parameter ε
Ensure: Matrix $Q(j) \in \mathbb{R}^{h \times |Y|}$ of privatized label proportions

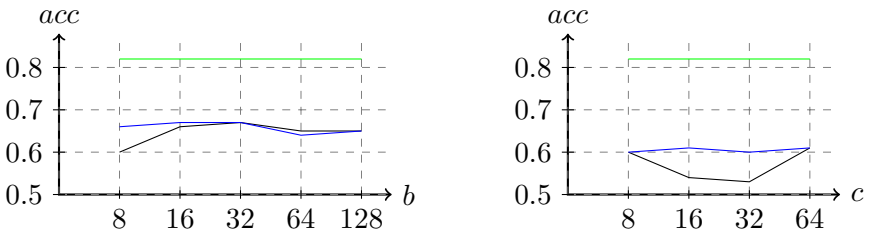
Initialize $Q(j)$ as an $h \times |Y|$ matrix filled with zeros
for $i \leftarrow 1$ **to** h **do** ▷ Process batch B_i
  for each label $y \in Y$ **do**
    Compute raw count $Q(j)_{i,y} \leftarrow \#\{(x, \tilde{y}) \in B_i \mid \tilde{y} = y\}$
  end for
  $m \leftarrow \sum_{y \in Y} Q(j)_{i,y}$ ▷ Batch size
  for each label $y \in Y$ **do**
    Sample $\eta \sim Lap\big(0, 1/\varepsilon\big)$
    $Q(j)_{i,y} \leftarrow Q(j)_{i,y} + \eta$ ▷ Add Laplace noise
    $Q(j)_{i,y} \leftarrow \max\{0.001, \min\{Q(j)_{i,y}, m\}\}$ ▷ Clip to $[0.001, m]$
  end for
  Normalize row $Q(j)_{i,*}$ so that $\sum_{y \in Y} Q(j)_{i,y} = 1$
end for
return $Q(j)$

5.3.5 Experimental Evaluation of Distributed Traffic Data

The proposed DP-LLP algorithm is evaluated based on real-world traffic data collected at junctions in the city of Dublin using the SCATS (McCann, 2014; Sachweh et al., 2021). The dataset comprises average traffic flow measurements at 15-minute intervals for several sensor locations over a period of one month. For experiments, a subset of four sensor nodes and their three nearest neighbors, determined by the Euclidean distance in geographic coordinates, is selected to emulate a small-scale distributed sensor network.

To construct the time series features, a sliding window of size $w = 5$ is applied, so that each feature vector $x_{i,t}$ contains the traffic flow values from the last five time steps at node n_i. The corresponding label $y_{i,t}$ is defined as the discretized traffic flow on the horizon $r = 1$, i.e., the subsequent time step, with continuous flow values mapped into five discrete categories representing different congestion regimes. Each local node dataset is then partitioned into batches of size b and the LLP as well as DP-LLP models are trained based on the proportions of the labels communicated between neighboring nodes. As baselines, two types of models are considered: (i) a centralized k-Nearest Neighbors (kNN) classifier trained on the union of all local datasets and (ii) the original LLP algorithm without DP.

The experimental protocol uses 10-fold cross-validation to estimate model performance. For kNN, the number of neighbors is fixed at $k = 16$, as in Sachweh et al., 2021. For LLP and DP-LLP, two sets of hyperparameters are varied: the batch size $b \in \{8, 16, 32, 64, 128\}$ while fixing the number of clusters to $k = 16$ and the number of clusters $k \in \{8, 16, 32, 64\}$ while fixing the batch size to $b = 50$. When applying DP-



(a) batch size variations ($k = 16$) **(b)** cluster size variations ($b = 50$)

Figure 5.2: Average accuracy of LLP and DP-LLP on the Dublin traffic dataset for varying batch sizes (a) and cluster sizes (b). Blue curves correspond to non-private LLP, black curves to DP-LLP with $\varepsilon = 0.1$ and the green line indicates the accuracy of a centralized kNN classifier with $k = 16$ as benchmark. Figure extracted from Sachweh et al., 2021.

LLP, the privacy parameter is typically set to $\varepsilon = 0.1$ to balance privacy and utility. The main metric of interest is the average classification accuracy over all cross-validation folds and sensor nodes.

Figure 5.2 summarizes the impact of batch size and cluster size on the model accuracy. The original LLP algorithm achieves an accuracy in the range of approximately 65%–67%, depending on the hyperparameters, while the centralized kNN baseline reaches around 82% on the same dataset. The DP-LLP variants with $\varepsilon = 0.1$ attain accuracy levels that are only slightly below the non-private LLP, particularly for batch sizes $b \geq 32$, where the difference typically remains within about one percentage point. In contrast, very small batches such as $b = 8$ lead to more pronounced degradation, because the noise added to each label count dominates the signal when the number of samples per batch is small. This observation is consistent with the sensitivity analysis of counting queries, where smaller batches increase the relative influence of individual examples and thus require effectively stronger perturbation.

A complementary set of experiments investigates the effect of varying $\varepsilon \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$ for fixed batch size and number of clusters, as depicted in Figure 5.3. For very small privacy budgets $\varepsilon \leq 0.05$, the injected Laplace noise is so large that the resulting models achieve only about $32\% - 48\%$ accuracy, making them practically unusable. Starting from $\varepsilon = 0.1$, accuracy increases sharply and approaches the non-private LLP baseline, reaching roughly $63\%-65\%$ for $\varepsilon \in [0.1, 1.0]$. This behavior is in line with broader empirical findings on differentially private learning, where moderate privacy parameters are often required to preserve adequate model utility in iterative algorithms (Abadi et al., 2016). In general, the experiments demonstrate that DP-LLP can achieve a favorable balance between privacy and prediction performance on realistic distributed time series data and thereby constitutes a viable building block for privacy-preserving analytics in federated Data Spaces.

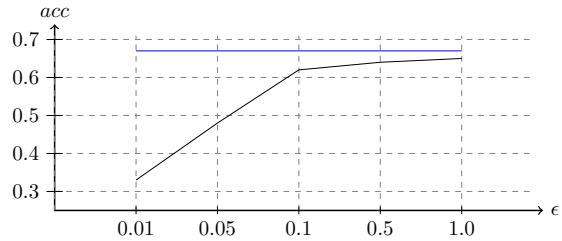


Figure 5.3: Accuracy of LLP (horizontal line) and DP-LLP for different privacy parameters $\varepsilon \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$ at fixed batch size and cluster size. Small values of ε provide stronger privacy but lead to substantial accuracy loss, whereas moderate values such as $\varepsilon = 0.1$ yield a favorable privacy–utility trade-off. Figure extracted from Sachweh et al., 2021.

5.3.6 Advantages in the Context of Federated Data Spaces

From a Data Space perspective, the DP-LLP algorithm provides a concrete instantiation of PPML tailored to fully distributed IoT deployments. It combines several desirable properties:

- **Data locality:** Raw time series data remain at the sensor nodes and are never shared with other participants or central servers.
- **Communication efficiency:** Only aggregated label proportions per batch are exchanged, resulting in significantly lower bandwidth requirements compared to gradient-based FL.
- **Formal privacy guarantees:** By applying the Laplace mechanism to batch-level label counts, the exchanged information satisfies individual-level DP under a tunable privacy budget ε.
- **Accuracy–privacy trade-off:** Empirical evaluation of real traffic data shows that moderate privacy parameters (e.g., $\varepsilon = 0.1$) preserve most of the predictive performance of non-private LLP, while providing meaningful protection against inference on individual trajectories.

These characteristics make DP-LLP a promising building block for privacy-aware analytics in Data Spaces, especially in scenarios where centralized aggregation is infeasible and nodes have limited computational and communication capabilities. In the context of this work, DP-LLP serves as a foundational technique for integrating DP into distributed learning workflows on time series data and its principles will be extended and combined with additional mechanisms in subsequent sections.

5.4 Differential Privacy Encoded in LSTM Model Architectures (DP-LSTM)

In this section, the DP-LLP framework from the previous section is extended to spatio-temporal time series and deep reinforcement models, following the approach proposed in Sachweh et al., 2022.

In contrast to the previous DP-LLP setting, which relies on a k-means based local learner and purely tabular label proportions, the proposed architecture integrates an LSTM-based model with differentially private label proportions exchanged between neighboring nodes, thereby combining temporal sequence modeling, spatial context and formal privacy guarantees.

This combination is particularly relevant for cross-enterprise environments and federated Data Spaces, where participants must jointly exploit spatio-temporal correlations without giving up data sovereignty or exposing raw time series and where more complex models and architectures can benefit from the distributed data gathering process beyond the capabilities of DP-LLP.

5.4.1 Positioning with respect to Previous Sections and Related Work

The DP-LLP algorithm introduced in the previous section extends classical LLP by injecting Laplace noise into batch-wise label counts before redistribution to neighboring nodes, thereby providing ε-DP for the exchanged label proportions.

However, its local learner is based on k-means clustering with majority-vote prediction and operates on static feature windows, which limits its ability to capture complex nonlinear and temporal dependencies that are characteristic for traffic time series and many industrial sensor streams.

Consequently, DP-LLP achieves privacy with low communication overhead, but does not fully exploit the temporal structure of cross-enterprise data nor modern Deep Learning capacity.

Existing deep LLP approaches, in particular the work of Dulac-Arnold et al., 2019, demonstrate that replacing simple local learners with deep models such as LSTMs can substantially improve performance, but they are evaluated on i.i.d. image data (MNIST) without temporal correlations and do not address distributed or cross-organizational settings.

Conversely, spatio-temporal graph neural network approaches and Federated Learning methods such as Flow-FL (Majcherczyk et al., 2021) focus on global centralized or parameter-server based models, often without strict privacy guarantees for intermediate updates and without explicit control over per-message privacy budgets.

The architecture proposed here differs from both lines of work in that it (i) maintains the fully distributed peer-to-peer label-proportion communication pattern of DP-LLP, (ii)

replaces the local k-means learner by an LSTM-based deep temporal encoder and (iii) fuses neighboring information only via differentially private label histograms in the final dense layer, instead of sharing gradients or model parameters.

The proposed architecture advances the DP-LLP framework by combining deep LSTM-based temporal modeling with rigorous DP guarantees in a fully distributed setting, thereby directly addressing **Research Question 2** on how to design expressive yet privacy-preserving learning algorithms.

By replacing the original k-means learner with an LSTM-based encoder, the model can represent complex non-linear and temporal dependencies, while all inter-node communication is restricted to differentially private label histograms, ensuring formal privacy protection for every exchanged message.

As a result, this section contributes an advanced privacy-preserving distributed algorithm that enables the collaborative training of complex deep models under strict DP constraints, providing a concrete and effective solution component for **Research Question 2**.

5.4.2 Cross-Enterprise Setting and Network Model

We consider a cross-enterprise setting in which multiple parties (e.g., municipalities, infrastructure operators, or industrial sites) participate in a federated Data Space and operate local sensors that continuously measure traffic-related quantities such as flow, density, or speed.

Each participant stores its raw time series data locally but agrees to exchange privacy-preserving aggregate label information with a subset of other participants to improve local prediction performance without violating sovereignty or contractual constraints.

The overall system is modeled as a graph $G = (V, E)$, where each node $j \in V$ represents a data provider (e.g., a sensor location or an enterprise site) and the edges $(i, j) \in E$ encode neighborhood relations derived from spatial proximity or operational topology.

The adjacency structure is given by an adjacency matrix and for a given node j, the neighborhood set N_j contains its direct neighbors $n_1(j), \ldots, n_{|N_j|}(j)$, as illustrated in Figure 5.4.

Communication is restricted to these direct neighbors in a peer-to-peer fashion and only involves label histograms; there is no global parameter server or central aggregator, which aligns with decentralized governance and data-supply chain concepts in federated Data Spaces.

Compared to the DP-LLP network in the previous section, this setting is identical at the communication level but differs at the local computation level: instead of a shallow clustering-based classifier, each node now hosts an RNN model that explicitly exploits

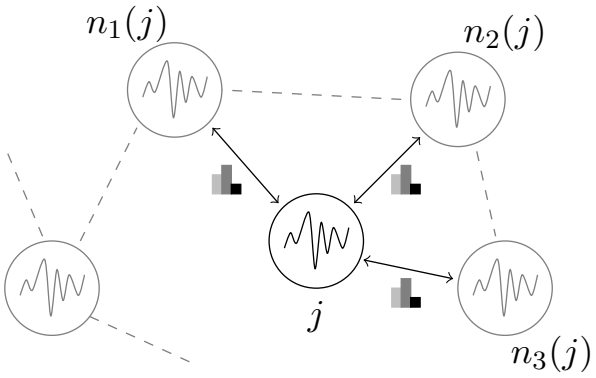


Figure 5.4: Distributed network topology and neighborhood structure. Each node j has a set of direct neighbors $n_1(j), n_2(j), n_3(j)$ connected via the adjacency matrix. Only aggregated histograms (buckets) are exchanged between neighbors; raw measurements and model parameters remain local. Figure originates from Sachweh et al., 2022.

temporal dependencies and is able to integrate neighbor information as an additional feature source.

5.4.3 Time-Series Label Proportions and Differential Privacy

Each node j observes multivariate time series $x_t \in \mathbb{R}^d$ at fixed sampling intervals (e.g., 5 minutes), including features such as traffic density or speed. To build training instances, the time series is partitioned into non-overlapping time windows of length w, each yielding a sequence $(x_t, \ldots, x_{t+w-1})$ and an associated target value derived from a future time step (e.g., density or speed at a prediction horizon). The targets are discretized into a finite set of classes or bins, so that each window receives a discrete label, analogous to the label-construction step in DP-LLP.

Instead of exchanging these labels directly, the nodes aggregate the labels into histograms in windows, resulting in label proportions that summarize the frequency of each class within an aggregation interval. For discrete-valued targets, histogram counts are computed directly, while for continuous targets, the value range is discretized into b bins and each observed value contributes to the corresponding bin count. This procedure is schematically shown in Figure 5.5, where sensor measurements are first windowed and then mapped to label histograms.

Formally, each histogram can be viewed as the result of a counting query with l_1-sensitivity $\Delta f = 1$, because changing a single data point affects at most one bin by one count. To ensure $(\varepsilon, 0)$-DP for each histogram, Laplace noise with scale parameter $\sigma = \Delta f / \varepsilon = 1/\varepsilon$ and mean 0 is added independently to each bin:

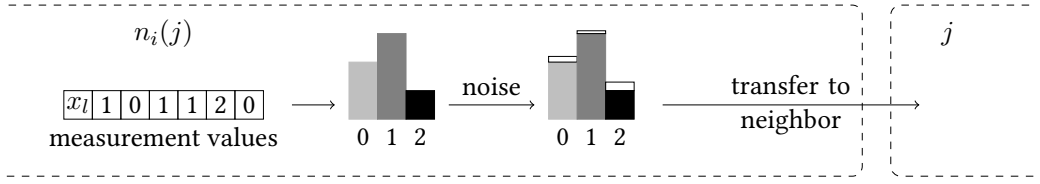


Figure 5.5: Construction and privatization of label histograms for distributed exchange. Local measurements at node j are grouped into fixed-size windows of length w and aggregated into histograms over discretized labels. Laplace noise calibrated to ε-DP is then added independently to each bin before normalization and transmission to neighbors. Figure originates from Sachweh et al., 2022.

$$lap(x \mid \tfrac{1}{\varepsilon}, 0) = \frac{\varepsilon}{2} e^{-|x|\varepsilon}. \tag{5.14}$$

After adding noise, the resulting values are clipped to reasonable bounds and normalized, producing a valid distribution of label proportions that can be transmitted without violating per-window privacy budgets. As in the previous DP-LLP variant, non-overlapping windows ensure that each raw observation contributes to at most one aggregate, preventing privacy loss due to overlapping queries and simplifying composition over time.

The crucial difference from DP-FL type approaches discussed earlier is that privacy noise is injected at the level of low-dimensional label histograms rather than in high-dimensional model gradients or parameters. This reduces sensitivity and communication size, which in turn allows smaller ε for a comparable utility loss and leads to better interpretability of the shared objects, as each histogram entry retains a clear semantic meaning.

Because all downstream processing (aggregation over neighbors, model training and prediction) operate exclusively on these noisy histograms, the entire algorithm remains ε-Differentially Privacy by the post-processing property from Dwork and Roth, 2014.

5.4.4 Model Architecture: Local LSTM with Spatial Aggregates

The core methodological contribution compared to the previous sections is the replacement of the shallow DP-LLP learner by a two-stage recurrent model that cleanly separates the local temporal encoding at node j from the differentially private aggregation of neighbor information.

Figure 5.6 summarizes this architecture: the left dashed box represents the computation at node j, while the right dashed box represents an arbitrary neighbor $n_i(j)$ that only contributes to noisy label statistics.

On the local side, node j observes a sequence x of windowed time-series measurements that are fed into a *LSTM Block*. This block encapsulates all temporal processing (LSTM,

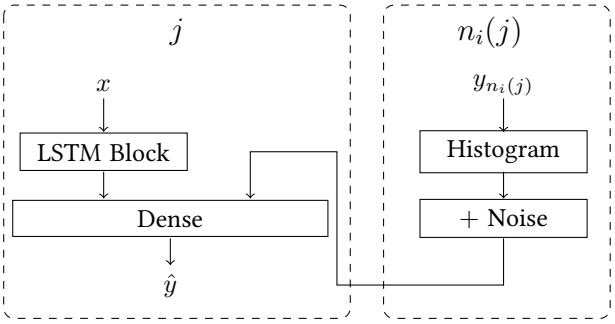


Figure 5.6: Model architecture for node j with local temporal features and spatially aggregated label proportions from neighbors $n_i(j)$. The LSTM Block processes local sequences x and provides a temporal representation to the Dense layer. Each neighbor transforms future labels $y_{n_i(j)}$ into a histogram, adds Laplace noise and sends the noisy histogram to j. The Dense layer fuses local features and aggregated noisy histograms to produce the prediction $\hat{y}$. Figure extracted from Sachweh et al., 2022.

non-linearities and the local linear projection) and produces a fixed-size feature representation that is forwarded to a *Dense* layer. The Dense layer also receives, as an additional input, the aggregated neighbor information and outputs the final prediction $\hat{y}$ for the next time step at node j.

On the neighbor side, each node $n_i(j)$ first maps its own future labels $y_{n_i(j)}$ into a *Histogram* over discretized label bins. The Laplace noise calibrated to the chosen privacy parameter ε is then added to the *+ Noise* block, producing a differentially private label histogram that is transmitted to node j. The incoming noisy histograms from all neighbors are averaged and injected only into the Dense layer of the node j, as indicated by the arrow crossing from the right to the left dashed box. This explicit separation of the LSTM Block (purely local, operating on raw x) and the noisy neighbor histograms (purely aggregate, injected at the last layer) makes it clear that learning of temporal representations is shielded from DP noise, while spatial context is incorporated in a privacy-preserving and communication-efficient manner.

Local Node Learning

The local node learning block processes sequences of length w (set to 12 time steps for PEMS-BAY, corresponding to roughly one hour) using an LSTM network that extracts temporal dependencies of each feature channel. In contrast to DP-LLP, which relies on static batch features, the LSTM maintains an internal state that can model long-range temporal correlations, which are essential for phenomena such as rush hours, daily patterns and delayed congestion propagation.

Figure 5.7 details how the raw LSTM activations are mapped into a channel-wise linear representation at node j. Each row of the matrix on the left represents the LSTM+ReLU output for one feature channel (e.g., x_{speed} and x_{density}) over the w time steps, so that the

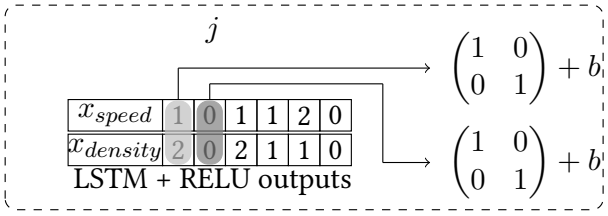


Figure 5.7: Local temporal feature extraction and linear projection at node j. The LSTM+ReLU stack produces a matrix of activations over time for each feature channel (rows). For the current prediction window (grey columns), the corresponding activation vectors are multiplied with a 2×2 weight matrix that is initially the identity and shifted by a bias b_t. This Local Linear Layer thus starts as a pass-through and gradually learns a channel-wise projection that refines the temporal representation before fusion with neighbour information. Figure extracted from Sachweh et al., 2022.

gray-shaded columns correspond to one particular prediction window. For each channel, the highlighted vector is multiplied by a weight matrix 2×2 that is initially set to the identity,

$$W^{(0)} = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}, \tag{5.15}$$

to which an additive bias b_t is applied, as indicated by the terms $(\cdot) + b_t$ on the right-hand side of the figure. This initialization guarantees that, at the beginning of training, the Local Linear Layer does not distort the temporal patterns learned by the LSTM, so that the network behaves like a pure LSTM+ReLU encoder.

During training, the entries of W and b_t are adapted via backpropagation. As a result, the Local Linear Layer learns to re-weight and mix the two channels (and, in general, all feature channels) in a data-driven manner, for example by emphasizing speed in congestion phases or density in free-flow phases. Because this transformation is applied independently per node and purely to local LSTM output, it can be interpreted as a lightweight, learnable feature projection that prepares the local representation for subsequent fusion with neighbor histograms in the Dense layer.

This local block corresponds to purely intra-node learning and can be seen as the LSTM-based analog of the DP-LLP k-means learner. It maps local time series to a latent representation that is later combined with neighbor aggregates. Because it is trained via standard backpropagation rather than discrete cluster label search, it can exploit richer loss signals from continuous targets and is less sensitive to heuristic cluster initialization effects as in DP-LLP.

Learning with Aggregated Spatial Information

In the second block, node j augments its temporal features with spatial context provided by the differentially private histograms of its neighbors.

For each time window, every neighbor $n_i(j) \in N_j$ sends a noisy label histogram derived from its own labels. The node j then averages these histograms to obtain a fixed-length vector with mean label proportions between neighbors, which ensures invariance to the number of neighbors and supports heterogeneous degrees in the underlying graph of distributed nodes. This averaged histogram is concatenated with the output of the local block (LSTM + ReLU + Local Linear Layer) and fed into a Dense layer that is applied channel-wise, aligning histograms of different feature channels (e.g., speed and density) with their corresponding temporal features.

The flow of data from local sequences and neighbor histograms through the Dense layer is indicated by arrows in Figure 5.6. The output $\hat{y}$ of the Dense layer is the final prediction of the target variable (e.g., future traffic speed) and all weights in both the local and spatial blocks are updated jointly by backpropagation.

Compared to graph convolutional approaches that operate on complete global graphs (Yu et al., 2018), this design has two important consequences. First, only low-dimensional DP-protected histograms are shared instead of node-level features or labels and second, spatial information is injected only at the final prediction stage, which reduces the effective depth of the model with regard to noisy inputs and thus mitigates the amplification of DP noise through multiple layers.

Loss Function and Optimization

Training is performed with the Mean Squared Error (MSE) loss between predicted and actual values, averaged over batch elements:

$$loss_{\text{MSE}} = \frac{1}{|B|} \sum_{i}^{|B|} (\hat{y}_i - y_i)^2. \tag{5.16}$$

For evaluation across all nodes and time windows, an MSE adapted to the fully distributed setting is used:

$$MSE = \frac{1}{|y|} \sum_{j=0}^{|N|} \sum_{i=0}^{|D_{\text{test}}|} \sum_{t=0}^{\frac{|D_{\text{test}}|}{w}} \left(y_{ijt} - \hat{y}_{ijt}\right)^2. \tag{5.17}$$

Inputs are normalized via instance normalization for each feature channel,

$$norm_{\text{inst}} = \frac{x - \overline{x}}{\sigma(x)}, \tag{5.18}$$

and restored to the original scale for analysis.

The models are implemented in PyTorch and PyTorch Geometric and optimized with the Adam optimizer using a learning rate of 0.01, reusing the same hyperparameter philosophy as in the DP-LLP experiments to ensure comparability.

5.4.5 Distributed Data Exchange in Federated Data Spaces

Data exchange follows the distributed network structure described above and is restricted to the exchange of noisy histograms between direct neighbors. For each time window, node j constructs a label histogram, adds Laplace noise according to the chosen privacy parameter ε, normalizes the result to obtain label proportions and transmits this vector to all $n_i(j) \in N_j$.

Simultaneously, j receives analogous histograms from its neighbors and uses their average as an additional input to its Dense layer. Compared to FL schemes for gradient-sharing discussed in Section 5.2, this protocol has three scientific advantages in Data Spaces between enterprises:

- **Bounded and interpretable leakage:** Only label histograms with formally quantified ε-DP leakage are shared, which is easier to reason about than gradient-based leakage from high-dimensional models.
- **Low communication cost:** Histogram vectors have fixed and small dimensionality independent of model size, which is relevant for bandwidth-constrained or intermittently connected industrial partners.
- **Architectural compatibility:** The peer-to-peer pattern avoids a central coordinator and fits naturally with Data Space connectors that mediate bilateral contracts and usage policies.

From an architectural perspective, the histogram service at each node can be realized as a Data Space connector interface that returns differentially private label proportions for authorized partners, with privacy budgets and label schemas negotiated and governed at the federation level.

5.4.6 Experimental Evaluation

The following experimental evaluation investigates how the proposed DP-LSTM architecture behaves in realistic cross-enterprise traffic scenarios. The analysis focuses on three main aspects: the accuracy of the prediction relative to established baselines, the scalability to larger sensor networks and the impact of the privacy parameter ε on utility. To this end, the following subsections describe the evaluated datasets and model variants, report quantitative results and provide a qualitative assessment of the prediction behavior in selected time series.

Datasets and Model Variants

The approach is evaluated on three spatio-temporal traffic datasets that differ in network topology, scale and data origin: PEMS-BAY, METR-LA and LuST (Codecá et al., 2017; Y. Li et al., 2017). The experimental protocol mirrors that of the previous DP-LLP evaluation

dataset	model	ε	test loss
LuST	kNN	x	0.470
	LabelProportionLocal	x	**0.377**
	LabelProportionToDense	x	0.379
		0.5	0.381
		0.1	0.380
METR-LA	kNN	x	1.020
	LabelProportionLocal	x	0.760
	LabelProportionToDense	x	**0.480**
		0.5	0.664
		0.1	0.730
PEMS-BAY	kNN	x	0.369
	LabelProportionLocal	x	0.496
	LabelProportionToDense	x	**0.305**
		0.5	0.487
		0.1	0.542

Table 5.1: Test MSE for kNN, local LSTM (LabelProportionLocal) and the proposed LSTM with neighbor histograms (LabelProportionToDense) on LuST, METR-LA and PEMS-BAY for different privacy levels ε (x indicates no Differential Privacy).

where possible but replaces the local learner and adds neighbor histograms to isolate the architectural effect.

Three main model variants are compared:

- **kNN (centralized baseline):** A centralized k-nearest-neighbor model with $k = 1$ that has access to all nodes' data and predicts the target by finding the nearest historical graph sequence in Euclidean distance.
- **LabelProportionLocal:** A fully decentralized model with the architecture LSTM $\rightarrow$ ReLU $\rightarrow$ Local Linear Layer $\rightarrow$ Dense layer that uses only local data without neighbor histograms.
- **LabelProportionToDense:** The full proposed model that integrates neighbor label histograms into the Dense layer, evaluated with no privacy (no noise) and with ε-DP (e.g. $\varepsilon = 0.5$ and $\varepsilon = 0.1$).

Quantitative Results and Comparison to Previous Approaches

Table 5.1 summarizes the test MSE for all model variants and datasets. The results indicate that the LSTM-based models generally outperform the centralized kNN baseline, especially on METR-LA and LuST, where LabelProportionLocal achieves lower MSE than kNN. For PEMS-BAY, the full LabelProportionToDense model without DP achieves the best performance (MSE ≈ 0.305), outperforming both the local LSTM and kNN, which

shows that incorporating neighbor histograms can significantly improve accuracy for dense sensor networks.

From a methodological standpoint, these improvements can be traced back to two effects:

- Replacing k-means by LSTM + Dense allows the model to approximate non-linear mappings from temporal sequences to future states and to exploit long-range dependencies that are invisible to batch-wise clustering.
- Adding DP-protected neighbor histograms effectively provides a structured, low-dimensional summary of the local spatial neighborhood, which allows the model to disambiguate local temporal patterns that differ across spatial contexts.

Compared to the original DP-LLP experiments, which operate on a smaller Dublin dataset and report accuracies in the range of 65–$67\,\%$ for LLP versus slightly lower values for DP-LLP at $\varepsilon = 0.1$, the LSTM-based approach achieves clearly lower MSE on larger and more complex datasets while preserving the same DP mechanism at the histogram level. This empirically supports the claim that the main bottleneck of DP-LLP in cross-enterprise time series is not the DP mechanism itself, but the expressive power of the local learner, which the proposed architecture addresses.

As expected, introducing DP noise by decreasing ε increases MSE, reflecting the loss of information in neighbor aggregates. For strong privacy ($\varepsilon = 0.1$), the advantage over the purely local model can disappear, highlighting the privacy–utility trade-off in cross-enterprise deployments. In particular, for PEMS-BAY the LabelProportionToDense model with $\varepsilon = 0.1$ behaves similarly to a mean predictor on some nodes, which is consistent with the intuition that heavily noised histograms contain little usable spatial information after normalization.

Qualitative Analysis of Prediction Behavior

To analyze prediction behavior in more detail, Figure 5.8 shows a zoomed in section of 200 time steps (approximately 17 hours) for a single node on the PEMS-BAY dataset. This slice contains a complete day-to-night cycle that includes rush-hour periods and is chosen from a region where the MSE is comparatively low to illustrate the best-case performance of each method.

The detailed view reveals that the kNN baseline often underestimates peaks and exhibits a step-like behavior, while the local LSTM-based LabelProportionLocal tracks the ground truth more closely and reacts better to steep drops. The LabelProportionToDense model without DP is closest to the ground truth, missing only minor fluctuations, whereas moderate DP with $\varepsilon = 0.5$ introduces occasional spurious peaks and slight misalignments. High DP with $\varepsilon = 0.1$ leads to a nearly constant prediction around the mean, indicating that neighbor histograms become too noisy to add useful information. Compared to the previous DP-LLP plots on the Dublin dataset (Sachweh et al., 2021), these results

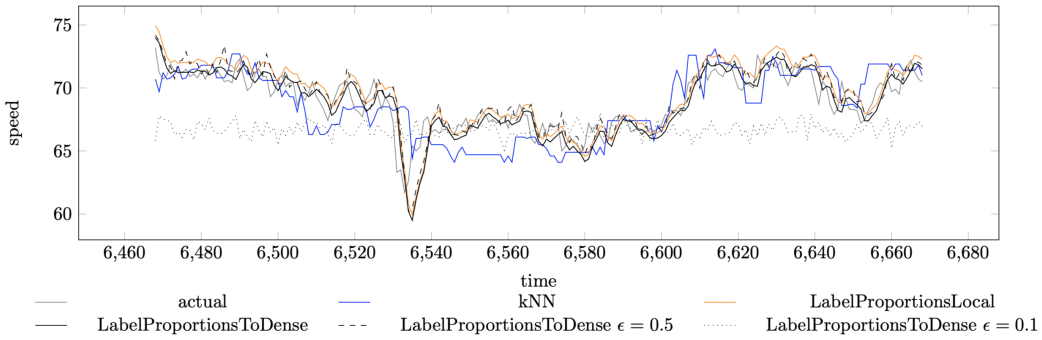


Figure 5.8: Detailed test slice of 200 time steps for one node on the PEMS-BAY dataset. The ground truth is shown in gray, the kNN baseline in blue, the local LabelProportionLocal model in orange and the LabelProportionToDense model in black. Variants with ε-DP are indicated by dashed ($\varepsilon = 0.5$) and dotted ($\varepsilon = 0.1$) black curves. Figure extracted from Sachweh et al., 2022.

from Sachweh et al., 2022 highlight that the proposed architecture scales to larger networks and still yields interpretable degradation as a function of ε, which is an important property for privacy-budget governance in federated Data Spaces.

5.4.7 Advantages in the Context of Federated Data Spaces

Compared to the earlier DP-LLP algorithm, the contribution of DP-LSTM lies in the integration of a deep temporal encoder with DP-protected neighbor features, which significantly improves predictive performance on realistic spatio-temporal datasets without changing the communication and privacy model. Therefore, DP-LSTM strongly contributes to **Research Question 2**, as it allows for an increase in model complexity by keeping the privacy guarantees from DP-LLP.

Compared to DP-enhanced FL or HE-based approaches discussed in the related work, the method offers a light-weight alternative that requires only histogram exchange and local ML model training, avoiding the complexity of encrypted gradients or global model aggregation while still providing formal privacy guarantees for all communicated information.

These properties make the architecture particularly suitable as a building block for Data Space deployments, where heterogeneous organizations contribute time series data, require strong guarantees about information leakage and at the same time expect competitive model performance for operational decision support.

Although DP-LSTM integrates DP and allows limited knowledge sharing through label proportions, it does not enable joint model training. Each participant must train its own model independently, preventing a unified optimization across all parties. To overcome this limitation, the next section introduces HEX-FL, which supports collaborative and privacy-preserving joint model training with FL.

5.5 Homomorphic Encryption in Federated Learning environments (HEX-FL)

Cross-enterprise ML in Data Spaces demands mechanisms that enable collaboration across organizational boundaries without exposing sensitive assets, like raw or personal data. As shown in the previous sections, ϵ-DP can limit information leakage in principle, but in practice it often requires intrusive adaptations of model architectures and training pipelines, which complicates integration into existing systems and hinders model reuse. Typical FL keeps data local while sharing model updates, yet unprotected gradients and weights can still leak sensitive information, making the system vulnerable to reconstruction and membership-inference attacks (Shokri et al., 2017b; Zhu et al., 2019). By leveraging HE and in particular the CKKS scheme for approximate arithmetic, it becomes possible to perform linear algebra directly on encrypted real-valued data. This allows the model parameters to remain encrypted throughout the training and aggregation, so updates do not need to be revealed, thus providing a possible solution to this privacy issue (Cheon et al., 2017; Lee et al., 2022).

The first relevant paper for this section demonstrates how CKKS-based HE can be integrated into a Federated Data Space to provide encrypted data provisioning for ML services (Sachweh et al., 2023). In this approach, data producers encrypt raw sensor streams using an externally supplied *Encryption Service* built on TenSEAL/SEAL and share only ciphertexts with data consumers, who then perform local analytics on the encrypted data. This method primarily addresses secure data exchange and encrypted inference or training at a single consumer, but does not yet provide collaborative FL across multiple data holders. The second work extends this idea to a full FL framework, *HEX-FL*, in which multiple organizations collaboratively train a shared model while all model updates are processed under CKKS-based MHE (Sachweh et al., 2026). HEX-FL decomposes the system into a Federated Aggregator, Homomorphic Encryption Services per client and local Machine Learning Services, orchestrated via asynchronous REST APIs and implemented with the Lattigo CKKS backend (C. Mouchet et al., 2021; Sachweh et al., 2026).

The following subsections first provide a systematic survey of different CKKS libraries that underpin both frameworks, followed by a detailed description of the HEX-FL architecture and protocol and an in-depth discussion of the empirical runtime, memory and accuracy characteristics for a federated classification task (Sachweh et al., 2023, 2026).

5.5.1 Survey of CKKS Libraries

Since several implementations of the conceptual CKKS methodology exist, each offering different features, performance characteristics and integration options. This section compares the most relevant libraries used within the studied frameworks, focusing on their functional capabilities and suitability for the respective application contexts.

Requirements from the Frameworks

All requirements for homomorphic encrypted ML applications in Data Spaces are derived from Sachweh et al., 2023 and Sachweh et al., 2026, which together impose concrete functional and non-functional requirements on their design and implementation.

From a functional perspective, the libraries must support the CKKS scheme with approximate arithmetic over real vectors, including SIMD packing, rescaling and rotations, to implement linear-algebra operations used in neural networks (Cheon et al., 2017; Lee et al., 2022). For HEX-FL, multiparty extensions are crucial, including collective key generation, relinearization-key generation and key-switching protocols that realize MHE (C. Mouchet et al., 2021; Sachweh et al., 2026). Furthermore, the libraries must exhibit reasonable runtime and memory characteristics for repeated encrypt–aggregate–decrypt cycles under realistic RLWE-based security parameters and precision requirements (Tsuji and Oguchi, 2024; Xie et al., 2024).

From a Data Space integration perspective, the programming language remains an important factor, as the services that need to interact with the Data Space ideally communicate directly within the connector or can be easily deployed next to a running instance. However, this is only one of several criteria, along with previous aspects such as available functionalities and performance. These considerations motivate the evaluation of SEAL (via TenSEAL), Lattigo and OpenFHE for both papers and finally HEX-FL. In the following sections, their main properties, advantages and disadvantages are summarized with a focus on CKKS-style approximate arithmetic, multiparty capabilities and practical programmability.

Overview of the Libraries

Microsoft SEAL is a C++17 HE library that provides efficient implementations of BFV and CKKS, optimized low-level arithmetic (optionally via Intel HEXL) and extensive configuration options for performance tuning (H. Chen et al., 2017). TenSEAL builds a tensor-oriented Python interface on top of SEAL and adds high-level encrypted vector and matrix operations, making CKKS/BFV usable directly from Python-based ML workflows (Benaissa et al., 2021).

Lattigo is a pure Go module that implements the full-RNS RLWE-based BFV, BGV and CKKS schemes together with their multiparty variants, designed for server-side and distributed applications with the Go concurrency model and WebAssembly (WASM) compilation (EPFL-LDS, Tune Insight SA, 2024; C. V. Mouchet et al., 2020).

OpenFHE is a C++ library that consolidates and extends the capabilities of earlier projects (e.g., PALISADE and HElib) and supports BFV, BGV, CKKS and Fast Fully Homomorphic Encryption (TFHE)/Fast Fully Homomorphic Encryption over Torus (FHEW)-style schemes, including advanced features such as functional and interactive bootstrapping and scheme switching (Badawi et al., 2022).

Features and Scheme Support

SEAL focuses on single-key BFV and CKKS with high-quality implementations but does not natively offer multiparty or threshold variants in the main library. TenSEAL exposes essentially the same scheme set, but tailored to tensor operations in Python. This design is attractive for single-party or client server scenarios where one party controls the secret key and where simplicity and performance of approximate arithmetic are prioritized over distributed key management.

Lattigo explicitly targets multiparty homomorphic encryption and provides full-RNS BFV/BGV/CKKS together with dedicated packages for local multiparty key-generation and key-switching, and supports dense-key and sparse-key bootstrapping procedures.

OpenFHE offers the broadest feature set: in addition to BFV/BGV/CKKS it includes TFHE-style Boolean schemes, threshold FHE for all major schemes, proxy re-encryption and sophisticated scheme-switching and functional bootstrapping between CKKS and FHEW/TFHE or RLWE.

Language Integration and Usability

TenSEAL's main advantage is its Python-first API that embeds CKKS/BFV operations into a tensor abstraction, supports encrypted vector and matrix operations, and exposes the SEAL functionality via a Python binding layer, which greatly simplifies prototyping with PyTorch or NumPy. Nonetheless, it inherits SEAL's single-key model and adds overhead from the Python/C++ binding system, making it less suitable for complex multiparty deployments or tightly optimized services.

SEAL itself is a C++17 library intended for integration into performance-critical backends; it can be compiled with modern compilers for better runtime and optionally uses hardware acceleration and compressed serialization for improved speed and memory behavior. Lattigo, being pure Go, integrates naturally into Go-based microservices and allows cross-platform and WASM builds without external native dependencies, but Go is less common in the ML ecosystem than Python or C++, which can raise the entry barrier for data science teams. OpenFHE exposes a C++ API and can be built with recent GCC or Clang, but its conceptual and API surface is comparatively large and integration often requires more engineering effort, especially when bridging into other languages via bindings or RPC.

Performance Considerations

Benchmark studies and vendor documentation indicate that SEAL provides highly competitive performance for CKKS and BFV, particularly when compiled with suitable compilers and with hardware acceleration enabled, and several comparative works report SEAL as one of the fastest libraries for core ring operations (Pambudi et al., 2025).

TenSEAL adds overhead from Python but remains practical for medium-scale encrypted inference and training workloads, especially when most heavy computations are offloaded to SEAL's C++ backend.

Lattigo's authors report comparable performance to state-of-the-art C++ libraries for BFV/CKKS, supported by optimized power-of-two ring arithmetic and advanced key-switching strategies. Published benchmarks show that Go implementations can reach timings similar to those of C++ within the ranges of parameters tested.

The performance of OpenFHE is more heterogeneous because it supports many schemes and advanced operations such as functional bootstrapping and scheme switching, which are inherently expensive. Comparative studies show that OpenFHE can be slower than SEAL for basic CKKS operations but is competitive when its advanced features are exploited.

Strengths and Weaknesses

The main advantages of SEAL/TenSEAL are mature CKKS/BFV implementations, strong documentation and excellent integration with Python via TenSEAL, which makes them attractive for prototyping and single-key PPML workloads. The downsides are limited native multiparty functionality and reliance on C++/Python, which may complicate deployment in some backend environments.

Lattigo's strengths lie in its built-in multiparty extensions, bootstrapping support and seamless use in Go-based distributed services, while its disadvantages include a smaller ecosystem and a less direct alignment with the main Python ML tooling.

OpenFHE stands out by offering the richest cryptographic feature set, including threshold FHE, proxy re-encryption, scheme switching and TFHE-style Boolean capabilities, which enable advanced protocols beyond straightforward encrypted aggregation. However, this comes at the cost of a steeper learning curve, a more complex API surface and higher engineering effort for integration and tuning, especially when only plain CKKS-style secure aggregation is required.

5.5.2 HEX-FL Architecture and Multiparty Protocol

HEX-FL is introduced as a generic service-oriented framework for PPML in cross-enterprise environments (Sachweh et al., 2023, 2026). It combines MHE based on CKKS with a modular microservice architecture to enable encrypted model aggregation while keeping raw data and plaintext model updates local to each participant. In contrast to purely conceptual architectures, HEX-FL is implemented as a concrete prototype that can be integrated with existing ML stacks via a narrow vector-based interface, allowing existing models to participate in encrypted FL with minimal code changes (Sachweh et al., 2026).

The framework consolidates and extends the contributions of the two preceding works (Sachweh et al., 2023, 2026). The earlier publication focuses on homomorphically encrypted analytics in Data Space scenarios and demonstrates how CKKS can be used to protect model inference and simple aggregation in a single-key setting, integrated with data sovereignty mechanisms of federated Data Spaces (Sachweh et al., 2023). The later work generalizes this approach towards full multiparty homomorphic encryption for FL. It introduces the HEX-FL architecture, instantiates a distributed CKKS protocol using Lattigo and provides an end-to-end implementation and evaluation of encrypted federated model training across multiple organizations (Sachweh et al., 2026).

Within this work, HEX-FL is therefore positioned as the unifying framework that operationalizes the conceptual ideas of Sachweh et al., 2023 and the protocol-level advances of Sachweh et al., 2026 in a single architecture. From Sachweh et al., 2023, it inherits the focus on cross-enterprise deployments and integration with federated Data Space infrastructures, while from Sachweh et al., 2026 it adopts the concrete multiparty CKKS construction, the service decomposition and the quantitative evaluation of runtime, memory and accuracy characteristics. The following subsections detail the resulting HEX-FL architecture, its service decomposition, the underlying multiparty protocol and relate them to the provided deployment diagram and algorithms.

Service Decomposition

HEX-FL follows a service-oriented design that decomposes the system into three main service types. A central *Federated Aggregator*, a per-client *Homomorphic Encryption Service* and a local *Machine Learning Service*. This decomposition aims to separate cryptographic concerns from model training logic and orchestration, allowing independent evolution of the HE backend, the FL protocol and the domain-specific ML components. The overall deployment and interaction of these services are shown in Figure 5.9, which also serves as a visual reference for the algorithms introduced later in this section.

Federated Aggregator The Federated Aggregator constitutes the core of the HEX-FL cryptographic processing and coordination. It exposes HTTP(S) endpoints for client registration and deregistration, distribution of CKKS parameters, submission of encrypted model updates and retrieval of global model versions. Internally, it maintains stateful caches of encrypted weight vectors keyed by client identifier and model version, validates dimensional consistency of submitted updates, maintains monotonically increasing global model version numbers and schedules aggregation rounds through a periodic cron-like scheduler. The aggregator is implemented in Go and uses Lattigo as its HE backend, leveraging Lattigo's CKKS and multiparty APIs to perform ciphertext aggregation, relinearization and the generation and application of collective keys for the MHE protocol (EPFL-LDS, Tune Insight SA, 2024; C. Mouchet et al., 2021; Sachweh et al., 2026).

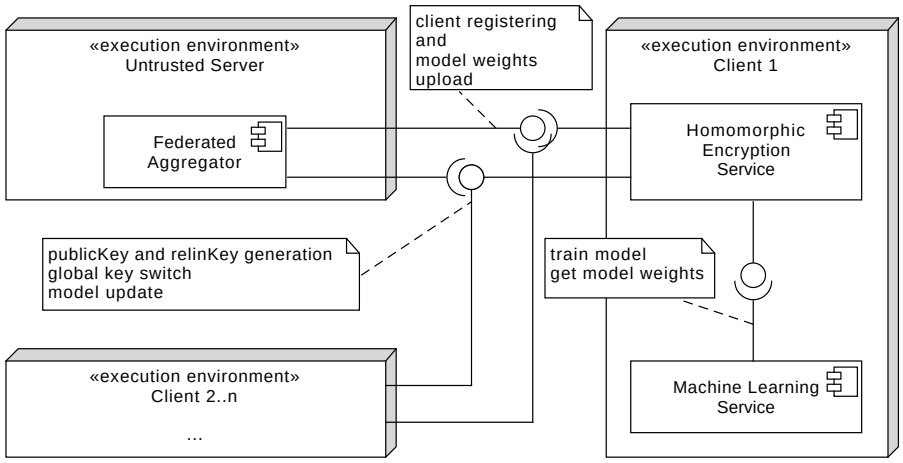


Figure 5.9: Deployment diagram of the HEX-FL Framework software components and distributed execution. Figure extracted from Sachweh et al., 2026.

Homomorphic Encryption Service The Homomorphic Encryption Service is deployed on each client's network and acts as a dedicated cryptographic adapter between the local ML environment and the aggregator. During the registration workflow, the HE service obtains the set of CKKS parameters and public information from the aggregator, generates a local secret-key share and participates in the distributed key-generation procedures for the collective encryption and relinearization keys. During training, it interacts with the ML service via a narrow vector-based interface. It requests the current model parameters as a flat vector, then encodes and encrypts this vector under the collective public key and finally uploads the resulting ciphertext to the aggregator. In addition, it exposes local endpoints to provide partial key-switch shares when requested by the aggregator and to deliver updated global weight vectors to the ML service at the end of each aggregation round.

Machine Learning Service The Machine Learning Service encapsulates the client's model and dataset. It remains largely agnostic to HE and multiparty protocol details. In the reference implementation, the ML service is realized as a Python application built on PyTorch and Flask, which provides endpoints to trigger local training epochs, export the current model parameters as a real-valued vector and import updated global weights. The evaluation prototype uses a compact convolutional neural network with two convolutional layers, ReLU activations and two fully connected layers trained on the MNIST/CIFAR dataset, chosen to provide a non-trivial yet computationally manageable workload for encrypted aggregation. Because the HEX-FL interface only requires vector export and import of parameters, the same service design can be applied to other architectures and datasets without changing the HE or aggregation logic.

Algorithm 5.2 Multiparty Homomorphic Encryption Scheme Initialization with A representing the aggregator and C represents every client (iterative call on every client).

```
C.RegisterClient(c_url);
C.GenPrivateKey();
A.GeneratePublicKey() {
    pk := ckks.PublicKeyShare();
    pk := C.PartialPublicKeyShare();
};
A.PublishPublicKey();
A.GenerateRelinearizationKeys() {
    rlk := ckks.RelinearizationKeyShare();
    rlk := C.PartialRelinKeyAggregation();
}
```

In combination, this service decomposition isolates cryptographic and orchestration complexity in the Federated Aggregator and Homomorphic Encryption Services, while keeping the Machine Learning Service close to existing ML practice. This makes it possible to integrate legacy or pre-existing models into HEX-FL by implementing the vector-based adapter at the ML service boundary, without modifying the core training loop or compromising the end-to-end encryption guarantees for model updates.

Multiparty CKKS Protocol

On the cryptographic level, HEX-FL instantiates the CKKS scheme in a multiparty setting so that model updates remain encrypted end-to-end while multiple clients jointly contribute to and benefit from a shared model. Each client i has a secret key share sk_i and the system collectively derives a public key pk and auxiliary evaluation keys that enable homomorphic computation as well as key switching without revealing any secret key of any party. The protocol is organized into two main workflows. A *registration phase* covering collective key generation and a *training and model distribution phase* covering encrypted upload, homomorphic aggregation and collective key switching, which are formalized in Algorithms 5.2 and 5.3.

Client registration and configuration Clients participate in HEX-FL by explicitly registering with the Federated Aggregator before any collective key material is established. In this registration step, each client sends an HTTP request containing a unique client identifier, a callback URL (*c_url*) for its Homomorphic Encryption Service, as well as metadata describing the type of local model and the current version of the model. This corresponds to the call to `C.RegisterClient(c_url)` in line 1 of Algorithm 5.2, which conceptually iterates over all clients C and populates the aggregator's registry with the necessary connection and model information. The aggregator uses this registry to deter-

mine the set of active participants, to route subsequent protocol messages and to decide which clients are included in the collective key generation and aggregation rounds.

Generation of a collective encryption key (GeneratePublicKey) After all participating clients have registered themselves, each client locally generates a secret key by invoking `C.GenPrivateKey()` as shown in line 2 of Algorithm 5.2. On this basis, the aggregator initializes a CKKS context with a globally agreed parameter set, including the ring dimension, the modulus chain, as well as the default scaling factor and invokes `A.GeneratePublicKey()` (line 3). Inside this routine, the aggregator first allocates data structures for a public-key share, denoted as `ckks.PublicKeyShare()` and then collects partial public-key contributions from each client via `C.PartialPublicKeyShare()` (line 5), which internally use the clients' local secret-key shares sk_i through Lattigo's multiparty library. The Federated Aggregator then aggregates these partial shares into the collective public key pk and publishes it to all clients through `A.PublishPublicKey()` in line 7, thereby enabling subsequent encryption while keeping individual secret keys confined to their respective clients.

Generation of collective relinearization keys (GenerateRelinearizationKeys) Using the same secret-key shares, clients further participate in a distributed protocol to generate a collective relinearization key rlk. This process is initiated on the aggregator side by calling `A.GenerateRelinearizationKeys()` (line 8 of Algorithm 5.2), which constructs an empty relinearization key share via `ckks.RelinearizationKeyShare()` and then iteratively incorporates partial relinearization contributions received from clients through `C.PartialRelinKeyAggregation()` (line 10). The resulting global rlk allows the relinearization of ciphertexts after homomorphic multiplications, thus preserving a compact ciphertext representation and controlling noise growth. In the HEX-FL architecture, this relinearization key is retained solely by the aggregator, since all homomorphic aggregation and multiplication operations are carried out on the server side.

Local encryption of model weights (Encrypt) At the beginning of each aggregation round, every ML service performs one or more local training epochs on its private dataset and exposes the updated model parameters as a single real-valued vector through the vector-based interface. The local HE service then performs the encryption step represented by `w := C.EncryptModelWeights();` in line 1 of Algorithm 5.3. It encodes the model-parameter vector into a CKKS plaintext using Lattigo's encoder and encrypts it under the collective public key pk to obtain a client-specific ciphertext w. Subsequently, the client uploads `w` and associated metadata, such as client identifier and global model version, to the aggregator via `C.UploadUpdatedModelWeights();` in line 2, ensuring that no plaintext model parameters ever leave the client's local network and thus preserving the confidentiality of local updates against an honest-but-untrusted aggregator.

Algorithm 5.3 HEX-FL process flow for training, updating and rolling out new model weights to all parties. A representing the aggregator and C represents every client (iterative call on every client).

```
w := C.EncryptModelWeights();
C.UploadUpdatedModelWeights();
A.CronModelUpdate() {
    if A.ValidateModelWeights() ? return : null;
    w := A.AggregateAndAverageWeights();
    pcks := C.PartialKeySwitchShareGen(sk, pk, w);
    pcksC := ckks.PublicKeySwitchShare();
    pcksC := C.AggregateKeySwitchShare(pcks, pcksC);
    wD := A.PublicKeySwitch(pcksC, w);
    A.SendUpdatedModelVersion(wD);
}
```

Homomorphic aggregation Once the aggregator's scheduler detects that a sufficient number of fresh ciphertexts has been collected, it periodically invokes `A.CronModelUpdate()` as shown in line 3 of Algorithm 5.3. In this routine, the aggregator first validates the received model updates via `A.ValidateModelWeights()` (line 4), which checks the consistency of the dimensional, the alignment of the versions and the completeness of the updates, and aborts the round early if validation fails. For all valid ciphertexts w, the aggregator then computes a homomorphic aggregate using `A.AggregateAndAverageWeights()` on line 5, which implements a variant encoded by CKKS of Federated Averaging by adding the ciphertexts and multiplying by an appropriate scalar factor, such as $1/|C|$ for uniform weighting or a weighted factor reflecting the size of the local dataset. The resulting ciphertext `w` in line 5 corresponds to an aggregated ciphertext that encodes the next global weight vector but remains encrypted under the collective key.

Collective key switching and decryption (ColPubKeySwitch) To make the global model usable for clients, the aggregator must convert the aggregated ciphertext `w` to a decryptable ciphertext under a designated target key or into a global plaintext weight vector. For this purpose, it initiates a collective public-key-switching protocol embedded in the subsequent lines of Algorithm 5.3. Each client computes a partial key-switch contribution via `C.PartialKeySwitchShareGen(sk, pk, w);` in line 6, using its secret-key share sk_i, the collective public key pk and the aggregated ciphertext `w` as specified by the multiparty CKKS construction. The aggregator initializes an empty public key-switching share `pcksC := ckks.PublicKeySwitchShare();` in line 7 and then aggregates the clients' partial contributions by calling `C.AggregateKeySwitchShare(pcks, pcksC);` in line 8, resulting in a combined key-switching key `pcksC`. Finally, the aggregator applies this key-switching key to `w` via `A.PublicKeySwitch(pcksC, w);` in line 9, obtaining a ciphertext that can be decrypted under the target key and subsequently decrypting it locally to de-

rive the global plaintext weight vector `wD`, denoted in the same line. A fully distributed variant in which the final decryption is jointly performed by the clients is conceptually possible but is left as future work.

Model distribution After decryption, the aggregator persists `wD` as the new global model version and exposes it through a dedicated endpoint in the Federated Aggregator service, as represented by the call to `A.SendUpdatedModelVersion(wD);` in line 10 of Algorithm 5.3. Each Homomorphic Encryption Service periodically polls this endpoint, retrieves the latest global weights and forwards them to the local ML service, which replaces or updates its model parameters accordingly and continues training from the new global starting point. In this way, the `GeneratePublicKey` and `GenerateRelinearizationKeys` phases are executed once during registration (see Algorithm 5.2), while the train and update cycle captured in Algorithm 5.3 is executed recurrently and asynchronously across clients, providing robustness to heterogeneous client speeds and intermittent connectivity while preserving the end-to-end encryption guarantees of the multiparty CKKS protocol.

5.5.3 Evaluation

The performance of HEX-FL is experimentally evaluated using a classification workload on the MNIST dataset. The environment consists of a single Apple MacBook Pro 2021 with an M1 Max SoC and 32 GB RAM, on which all services (aggregator, Homomorphic Encryption Services and ML Services) run as separate host processes without containerization to avoid virtualization overhead. Each service listens on a distinct TCP port and inter-service communication uses HTTP over the local network stack.

The ML model is a small Convolutional Neural Network (CNN) implemented in PyTorch, with two convolutional layers (each followed by ReLU and pooling) and two fully connected layers, as commonly used in MNIST experiments. The MNIST data are partitioned among clients by splitting the training set into equally sized subsets per client, with an 80–20 train-test split at each client. For each client, one local training epoch is performed between global aggregation steps.

The metrics collected in the evaluation include:

- Runtimes per protocol phase (key generation, encryption, upload, aggregation, key-switch generation, client-side key switch, model update), measured in milliseconds.
- Memory consumption per service and per phase, measured in megabytes.
- Classification accuracy on the MNIST test set after a specified number of global aggregation rounds.

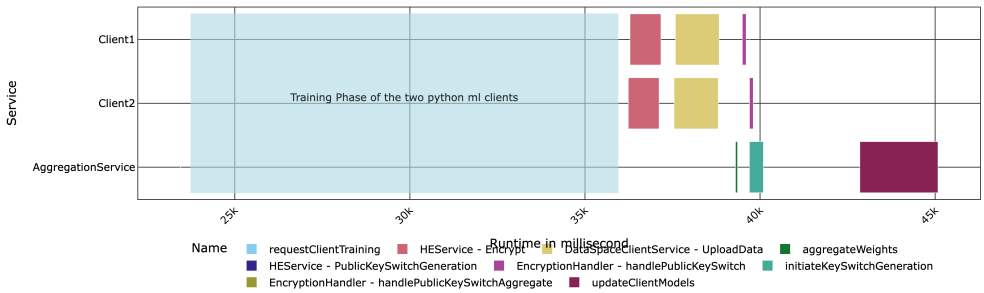


Figure 5.10: Timeline of one HEX-FL training and update cycle with two clients, showing the relative position of training, encryption, aggregation, key switching and model update phases. Results from Sachweh et al., 2026.

Runtime Behavior

To characterize the runtime behavior of HEX-FL, this section analyzes the temporal decomposition of a complete training and update cycle with two participating clients and evaluates the scalability with multiple clients.

Phase Decomposition for Two Clients A first set of experiments decomposes the end-to-end execution into protocol phases and measures their runtimes with two clients. For this purpose, each service logs timestamps at the beginning and end of relevant phases and a timeline is constructed to visualize their temporal ordering, as shown in Figure 5.10.

The initial client registration and local secret-key generation steps are fast relative to subsequent phases and contribute negligibly to the overall runtime. In contrast, the `PublicAndRelinKeyGen` phase, in which the aggregator coordinates collective public-key and relinearization-key generation, dominates initialization. For two clients, this phase takes approximately 5,299 ms and must be executed once per client registration cycle and every time the federation membership changes. The cost of key generation therefore scales roughly linearly with the number of registration events, although the cost per-event grows only mildly with $|C|$. Detailed runtime numbers to the relevant phases of HEX-FL are shown in Table 5.2.

After registration, the services remain idle until local training is complete, since HEX-FL uses a push-based interface from the ML Service to the Homomorphic Encryption Service. In the reported experiments, the longest idle period occurs between roughly 20,000 ms and 33,000 ms, during which the ML Services train their local MNIST models. Once training finishes, the Homomorphic Encryption Services perform the `Encrypt` and upload phases. For two clients, encrypting the model weights takes about 861 ms per client and uploading the resulting ciphertexts to the aggregator takes about 1,170 ms,

Agg Server	2	3	4	8	12
UpdateModels	2216	3342	4485	9165	16574
KeySwitchGen	367	606	790	1704	6954
EncryptionSetup	5299	5408	5515	5883	6262
AggregateWeights	55	66	76	123	352
Client					
UploadWeights	1170	1079	1270	1517	2687
Encrypt	861	896	918	1045	1956
PublicKeySwitch	125	149	149	168	463

Table 5.2: Runtime on the aggregator for homomorphic operations and model updates (in ms). Measurements were taken from Sachweh et al., 2026.

resulting in approximately 2,031 ms from the end of training to completion of upload per client when overheads are included.

The model update is triggered asynchronously by the aggregator's `CronModelUpdate` task, which is scheduled periodically. After this cron job detects new encrypted weights and validates them, it executes homomorphic aggregation and initiates the collective key switch. For two clients, the server-side collective key-switch generation (`KeySwitchGen`) requires around 367 ms, while the client-side `PublicKeySwitch` calls, which compute partial key-switch shares, take approximately 125 ms per client. Following the key switch, the aggregator decrypts the aggregated ciphertext and executes `UpdateClientModel` to push the updated weights to the clients.

An entire cycle (local training, encryption and upload, aggregation, key switch and model update) takes about 45,307 ms in the two-client scenario. When subtracting HE-specific overheads and keeping the asynchronous structure unchanged but without encryption, a comparable cycle would take approximately 33,789 ms. In this setting, local training accounts for about 18,222 ms. Thus, the integration of HE into the FL loop increases runtime per cycle by roughly one third, primarily due to key generation and collective key-switch operations.

Scalability with the Number of Clients To analyze scalability, the same phase-wise measurements are conducted for configurations with 2, 3, 4, 8 and 12 clients. On the client side, the runtimes of `Encrypt`, `UploadWeights` and `PublicKeySwitch` increase with $|C|$ but not strictly linearly. The detailed values are summarized in Table 5.2. The table is divided into two sections. One represents the measured results from the Aggregation Server phases and one for the client side phases. All measurements are averaged over multiple runs and all clients.

For example, the execution time of `PublicKeySwitch` increases from around 125 ms for two clients to about 463 ms for twelve clients, corresponding to a factor of approximately 3.7. Encryption and upload times per client grow by a factor of about 2.3 between two

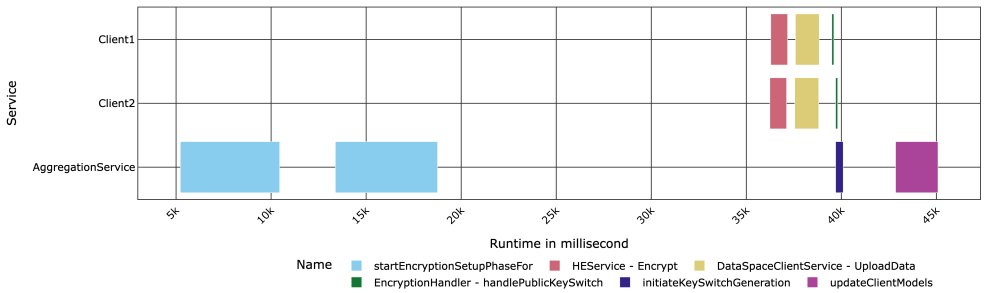


Figure 5.11: Runtime profile of the initial training and update sequence, highlighting the relative cost of encryption, aggregation, key switching and model update. Results from Sachweh et al., 2026.

and twelve clients. This indicates moderate per-client overhead growth driven by larger global models and network contention.

On the aggregator side, phases exhibit distinct scaling patterns. The `UpdateModels` phase, which distributes decrypted global weights, shows a nearly linear increase, with the execution time for twelve clients being about 7.47 times that for two clients. The `AggregateWeights` phase, which sums and scales ciphertexts, behaves similarly, with a runtime factor of approximately 6.4 between the two- and twelve-client setups. In contrast, `EncryptionSetup` (collective key generation) only increases slightly per run but is executed once per client registration and thus accumulates roughly linearly over the lifetime of the system.

The most critical phase for scalability is `KeySwitchGen`. In this phase, the runtime almost doubles when moving from two to three clients and scales by a factor of approximately 18.9 when comparing two and twelve clients. This suggests an approximate quadratic relationship between the number of clients and the cost of collective key switching, consistent with the structure of MHE key-switching algorithms. As a result, HEX-FL can accommodate a moderate number of clients with reasonable overhead but becomes increasingly expensive when the federation grows large, particularly in terms of server-side key-switch generation time.

To illustrate the temporal distribution of these phases in the first training run, Figure 5.11 shows a detailed Gantt-style view.

Memory Consumption

Memory evaluation measures per-service memory usage during different protocol phases and how this changes with $|C|$. With two clients, memory usage during registration and key generation is relatively low, below 100 MB for each service. During encryption and upload, each Homomorphic Encryption Service allocates buffers for CKKS plaintexts and

ciphertexts, resulting in a peak memory footprint of roughly 550 MB per client. The aggregator's memory usage is highest during `AggregateWeights`, where it holds several ciphertexts simultaneously. In the two-client scenario, this can exceed 1 GB.

The phases `KeySwitchGen` and `UpdateClientModel` also exhibit transient memory spikes. During `KeySwitchGen`, temporary buffers are allocated for partial key-switch shares and intermediate ciphertexts, increasing the aggregator's memory usage, while clients temporarily store shares and updated ciphertexts. After model update and subsequent garbage-collection cycles in Go, memory usage decreases but rarely returns exactly to baseline, reflecting the behavior of the runtime's allocator.

The temporal evolution of memory usage for the aggregator and two clients is visualized in Figure 5.12, annotated with the main phases of the protocol.

When increasing $|C|$, the usage of memory per-client in the corresponding phases remains relatively constant, which implies that the total memory requirement on the client side grows approximately linearly with the number of clients. On the aggregator, memory usage in phases that buffer data from all clients, particularly `AggregateWeights` and `UpdateModels`, increases significantly. This trend and the concrete numbers for the varying amounts of clients can be seen in Table 5.3. For instance, when moving from two to eight clients, memory consumption in `AggregateWeights` grows by about 3.8× and in `UpdateModels` by about 4.2×. These factors are consistent with the need to store and process ciphertexts from more clients concurrently.

In general, the aggregator emerges as the main memory bottleneck in HEX-FL, whereas individual clients remain relatively lightweight. In practical deployments, this suggests provisioning sufficient RAM for the aggregator, especially when targeting larger numbers of clients or more complex models than the MNIST CNN used in the evaluation.

Figure 5.12: Memory usage of aggregator and clients over time, annotated with registration, training, encryption and aggregation phases. Results from Sachweh et al., 2026.

Agg Server	2	3	4	8	12
UpdateModels	529	575	1085	1745	2229
KeySwitchGen	1225	1068	1605	2324	3107
EncryptionSetup	43	75	88	80	95
AggregateWeights	1079	955	1928	2604	4145

Table 5.3: Memory measurements of longest running HEX-FL processes on the aggregation service. Measured in megabytes (MB). Columns numbered (2, 3, 4, 8 and 12) specify the amount of clients used in the test. Measurements from Sachweh et al., 2026.

Model Accuracy and Convergence

The evaluation of model quality contrasts HEX-FL with a non-private centralized baseline and examines how accuracy behaves for different client configurations and aggregation rounds. The centralized baseline (*Global training*) trains a single instance of the CNN on the full MNIST training set without HE or FL, reaching 98.7% test accuracy after one epoch. Further epochs bring negligible improvements. This baseline therefore represents an upper bound on achievable accuracy with the given architecture and a lower bound in terms of computational and communication overhead.

HEX-FL is evaluated for a varying number of clients $|C| \in \{2, 3, 4, 8, 12\}$ to study the combined impact of federation size, data partitioning and encrypted aggregation on predictive performance. For each configuration, all clients perform one local training epoch per round and then participate in a HEX-FL exchange, during which encrypted model updates are aggregated and redistributed using the CKKS-based protocol described in Section 5.5.2. Figure 5.13 summarizes the resulting test accuracies across client counts and aggregation exchanges, showing both the centralized baseline and the global HEX-FL model after up to four rounds.

For small and medium federation sizes (2–8 clients), the curves for the first and subsequent HEX-FL exchanges remain close to the centralized baseline, indicating that encrypted federated averaging preserves the precision of the underlying CNN when each client still owns a sufficiently large local dataset. In a two-client setup, each client trains on half of the dataset and the global model after the first aggregation is only slightly below 98.7% accuracy. For three clients, final accuracies remain in the high 98% range, demonstrating that moderate client counts do not significantly harm model quality in this setting.

For four and eight clients, each client's local dataset becomes smaller, which increases overfitting and reduces the diversity of each local training run. Initial local accuracies before aggregation are in the mid-90% range (approximately 94.5% for eight clients) and the global accuracy after the first averaging step remains close but slightly lower, with only a minor additional loss attributable to approximate aggregation based on CKKS. In these regimes, the dominant factor that affects accuracy is the statistical effect of data partitioning rather than the use of HE.

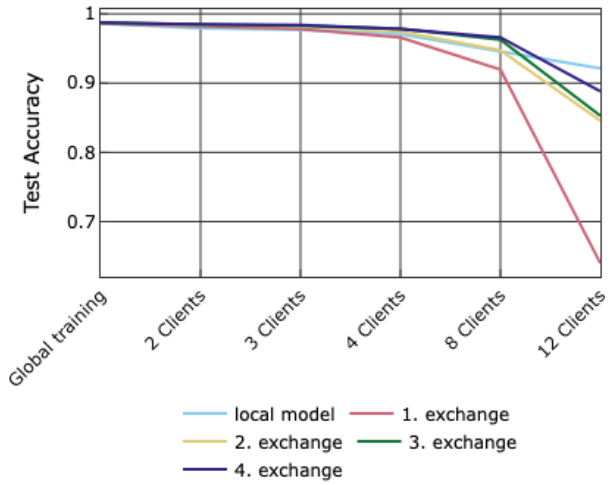


Figure 5.13: Test accuracy of centralized training ("Global training") and HEX-FL with 2, 3, 4, 8 and 12 clients over up to four aggregation exchanges. Results from Sachweh et al., 2026.

The most challenging configuration is the twelve-client scenario, where each client trains on a relatively small subset of MNIST and, therefore, strongly overfits to its local data distribution. After one epoch, local models achieve approximately 92.1% accuracy in their respective test splits, but the aggregated global model after the first HEX-FL exchange reaches only approximately 63.9% accuracy in the complete MNIST test set, indicating poor initial generalization. This pronounced drop is mainly caused by the mismatch between many small, overfitted local models and the global distribution, rather than by numerical approximation errors introduced by CKKS.

Crucially, additional HEX-FL rounds steadily improve the global model in the twelve-client case. After the second aggregation, test accuracy increases to around 84.5% and after the fourth aggregation it reaches approximately 88.8%, as reflected by the successive "2. exchange" to "4. exchange" curves in Figure 5.13. Later exchanges thus narrow the gap between local and global performance, although they do not completely close it within the limited number of rounds studied.

5.5.4 Advantages in the Context of Federated Data Spaces

In summary, HEX-FL preserves the fundamental convergence behavior of Federated Averaging while providing strong cryptographic protection for model updates. For moderate numbers of clients with sufficient local data, accuracy close to the centralized baseline can be achieved within a small number of encrypted aggregation rounds and the impact of CKKS's approximation error on model quality is negligible compared to statistical effects.

For larger federations with small local datasets, more HEX-FL exchanges are required to reach acceptable accuracy and some residual performance gap to centralized training remains due to data fragmentation rather than encryption-specific artifacts.

Beyond convergence aspects, the main contribution of HEX-FL lies in introducing model-level privacy through HE of model parameters. In contrast, DP-LLP and DP-LSTM ensured data-level privacy by perturbing data representations or local gradients using DP. HEX-FL therefore extends these approaches by protecting both the training process and the exchanged model weights, ensuring that sensitive information about local data can not be inferred from shared parameters.

Furthermore, HEX-FL has been specifically designed to comply with Federated Data Space requirements, making it deployable in cross-enterprise environments where organizations maintain full data sovereignty while still contributing to a shared model. This architectural compatibility enables secure and transparent collaboration across industrial or institutional boundaries without compromising intellectual property or regulatory constraints.

Consequently, HEX-FL fully addresses **Research Question 2**, which investigates how advanced privacy-preserving mechanisms can enable FL in complex, multi-stakeholder ecosystems. By combining practical scalability with cryptographic security guarantees, HEX-FL establishes a bridge between theoretical HE research and real-world Data Space implementations.

5.6 Summary

This chapter has examined how privacy-preserving learning can be realized in cross-enterprise Data Spaces by combining algorithmic and cryptographic protection mechanisms. Building on the foundations of DP and HE, it presented DP-LLP and DP-LSTM as fully distributed, GDPR-aligned learning schemes for time series forecasting and contrasted them with the HEX-FL framework, which applies multiparty CKKS-based encryption to secure federated model aggregation. The following subsections synthesize these contributions, compare their architectural and privacy characteristics, and outline how differentially private LLP approaches and HE-based FL can be jointly deployed to achieve multi-level privacy on both data and model representations.

5.6.1 Differentially Private Learning from Label Proportions (DP-LLP)

The first main section of this chapter introduced DP-LLP as a privacy-preserving learning scheme explicitly designed for cross-enterprise Data Spaces in which multiple organizations operate sensor nodes and must comply with data protection regulations such as the GDPR while still exploiting shared information for forecasting tasks. The core novelty is the integration of a tunable DP privacy factor ε into LLP. For the algorithm, batch-wise label counts are perturbed with Laplace noise calibrated to the sensitivity of counting

queries, producing noisy label proportions that satisfy individual-level ε-DP and can be safely exchanged between enterprises.

Architecturally, DP-LLP realizes a fully distributed, peer-to-peer communication pattern. Each node partitions its local time series into batches, computes label histograms, privatizes them and shares only these noisy aggregates with a small neighborhood of collaborating partners, without any central coordinator. This directly supports GDPR principles such as data minimization and purpose limitation, because first, raw trajectories never leave organizational boundaries, second, exchanged data is reduced to the minimum necessary (low-dimensional label proportions) and as last, formal DP guarantees bound the risk of re-identifying individuals from cross-enterprise analytical artifacts.

The use of a single privacy parameter ε provides a transparent control instance: smaller ε yields stronger protection at the cost of more noise, while moderate values (e.g. $\varepsilon = 0.1$) preserve most of the LLP utility on real-world traffic data. In cross-enterprise deployments, this enables contract- and policy-aligned privacy budgeting, where partners can agree on acceptable ε ranges that jointly satisfy regulatory and business requirements. At the same time, communication traffic is minimal because only compact histograms are exchanged instead of full gradients or models, which is beneficial for heterogeneous industrial networks with limited bandwidth.

5.6.2 Differential Privacy Encoded in LSTM Model Architectures (DP-LSTM)

The DP-LSTM architecture extends DP-LLP to deep temporal models while preserving its cross-enterprise and GDPR-aligned design principles. The key innovation is that each company hosts a local LSTM that encodes temporal patterns in its own time series and only differentially private label histograms, parameterized by a privacy factor ε, are shared across organizational boundaries as spatial context. This preserves the strong data minimization and locality properties of DP-LLP while upgrading expressiveness from shallow clustering to deep ML modeling.

In terms of architecture, the system maintains a fully distributed, graph-based communication model across enterprises. Nodes in the graph represent organizational sites or sensor locations and edges denote bilateral agreements to exchange noisy histograms. No central parameter server is required and raw data, model parameters and gradients remain strictly local, which simplifies GDPR-compliant governance because each partner retains full control over its data processing and exposure surface. Only small, DP-protected label proportion vectors are transmitted (similar to DP-LLP), drastically reducing network traffic compared with gradient-based FL schemes and making the approach suitable for constrained cross-enterprise links.

The novel design choice is to inject DP noise only at the level of low-dimensional label histograms and to fuse these aggregates with local LSTM features in a final Dense layer, rather than perturbing high-dimensional gradients or model parameters. This first reduces sensitivity and thus the amount of noise needed for a given ε, second, avoids

amplifying noise through deep stacks of layers and finally keeps the shared quantities semantically interpretable for auditing and contractual reasoning (e.g., *distribution over congestion levels* instead of opaque numeric gradients). For cross-enterprise Data Spaces, this combination of deep temporal modeling, explicit DP control via ε, full locality of raw data and parameters and lightweight message exchange makes DP-LSTM a GDPR-friendly alternative to classical FL.

5.6.3 Homomorphic Encryption-based Federated Learning (HEX-FL)

The HEX-FL section describes a complementary approach in which privacy is enforced cryptographically through HE rather than statistically through DP noise on labels. In a typical cross-enterprise FL setup, each organization trains a local model on its own data and encrypts model updates under an HE scheme before sending them to an aggregator, which performs secure aggregation directly on ciphertexts and only produces a decrypted global model or update at the end. This protects individual enterprise updates from an honest but untrusted server and other parties, contributing to GDPR-compliant confidentiality during transport and aggregation.

Compared to DP-LLP and DP-LSTM, HEX-FL preserves full model expressivity and allows reuse of standard gradient-based training pipelines, but usually at the cost of significantly increased computational load and communication volume due to the large size and complexity of HE ciphertexts. In cross-enterprise architectures, this can be challenging for partners with limited compute or bandwidth and it often implies a more centralized topology with an explicit aggregation service, which increases the organizational and technical attack surface compared to fully peer-to-peer schemes. While HE ensures that the aggregator cannot inspect raw updates, it does not by default bound the information that can be inferred from the final global model. Thus, HEX-FL is naturally complementary to DP and may need to be combined with DP mechanisms to fully address GDPR's requirements on inference risks.

Nevertheless, HEX-FL aligns with GDPR's data minimization and confidentiality principles in a different manner. No raw data leaves the company, only encrypted updates are shared and access to plaintext aggregates can be tightly controlled through key management and decryption policies. Unlike DP-LLP and DP-LSTM's fully distributed label-proportion exchange, HEX-FL optimizes for strong cryptographic guarantees in a (possibly) centralized or hierarchical aggregation architecture, making it particularly suitable when regulatory or contractual constraints emphasize secure computation at a central service over fully decentralized collaboration.

5.7 Discussion

The discussion in this chapter compares and integrates the different privacy-preserving learning approaches developed in this work in order to answer **Research Question 2** on how expressive yet privacy-compliant analytics can be realized in cross-enterprise Data

Category	Criteria	1	2	3	DP-LLP	DP-LSTM
LLP	Label propor. opt.	✓	-	-	✓	✓
Distributed	Label propag. opt.	(✓)	(✓)	-	(✓)	✓
Distributed	Spatial aware	-	✓	✓	(✓)	✓
Distributed	Peer to Peer	(✓)	-	(✓)	✓	✓
Privacy	Privacy	-	-	-	✓	✓
Non-Linear	Deep Learning	✓	✓	(✓)	-	✓
Time	Bags are ordered	-	(✓)	-	✓	✓
Time	Time features	-	✓	✓	-	✓
Time	Incremental	-	✓	-	-	✓
Time	Online Learning	-	-	-	-	-
Application	Traffic specific	-	✓	-	✓	✓
Performance	Accurate	✓	✓	(✓)	-	✓
Energy	E. Consumption	-	-	-	✓	-

Table 5.4: Comparison of related works with methods proposed for DP-Data and DP-Model based learning. 1: Dulac-Arnold et al., 2019, 2: Yu et al., 2018, 3: Majcherczyk et al., 2021, DP-LLP: Sachweh et al., 2021, DP-LSTM: Sachweh et al., 2022 (table derived from Sachweh et al., 2022).

Spaces. Building on the criteria summarized in Table 5.4, the first part contrasts classical LLP techniques with the proposed DP-LLP and DP-LSTM variants, highlighting their respective capabilities in terms of label-proportion based learning, spatial and temporal modeling, formal DP and resource efficiency in distributed settings. The second part analyzes the performance trade-offs between the Lattigo- and OpenFHE-based instantiations of the HEX-FL framework, demonstrating that both provide equivalent homomorphic protection of model updates while exhibiting markedly different runtime characteristics. Finally, the chapter discusses the combination of DP-LLP, DP-LSTM and HEX-FL into a unified multi-level privacy architecture, showing how their complementary strengths jointly provide data-level and model-level protection and thus contribute a concrete, technically grounded answer to **Research Question 2**.

5.7.1 Comparison of DP-LLP and DP-LSTM

In addition to the general explainability of the applied methodologies, it is also crucial to compare their specific characteristics. The criteria summarized in Table 5.4 from Sachweh et al., 2022 contrast state-of-the-art LLP approaches with the proposed variants DP-LLP and DP-LSTM developed in this work. The listed dimensions include relevant aspects for GDPR-aware cross-enterprise Data Spaces and serve as a foundation for evaluating the proposed methods. They capture whether a method supports label-proportion based learning, distributed and peer-to-peer operations, spatial awareness, formal privacy protection, non-linear deep models, temporal structure, application specificity to traffic scenarios, predictive performance and energy efficiency.

With respect to *LLP and distributed learning*, both DP-LLP and DP-LSTM employ optimization over aggregated label proportions as their primary learning signal. This enables model training across organizations without requiring access to instance-level labels and aligns well with the decentralized architecture of Data Spaces. Moreover, the peer-to-peer design of these methods eliminates the need for a central coordinator, strengthening data sovereignty and facilitating compliance with GDPR requirements. Nevertheless, the degree of spatial awareness and label propagation capabilities differs between the two approaches: while DP-LLP models spatial dependencies via neighborhood-based label exchange, DP-LSTM integrates spatially aggregated, differentially private histograms directly into its temporal encoding, embedding spatial context explicitly into the learned representation.

Regarding *privacy and non-linearity*, both frameworks incorporate formal DP mechanisms to ensure protection against inference attacks. Privacy is controlled by the parameter ε, which governs the magnitude of Laplace noise added to the aggregated label statistics at each node. The DP-LLP approach introduces this DP-protected aggregation on top of a lightweight, k-means-based model, while DP-LSTM combines these privacy-enhanced label aggregates with a non-linear deep architecture based on recurrent networks. This structural difference allows DP-LSTM to capture more complex spatio-temporal dependencies, thereby improving predictive accuracy, which comes at the cost of higher computational complexity.

In terms of *temporal structure and incremental learning behavior*, DP-LLP organizes the data in ordered batches to reflect temporal progressions, but lacks an explicit mechanism to encode time dependencies within the model architecture. By contrast, DP-LSTM directly processes time series data. It incorporates both spatial and temporal awareness through sequential modeling and aggregated spatio-temporal histograms. This design is particularly advantageous for cross-enterprise applications with continuously arriving sensor data or streaming contexts, such as traffic forecasting. However, both methods remain limited in supporting true online learning, as none is fully incremental.

Taking into account *application specificity, performance and energy efficiency*, both methods demonstrate their applicability in cross-enterprise traffic forecasting, a canonical use case for federated Data Spaces due to the high degree of temporal and spatial correlation in mobility data. The table characterizes DP-LSTM as accurate, benefiting from its deep temporal modeling, while DP-LLP provides competitive performance under strict privacy budgets and is marked for low energy consumption. This efficiency results from its shallow model structure and simple aggregation logic, which makes it suitable for edge and IoT environments where computing and energy resources are limited.

In general, DP-LLP offers a formally grounded privacy mechanism through the tunable DP parameter ε, minimal data exchange through the share of label proportions and low computational overhead. It naturally fits peer-to-peer learning paradigms and aligns with GDPR principles of data locality and minimization. Its main limitations lie in the restricted modeling capacity of its shallow architecture and the indirect handling of temporal and spatial dependencies. Conversely, DP-LSTM preserves the same formal DP protections

and distributed data sovereignty, but adds deep spatio-temporal encoding and non-linear learning capacity. This enables higher predictive accuracy but requires greater computational effort and may lead to increased energy consumption at the edge.

Beyond these specific implementations, it is important to note that LLP represents a flexible and model-agnostic paradigm. As long as the underlying model type accepts LLP-type feature inputs, LLP can be employed with a broad variety of model families, from shallow statistical methods to complex neural networks. Integrating LLP with more complex architectures, such as LSTM-based models, demands additional engineering effort, but, as demonstrated by the DP-LSTM case, the underlying aggregation and optimization principles remain general and well-suited for distributed and privacy-aware learning environments. The aggregation concept central to LLP provides an elegant, easily implementable and privacy-guaranteed mechanism that aligns strongly with the technical and legal requirements of cross-enterprise Data Spaces. However, the success of LLP depends on the nature of the data that can be meaningfully aggregated. This condition typically applies to temporally correlated or streaming data, as in traffic or sensor systems, but may not hold for arbitrary datasets without inherent structure or homogeneity suitable for aggregation.

5.7.2 Performance Differences between Homomorphic Encryption Libraries

Within the HEX-FL framework, both Lattigo and OpenFHE are instantiated with CKKS-based multiparty key generation, encrypted model upload, homomorphic aggregation and collective key switching, so that in both cases all model updates remain encrypted throughout the federated training workflow described in Section 5.5.2. In other words, both library backends realize the same qualitative privacy guarantees for cross-enterprise FL in Data Spaces, namely end-to-end protection of gradients and weights against an honest-but-curious aggregator and external adversaries, analogous to other CKKS-based PPML frameworks in the literature.

However, the execution-time measurements in Table 5.5 show that the two CKKS implementations lead to markedly different performance profiles when integrated into the concrete HEX-FL prototype. Table 5.5 summarizes the average runtimes per phase (in ms) for the HE Client and Aggregation Server components, using the same experimental setup as Table 5.2, but evaluated once with a Lattigo backend and once with OpenFHE under comparable RLWE security parameters and model configuration.

On the client side, encryption and collective public-key switching dominate the runtime overhead for both libraries, but Lattigo consistently outperforms OpenFHE by one to two orders of magnitude in these phases in the given configuration. For example, `Encrypt` requires approximately 0.86s with Lattigo but around 25.1s with OpenFHE and `PublicKeySwitch` increases from roughly 0.13s to about 6.6s when replacing Lattigo by OpenFHE. These effects are in line with independent CKKS benchmarks that report Lattigo as particularly efficient for moderate multiplicative depths and vector sizes, whereas

HE Client	Lattigo	OpenFHE
Encrypt	861	25095
PublicKeySwitch	125	6579
UploadWeights	1170	1107
GetParameters	8	464
GetPublicKey	1	329
ClientRegister	15	28
Agg Server		
PublicAndRelinKeyGen	5299	89905
UpdateClientModels	2216	19687
AggregateWeights	55	16483
InitiateKeySwitchGeneration	367	1868
AddClient	0	1

Table 5.5: Phase-wise runtime comparison of Lattigo and OpenFHE within the HEX-FL prototype (ms per phase, same hardware and model as in Section 5.5.3). Measurements are extracted from Sachweh et al., 2026.

OpenFHE tends to incur higher costs for similar CKKS parameter sets when its more general feature set is not fully exploited.

On the aggregation server, the relative differences are even more pronounced in the initialization and aggregation-centric phases that are central to HEX-FL. The joint `PublicAndRelinKeyGen` step, which corresponds to the combined `GeneratePublicKey` and `GenerateRelinearizationKeys` procedures in Section 5.5.2, completes in about 5.3s with Lattigo but takes close to 90s with OpenFHE under the same security and precision settings, making it a major bottleneck whenever the federation membership changes. Similarly, the `AggregateWeights` and `UpdateClientModels` phases, which implement the homomorphic averaging and subsequent distribution of the global model as described in Section 5.5.2, grow from tens or low thousands of milliseconds with Lattigo to more than an order of magnitude higher with OpenFHE. These tendencies are consistent with broader cross-library studies, which show that OpenFHE trades higher per-operation latency for a richer set of bootstrapping and scheme-switching capabilities, whereas Lattigo optimizes CKKS/BFV/BGV for server-side use with relatively shallow circuits.

From a privacy and accuracy perspective, both HEX-FL versions achieve the same qualitative guarantees. The CKKS-based multiparty protocol, the end-to-end encryption of model updates and the training dynamics of the CNN on MNIST remain unchanged and no statistically significant accuracy differences are observed between the two backends in the evaluated setting. In that sense, either library can be used to implement the encrypted aggregation pipeline required for cross-enterprise FL in Data Spaces and the choice does not affect the underlying threat model or convergence behavior.

However, for the concrete HEX-FL prototype and workload, the quantitative results clearly indicate that Lattigo is the preferable backend when the primary objective is ef-

ficient encrypted aggregation with moderate multiplicative depth and without advanced features such as functional bootstrapping or scheme switching. Across all critical phases (collective key generation, encryption, homomorphic aggregation and public-key switching) Lattigo reduces execution times by one to two orders of magnitude compared to OpenFHE in the presented experiments, while providing equivalent privacy protection for model updates. Consequently, within the scope of this work, Lattigo emerges as the more suitable HE backend for privacy-preserving FL in Data Spaces when runtime efficiency is the dominant requirement and the extended functionality of OpenFHE is not required. Therefore, the main section of this work only presented the Lattigo implementation of HEX-FL, as it is the more suited implementation protocol for use-cases in federated Data Spaces and ecosystems.

5.7.3 Combining DP-LLP, DP-LSTM and Homomorphic Encryption for Multi-Level Privacy

The previous sections have introduced three complementary classes of privacy-preserving techniques for cross-enterprise analytics in Data Spaces: **DP-LLP** for differentially private learning from label proportions, **DP-LSTM** for deep spatio-temporal modeling under DP and **HEX-FL** for CKKS-based HE of federated model updates. Together, these methods form a toolbox that targets distinct attack surfaces, data-level leakage via statistical inference, model-level leakage via gradients and weights and architectural limitations of purely local training, thus enabling a multi-faceted response to **Research Question 2**.

From a data-centric perspective, DP-LLP and DP-LSTM implement formal (ε, δ)-DP mechanisms that limit the influence of individual records while still supporting distributed learning with only noisy label aggregates exchanged between organizations. DP-LLP achieves this by injecting calibrated noise into batch-wise label counts before redistribution, whereas DP-LSTM integrates differentially private spatio-temporal histograms directly into a non-linear recurrent model, thereby combining privacy guarantees with expressive temporal representation learning. The key contribution at this level is that both algorithms demonstrate how DP can be embedded into LLP-style and LSTM-based learning pipelines without sacrificing the ability to capture relevant spatial and temporal patterns in realistic traffic forecasting scenarios.

At the model- and communication-centric level, the HEX-FL framework extends these protections by ensuring that all model parameters and updates exchanged during federated training remain encrypted under a multiparty CKKS scheme. In contrast to DP-LLP and DP-LSTM, which protect the information content of local statistics and gradients, HEX-FL directly secures the transport and aggregation layers, preventing honest-but-curious aggregators or external adversaries from observing plaintext model states. An important design result is that HEX-FL preserves the convergence behavior of standard federated averaging while enabling practical encrypted aggregation with a suitable ho-

momorphic backend, thereby turning homomorphic protection from a purely theoretical construct into an executable protocol for Data Spaces.

The combined architecture that emerges from this work applies DP-LLP or DP-LSTM locally and HEX-FL globally, yielding a layered privacy design with clearly separated responsibilities. Concretely, each participant first trains a local model under DP, either using noisy label proportions or DP-enhanced gradient updates and then submits the resulting parameters in encrypted form to the HEX-FL aggregation protocol, where all subsequent operations on the global model occur over ciphertexts. This pipeline ensures that the contribution of any individual record to the local model is formally bounded and no plaintext model parameters are exposed during cross-enterprise synchronization, even in the presence of untrusted infrastructure or intercepted communication channels.

In terms of answering **Research Question 2**, the joint use of **DP-LLP**, **DP-LSTM** and **HEX-FL** represents a central contribution of this thesis, as it demonstrates that high expressiveness, formal privacy guarantees and distributed deployment can be reconciled within a single architectural framework. At the algorithmic level, **DP-LLP** and **DP-LSTM** show that LLP-based and deep temporal models can be trained under strict DP while still achieving competitive accuracy in realistic traffic forecasting tasks; at the system level, **HEX-FL** shows that fully encrypted joint model training is feasible with acceptable overhead when using an efficient HE backend such as Lattigo. Taken together, these results substantiate that DP-based local learning and HE-based federated aggregation are not competing alternatives but complementary building blocks and that their integration provides a concrete, technically realized solution pattern for designing expressive, privacy-preserving analytics in cross-enterprise Data Spaces as posed in **Research Question 2**.

6 Federated Unlearning with Data Spaces

The previous chapter on Privacy-Preserved Federated Learning *(Chapter 5) has shown that DP-LLP, DP-LSTM and HEX-FL provide strong protection of data during training and inference in industrial Data Spaces, but it has intentionally left open how already trained models can react to legal data removal demands. This chapter addresses the remaining part of the overall research problem, namely how GDPR requirements, particularly those related to the removal of previously used data, affect such FL frameworks and which additional mechanisms are needed to make them compliant in practice (see* ***Research Question 3****).*

To answer this question, Section 6.1 motivates the need for federated unlearning in the context of the GDPR with the right to be forgotten *and outlines why naive retraining is infeasible in federated settings. Section 6.2 positions the work with respect to existing privacy-preserving FL and introduces FedSSU as the algorithmic basis for unlearning. Section 6.3 describes how FedSSU is integrated into the HEX-FL and Data Space infrastructure, enabling the targeted removal of client contributions and propagation of updated models across all participants. Section 6.4 then presents a multi-layered certification mechanism that verifies code, models and execution traces to ensure that unlearned models are actually deployed and used by all clients. Section 6.5 evaluates the proposed method called* ***CeMUFl*** *in terms of runtime, unlearning quality, configurability and model divergence, thereby demonstrating that* ***CeMUFl*** *can operationalize GDPR-conform unlearning in federated Data Spaces. Finally, Section 6.7 discusses how these results relate back to the overarching* ***Research Question 3****, highlighting the main contributions of the chapter and clarifying to which extent* ***CeMUFl*** *closes the remaining gaps towards GDPR-conform FL.*

6.1 Introduction

In the previous chapters, the presented frameworks FedDScon, DP-LLP, DP-LSTM and HEX-FL jointly enable sovereign, privacy-preserving and interoperable data processing in industrial Data Spaces. FedDScon focuses on policy-based, contract-governed data exchange between organizations, while HEX-FL realizes homomorphically encrypted FL combined with DP-LLP and DP-LSTM which provide data level privacy. However, these mechanisms address confidentiality and privacy during training and inference, but do not provide explicit guarantees for the removal of information retrieved from already trained models.

From a regulatory perspective, this gap is crucial. The GDPR defines, among others, the *right to be forgotten* (Art. 17), which grants data subjects the right to request erasure of

their personal data and, by extension, the removal of its influence from downstream processing artifacts such as ML models. Although FL intuitively appears to be more aligned with this requirement, as raw data never leave the local environment, it does not automatically guaranty that the contribution of a client can be removed from the global model once training has been completed (Z. Liu et al., 2024). In particular, once local updates are aggregated, individual contributions are typically entangled in the global parameter space and cannot be trivially isolated or reversed. A naive way to comply with an erasure request would be to retrain the model from scratch on a sanitized dataset that excludes the to-be-forgotten data, but in federated settings such full retraining is usually prohibitively expensive in terms of computation, communication and time, especially when unlearning requests occur repeatedly or when models and datasets are large (Z. Liu et al., 2024).

Data Spaces, as instantiated through FedDScon, enforce data sovereignty by controlling access and usage policies at the level of data exchange contracts. However, they do not provide primitives for retrospectively invalidating model states that were trained on data accessed under previous contracts. Similarly, HEX-FL and the DP-LLP/DP-LSTM extensions ensure that training and inference are privacy-preserving but do not implement any form of machine or federated unlearning. To provide a complete path to GDPR-conformity for the overall architecture, the FL pipeline must be extended with unlearning capabilities that can be triggered by erasure requests and whose correct execution can be attested or certified.

Therefore, this chapter addresses the missing link between privacy-preserving training and regulatory erasure requirements. It directly contributes to answering **Research Question 3**, which asks what influence GDPR can have on such modern frameworks, whether the provided framework can be made fully GDPR-compliant and what is required to ensure such compliance.

6.2 Related Work

This section positions the proposed approach relative to existing privacy-preserved FL and FedSSU as an unlearning methodology. The first subsection briefly revisits privacy-enhancing techniques in FL and their limitations with respect to unlearning, while the second part summarizes FedSSU as a concrete federated unlearning framework that serves as the algorithmic basis for this chapter.

6.2.1 Privacy-Preserved Federated Learning

FL enables multiple parties to collaboratively train ML models without sharing raw data and is commonly combined with techniques such as secure aggregation, HE and DP to strengthen privacy guarantees (Z. Liu et al., 2024). In the context of this thesis, the previous chapter has introduced HEX-FL as a concrete framework that instantiates these concepts in industrial Data Spaces, combining homomorphically encrypted training with

the DP-LLP and DP-LSTM components to provide local DP for tabular and sequential models, respectively.

These extensions substantially improve confidentiality and limit information leakage during training and inference, but they do not offer mechanisms to remove the influence of specific clients or data points from an already trained model (Z. Liu et al., 2024). Most privacy-preserving FL schemes, including the HEX-FL-based solution presented in Chapter 5, treat unlearning as an orthogonal problem. This motivates the integration of explicit unlearning algorithms, such as FedSSU, into the existing FL pipeline to preserve GDPR-compliance.

6.2.2 FedSSU

FedSSU is a decentralized federated unlearning framework designed to efficiently remove the influence of client contributions from a global model while maintaining scalability and practicality (Leng et al., 2025). Rather than relying on central retraining, FedSSU enables clients to initiate and execute unlearning requests locally and collaboratively, making it suitable for settings where clients operate under heterogeneous conditions and where centralized access to raw data is not possible. FedSSU considers different types of unlearning, including client-level and class-level unlearning (FedSSU-Class) and leverages local computations to approximate the model that would have been obtained had the to-be-forgotten data never participated in training.

Conceptually, FedSSU builds on the standard FL process but augments it with targeted unlearning procedures. By leveraging model saliency and similarity constraints, FedSSU identifies and suppresses the influence of specific clients or classes on the global model (Leng et al., 2025). The framework introduces specialized unlearning rounds, during which participating clients apply modified local objectives that steer the model towards a counterfactual state that approximates the absence of the forgotten data. In practice, this is achieved by adapting the local training process and using stored information from previous rounds to guide the unlearning process.

A central component of FedSSU is its local loss function L_{local}, which balances three aspects:

- preserving performance on the remaining data,
- removing or attenuating the influence of the data to be forgotten and
- maintaining stability with respect to the previously learned model.

Although the exact definition of L_{local} depends on the specific unlearning scenario (e.g., client-level versus class-level), it typically extends the standard empirical risk term with additional regularization or penalty components that enforce unlearning constraints (Leng et al., 2025). Intuitively, L_{local} encourages parameter updates that reduce the model's dependence on forgotten data while avoiding catastrophic degradation on the

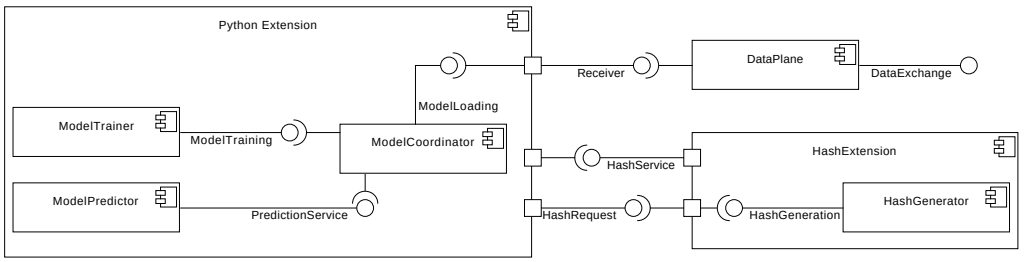


Figure 6.1: Component diagram for CeMUFl (FedSSU logic and certification process with hash extension) integration on client side HEX-FL modules.

remaining dataset. This approach has been shown to provide a practical trade-off between unlearning effectiveness and computational efficiency, making FedSSU a suitable base for integration into existing FL frameworks.

Despite these advantages, FedSSU, as originally proposed, focuses on the algorithmic aspect of unlearning and does not address how to certify that unlearning has actually been performed in potentially adversarial client environments. In particular, there is no mechanism to prevent a client from using old model versions, manipulating unlearning routines, or bypassing unlearned models during inference. Addressing these limitations requires additional mechanisms that can attest to the integrity and correct execution of the unlearning process, especially in the context of cross-enterprise Data Spaces.

6.3 FedSSU Integration in HEX-FL

Integrating FedSSU into the HEX-FL framework requires both algorithmic and architectural adaptations. On the algorithmic side, the FedSSU unlearning routines and the associated local loss L_{local} must be reimplemented, validated and embedded into the existing training workflow. On the architectural side, the decentralized unlearning operations need to be exposed as services within the Data Space and orchestrated through the existing HEX-FL components, namely the *ModelTrainer*, *ModelPredictor* and *ModelCoordinator*, as illustrated in the component diagram in Figure 6.1.

The original FedSSU publication does not provide a reference implementation, which necessitated an independent realization of the framework. Based on the algorithmic description in the paper, the FedSSU logic was reconstructed, including the computation of the saliency information, the formulation of the unlearning loss and the update rules for the model parameters during the unlearning rounds (Leng et al., 2025). The reimplementation was validated by reproducing key experimental results reported in the paper, ensuring that the behavior of the unlearning process is consistent with the original design. To facilitate maintainability and reuse, the implementation was modularized, separating the core unlearning logic from the HEX-FL-specific interfaces that are depicted as individual components inside the *HEX-FL extension* in Figure 6.1.

From the perspective of HEX-FL, FedSSU-specific functionality is primarily integrated into the *ModelTrainer* module. In the component diagram, the ModelTrainer is represented as a separate component inside the HEX-FL extension and offers the *ModelTraining* interface to the *ModelCoordinator*. The ModelTrainer is extended with additional routines that implement FedSSU's unlearning rounds, including:

- identification and configuration of unlearning targets (e.g., which client or class to forget),
- adaptation of the local training loop to use L_{local} during unlearning and
- coordination with the global aggregation process to incorporate unlearning updates into the global model.

To maintain compatibility with HE and DP mechanisms, care is taken to ensure that unlearning-related updates can be processed within the same cryptographic and statistical privacy framework used for HEX-FL's default training process.

At the orchestration level, the *ModelCoordinator* is extended to manage unlearning workflows. As shown in Figure 6.1, the ModelCoordinator connects the *ModelTraining* and *PredictionService* interfaces of the *ModelTrainer* and *ModelPredictor* with the *ModelLoading* interface towards the Data Space. When an unlearning request is issued, for example due to a GDPR erasure request from a data subject or the revocation of a client, the coordinator initiates a FedSSU-based unlearning round. This involves distributing the current global model and unlearning configuration to participating clients, triggering the FedSSU routines in their local ModelTrainers and aggregating the resulting unlearning updates into a new global model. The coordinator also ensures that this unlearned model is disseminated to all clients via the *Receiver* port to the *DataPlane* component and that versioning information is updated accordingly.

To integrate FedSSU into the Data Space context, additional HTTP endpoints are introduced in the HEX-FL extensions. In the component diagram, these correspond to the service ports exposed by the HEX-FL extension and the separate *HashExtension*. These endpoints expose operations such as initiating an unlearning request, querying unlearning status and retrieving the latest unlearned model. All communication follows the protocols and security constraints of the EDC, ensuring that unlearning operations respect the same data sovereignty and contract-based access controls as standard training and prediction. This design allows FedSSU-based unlearning to be seamlessly incorporated into existing Data Space workflows and to be triggered as part of contractual or regulatory processes.

6.4 Certification of the Unlearning Model

While integrating FedSSU into HEX-FL enables the algorithmic removal of client contributions from the global model, it does not by itself guarantee that all clients actually execute the resulting unlearned model in practice. In adversarial or misconfigured

environments, clients could circumvent the intended unlearning behavior by retaining and reusing obsolete models or by modifying the prediction and training code. From a GDPR perspective, such behavior would undermine the right to be forgotten, as predictions might still be generated from models that encode information that should have been erased (Z. Liu et al., 2024).

To address this challenge, this chapter introduces a certification mechanism that allows verification that a given client is executing the correct, unlearned model in a trustworthy way. The central idea is to augment the HEX-FL architecture with a hash-based verification process that checks three aspects of the local prediction pipeline:

- the structure of the prediction program,
- the identity and integrity of the ML model object and
- the dynamic execution trace of a sample prediction.

These checks are coordinated by a server-side verification service that operates as a certified EDC extension and maintains canonical hash values for each certified model version. The proposed mechanism can be seen as an instantiation of verifiable FL concepts at the level of local model execution, complementing existing work on verifying aggregated models and server behavior.

6.4.1 Attack Scenarios

A first step in designing a certification mechanism is to explicate the threat model and the attack scenarios it aims to mitigate. In the context of federated unlearning in Data Spaces, clients are assumed to have control over their local execution environment and may behave maliciously or negligently. This includes the ability to modify code, store and load model files and interfere with the deployment process. The goal of the adversary is to preserve access to models that still encode information that should have been forgotten, or to bypass unlearning routines while appearing compliant to the federation.

One straightforward attack is the persistent storage of old model states. Before an unlearning operation is performed, a client may save the current global model to disk. After the unlearning round has completed and a new unlearned model has been deployed, the client could simply ignore the updated model and reload the old checkpoint for local inference. If there is no mechanism to detect such behavior, the client can continue to serve predictions based on a non-compliant model, effectively violating the unlearning requirement (Z. Liu et al., 2024). A variant of this attack is to maintain multiple model versions and dynamically choose which one to use depending on the input, thereby selectively reintroducing forgotten information.

A second category of attacks targets the prediction code itself. The client could modify the prediction service to remove or bypass the function that replaces the model with the newly unlearned version, or to add additional methods that load custom weights prior to prediction. Even if the unlearned model is present on disk, the modified code may

never use it, instead delegating predictions to an older or different model. Alternatively, the model class definition could be changed to inject hooks that overwrite parameters or alter control flow during inference, effectively undermining the semantics of unlearning while maintaining the same public API.

A third attack class operates at the level of the internal call hierarchy and dynamic behavior. Even if the program structure and model object superficially match the certified versions, the client could introduce additional layers of indirection or conditional logic that reroute predictions through different functions or models depending on runtime conditions. Such manipulations may be difficult to detect by static inspection alone but can have a significant impact on whether the executed model truly reflects the unlearned state. These scenarios motivate a certification mechanism that verifies not only static artifacts (files, signatures) but also the dynamic execution behavior of the prediction module.

6.4.2 Interaction Between Certification Components

The certification mechanism is realized by extending the existing HEX-FL components with an additional hash extension that implements hash generation and verification. In the component diagram shown in Figure 6.1, all certification-related interactions are modeled as a data flow between the HEX-FL extension (containing the ModelTrainer, ModelPredictor and ModelCoordinator), the hash extension and the EDC Data Plane.

On the client side, the certification process flow starts in the *ModelPredictor*. Before serving predictions, the predictor needs to load the model, for which it is mandatory to verify the correctness of the model. Therefore, it invokes the *HashGeneration* interface of the hash extension via the *HashRequest* port. This triggers the *HashGenerator* to compute cryptographic hashes over the prediction module, the local model and a sample execution trace. The resulting hash values are then exposed through the *HashService* port of the hash extension and forwarded to the ModelCoordinator inside the HEX-FL extension, which is responsible for preparing and sending verification messages.

The ModelCoordinator connects the certification data flow to the Data Space. As indicated by the ports *ModelLoading* and *Receiver* in Figure 6.1, the coordinator uses the EDC Data Plane to exchange model artifacts and hash-based verification messages with the server-side verification service. Outgoing messages contain the locally computed hashes, whereas incoming responses carry the verification decisions associated with the currently deployed model version. In this way, the Data Plane mediates certification-specific communication alongside regular model distribution, without bypassing the existing data sovereignty and contract mechanisms.

The hash extension thus acts as a local certification interface for model and code fingerprints, while the HEX-FL extension components route these fingerprints through the Data Plane to the trusted verification service and consume its responses. Therefore, the HEX-FL extension is extended by the unlearning logic to be able to verify model ver-

sions and start unlearning procedures. If the verification succeeds, the ModelCoordinator allows the ModelPredictor to continue serving predictions. If verification fails, the central ML service aggregator (not shown in Figure 6.1) can block prediction requests and exclude the client from further participation. By structuring the certification logic as an explicit data flow between these components, the architecture cleanly separates hash generation, orchestration and cross-organizational communication while ensuring that all certification-related interactions remain under Data Space governance. The verification process, which allows to certify client compliance after unlearning rounds is detailed in the following.

6.4.3 Hash-Based Verification Process

The hash-based verification process is structured as a three-level certification protocol, corresponding to program structure verification, ML model verification and trace validation. The corresponding message flow is depicted in the sequence diagram (see Figure 6.2). The overarching goal is to ensure that the locally running prediction service corresponds to a certified configuration known to the server-side verification service.

At a high level, the verification process proceeds as follows. Before serving predictions, the client-side hash extension first computes the program structure hash `H1`, describing the structural properties of the prediction module. This hash is sent to the server-side verification service, which compares it to the canonical `H1` value of the certified prediction module. Depending on the result, the server responds with either an acceptance or a denial. If `H1` is accepted, the client proceeds to load the model in a temporary state to be further validated and computes the model hash `H2` and the trace hash `H3`. These hashes are again sent to the server-side verification service, which compares them to the canonical `H2` and `H3` values and returns a final decision on whether the model and its execution behavior match the certified configuration.

The server-side verification service acts as a trusted third party within the Data Space. It is assumed to be non-malicious and to operate on a certified code base and configuration that cannot be manipulated by regular participants. This assumption is analogous to the trust placed in central aggregation servers or verifiable FL components in related work, where verification logic is anchored in a trusted environment. For each global model version, including those resulting from unlearning operations, the verification service maintains a record of the canonical hash values `H2` and `H3`. These values are derived from a reference deployment that has been audited and approved. The verification protocol ensures that only clients whose local hashes match these values are authorized to serve predictions using the corresponding model.

Program Structure Verification

Program structure verification constitutes the first line of verification against code manipulation. In this phase, the HashGenerator computes a hash `H1` over the structural ele-

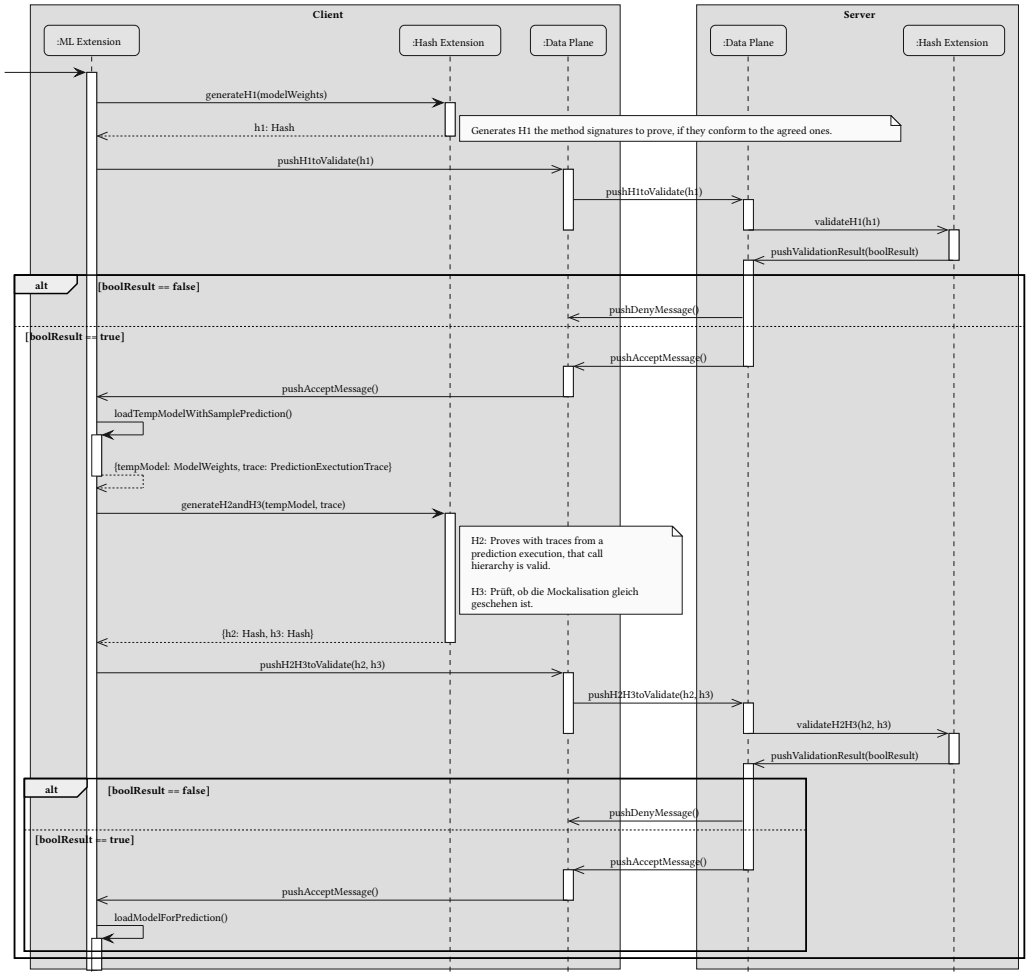


Figure 6.2: Sequence diagram of a client verification process prior to ML model usage.

ments of the ML prediction module, including the signatures of all public functions, the existence of mandatory methods prescribed by the HEX-FL framework and potentially other structural metadata (such as module names or class hierarchies). The aim is to ensure that the client has not added or removed functions or altered function signatures in a way that could bypass or subvert the certified prediction pipeline.

Operationally, the client-side hash extension extracts a representation of the prediction module's structure, by introspecting Python modules and classes and computes `H1` using a cryptographic hash function. This hash is then transmitted to the server-side verification service via the Data Plane. The verification service compares the received `H1` value with the canonical value stored for the certified prediction module. If the values differ, the verification service sends a denial response and the client aborts the prediction process without loading any model or proceeding to further verification steps. If the values match,

the verification service sends an acceptance response and the client proceeds to the next phase.

In the sequence diagram (see Figure 6.2), this interaction occurs before the first `alt` block, representing an early decision point. By enforcing program structure verification at this stage, the system prevents attacks that attempt to modify the API or to introduce additional entry points into the prediction service. For example, if an attacker adds a function that loads an older or wrong model and routes incoming prediction requests to it, the resulting change in function signatures will modify `H1` and cause verification to fail.

ML Model Verification

Once the program structure has been verified, the next step is to ensure that the ML model loaded by the prediction service matches the certified unlearned model (latest model version from the aggregation server). In this phase, the HashGenerator computes a hash `H2` of the model object, which is implemented as a PyTorch module in the HEX-FL framework. The model object includes the ML architecture definition (layers, connectivity), parameter tensors (weights and biases) and any additional methods that could influence the forward pass.

To compute `H2`, the hash extension serializes the corresponding information of the model object and applies a cryptographic hash function. Relevant for the hash generation is to ensure deterministic serialization so that models that are semantically identical produce the same `H2` value. The resulting hash is then transmitted to the server-side verification service, which compares it against the canonical `H2` value for the current certified model version. A mismatch indicates that the client is not using the correct model. One reason could be that it has loaded an old, non-unlearned model, or because the model has been modified locally by reinitializing certain layers or injecting backdoors.

If verification of `H2` fails, the server responds with a denial decision and the client aborts the prediction process, discarding the loaded model and marking its local environment as non-compliant. If the verification of `H2` is successful, the client goes to the final verification step. This two-stage approach ensures that both the static structure of the prediction code and the specific model parameters are consistent with the certified unlearned configuration, addressing attacks that manipulate either the code or the model independently.

Trace Validation

The third verification step focuses on the dynamic behavior of the prediction pipeline. Even if the program structure and model object match the certified versions, a malicious client could still manipulate the internal control flow, for example by inserting additional function calls, conditionally swapping models at runtime, or altering the way inputs are processed and forwarded through the network. To detect such manipulations, the cer-

tification mechanism performs trace validation based on a hash H3 computed from the execution trace of a sample prediction.

In this phase, the client performs a controlled prediction using synthetic or random input data that do not reveal any private information. During this prediction, the hash extension instruments the prediction module to log the sequence of function calls, method invocations and potentially key internal events (such as layer activations or branching decisions). The resulting trace is then compressed into a canonical representation and hashed to obtain H3. Importantly, the same instrumentation and input configuration is used on the server-side reference deployment to obtain the canonical H3 value for the certified model version.

The client sends H3 to the verification service, which compares it to the canonical H3. If the values differ, this indicates that the dynamic execution behavior of the client's prediction pipeline deviates from the certified behavior, even if H1 and H2 matched. This could be the case, for instance, if the client inserted additional calls that overwrite model parameters just before the forward pass, or if it routes the prediction through an alternative model depending on runtime conditions. In such cases, the verification service denies the request and the client must not serve predictions. If H3 matches, the verification service concludes that the program structure, model object and dynamic behavior are all consistent with the certified configuration.

In the sequence diagram, the validation of H2 and H3 is represented as a combined step, in which a concatenation or joint representation of H1, H2 and H3 is validated on the server. A successful verification yields a positive decision that the client can use to unlock prediction functionality for the current model version. This three-level hashing scheme is inspired by broader work on verifiable FL, which emphasizes the need to verify not only aggregated parameters, but also the integrity and behavior of local computations.

Transfer to HEX-FL Process

The outcome of the verification process is directly integrated into the HEX-FL process and influences both the training and inference. If the server-side verification service returns an invalid hash response, the client is treated as non-compliant. The ModelCoordinator removes the client from the list of active participants, preventing it from participating in future FL or unlearning rounds. In addition, the client is instructed to delete all copies of the model from memory and persistent storage, reducing the risk that it continues to use non-compliant models outside the CeMUFl.

A non-compliant client is also prevented from performing further inference under the FL contract. Since prediction requests in the Data Space are mediated through EDC connectors and HEX-FL extensions, the absence of a valid verification token or positive verification state causes the prediction endpoints to reject incoming requests. This ensures that only clients that have successfully passed verification can provide predictions as part of

the Data Space, reinforcing a system-wide guaranty that only certified unlearned models are used in production.

Conversely, if the verification process completes successfully, the ModelPredictor is authorized to load the verified model in the final inference state and to expose its prediction API to external consumers. The model is then handed over to the standard HEX-FL prediction pipeline and inference proceeds as usual. From this point on, the local prediction service operates under a verified configuration until a new model version is deployed. When a new global model is rolled out, for example, as the result of a FedSSU unlearning round followed by federated averaging, the ModelCoordinator updates the canonical hashes in the verification service and triggers a new verification cycle for each client. Only clients that successfully verify against the new hashes are allowed to continue serving predictions. This updates on the server-side verification process is not shown in Figure 6.2.

Through this integration, the certification mechanism complements the algorithmic unlearning capabilities of FedSSU and the privacy-preserving properties of HEX-FL and DP-based extensions. It ensures that the unlearned models are not only computed, but also reliably deployed and enforced in a heterogeneous and potentially adversarial federation, thereby moving the overall architecture closer to operational compliance with the right to be forgotten in federated Data Spaces (Z. Liu et al., 2024).

6.5 Evaluation of Unlearning Capabilities

To validate whether the FedSSU approach was implemented correctly and to assess the effectiveness and completeness of the unlearning process, the CeMUFl is evaluated on several metrics, all based on the CIFAR-10 dataset (see Section 3.3). The evaluation follows the same experimental setup and hardware configuration as the HEX-FL experiments. All measurements are conducted on a MacBook Pro 2021 with an M1 Max SoC and 32 GB RAM. As a baseline, a complete retraining is performed on the *unlearn dataset*. The unlearn dataset is constructed by randomly removing $n = 200$ data points from the original CIFAR-10 training set, representing the data to be forgotten.

Using this unlearn dataset and the baseline, the proposed CeMUFl framework is evaluated in multiple dimensions. First, the time-related performance of the FedSSU-based unlearning process is compared against the full retraining baseline. Second, the quality of unlearning is analyzed using the Anamnesis Index (AIN), which captures how quickly a model can reacquire forgotten information (Chundawat et al., 2023). Third, the influence of the FedSSU correction term λ on the accuracy of the model is investigated. Finally, the Jensen–Shannon distance (Jensen-Shannon Divergence (JSD)) between the original and unlearned models is studied along variations of λ, providing insight into how far the unlearned model drifts away from the original parameter configuration (Corander et al., 2021; Menéndez et al., 1997).

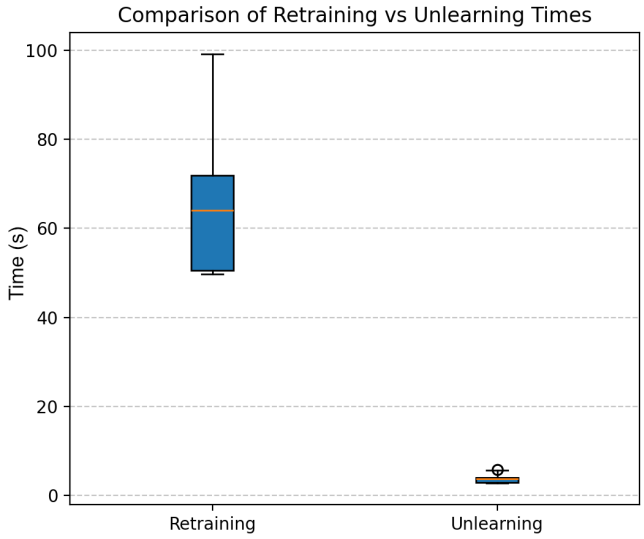


Figure 6.3: Runtime comparison between full retraining on the unlearn dataset (baseline) and the FedSSU-based unlearning procedure.

6.5.1 Time-Related Performance Comparison

The first set of experiments focuses on the runtime behavior of the unlearning process compared to the baseline of full retraining. The scenario is as follows. Both the baseline and the proposed approach use 12 training epochs with a learning rate of 0.001. Two approaches are compared:

1. **Retraining model (baseline)**: A newly initialized model is trained from scratch on the unlearn dataset.
2. **Unlearning (proposed)**: A model initially trained on the full CIFAR-10 dataset is updated using the FedSSU unlearning procedure with a fixed $\lambda = 0.5$ to remove the influence of the 200 forgotten samples.

Figure 6.3 shows the distribution of runtimes for both approaches over 10 independent runs in the form of box plots. The baseline (*Retraining model*) consistently exhibits substantially higher runtimes, with all observed values above 50 seconds for the 12 training epochs. In contrast, the FedSSU-based unlearning approach (*Unlearning*) is completed in approximately 10 seconds in all runs. Thus, the proposed unlearning procedure is at least five times faster than full retraining on this setup, demonstrating a clear efficiency advantage.

It is important to note that CIFAR-10 is a relatively small benchmark dataset. In realistic industrial scenarios with larger models and datasets, the absolute cost of complete retraining would increase significantly, further amplifying the practical benefits of the

lighter unlearning procedure. The results in Figure 6.3 therefore primarily demonstrate that the proposed approach is runtime-efficient, while the subsequent evaluations focus on the quality and completeness of the unlearning process.

6.5.2 Anamnesis Index

Although the previous evaluation quantifies runtime, it does not yet provide insight into how effectively information associated with the forgotten data is removed. To evaluate the quality of unlearning, the Anamnesis Index (AIN) proposed by Chundawat et al., 2023 is used. The AIN measures how quickly a model can reacquire forgotten information when retrained on the full dataset and is thus an indicator of how much residual knowledge about the forgotten data remains after unlearning.

The evaluation setup is as follows. First, a standard model is trained on the full CIFAR-10 dataset for 12 epochs and the resulting test accuracy is recorded; this accuracy defines a reference performance level. Second, two models are constructed for comparison:

- **Retrained model (baseline)**: A freshly initialized model is trained on the unlearn dataset.
- **Unlearned model**: A model is first trained on the full CIFAR-10 dataset, then subjected to the FedSSU-based unlearning procedure with the unlearn dataset and finally evaluated.

The AIN is then computed by retraining both retrained and unlearned models on the *full* CIFAR-10 dataset and recording how many epochs are required to reach a target accuracy. In this evaluation, the target accuracy is defined as the standard model's accuracy minus a threshold of 10%, which corresponds to the horizontal reference level of approximately 37% shown in Figure 6.4.

Figure 6.4 plots the test accuracy of the retrained and unlearned models during this retraining phase over 12 epochs. The curves reveal that the unlearned model regains accuracy substantially faster than the retrained baseline. It reaches the target accuracy already in epoch 4, whereas the baseline requires the full 12 epochs to achieve comparable performance. This indicates that the unlearned model retains some latent structure or representations that facilitate faster re-learning of the forgotten information.

One potential explanation is that the unlearned model, which has been trained on the full dataset before unlearning, has encountered a more diverse set of examples and thus only undergoes relatively localized weight updates during the unlearning phase. As a result, it can adapt back more quickly to the pre-unlearning performance when retrained on the full dataset. From an unlearning perspective, this suggests that while the specific decision boundary associated with the forgotten samples is modified, broader features and representations that indirectly encode information about them may still be present, enabling rapid recovery of performance when the full data is reintroduced.

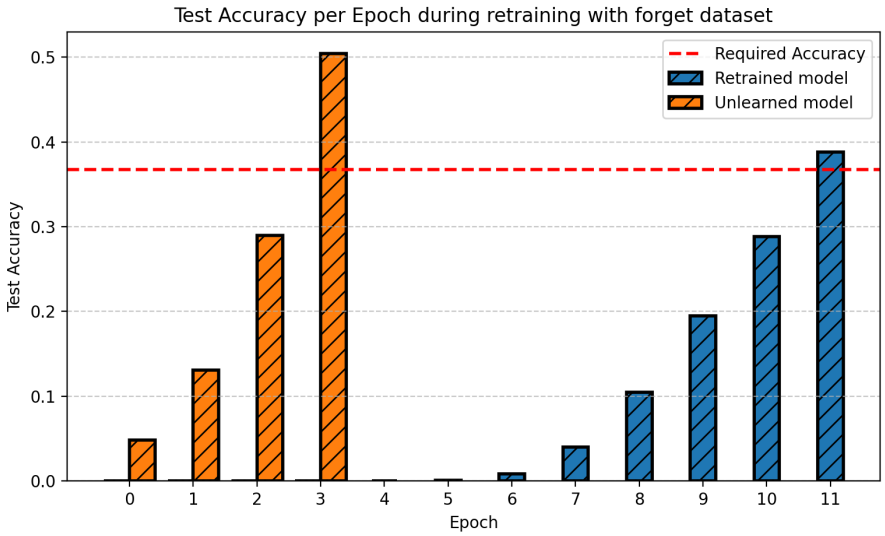


Figure 6.4: Evaluation of the Anamnesis Index (AIN): test accuracy of the retrained model (baseline) and the unlearned model when retrained on the full CIFAR-10 dataset after the initial unlearning phase.

6.5.3 Unlearning Performance corresponding to λ

Beyond runtime and AIN, it is informative to analyze how the FedSSU correction term λ influences the overall predictive performance of the unlearned model. In FedSSU, λ controls the trade-off between adhering to the original model and enforcing the unlearning constraints. Intuitively, smaller values of λ allow the model to deviate more strongly from its original parameters (potentially achieving stronger forgetting), while larger values encourage staying closer to the initial model (potentially preserving more utility).

Figure 6.5 shows the test accuracy of the unlearned model on the full CIFAR-10 dataset for different values of λ between 0 and 1 in steps of 0.1. For each value of λ, box plots over 10 runs are shown, capturing the variability due to random initialization and sampling. The results indicate a clear monotonic trend: the arithmetic mean accuracy increases from approximately 37.4% at $\lambda = 0$ to around 44.2% at $\lambda = 1$.

This behavior confirms that λ serves as a control knob to balance the unlearning strength and the utility of the model. Low λ values enable more aggressive deviations from the original model, which can enhance forgetting but also reduce the accuracy of the retained data. High λ values keep the model closer to its original state, achieving higher accuracy but weaker unlearning. The role of λ is therefore analogous to the privacy parameter ε in DP, where larger values of ε correspond to weaker privacy guarantees and greater utility.

For the CIFAR-10 experiments, Figure 6.5 suggests that a moderate λ in the range of 0.2 to 0.3 yields a reasonable trade-off. The loss in accuracy relative to the original model is

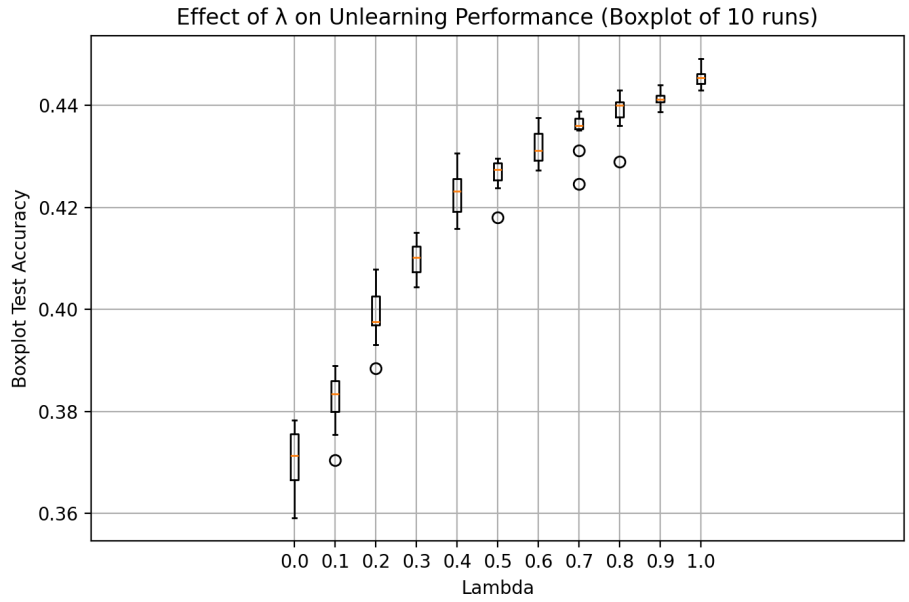


Figure 6.5: Influence of the FedSSU parameter λ on the test accuracy of the unlearned model on the full CIFAR-10 dataset.

approximately 3–4%, which appears acceptable given that only a small number of data points (200 samples) have been removed. At the same time, this range of λ still allows substantial deviation from the original model, which is desirable to ensure meaningful forgetting. Importantly, λ can be configured individually for each round of unlearning and client, allowing stakeholders to adapt the strength of unlearning to their specific requirements.

6.5.4 Jensen–Shannon Distance per λ

As a final metric, the Jensen–Shannon distance (JSD) is used to quantify how far the unlearned model deviates from the original model in parameter space. The Jensen–Shannon divergence is a symmetrized and bounded variant of the Kullback–Leibler divergence that measures the dissimilarity between two probability distributions (Corander et al., 2021; Menéndez et al., 1997). The Jensen–Shannon distance is defined as the square root of the Jensen–Shannon divergence and yields a true metric with values between 0 (identical distributions) and 1 (maximally different) when logarithms with base 2 are used (Corander et al., 2021; Menéndez et al., 1997). In this evaluation, the JSD is computed over probability distributions derived from the model weights, providing a scalar measure of how strongly the parameter configuration changes due to unlearning.

Concretely, for each experimental run, the pairwise distance between the original model (trained on the full CIFAR-10 dataset) and the corresponding unlearned model is com-

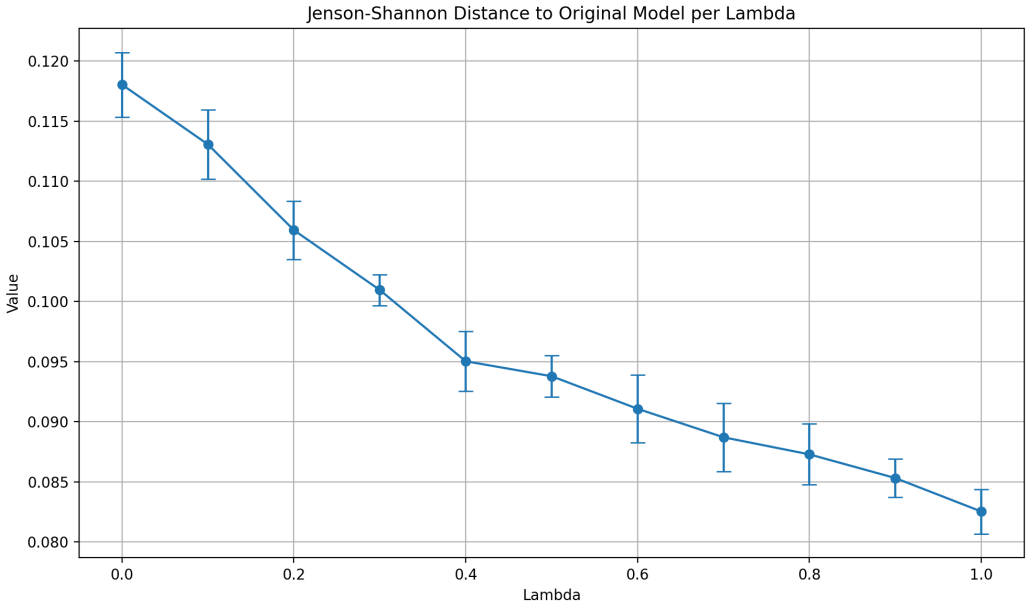


Figure 6.6: Jensen–Shannon distance between the original model and the unlearned model as a function of λ.

puted and summed over all parameters. Figure 6.6 summarizes the results per λ configuration. For each $\lambda \in \{0.0, 0.1, \ldots, 1.0\}$, 10 runs are conducted. The figure reports the minimum and maximum distances as horizontal bars and the median distance as a dot, which is connected between the values of λ to indicate the overall trend.

The results show that the median JSD decreases from approximately 0.117 at $\lambda = 0$ to around 0.082 at $\lambda = 1$. This pattern is consistent with the interpretation of λ as controlling proximity to the original model. For high λ, the unlearned model is constrained to stay closer to the initial parameter configuration, resulting in smaller distances and weaker forgetting. In contrast, for small λ, the unlearning process has more freedom to adjust weights, producing larger distances and stronger deviations from the original model.

Together with the accuracy analysis in Section 6.5.3, the JSD results suggest that a moderate value of λ of around 0.2 to 0.3 provides a good compromise. In this regime, the distance from the original model is large enough to enable meaningful unlearning, while the accuracy of the retained data remains high. Larger λ values reduce the distance and improve accuracy, but risk to weak unlearning, whereas very small λ values may induce excessive changes that harm overall performance. In practice, the choice of λ can therefore be guided by both utility requirements and the desired unlearning strength.

6.6 Summary

This chapter has extended the privacy-preserving FL architecture introduced in the previous chapters with explicit federated unlearning capabilities and a certification mechanism for enforcing GDPR-related erasure requirements in industrial Data Spaces. The central idea is to combine the FedSSU unlearning framework with the HEX-FL infrastructure and to augment this integration by a hash-based verification process that attests correct model usage on each client. The following summarizes the main contributions and findings along the three core aspects of the chapter including FedSSU integration, certification and the empirical evaluation.

From an algorithmic perspective, FedSSU is adopted as the unlearning backbone to remove the influence of client contributions from the global model without resorting to full retraining (Leng et al., 2025). Its local loss function L_{local} balances preservation of accuracy on retained data, suppression of forgotten data and stability with respect to the original model, providing a practical trade-off between unlearning effectiveness and computational efficiency. Within HEX-FL, the FedSSU routines are implemented and embedded into the *ModelTrainer*, while the *ModelCoordinator* is extended to orchestrate unlearning rounds, manage model versioning and distribute updated unlearned models across the Data Space via the EDC Data Plane. This design allows unlearning to be requested and executed as a first-class operation in the federated pipeline, thus addressing the algorithmic gap identified in Section 6.1.

In addition, this chapter introduces a certification mechanism that ensures that clients actually deploy and execute the unlearned models produced by FedSSU. The proposed solution consists of a hash extension and a server-side verification service that jointly implement a three-level verification protocol: program structure verification (`H1`), ML model verification (`H2`) and trace validation (`H3`). As detailed in Section 6.4, these components are integrated into the existing HEX-FL architecture (see Figure 6.1) and interact via the EDC Data Plane to exchange hash values and verification decisions. Only clients whose local hashes match the canonical values maintained by the trusted verification service are allowed to serve predictions, while non-compliant clients are excluded from future training and inference rounds. In this way, the certification layer strengthens the operational realization of the GDPR *right to be forgotten* by ensuring that unlearning is not only computed but also enforced in potentially adversarial environments.

The empirical evaluation in Section 6.5, conducted in CIFAR-10 with a controlled unlearn dataset of 200 samples removed, demonstrates both the efficiency and limitations of the proposed approach. First, time-related experiments (Section 6.5.1) show that FedSSU-based unlearning is at least five times faster than full retraining in the unlearn dataset, highlighting its practical suitability for repeated unlearning requests. Second, the AIN experiments (Section 6.5.2) reveal that unlearned models can regain forgotten information significantly faster than freshly retrained baselines, indicating that high-level representations remain that facilitate recovery. Third, the analysis of accuracy and JSD of the FedSSU parameter λ (see Sections 6.5.3 and 6.5.4) show that λ acts as a tunable parame-

ter controlling the trade-off between utility and forgetting, with moderate values around 0.2–0.3 yielding a reasonable compromise between performance degradation and deviation from the original model. Overall, these results confirm that the integrated FedSSU and certification framework advances the GDPR alignment of Data-Space-based FL, while also making explicit the residual challenges of approximate unlearning and the need for careful parameter tuning.

6.7 Discussion

The integration of FedSSU and the certification mechanism into the Data Space-oriented FL stack changes the role of unlearning from a theoretical add-on to a practically usable capability that can be embedded into operational workflows. Rather than treating erasure requests as exceptional events that require ad-hoc engineering effort, CeMUFl turns them into first-class operations that can be triggered, parameterized and audited in a standardized way. In this sense, this chapter demonstrates that unlearning can be aligned with the architectural principles of industrial Data Spaces (decentralization, contract-based interaction and connector-based deployment) without sacrificing performance or manageability.

A central benefit of the proposed approach is that it makes unlearning *operationally feasible* in settings where full retraining would be unrealistic. The use of FedSSU at the level of the HEX-FL *ModelTrainer* and *ModelCoordinator* ensures that unlearning can be executed within the same control plane that already manages regular training rounds, contracts and model deployments. This tight integration reduces the operational friction of handling repeated erasure requests and supports scenarios in which unlearning is part of continuous lifecycle management, rather than a rare, disruptive exception.

Beyond feasibility, CeMUFl improves the *regulatory expressiveness* of the FL pipeline. Previous chapters already established confidentiality guarantees (through HE and DP) and data sovereignty (through FedDScon), but they lacked explicit mechanisms to selectively erase the influence of individual clients. By adding targeted unlearning, the architecture can now distinguish between contracts that merely govern access to training and inference and contracts that also include obligations to remove past contributions upon request. The possibility of propagating unlearned models to all clients through the Data Plane turns such obligations into system-level behavior, thus strengthening the alignment with GDPR's *right to be forgotten.*

The configurability introduced by the FedSSU parameter λ also has implications that go beyond the quantitative trade-offs analyzed in the evaluation. In practice, λ can be interpreted as a policy variable that allows stakeholders to encode different unlearning strategies into the system: conservative settings that favor utility, aggressive settings that prioritize unlearning strength, or intermediate settings negotiated as part of contractual terms. Because λ can be chosen per unlearning round, it becomes possible to tailor the

degree of forgetting to the sensitivity of the underlying data or to specific regulatory or business requirements, rather than enforcing a single, rigid global policy.

The certification mechanism further shifts the trust assumptions in federated unlearning. Instead of implicitly assuming that clients faithfully deploy unlearned models, CeMUFl provides a concrete means to *test* this assumption for every client and every model version. The three-layer hashing protocol ensures that compliance is not merely a matter of configuration but is backed by verifiable evidence on code structure, model parameters and execution traces. This is especially relevant in heterogeneous industrial federations, where participants may run different stacks, operate under different internal governance processes, or have varying incentives to comply.

Finally, the coupling of certification with model usage establishes an enforcement channel that complements contractual and organizational measures. By requiring successful verification as a precondition for serving predictions, CeMUFl effectively ties economic value (the ability to participate in the FL ecosystem) to technical compliance with unlearning and model update requirements. This creates a feedback loop in which non-compliant clients are automatically isolated at the technical level, without requiring manual intervention in every case. In combination, these aspects mean that the contributions of this chapter do not only include **CeMUFl**, but also reshape how unlearning is governed, negotiated and enforced within Data Spaces. Consequently, the proposed design directly operationalizes the additional GDPR dimensions highlighted in **Research Question 3** and shows that they can be addressed within a coherent and implementable FL architecture.

7 Integrated Framework for GDPR-Conform Federated Learning

The preceding chapter introduced CeMUFl as an approach for certified machine unlearning *in federated infrastructures, motivated by the observation that GDPR compliance in cross-enterprise FL cannot be reduced to training-time protection alone, but also requires enforceable lifecycle interventions (e.g., verified removal of learned influence after a deletion request). In this context, certified unlearning is positioned as a technical response to the* Right to Erasure *(GDPR Article 17), which requires erasure of personal data under specified conditions and thus motivates mechanisms that go beyond local storage deletion in ML-driven processing pipelines.*

Building on this foundation, the present chapter addresses ***Research Question 4****, which asks how a holistic framework could be defined, that combines GDPR regulations and FL methodologies by integrating CeMUFl with the infrastructure and privacy-preserving training mechanisms from the earlier chapters into* ***Feder****, a holistic and modular framework for GDPR-conform FL across enterprises in sovereign* Data Spaces*. After motivating the lack of standardized and interoperable end-to-end FL architectures for cross-organizational settings in Section 7.1, the chapter introduces the overall* ***Feder*** *architecture concept in Section 7.2 and its modular extension points, distinguishing client-side and server-side responsibilities. Subsequently, the contributions FedDScon, HEX-FL and CeMUFl are positioned as the three integrated building blocks and are consolidated into the finalized* ***Feder*** *framework in Section 7.6. Finally, a real-world-inspired logistics scenario derived from GAIA-X 4 ROMS is used to evaluate practicability, grounding the architecture in a representative mobility and logistics context that targets cooperative, data-sovereign ecosystems based on Gaia-X principles.*

7.1 Introduction

Cross-organizational collaboration is increasingly based on heterogeneous multimodal process chains that integrate data and ML models from different domains. Although modern ML approaches have become more data-driven, their adoption in enterprises lacks standardization with respect to data protocols, privacy preservation and interoperability. Consequently, many approaches focus primarily on developing ad-hoc interfaces to connect multiple proprietary systems or models, which hinders scalability and reproducibility (Yang et al., 2019).

In the FL domain, a major limitation remains the absence of a standardized architecture for data exchange and privacy preservation. Most current FL solutions, such as those offered by *TensorFlow Federated* or *NVIDIA Clara*, use proprietary implementations that do not integrate seamlessly with open Data Spaces architectures. Although *Data Spaces* (as conceptualized in the *European Data Space* initiatives and by IDSA offer a standardized foundation for data sovereignty and secure sharing, they focus primarily on data-level interoperability rather than end-to-end integration of ML pipelines (Otto and Jarke, 2019).

In addition, both the deployment of Data Spaces and the integration of high-assurance privacy mechanisms within the FL workflows remain complex and resource-intensive processes. Existing methods for PPML or *privacy-preserving FL*, such as differentially private optimization or secure aggregation, provide valuable building blocks but do not constitute modular GDPR-aligned systems for cross-enterprise environments.

This chapter addresses **Research Question 4** by introducing a standardized architecture approach conceptualized with transparency, sovereignty and extensibility as its main design principles. The central contribution is the *Feder* framework (*Privacy-preserved Federated Learning Across Enterprises*), which integrates contributions from previous chapters like FedDScon, HEX-FL and CeMUFl into a holistic, modular and GDPR-conform solution architecture for *privacy-preserving FL* in Data Spaces.

7.2 Architecture Concept

Figure 7.1 illustrates the overall system architecture of *Feder*, depicting the interaction among its core components. The previous chapters introduced individual components addressing specific aspects of GDPR-compliant *Federated Learning* in cross-enterprise environments. *Feder* synthesizes these contributions into an end-to-end framework through modular and extensible interfaces that ensure seamless interoperability.

At its foundation lies FedDScon, which establishes the *Federated Data Space* infrastructure using the EDC. This component provides REST-based interfaces and extension mechanisms that enable embedding of custom *privacy-preserving* functionalities as EDC connector libraries, as indicated by the `<<extends>>` association in the figure.

The design distinguishes between server-side (*FederHub*) and client-side (*FederClient*) extensions, reflecting the canonical FL architecture comprising *clients* and an *aggregation service*. *FederHub* extensions encompass model orchestration, lifecycle management, versioning in the *Model Management Module* and the HEX-FL secure aggregation mechanism in the *FL Aggregation Module*. The *FederClient* extensions integrate privacy-preserving modules for HE (*HE FL Client Module*), DP (*DP Preprocessing Module*) and Certified Machine Unlearning (CeMUFl) procedures in the *Unlearning Module*, ensuring preservation of local privacy prior to any aggregation. As communication uses the *Federated Data Space* as base layer, the *FederHub* and *FederClients* communicate over a customized protocol for secure FL tasks based on HEX-FL, namely *FL DS Protocol*. Additionally, *UL-Cert*

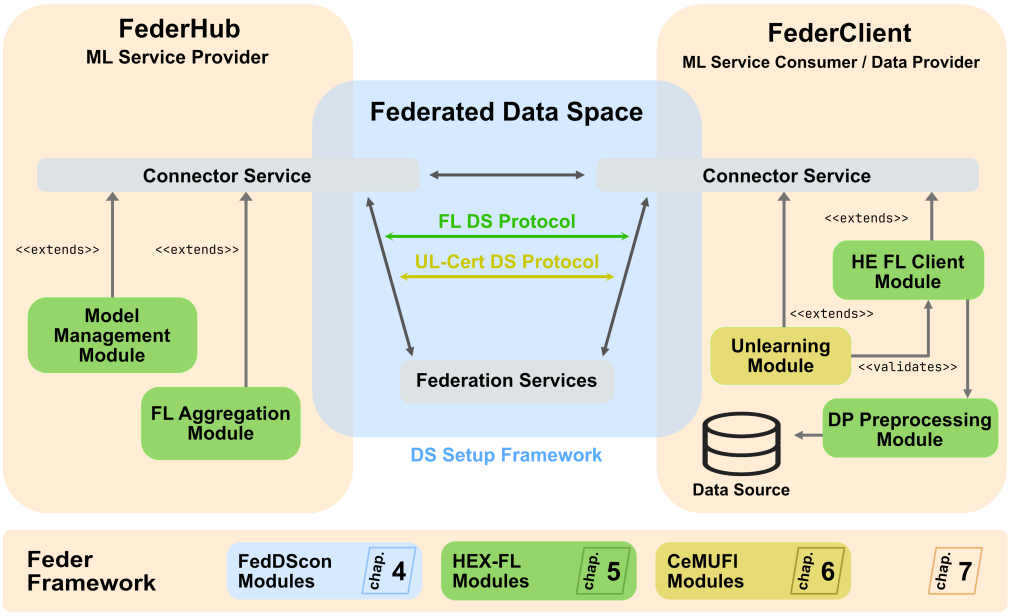


Figure 7.1: Finalized *GDPR*-conform framework architecture (Feder).

DS Protocol is defined separately, which is used for the necessary communication for unlearning tasks based on CeMUFl.

The module-based encapsulation of all *privacy-preserving* mechanisms and *unlearning modules* facilitates the rapid replacement or enhancement of components in response to evolving *privacy-preserving* frameworks or regulatory requirements. Adherence to *Feder* standard interfaces guaranties interchangeability and long-term maintainability of the system.

7.3 FedDScon

The component FedDScon constitutes the infrastructure layer of *Feder*, which orchestrates the setup, provisioning and management of *Federated Data Spaces* between participating entities. Using the EDC framework, FedDScon automates the deployment of configured connectors with embedded extensions related to FL, the definition of *data assets* and the establishment of contractual relationships between *FederHub* and *FederClients*. This automation significantly reduces configuration complexity through policy-enforced contract negotiation, forming the essential trust and governance foundation required by GDPR and *Data Space* standards.

7.4 HEX-FL

HEX-FL forms the *privacy-preserving FL* layer within *Feder*, operating on top of FedDScon contracts to enable modular and compliant FL workflows. It implements a rigorous two-layer privacy paradigm: *data-level* protection through configurable ϵ*-DP* in the *DP Preprocessing Module* and *model-level* security through *Secure Federated Averaging* provided by the *HE FL Client Module*, ensuring that intermediate updates remain confidential.

HEX-FL further defines the standardized *FL DS Protocol*, structuring the exchange of *model weights*, *aggregation signals* and certificates while preserving the *Data Space* principles of sovereignty and traceability.

7.5 Certified Machine Unlearning in Federated Infrastructures (CeMUFl)

CeMUFl extends the HEX-FL interfaces to incorporate the *certified machine unlearning* capabilities compliant with GDPR *Article 17* ("*Right to be Forgotten*"). *Client-side unlearning procedures* remove specific data contributions locally, followed by cross-enterprise *model re-synchronization* and verification via an embedded *certification protocol*. Pre-inference *model validity checks* ensure certified state compliance, maintaining integrity and accountability in distributed environments.

7.6 Feder: Integrated Framework

Feder represents the comprehensive synthesis of FedDScon, HEX-FL and CeMUFl into a unified framework that holistically addresses GDPR-conformity challenges in FL tasks across enterprises.

The modular extension architecture integrates all components through open and well-defined standardized interfaces that connect internal mechanisms with the integrated *Data Space* infrastructure. This design inherently promotes extensibility, allowing researchers and practitioners to develop novel *privacy-preserving* or *unlearning procedures* that only need to conform to the interface specifications of *Feder*. Simultaneously, FedDScon's automation ensures *ease-of-use*, allowing non-experts to establish FL-capable *Data Spaces* among multiple parties with minimal configuration overhead.

The integrated framework delivers end-to-end GDPR compliance by combining FedDScon's *data sovereignty* mechanisms, HEX-FL's formal two-layer privacy guarantees (ϵ-DP in *data level*, *secure aggregation* in *model level*) and CeMUFl's *certified unlearning propagation*. *Model lifecycle management*, including updates, *versioning* and *unlearning*, is centrally orchestrated on the *server side* with full auditability, while zero-trust collaboration preserves organizational sovereignty without data leaving boundaries.

Key technical characteristics include:

- **Production-ready automation**: Single-command Data Space provisioning with automatic contract negotiation
- **Interoperability**: REST-based EDC integration compatible with IDSA/*GAIA-X* ecosystems
- **Scalability**: Vendor-agnostic support for $n \geq 2$ participants
- **Future-proof extensibility**: Library-based components with standardized interfaces

By eliminating proprietary lock-in and providing compliance auditability through traceable privacy parameters and procedures, *Feder* constitutes the first production-ready blueprint for GDPR-conform *Federated Learning* in sovereign *Data Spaces*. This architecture enables scalable cross-enterprise AI collaboration while maintaining regulatory compliance, technical sovereignty and innovation potential.

7.7 Feder Application

To demonstrate the usability of the *Feder* framework, this section evaluates its applicability along a real-world scenario derived from the GAIA-X 4 ROMS project, which investigates data-sovereign services for logistics and mobility in a federated Data Space. In this evaluation, an FL use case is considered, in which multiple logistics companies collaboratively forecast parcel delivery volumes without exchanging raw data, aligning with data sovereignty principles as promoted in GAIA-X Data Spaces. First, the use case is described, then the participating companies are introduced and finally the different phases of the evaluation are detailed, including the setup of the Data Space and its use for PPML training and prediction.

7.7.1 Use Case Description

In line with the real-world GAIA-X 4 ROMS scenario, the use case is situated in the transportation and logistics domain, where reliable forecasting of arrival times and capacities is a key driver for planning and fleet management in parcel delivery networks. The GAIA-X 4 ROMS project targets a federated data ecosystem that enables logistics actors to share data and services in a sovereign manner, for example, to improve planning for first- and last-mile parcel delivery and intermodal transport chains.

In the use case considered, multiple logistics companies collaborate to forecast future parcel delivery volumes based on historical contracts, orders and operational data, which is in line with ongoing research that uses FL to improve demand forecasting in supply chains while preserving data privacy. Concretely, the companies Krone, DHL and Rhenus act as logistics participants that each store their operational and contractual data locally in their respective corporate networks and have direct access only to their own data.

These companies aim to build a shared global prediction model, motivated by the hypothesis that a larger and more heterogeneous knowledge base leads to improved forecasting performance, as observed in collaborative FL scenarios in logistics and mobility. However, the underlying order, customer and operational data contain commercially sensitive and personal information, such that direct data sharing is restricted by GDPR and sectoral data sovereignty requirements, which is why they opt for a secure FL architecture within a trusted, sovereign GAIA-X-compliant Data Space.

7.7.2 Use Case Participants

As outlined in the use case description, the evaluation scenario involves three logistics companies as primary participants, namely Krone, DHL and Rhenus, which act as Data Providers and FL participants. Each of these company participants maintains sensitive but ML-relevant data, provides local storage and compute infrastructure for model training, as well as employs staff responsible for developing GDPR-compliant models in collaboration with the other companies. In the following, these companies are called **FederClients** representing the client side of the *Feder* framework.

Beyond the company participants, an additional organization is assumed that operates the federated Data Space and configures data exchange and governance services in line with GAIA-X principles. This participant, denoted as **FederHub**, provides the infrastructure for Data Space onboarding and management, as well as the federated coordinator responsible for global model orchestration and lifecycle management, analogous to central aggregators described in FL reference architectures for mobility data. As part of this use case, **FederHub** is represented by the TU Dortmund University.

7.7.3 Feder Usage Evaluation

After the use case and participants have been introduced, the subsequent subsections detail how *Feder* is used along the different phases of the scenario. The objective is to systematically evaluate whether *Feder* covers the required functionalities for privacy-preserving FL in a GAIA-X Data Space and to identify potential gaps with respect to provided privacy-preserving FL and federated unlearning approaches. In the following, the individual steps are described and the responsibilities of each participant are made explicit.

Phase 1: Data Space Setup

The evaluation starts with the setup of the Data Space that provides the trusted technical and organizational foundation for data exchange between the participants, consistent with current Mobility Data Space concepts. The three **FederClients** jointly conclude that only a privacy-preserving FL architecture such as *Feder* can enable collaborative parcel delivery volume forecasting without violating data protection and data sovereignty constraints.

To instantiate this architecture, the **FederClients** contact **FederHub**, which offers services to deploy and provision the necessary federated infrastructure components in accordance with GAIA-X data connector patterns. **FederHub** assists each **FederClient** in setting up the required local infrastructure, such as EDC-based data connectors, to enable standardized and policy-controlled access to the Data Space.

For this purpose, **FederHub** uses the FedDScon mechanism to deploy and configure the EDC connectors at each **FederClient** according to predefined security and governance requirements. Once the local infrastructures at all **FederClients** are configured, **FederHub** deploys the required federated services via FedDScon in order to instantiate the federated Data Space on top of the underlying GAIA-X infrastructure.

After these steps, each **FederClient** can participate in the Data Space, as they all expose a standardized access point through their EDC connector, configured using FedDScon features. Through this setup, a shared but data-sovereign environment is established, in which further FL-specific services can be provisioned.

Phase 2: FL Aggregator Service Provisioning

At this point, only the generic GAIA-X-compliant Data Space has been established, which supports sovereign data and service exchange but no FL-specific interactions yet. However, for the parcel delivery forecasting use case, it is necessary that **FederClients** can perform FL training and communicate model updates securely through their connectors.

To enable this, **FederHub** reconfigures each EDC connector with the required FL extensions. Concretely, the deployment configurations for each **FederClient** are extended with privacy-preserving FL modules provided by HEX-FL, as well as unlearning support from CeMUFl. These extensions are automatically loaded and become available after an orchestrated restart of each connector, which can be triggered using FedDScon automation features.

In addition to the client-side configuration, **FederHub** deploys the corresponding server-side components in its local infrastructure to support centralized but privacy-preserving FL aggregation. This includes a *Model Management Module* as well as a *FL Aggregation Module*, which implement encrypted aggregation and model management as discussed in the previous sections. The configuration of these modules and the restart of the **FederHub** connector are also controlled via FedDScon.

Phase 3: FederClient Contract Agreement

Once the infrastructure has been deployed for all participants, the next step is to define the concrete data flows and configure the data exchange for the parcel delivery volume prediction task. In line with FL principles, training is performed locally at each **FederClient**, so that only model parameters or updates, rather than raw data, are exchanged

through the Data Space. For this use case, an FL protocol similar to that described in HEX-FL is adopted, in which each **FederClient** communicates bidirectionally with **FederHub**, that acts as the central aggregator.

The bidirectional communication channels are realized via asset and contract agreement features provided by FedDScon, building on the asset- and contract-based interaction model of data connectors. **FederHub** configures the framework so that each EDC connector in both **FederClients** and **FederHub** exposes assets for a local model update on the **FederClient** side and a global model update on the **FederHub** side.

After creating assets on both sides, FedDScon is used to automatically conclude data usage contracts according to the defined assets, thus establishing explicit policies for the exchange of model updates. During contract negotiation, participating companies also agree on a shared ML model architecture, which ensures compatibility for federated averaging and avoids inconsistencies in the aggregation process. With these contracts in place, each **FederClient** can provide local model updates to the server aggregator and consume global model updates distributed by **FederHub**.

Since the use case involves three **FederClients**, the asset creation and contract conclusion steps are carried out individually for each participant. This mirrors typical multi-client FL setups in which each client maintains its own connector and contractual relation with the central aggregator within a federated data ecosystem.

Phase 4: FL Training

Immediately after successful contract agreements between **FederHub** and each **FederClient**, the *HE FL Client Module* embedded in the EDC connector is automatically triggered to start the initial training of the local model. This follows established FL workflows, where a global model is disseminated to clients, which then perform local training and send back encrypted updates.

The **FederClient**-side FL extension uses parameters defined in the contract (e.g., model architecture, number of epochs, learning rate) to autonomously execute local training without manual intervention. Training data is provided by each **FederClient** via an internal ML data service exposing a REST endpoint, from which the EDC connector retrieves the data. The target endpoint was configured during Phase 2 when the FL-related extensions were installed.

Once all required configuration information is available, the local training step as well as HE of the resulting model updates are carried out automatically, in line with the HEX-FL approach that protect model updates against inversion attacks. The encrypted local model weights are then transmitted to **FederHub**, which stores them until updates from all **FederClients** have been received in accordance with the agreed model definition.

After collecting all encrypted updates, **FederHub** initiates a secure federated averaging step equivalent to HE-based aggregation procedures described in Chapter 5. The resulting

updated global model is then redistributed to all **FederClients** through the Data Space using the previously configured assets and contracts, thus completing the first FL training round.

Phase 5: Prediction Request

To exemplify the prediction workflow from a client perspective, the following describes the process at **Krone**, which is representative for all **FederClients**. After one or multiple FL rounds have been completed and a new global model version has been distributed, **Krone** intends to use the model to forecast parcel delivery volumes in its operational planning.

For this purpose, **Krone** operates a local `business logic service` within its corporate network that is aware of the prediction endpoint provided by the *HE FL Client Module* of the EDC connector. The business service invokes this endpoint with the relevant input features of current parcel delivery demand, thereby delegating the prediction task to the locally hosted and regularly updated model.

Upon a prediction request, the *HE FL Client* first verifies the validity and certification status of the local model by communicating with services operated by **FederHub**, ensuring that the client uses the latest approved global model. This certification step corresponds to functionality provided by CeMUFl and is executed once per new model version in order to limit overhead while ensuring compliance.

After successful verification, the prediction request is forwarded to the local model, which computes the parcel demand forecast and returns the result to the business logic service, where it can be integrated into downstream planning and decision processes. This design allows operational services to benefit from collaboratively trained models without leaving the sovereign environment of the client.

Phase 6: Unlearning Request

During operation, it is possible that customers of **Rhenus** exercise their data protection rights under GDPR, requesting that their personal or contract-related data be removed from further processing. Such requests create the need for machine unlearning mechanisms that can remove the influence of specific data from an already trained model.

In response, **Rhenus** triggers a model unlearning procedure provided by a CeMUFl component that supports federated unlearning in a Data Space setting, that can forget specific samples, classes or clients. The unlearning component performs a local unlearning step according to the approach described in Chapter 6 and updates the model version that is registered at **FederHub**.

The continuously running *Model Management Module* detects the updated model version and enforces a model update across all **FederClients**, ensuring that each client replaces its local model with the unlearned variant. As part of this process, the updated model is

re-verified using the certification mechanisms described earlier, so that only compliant and unlearned models are used for future predictions, which aligns with regulatory expectations and technical proposals for FL unlearning. Consequently, after the enforced model update, all **FederClients** operate on a model that no longer encodes information from the removed **Rhenus** customer data.

Following iterations

The phases described above constitute a single representative iteration of the processes that *Feder* enables on top of Data Spaces and state-of-the-art privacy-preserving FL and unlearning techniques. In subsequent phases, additional clients can join the Data Space. Such newcomers must start at Phase 1 to perform onboarding and connector configuration, after which all **FederClients** may decide to trigger a renewed training cycle from Phase 4 to integrate the new data sources.

Similarly, if another participant requires the removal of specific data from the model, only Phase 6 has to be repeated, demonstrating that unlearning can be decoupled from the rest of the operational workflow. In practice, the training phase (Phase 4) is often executed multiple times in sequence to continuously improve model quality, which matches online or continual learning interfaces included in many modern FL frameworks for logistics and mobility forecasting. All phases can run largely independently from the perspective of the participants, except where the *Feder* framework automatically triggers follow-up actions such as aggregation, deployment, or unlearning.

7.8 Summary

This chapter consolidates the previously developed building blocks into a unified, deployable architecture and demonstrates that they compose into a working end-to-end solution for GDPR-conform FL across enterprises.

7.8.1 Feder: Integrated Framework

Feder integrates FedDScon, HEX-FL and CeMUFl into a unified and modular framework for privacy-preserving FL in sovereign Data Spaces. At the infrastructure layer, FedDScon automates Data Space provisioning by deploying and configuring EDC connectors, defining assets and establishing policy-enforced contract agreements that govern cross-enterprise exchange of FL artifacts. Building on this governance layer, HEX-FL provides standardized FL workflows via a two-layer privacy paradigm, combining configurable ϵ-DP (data-level) with HE-based secure aggregation for confidentiality of intermediate model updates (model-level). Finally, CeMUFl extends the lifecycle with certified federated unlearning, enabling verifiable removal of learned influence and thereby supporting the *Right to Erasure* under GDPR Article 17 in distributed settings. In general, the library-based extension design and standardized interfaces ensure interchangeability of privacy

modules and long-term maintainability under evolving technical and regulatory requirements.

7.8.2 Evaluation

The evaluation validated the usability of *Feder* through a phased, real-world-inspired logistics scenario motivated by GAIA-X 4 ROMS-style Data Space ecosystems, in which multiple companies collaboratively forecast parcel delivery volumes without sharing raw data. Three **FederClients** and a coordinating **FederHub** were used to exercise the complete operational chain: Data Space setup, deployment of client- and server-side FL extensions, automated contract agreement for bidirectional model-update exchange and execution of encrypted training with secure aggregation and redistribution of new global model versions. The demonstrator further covered operational inference, where clients consume the collaboratively trained model locally and it explicitly evaluated an unlearning event in which a participant triggers certified unlearning and the framework enforces resynchronization and compliant model replacement across participants. By covering training, deployment, inference and unlearning within one coherent workflow, the evaluation showcased that *Feder* functions as a holistic and practical blueprint for cross-enterprise GDPR-conform FL in Data Spaces.

7.9 Discussion

This chapter directly addresses **Research Question 4** by consolidating the previously introduced mechanisms into **Feder**, a single operational framework that spans infrastructure, privacy-preserving training and compliance-relevant lifecycle control.

The core contribution is the finalized integration of FedDScon, HEX-FL and CeMUFl into one coherent architecture that supports the complete cross-enterprise FL process chain: onboarding and governance, privacy-preserving training and aggregation, deployment and inference and special lifecycle events (e.g., unlearning). Holism in the sense of **Research Question 4** is achieved because **Feder** does not treat privacy as an isolated algorithmic add-on. Instead, privacy mechanisms and lifecycle enforcement are embedded into an interoperable Data Space workflow, implemented through standardized interfaces and modular extensions. This design choice is practically enabled by the extensibility and customizability of Eclipse Dataspace Components and the EDC connector ecosystem, which explicitly supports reusable components and integration via extensions. Moreover, CeMUFl elevates certified unlearning to a first-class operational procedure that supports the *Right to Erasure* (GDPR Article 17) at the model-lifecycle level, rather than limiting compliance to local data deletion, therefore strengthening trust in the privacy-guarantees.

The evaluation is designed as a practicability demonstration that positions **Feder** in a realistic logistics context motivated by GAIA-X 4 ROMS, i.e., a Gaia-X-related setting targeting data-sovereign, cross-actor ecosystems for mobility and logistics applications. By

grounding the framework in a representative logistics forecasting use case with multiple organizational participants, the evaluation provides evidence that **Feder**'s architectural integration is implementable and usable under realistic cross-enterprise constraints (sovereignty, governed exchange and compliance obligations), rather than remaining a purely conceptual reference architecture.

8 Summary and Outlook

The previous chapter presented Feder *as the main contribution of this thesis, introducing it as a holistic and modular framework that integrates FedDScon, HEX-FL and CeMUFl modules as well as two Data Space protocols (*FL DS Protocol *and* UL-Cert DS Protocol*) to enable GDPR-conform FL in cross-enterprise* Data Spaces.

This chapter concludes the dissertation by first summarizing how the thesis answers the four Research Questions*, linking each answer to the corresponding chapters and results in Section 8.1. Afterwards it will consolidate the resulting artifacts in a compact* Summary of Contributions. *Section 8.3 will provide a critical* Discussion *of the contributions followed by the* Outlook *in Section 8.4 of open challenges and promising directions for future work.*

8.1 Answers to Research Questions

This section summarizes how the thesis answers the four *Research Questions* introduced in the problem statement and it briefly links each answer to the corresponding technical results and thesis chapters.

8.1.1 Research Question 1

Research Question 1 asked how modern software technologies and architectures can be used to create a standardized and sovereign environment for cross-enterprise FL tasks. This thesis answered the question by designing and validating a *connector*-based *Federated Data Space* foundation that operationalizes interoperability, contract-governed exchange and policy enforcement as first-class architectural primitives. The corresponding results are consolidated in the FedDScon contribution, which is derived from a systematic evaluation of *Data Space* connector technologies and their suitability for FL-centric workflows (see Sections 4.3 and 4.4).

A core finding is that both IDS and the EDC can be adapted to cross-organizational analytics, but differ substantially in extensibility, protocol alignment and practical operability for advanced FL scenarios. In particular, the evaluation and demonstrators show that EDC's extension model and protocol flexibility provide a more sustainable basis for integrating additional FL building blocks in a modular manner (e.g., aggregation services, poisoning detection, or privacy-enhancing components) than approaches that rely on scenario-specific workarounds. These insights motivate FedDScon as a reusable EDC-based framework that automates *asset provisioning*, *catalog publication*, *contract nego-*

tiation and *transfer initiation*, thereby reducing the operational friction of establishing FL-capable *Data Spaces* (see Section 4.5).

8.1.2 Research Question 2

Research Question 2 asked how privacy-by-design can be guaranteed for FL tasks inside standardized and sovereign environments such as *Data Spaces*. This thesis answered the question by developing and implementing novel privacy-preserving FL approaches that integrate complementary mechanisms from PPML, namely *DP* and *HE*, directly into distributed learning protocols and FL architectures (see Chapter 5). The result is a layered privacy architecture for cross-enterprise *Data Spaces* that combines data-level protection (via DP) with confidentiality of model updates during federated aggregation (via HE).

On the data layer, DP-LLP extends *Learning from Label Proportions* by injecting calibrated noise into batch-wise label counts before exchanging label proportions between participants, thereby enabling fully distributed prediction while providing record-level privacy guaranties (see Section 5.3). Its empirical evaluation on distributed traffic data shows that moderate privacy budgets (e.g., $\epsilon \approx 0.1$) preserve most of the non-private utility, whereas very small privacy budgets lead to substantial accuracy degradation, making the privacy-utility trade-off explicit in realistic time-series settings. Building on this principle, DP-LSTM increases model expressiveness by combining deep temporal encoders with the same privacy-by-design idea that cross-organizational communication is restricted to differentially private aggregates, rather than gradients or raw records (see Section 5.4).

At the cryptographic layer, HEX-FL extends *Federated Averaging* by encrypting model updates under a multiparty CKKS-based HE scheme and performing aggregation directly on ciphertexts, thus protecting intermediate model updates during cross-enterprise coordination (see Section 5.5). Together, DP-LLP, DP-LSTM and HEX-FL form a coherent toolbox that demonstrates how privacy-preserving mechanisms can be embedded into FL workflows in *Data Spaces* without sacrificing practical deployability and interoperability.

8.1.3 Research Question 3

Research Question 3 asked what influence the GDPR has on FL frameworks, whether the proposed frameworks can be made GDPR-compliant and what is required to ensure such compliance. The thesis answered this question by showing that privacy-preserving training in federated *Data Spaces* alone is insufficient for operational GDPR compliance, because regulatory obligations also include lifecycle requirements, most notably the *Right to Erasure* (Art. 17), which motivates *federated unlearning* in practice (see Chapter 6). Consequently, the thesis contributes CeMUFl, an unlearning-enabled approach of the FL pipeline that combines an algorithmic unlearning procedure (FedSSU-based) with a cer-

tification mechanism that makes compliance verifiable in heterogeneous and potentially adversarial client environments.

On the algorithmic side, FedSSU is integrated into the findings of **Research Questions 1** and **2**, namely the FedDScon exectution environment and HEX-FL, allowing targeted removal of client contributions while avoiding complete retraining as the default response to erasure requests (see Section 6.3).

On the systems side, a novel hash-based certification protocol (program structure, model identity and execution trace) is introduced to verify that clients actually deploy and execute the correct unlearned model versions, rather than retaining or reusing obsolete non-compliant models (see Section 6.4). The evaluation demonstrates that the unlearning procedure can be substantially faster than full retraining in the reported setup while also making explicit the limitations of approximate unlearning (e.g. faster reacquisition of forgotten information), which motivates careful parameterization and governance-aware operation (see Section 6.5).

8.1.4 Research Question 4

Research Question 4 asked how a holistic framework can be developed that integrates cross-enterprise process chains, GDPR regulations and advanced ML methodologies. This thesis answered the question by consolidating the previously researched and evaluated approaches into *Feder*, a modular end-to-end framework for GDPR-conform FL in sovereign *Data Spaces* (see Sections 7.2 and 7.6). *Feder* composes

1. FedDScon as the infrastructure and governance layer,
2. HEX-FL as the privacy-preserving training and aggregation layer and
3. CeMUFl as the certified unlearning and compliance-relevant lifecycle layer,

thereby integrating sovereignty, privacy-by-design and enforceable lifecycle interventions into a single architecture.

At the architectural level, *Feder* separates client-side and server-side responsibilities via standardized extension interfaces, allowing privacy and compliance mechanisms (e.g., HE, DP and certified unlearning) to be deployed as reusable connector extensions while preserving *Data Space* interoperability. At the operational level, a GAIA-X-inspired logistics scenario is used to explain the practicability across the full chain. The *Data Space* setup, automated contract establishment for model-update exchange, encrypted training and aggregation, inference usage and an unlearning event with enforced resynchronization and verification across participants (see Section 7.7). As a result, *Feder* provides a concrete blueprint for cross-enterprise FL that remains privacy-preserving during training and inference and with its feature-set it can satisfy all relevant requirements that can be derived from GDPR.

8.2 Summary of Contributions

As detailed in the previous section, this thesis addressed all formulated *Research Questions* and thereby, the overarching problem statement of how an integrated, unified, privacy-compliant FL architecture could be designed (see Section 1.2). A central characteristic of the work is that each contribution is motivated by an extensive evaluation of relevant technologies and design alternatives prior to consolidating the final artifacts, ensuring that the resulting frameworks are derived from systematically assessed requirements and constraints.

This thesis advances the state of the art through four main contributions that together enable GDPR-conform FL in cross-enterprise *Data Spaces*. First, **FedDScon** contributes a reusable *Data Space* foundation that improves the practical usability of modern connector technology for cross-organizational deployments and provides the infrastructural baseline for governed, interoperable workflows.

Second, **HEX-FL** contributes to the privacy-preserving FL mechanisms by integrating DP and HE into FL architectures, enabling secure multi-party training under privacy-by-design constraints.

Third, **CeMUFl** contributes to the missing regulatory and lifecycle layer by integrating GDPR-derived requirements into federated pipelines, most notably through certified unlearning to operationalize erasure-related obligations in practice.

Finally, the holistic framework **Feder** constitutes the overarching synthesis of this thesis by integrating **FedDScon**, **HEX-FL** and **CeMUFl** into a modular, production-oriented end-to-end framework that can be applied to GDPR-conform FL tasks in cross-enterprise environments. Regarding **Feder**, one relevant contribution is, that it is tested and run in a production use case of GAIA-X 4 ROMS.

8.3 Discussion

This thesis investigated the practical feasibility of executing FL in modern data ecosystems, where cross-enterprise communication and interoperability are decisive criteria for industrial adoption. To this end, the work first evaluated and then developed methods on multiple architectural levels to operationalize FL under the governance and technical constraints of *Data Spaces*. A central driver throughout the thesis was the GDPR, which requires privacy-by-design, accountable processing and sovereign exchange mechanisms, thereby constraining how collaborative learning systems can be deployed across organizational boundaries. Consequently, the proposed artifacts and their integration in the holistic framework *Feder* explicitly target regulated cross-enterprise settings and aim to provide a coherent path towards GDPR-conform FL.

From a software-architecture perspective, the contributions were structured into three layers, reflecting typical separations between infrastructure, privacy-preserving learn-

ing and lifecycle extensions. At the infrastructure layer, the evaluation of the connector technologies and the resulting framework FedDScon address the core challenge of governed cross-enterprise communication and lower the operational barrier to establish *Data Spaces* that can host FL workflows. However, a current limitation is that FedDScon is bound to a specific EDC version used during development (v0.6.4). As developments of the EDC evolve, it is mandatory to update the framework using the updated EDC version for *Data Space* deployments.

Building on the infrastructure, the privacy-preserving learning layer demonstrates that FL can be strengthened by integrating state-of-the-art privacy-enhancing techniques based on noise injection (DP) and encrypted computation (HE), thereby addressing both data-level and model-level confidentiality requirements in federated settings. At the same time, the current HEX-FL realization assumes a trusted aggregation role, because the model weights are delivered in decrypted form to clients after the aggregation and decryption process. This trust assumption is a pragmatic design choice for the deployability resulting from the implementation of the *Multiparty Computation* protocol, but it creates a clear trust concentration. The *FederHub* (and its extension) becomes a critical component that must be governed, audited and technically protected to prevent retention or misuse of intermediate plaintext artifacts. Another point regarding HEX-FL is that the framework currently requires all participants to agree on a model architecture. This agreement must be done outside of the framework, so coordination at the human level is necessary first. On the one hand, this is an advantage as incorrect model structures cannot automatically be applied to each other, but on the other hand, it is an additional overhead that could be prevented through automation.

Finally, the compliance and lifecycle layer contributed by CeMUFl extends the privacy-preserving FL stack with certified unlearning capabilities to address GDPR-driven erasure requirements in operational federated environments. A key strength of this layer is its modularity. Although the presented realization integrates FedSSU as a concrete unlearning procedure, the architecture is not conceptually restricted to this algorithm and can incorporate alternative unlearning approaches if federation participants agree on another shared unlearning approach.

In summary, the thesis provides a multi-layered, extensible architecture that can deliver GDPR-conform FL in cross-enterprise environments under the practical precondition that participating organizations jointly commit to the shared technical stack and its governance assumptions as provided by *Feder*.

8.4 Outlook

The presented framework *Feder* provides a robust foundation for privacy-preserving FL in cross-enterprise environments, but several research and engineering directions remain open due to the fast pace of innovation in PPML and federated system design. In particular, future work can improve privacy-efficiency trade-offs (runtime, communication,

memory) by integrating novel DP approaches, more efficient cryptographic aggregation variants, or alternative privacy-preserving FL methodologies, while preserving the modular exchangeability of components envisioned in *Feder*. A further research direction concerns reducing or eliminating the trusted-aggregator assumption in HEX-FL, for example by extending the architecture towards stronger distributed trust, more decentralized decryption workflows, or verifiable enforcement mechanisms that reduce the reliance on a single trusted coordination point.

With respect to unlearning, future work can integrate emerging unlearning algorithms and certification approaches that aim to provide stronger evidence that the influence of deleted data is no longer present in the model, beyond the current unlearning approach and verification framework proposed by this thesis. Because the unlearning logic is architecturally modular, such advances can be incorporated as interchangeable modules, provided that all participating organizations agree on the updated procedure and its operational semantics.

Beyond technical extensions, *Feder* can serve as a foundation architecture for a larger EU-oriented ecosystem of certified ML and FL services, in which standardized certification and deployment processes reduce the adoption barrier for regulated industries. A concrete outlook is a tool-supported platform built on top of *Feder* that bundles compliance-relevant controls (e.g. auditable configurations, verifiable execution environments and standardized certification workflows) and enables companies to deploy certified models with less uncertainty regarding GDPR-conformity.

List of Figures

1.1 Overview of Contributions 4

3.1 Design Science Research Methodology 24
3.2 Gaia-X architecture and research project 27

4.1 Multi-Agent System overview 42
4.2 Sequence Diagram of the Contract Net Protocol 44
4.3 Sequence diagram for synchronous IDS communication 45
4.4 Data Space setup with IDS 47
4.5 Data Space setup with EDC 48
4.6 EDC Data Space communication 50
4.7 FedDScon: Sequence diagram for Asset Creation 53
4.8 FedDScon: Sequence diagram for Concluding a Contract 56

5.1 DP-LLP scenario and window creation 78
5.2 DP-LLP accuracy evaluation compared to LLP 81
5.3 DP-LLP accuracy evaluation with reference to ε 82
5.4 DP-LSTM distributed network architecture 85
5.5 DP-LSTM label proportion transfer 86
5.6 DP-LSTM model architecture 87
5.7 DP-LSTM linear layer LLP fusion 88
5.8 DP-LSTM visual evaluation of speed prediction 93
5.9 HEX-FL deployment diagram of framework components 99
5.10 HEX-FL evaluation of time consumption per phase 104
5.11 HEX-FL runtime evaluation for initial phases 106
5.12 HEX-FL memory evaluation 107
5.13 HEX-FL accuracy evaluation along clients 109

6.1 Component diagram for CeMUFl 122
6.2 Sequence diagram of a client verification process 127
6.3 Evaluation of the runtime 131
6.4 Evaluation of the Anamnesis Index 133
6.5 Evaluation of the λ influence 134
6.6 Evaluation of the Jensen-Shannon Distance 135

7.1 Finalized *GDPR*-conform framework architecture (Feder). 141

List of Tables

3.1 DSR Relevance and Rigor Cycles per Contribution 26

4.1 ATAM screening questions 37
4.2 ATAM screening questions (continued) 38
4.3 ATAM evaluation results 40

5.1 DP-LSTM loss evaluation 91
5.2 HEX-FL evaluation of runtime scaling along clients 105
5.3 HEX-FL memory evaluation along clients 108
5.4 Feature comparison of LLP-approaches 113
5.5 Library performance comparison between Lattigo and OpenFHE 116

List of Algorithms

4.1 EDC request body for creating an asset in the Data Space. 54
4.2 FedDSGcon tool calling 59
5.1 Computation of ε-differentially private label proportions at node n_j . . . 80
5.2 Multiparty Homomorphic Encryption Scheme - Initialization 100
5.3 Multiparty Homomorphic Encryption Scheme - Train and Update 102

List of Acronyms

AI Artificial Intelligence.

AISBL association internationale sans but lucratif.

API Application Programming Interface.

ATAM Architecture Trade-off Analysis Method.

BFV Brakerski-Fan-Vercauteren.

BGV Brakerski-Gentry-Vaikuntanathan.

CeMUFl Certified Machine Unlearning for Federated Learning.

CIFAR Canadian Institute for Advanced Research.

CKKS Cheon-Kim-Kim-Song.

CNN Convolutional Neural Network.

CNP Contract Net Protocol.

DAPS Dynamic Attribute Provisioning Service.

DFKI Deutsches Forschungszentrum für Künstliche Intelligenz.

DP Differential Privacy.

DP-LLP Differentially Private Learning from Label Proportions.

DP-LSTM Differentially Private Long Short-Term Memory.

DSP Dataspace Protocol.

DSR Design Science Research.

EDC Eclipse Dataspace Connector.

ETA Estimated Time of Arrival.

EU European Union.

FedDScon Federated Data Space Connector.

FedSSU Federated Secret Sharing Unlearning.

FHE Fully Homomorphic Encryption.

FHEW Fast Fully Homomorphic Encryption over Torus.

FL Federated Learning.

GDPR General Data Protection Regulation.

HE Homomorphic Encryption.

HEX-FL Homomorphic Encryption Extended Federated Learning.

HTTP Hypertext Transfer Protocol.

IDS International Data Spaces.

IDSA International Data Spaces Association.

IEEE Institute of Electrical and Electronics Engineers.

ISO International Organization for Standardization.

JSD Jensen-Shannon Divergence.

JSON JavaScript Object Notation.

LDP Local Differential Privacy.

LLP Learning from Label Proportions.

LSTM Long Short-Term Memory.

LWE Learning With Errors.

MAS Multi-Agent System.

MHE Multiparty Homomorphic Encryption.

ML Machine Learning.

MNIST Modified National Institute of Standards and Technology.

MPC Secure Multi-Party Computation.

MSE Mean Squared Error.

MVD Minimum Viable Demonstrator.

OCI Open Container Initiative.

ODRL Open Digital Rights Language.

PHE Partially Homomorphic Encryption.

PPML Privacy-Preserving Machine Learning.

RAM Reference Architecture Model.

REST Representational State Transfer.

RLWE Ring Learning With Errors.

RNS Residue Number System.

RQ Research Question.

SCATS Sydney Coordinated Adaptive Traffic System.

SEAL Simple Encrypted Arithmetic Library.

SGD Stochastic Gradient Descent.

SHE Somewhat Homomorphic Encryption.

SIMD Single Instruction, Multiple Data.

TFHE Fast Fully Homomorphic Encryption.

TRL Technology Readiness Level.

UI User Interface.

WASM WebAssembly.

Bibliography

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. https://doi.org/10.1145/2976749.2978318

Abowd, J. M. (2018). The u.s. census bureau adopts differential privacy. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2867. https://doi.org/10.1145/3219819.3226070

Agrawal, R., & Joshi, A. (2023). *On architecting fully homomorphic encryption-based computing systems*. Springer International Publishing.

Badawi, A. A., Alexandru, A., Bates, J., Bergamaschi, F., Cousins, D. B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Pascoe, C., Polyakov, Y., Quah, I., R.V., S., Rohloff, K., Saylor, J., … Zucca, V. (2022). OpenFHE: Open-source fully homomorphic encryption library.

Bao, W., Wang, H., Wu, J., & He, J. (2023). Optimizing the collaboration structure in cross-silo federated learning. *Proceedings of the 40th International Conference on Machine Learning*, 1718–1736.

Benaissa, A., Retiat, B., Cebere, B., & Belfedhal, A. E. (2021). Tenseal: A library for encrypted tensor operations using homomorphic encryption. *ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML 2021)*.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., & Papernot, N. (2021). Machine unlearning. *2021 IEEE symposium on security and privacy (SP)*, 141–159.

Brakerski, Z. (2012). Fully homomorphic encryption without modulus switching from classical gapsvp. *Advances in Cryptology – CRYPTO 2012*, 868–886.

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*. https://doi.org/10.1145/2633600

Brakerski, Z., & Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-lwe and security for key dependent messages. *Advances in Cryptology – CRYPTO 2011*, 505–524.

Brakerski, Z., & Vaikuntanathan, V. (2014). Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 831–871. https://doi.org/10.1137/120868669

Bun, M., & Steinke, T. (2016). Concentrated differential privacy: Simplifications, extensions, and lower bounds. *Theory of Cryptography*, 635–658.

Cavoukian, A., et al. (2009). Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada*, 12.

Chen, C., Wei, L., Xie, J., & Shi, Y. (2025). Privacy-preserving machine learning based on cryptography: A survey. *ACM Trans. Knowl. Discov. Data*. https://doi.org/10.1145/3729234

Chen, H., Laine, K., & Player, R. (2017). Simple encrypted arithmetic library - seal v2.1. *Financial Cryptography and Data Security*, 3–18.

Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. *Advances in Cryptology – ASIACRYPT 2017*, 409–437.

Chundawat, V. S., Tarun, A. K., Mandal, M., & Kankanhalli, M. (2023). Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 2345–2354. https://doi.org/10.1109/TIFS.2023.3265506

Codecá, L., Frank, R., Faye, S., & Engel, T. (2017). Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 52–63.

Corander, J., Remes, U., & Koski, T. (2021). On the jensen-shannon divergence and the variation distance for categorical probability distributions. *Kybernetika*, 879–907.

Damgård, I., Pastro, V., Smart, N., & Zakarias, S. (2012). Multiparty computation from somewhat homomorphic encryption. *Advances in Cryptology – CRYPTO 2012*, 643–662.

De Vos, M., Kirrane, S., Padget, J., & Satoh, K. (2019). Odrl policy modelling and compliance checking. *Rules and Reasoning*, 36–51.

Diao, E., Ding, J., & Tarokh, V. (2020). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

Dimitrov, D. I., Balunovic, M., Konstantinov, N., & Vechev, M. (2022). Data leakage in federated averaging. *Transactions on Machine Learning Research*.

Ding, B., Kulkarni, J., & Yekhanin, S. (2017). Collecting telemetry data privately. *Advances in Neural Information Processing Systems*.

Dulac-Arnold, G., Zeghidour, N., Cuturi, M., Beyer, L., & Vert, J. (2019). Deep multi-class learning from label proportions. *CoRR*.

Dwork, C. (2006). Differential privacy. *Automata, Languages and Programming*, 1–12.

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and TrendsÂ® in Theoretical Computer Science*, 211–407. https://doi.org/10.1561/0400000042

EPFL-LDS, Tune Insight SA. (2024, August). Lattigo v6 [EPFL-LDS, Tune Insight SA].

Erlingsson, Ú., Pihur, V., & Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 1054–1067. https://doi.org/10.1145/2660267.2660348

European Parliament & Council of the European Union. (2016, May 4). *Regulation (EU) 2016/679 of the European Parliament and of the Council* [Of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General

Data Protection Regulation)]. Retrieved April 13, 2023, from https://data.europa.eu/eli/reg/2016/679/oj

Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption.

Fawaz, S. M., Belal, N., ElRefaey, A., & Fakhr, M. W. (2021). A comparative study of homomorphic encryption schemes using microsoft seal. *Journal of Physics: Conference Series*, 012021. https://doi.org/10.1088/1742-6596/2128/1/012021

Fazio, N., Gennaro, R., Jafarikhah, T., & Skeith, W. E. (2017). Homomorphic secret sharing from paillier encryption. *Provable Security*, 381–399.

Ficek, J., Wang, W., Chen, H., Dagne, G., & Daley, E. (2021). Differential privacy in health research: A scoping review. *Journal of the American Medical Informatics Association*, 2269–2276. https://doi.org/10.1093/jamia/ocab135

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 169–178.

Gösling, H., Maecker, D., Pieper, T., Sachweh, T., & Heinbach, C. (2024). A procedure for conceptualizing and implementing spade agents. *Engineering Multi-Agent Systems*.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 4.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 75–105.

Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., & Zhang, Y. (2021). Personalized cross-silo federated learning on non-iid data. *Proceedings of the AAAI conference on artificial intelligence*, 7865–7873.

Jagielski, M., Kearns, M., Mao, J., Oprea, A., Roth, A., -Malvajerdi, S. S., & Ullman, J. (2019). Differentially private fair learning. *Proceedings of the 36th International Conference on Machine Learning*, 3000–3008.

Jin, X., Chen, P.-Y., Hsu, C.-Y., Yu, C.-M., & Chen, T. (2021). Cafe: Catastrophic data leakage in vertical federated learning. *Advances in Neural Information Processing Systems*, 994–1006.

Jung, C., & Dörr, J. (2022). Data usage control. In *Designing data spaces : The ecosystem approach to competitive advantage* (pp. 129–146). https://doi.org/10.1007/978-3-030-93975-5_8

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 1–210.

Kamarinou, D., Millard, C., & Singh, J. (2016). Machine learning with personal data: Profiling, decisions and the eu general data protection regulation. *29th Conference on Neural Information Processing Systems (NIPS 2016)*.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., & Carriere, J. (1998). The architecture tradeoff analysis method. *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193)*, 68–78. https://doi.org/10.1109/ICECCS.1998.706657

Koen, P., Kollenstart, M., Marino, J., Pampus, J., Turkmayali, A., Steinbuss, S., & Weiß, A. (2025, July). Eclipse dataspace protocol 1.0.0 [08.10.2025].

Kremer, M., Pohling, L., Gösling, H., Heinbach, C., Sachweh, T., Gogineni, S., & Berger, K. (2023). An intelligent arrival time prediction service in a federated data ecosystem: The minimum viable demonstrator of the GAIA-x 4 ROMS research project. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4331859

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

Lee, J.-W., Kang, H., Lee, Y., Choi, W., Eom, J., Deryabin, M., Lee, E., Lee, J., Yoo, D., Kim, Y.-S., & No, J.-S. (2022). Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 30039–30054. https://doi.org/10.1109/ACCESS.2022.3159694

Leng, Y., Xu, L., Liu, J., Zhang, X., Mei, L., Qu, Y., & Xu, C. (2025). Fedssu: Flexible and efficient decentralized unlearning for federated learning. *The Journal of Supercomputing*. https://doi.org/10.1007/s11227-025-07478-2

Li, Q., Yu, W., Xia, Y., & Pang, J. (2025). From centralized to decentralized federated learning: Theoretical insights, privacy preservation, and robustness challenges. *arXiv preprint arXiv:2503.07505*.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Graph convolutional recurrent neural network: Data-driven traffic forecasting. *CoRR*.

Liu, K., Hu, S., Wu, S. Z., & Smith, V. (2022). On privacy and personalization in cross-silo federated learning. *Advances in Neural Information Processing Systems*, 5925–5940.

Liu, Z., Jiang, Y., Shen, J., Peng, M., Lam, K.-Y., Yuan, X., & Liu, X. (2024). A survey on federated unlearning: Challenges, methods, and future directions. *ACM Comput. Surv.* https://doi.org/10.1145/3679014

Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. *Advances in Cryptology – EUROCRYPT 2010*, 1–23.

Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 3–es. https://doi.org/10.1145/1217299.1217302

Maecker, D., Gösling, H., & Sachweh, T. (2024). Setting up a ros2-based multi-agent system implementing the contract net protocol and ids connectors. *Engineering Multi-Agent Systems*.

Maecker, D., Harenbrock, F., Gösling, H., Sachweh, T., & Thomas, O. (2024). Synergizing trust and autonomy: Gaia-x enabled multi-agent ecosystems for advanced freight fleet management. *Engineering Multi-Agent Systems*, 82–92.

Majcherczyk, N., Srishankar, N., & Pinciroli, C. (2021). Flow-fl: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 8836–8842.

McCann, B. (2014). A review of scats operation and deployment in dublin. *Proceedings of the 19th JCT traffic signal symposium & exhibition*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data.

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 1273–1282.

Melis, L., Song, C., De Cristofaro, E., & Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 691–706. https://doi.org/10.1109/SP.2019.00029

Menéndez, M., Pardo, J., Pardo, L., & Pardo, M. (1997). The jensen-shannon divergence. *Journal of the Franklin Institute*, 307–318. https://doi.org/https://doi.org/10.1016/S0016-0032(96)00063-4

Mironov, I. (2017). Rényi differential privacy. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 263–275. https://doi.org/10.1109/CSF.2017.11

Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.-P., & Hubaux, J.-P. (2021). Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies*.

Mouchet, C. V., Bossuat, J.-P., Troncoso-Pastoriza, J. R., & Hubaux, J.-P. (2020). Lattigo: A multiparty homomorphic encryption library in go. *Proceedings of the 8th Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, 64–70.

Nasr, M., Shokri, R., & Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 739–753. https://doi.org/10.1109/SP.2019.00065

Neidhardt, E. (2024). Gaia-x compliant data exchange for autonomous driving vehicles using the eclipse dataspace connector. *30th ITS World Congress*.

Otto, B., & Jarke, M. (2019). Designing a multi-sided data platform: Findings from the international data spaces case. *Electronic markets*, 561–580.

Otto, B., ten Hompel, M., & Wrobel, S. (2019). International data spaces: Reference architecture for the digitization of industries. In *Digital transformation* (pp. 109–128).

Otto, B., ten Hompel, M., & Wrobel, S. (2022). *Designing data spaces: The ecosystem approach to competitive advantage*. Springer Nature.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology — EUROCRYPT '99*, 223–238.

Palanca, J., Terrasa, A., Julian, V., & Carrascosa, C. (2020). Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access*, 182537–182549.

Pambudi, D. S., Studiawan, H., Pratomo, B. A., & Purwitasari, D. (2025). Performance analysis of leading homomorphic encryption libraries: A benchmark study of seal, helib, openfhe, and lattigo. In *Proceedings of the 2025 4th international conference on cyber security, artificial intelligence and the digital economy* (pp. 25–30).

Papernot, N., Thakurta, A., Song, S., Chien, S., & Erlingsson, Ã. (2021). Tempered sigmoid activations for deep learning with differential privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 9312–9321. https://doi.org/10.1609/aaai.v35i10.17123

Pretzsch, S., Drees, H., & Rittershaus, L. (2022). Mobility data space. In *Designing data spaces : The ecosystem approach to competitive advantage* (pp. 343–361). https://doi.org/10.1007/978-3-030-93975-5_21

Rambau, Y. (2026). *Evaluation von edc und ids für den einsatz im federated learning (eng.: Evaluation of edc and ids for use in federated learning)* [BSc Thesis]. TU Dortmund University.

Reiberg, A., et al. (2023). Building a dataspace: Technical overview [Gaia-X Hub Austria White Paper].

Ren, H., Deng, J., & Xie, X. (2022). Grnn: Generative regression neural network—a data leakage attack for federated learning. *ACM Trans. Intell. Syst. Technol.* https://doi.org/10.1145/3510032

Rieke, N., Hancox, J., Li, W., Milletarì, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., Ourselin, S., Sheller, M., Summers, R. M., Trask, A., Xu, D., Baust, M., & Cardoso, M. J. (2020). The future of digital health with federated learning. *NPJ Digit. Med.*, 119.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 120–126. https://doi.org/10.1145/359340.359342

Rone, J. (2024). 'the sovereign cloud' in europe: Diverging nation state preferences and disputed institutional competences in the context of limited technological capabilities. *Journal of European Public Policy*, 2343–2369. https://doi.org/10.1080/13501763.2024.2348618

Ryu, J., Kim, K., & Won, D. (2023). A study on partially homomorphic encryption. *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 1–4. https://doi.org/10.1109/IMCOM56909.2023.10035630

Sachweh, T., Boiar, D., & Liebig, T. (2021). Differentially private learning from label proportions. In *Communications in computer and information science* (pp. 119–127). https://doi.org/10.1007/978-3-030-93736-2_11

Sachweh, T., Boiar, D., & Liebig, T. (2022). Distributed lstm-learning from differentially private label proportions. *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1071–1078. https://doi.org/10.1109/ICDMW58026.2022.00139

Sachweh, T., Kuhlmann, H., Gösling, H., & Liebig, T. (2025a). *FedDSGcon: PoC and tools for Data Spaces with EDC* (Version 1.0.0) [Accessed: 14.01.2025].

Sachweh, T., Kuhlmann, H., Gösling, H., & Liebig, T. (2025b). Federated data spaces as an enabler for smart city services - logistics use-case example. *2025 IEEE European Technology and Engineering Management Summit (E-TEMS)*, 80–85. https://doi.org/10.1109/E-TEMS64751.2025.11239243

Sachweh, T., Kuhlmann, H., & Liebig, T. (2023). Empowering data owners with homomorphic encrypted federated learning in decentralized data spaces. *International Symposium on Location-Based Big Data and GeoAI 2023 (LocBigDataAI 2023)*.

Sachweh, T., Kuhlmann, H., & Liebig, T. (2026). *Homomorphic encryption schemes for federated learning in cross-enterprise environments* [In Review].

Sang, G. M., Xu, L., de Vrieze, P., Bai, Y., & Pan, F. (2021). Predictive maintenance in industry 4.0. *Proceedings of the 10th International Conference on Information Systems and Technologies*. https://doi.org/10.1145/3447568.3448537

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017a). Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. https://doi.org/10.1109/SP.2017.41

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017b). Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. https://doi.org/10.1109/SP.2017.41

Silva, P. R., Vinagre, J., & Gama, J. (2023). Towards federated learning: An overview of methods and applications. *WIREs Data Mining and Knowledge Discovery*, e1486. https://doi.org/https://doi.org/10.1002/widm.1486

Song, S., Chaudhuri, K., & Sarwate, A. D. (2013). Stochastic gradient descent with differentially private updates. *2013 IEEE Global Conference on Signal and Information Processing*, 245–248. https://doi.org/10.1109/GlobalSIP.2013.6736861

Stolpe, M., Liebig, T., & Morik, K. (2015). Communication-efficient learning of traffic flow in a network of wireless presence sensors. *Proceedings of the Workshop on Parallel and Distributed Computing for Knowledge Discovery in Data Bases (PDCKDD 2015)*.

Stolpe, M., & Morik, K. (2011). Learning from label proportions by optimizing cluster model selection. *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III*, 349–364. https://doi.org/10.1007/978-3-642-23808-6_23

SWEENEY, L. (2002). K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 557–570. https://doi.org/10.1142/S0218488502001648

Taddeo, M., & Floridi, L. (2018). How ai can be a force for good. *Science*, 751–752. https://doi.org/10.1126/science.aat5991

Tikkinen-Piri, C., Rohunen, A., & Markkula, J. (2018). Eu general data protection regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, 134–153.

Timan, T., & Mann, Z. (2021). Data protection in the era of artificial intelligence: Trends, existing solutions and recommendations for privacy-preserving technologies. In *The elements of big data value: Foundations of the research and innovation ecosystem* (pp. 153–175).

Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1958–1970. https://doi.org/10.1109/TPAMI.2008.128

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., & Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 1–11. https://doi.org/10.1145/3338501.3357370

Truex, S., Liu, L., Gursoy, M. E., Yu, L., & Wei, W. (2021). Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 2073–2089. https://doi.org/10.1109/TSC.2019.2897554

Tsuji, A., & Oguchi, M. (2024). Comparison of fhe schemes and libraries for efficient cryptographic processing. *2024 International Conference on Computing, Networking and Communications (ICNC)*, 584–590.

Usländer, T., & Teuscher, A. (2022). Industrial data spaces. In *Designing data spaces : The ecosystem approach to competitive advantage* (pp. 313–327). https://doi.org/10.1007/978-3-030-93975-5_19

Voigt, P., & Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10–5555.

Wu, L., Guo, S., Wang, J., Hong, Z., Zhang, J., & Ding, Y. (2022). Federated unlearning: Guarantee the right of clients to forget. *IEEE Network*, 129–135. https://doi.org/10.1109/MNET.001.2200198

Xie, Q., Jiang, S., Jiang, L., Huang, Y., Zhao, Z., Khan, S., Dai, W., Liu, Z., & Wu, K. (2024). Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: A brief survey. *IEEE Internet of Things Journal*, 24569–24580. https://doi.org/10.1109/JIOT.2024.3382875

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* https://doi.org/10.1145/3298981

Ye, M., Fang, X., Du, B., Yuen, P. C., & Tao, D. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Comput. Surv.* https://doi.org/10.1145/3625558

Yousuf, H., Lahzi, M., Salloum, S. A., & Shaalan, K. (2021). Systematic review on fully homomorphic encryption scheme and its application. In *Recent advances in intelligent systems and smart applications* (pp. 537–551). https://doi.org/10.1007/978-3-030-47411-9_29

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 3634–3640. https://doi.org/10.24963/ijcai.2018/505

Yurdem, B., Kuzlu, M., Gullu, M. K., Catak, F. O., & Tabassum, M. (2024). Federated learning: Overview, strategies, applications, tools and future directions. *Heliyon*.

Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., Jiang, Z., Zhong, S., & Hu, X. (2025). Data-centric artificial intelligence: A survey. *ACM Comput. Surv.* https://doi.org/10.1145/3711118

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., & Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, 106775. https://doi.org/https://doi.org/10.1016/j.knosys.2021.106775

Zhu, L., Liu, Z., & Han, S. (2019). Deep leakage from gradients. *Advances in Neural Information Processing Systems*.