Protecting Privacy in Federated Time Series Analysis: A Pragmatic Technology Review for Application Developers*

Daniel Bachlechner[1][a], Ruben Hetfleisch[1][b], Stephan Krenn[2][c], Thomas Lorünser[2,3][d] and Michael Rader[1][e]

[1]*Fraunhofer Austria Research GmbH, Vienna/Wattens, Austria*

[2]*AIT Austrian Institute of Technology GmbH, Vienna, Austria*

[3]*Digital Factory Vorarlberg GmbH, Dornbirn, Austria*

fl

Keywords: Federated Time Series, Privacy-Preserving Technologies, Technology Review.

Abstract: The federated analysis of sensitive time series has huge potential in various domains, such as healthcare or manufacturing. Yet, to fully unlock this potential, requirements imposed by various stakeholders must be fulfilled, regarding, e.g., efficiency or trust assumptions. While many of these requirements can be addressed by deploying advanced secure computation paradigms such as fully homomorphic encryption, certain aspects require an integration with additional privacy-preserving technologies.

In this work, we perform a qualitative requirements elicitation based on selected real-world use cases. We match the derived requirements categories against the features and guarantees provided by available technologies. For each technology, we additionally perform a maturity assessment, including the state of standardization and availability on the market. Furthermore, we provide a decision tree supporting application developers in identifying the most promising candidate technologies as a starting point for further investigation. Finally, existing gaps are identified, highlighting research potential to advance the field.

1 INTRODUCTION

The proliferation of data-driven technologies has led to an exponential increase in the collection and analysis of time series data across multiple domains.[1] Time series data, characterised by its sequential and temporal nature, provides critical insights that drive decision-making processes in sectors such as healthcare, manufacturing and smart infrastructure. However, the sensitivity of this data often includes personal, confidential or proprietary information, requiring strict privacy measures.

In healthcare, time series data from patient monitoring systems and electronic health records can reveal important trends and treatment outcomes, contributing significantly to medical research and patient care. However, patient privacy must be protected to comply with regulatory standards such as HIPAA and GDPR, while still allowing for comprehensive data analysis. This delicate balance between data utilization and privacy is a significant challenge.

In the manufacturing sector, often referred to as Industry 4.0, time-series data collected from sensors and machines is essential for optimising production processes, predictive maintenance and improving operational efficiency. If this data is compromised, it can lead to significant competitive and operational risks. It is therefore critical to implement privacy technologies that can protect proprietary information while facilitating the analysis required for innovation and improvement.

Smart buildings, a specialised area of smart infrastructure that integrates IoT devices and sensors to monitor and control various systems such as lighting, heating and security, generate large amounts of time series data. This data can improve energy efficiency, enhance occupant comfort and ensure safety.

[a] https://orcid.org/0000-0001-7726-9065
[b] https://orcid.org/0000-0002-9428-3835
[c] https://orcid.org/0000-0003-2835-9093
[d] https://orcid.org/0000-0002-1829-4882
[e] https://orcid.org/0009-0000-1819-5246

*Authors are listed in alphabetical order.

[1]https://www.verifiedmarketresearch.com/product/time-series-analysis-software-market/

Bachlechner, D., Hetfleisch, R., Krenn, S., Lorünser, T. and Rader, M.
Protecting Privacy in Federated Time Series Analysis: A Pragmatic Technology Review for Application Developers.
DOI: 10.5220/0013356100003950

In *Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER 2025)*, pages 198-210
ISBN: 978-989-758-747-4; ISSN: 2184-5042

However, it also contains sensitive information about the behaviour and preferences of building occupants, requiring robust data protection measures to prevent unauthorised access and misuse.

What all these domains have in common is that valuable insights can be gained by combining data coming from different – potentially mutually distrusting – entities. Thus, privacy-preserving technologies (PPTs) providing formal security guarantees are required to unlock the potential of the data without compromising confidentiality. This is also in line with the paradigm of data sovereignty, as also pushed forward, e.g., by the European Data Spaces, guaranteeing users and organizations full control over their own sensitive data.

Besides social acceptance, the consideration of PPTs is also mandated by legal frameworks. For instance, Article 32 GDPR requires to consider the available state of the art when selecting measures to ensure a level of security appropriate to the identified risks. In this context, as clarified, e.g., by the European Union Agency for Cybersecurity (ENISA), a technology is considered as "state of the art" when it reaches market maturity, which, among others, depends on its availability and level of standardization.[2]

However, the landscape of PPTs is complex and rapidly evolving, making it challenging for both practitioners and researchers to navigate. Identifying technologies that have the necessary properties and are mature enough to be implemented in real-world scenarios remains a daunting task. This difficulty is compounded by the diversity of PPTs, ranging from secure multi-party computation to federated learning, each with its own strengths and limitations. As a result, there is an urgent need for comprehensive evaluations and guidelines to assist in the selection and use of the most appropriate technologies for specific time series analysis applications.

Our Contribution. This research explores the practical applicability of PPTs in real-world use cases, identifies research gaps, proposes a high-level research agenda and provides practical guidance for both practitioners and researchers aiming to implement PPTs in sensitive time series analysis.

More precisely, we provide developers of collaborative time-series analysis applications with a comprehensive overview of the available computing paradigms and complementary additional PPTs. For each technology, we provide an analysis of its properties with respect to the most relevant requirement categories derived from real-world use cases and based on discussions with relevant players in the respective domains, and support the developer by assessing the maturity level of each candidate.

Furthermore, we provide practical guidance in the identification of suitable candidate technologies for given scenarios.

Finally, we identify future research potential and propose a concise research agenda, to further increase the real-world applicability of available technologies.

Related Work. For an introduction to time-series analysis in general, we refer to the academic literature, e.g., (Brockwell and Davis, 2016).

There are systematic overviews of PPTs and guidance for the selection of such technologies both in general (Bachlechner et al., 2018) and for different application domains, e.g., for health data (Jordan et al., 2022) and production (Sielaff et al., 2023). A comprehensive overview in the context of official statistics has recently been published by the UN Statistics Division (United Nations, 2023). Support in the maturity assessment was offered by ENISA.[3][4]

However, to the best of our knowledge, no comprehensive overview supporting developers in assessing the most promising technologies for specific application domains of (federated) time series exists.

Paper Outline. The paper is structured as follows: The results of a requirements analysis are presented in Section 2. Advanced secure computation paradigms as well as complementary PPTs are introduced in Section 3, discussing their properties and maturity. Section 4 then discusses the pros and cons of all approaches and provides support for technology selection. Section 5 concludes the paper with a research agenda and practical guidelines for practitioner and researchers.

2 REQUIREMENT CATEGORIES

In the following we introduce a set of exemplary application scenarios in the context of federated time series analysis. Supported by discussions with stakeholders from the domains, we performed a qualitative requirements elicitation for each of these uses.

The use cases were chosen from complementary domains to highlight the diverse needs and expectations associated with federated time series. Although numerous specific use cases exist within each domain, the selected examples represent a wide spectrum of requirements, encompassing a broad range of applications in federated time series analysis.

[2]https://www.enisa.europa.eu/news/enisa-news/what-is-state-of-the-art-in-it-security

[3]https://www.enisa.europa.eu/publications/pets

[4]https://www.enisa.europa.eu/publications/pets-maturity-tool

Based on this, Section 2.2 then presents a clustering of the found requirements, showing clear similarities and differences in the needs of our use cases.

2.1 Motivating Use Cases

2.1.1 Domain 1: Medical Research

This use case considers the scenario of multiple hospitals collaborating on joint medical studies. This is particularly relevant in the context of rare diseases, where a sufficiently large number of patients can only be monitored and analyzed by involving geographically distributed medical facilities.

A concrete example is related to post-operative pain management and the required amount of Fentanyl for treatment, as studied by (Chiang et al., 2016), based on time series containing time stamps and dosage. Assuming that not all patients are treated in the same hospital, calculating the significance of different drug dosages can be challenging. Given that medical data is classified as special category data according to Article 10 of the GDPR, any computation or pooling of such information is constrained by stringent regulatory and ethical concerns.

Requirements. To facilitate such studies, it would be advantageous if data controlled by a hospital never leaves its facility, not even in encrypted form, as this still constitutes personally identifiable information.

Given that different studies typically involve various types of data and computations, a highly versatile setup is required. However, efficiency is only of secondary importance, and no real-time requirements exist. A reasonable IT infrastructure in terms of computational and bandwidth capacity is assumed.

Since hospitals are highly regulated, they are unlikely to maliciously deviate from protocol specifications. Nonetheless, because sensitive decisions (e.g., regarding treatment strategies) may depend on the outcomes, auditability would still be beneficial to the greatest extent possible.

2.1.2 Domain 2: Industry 4.0

Among other goals, Industry 4.0 aims at creating smart factories where machines, operators, and manufacturers are interconnected through the Internet of Things (IoT). Machines generate vast amounts of data which they transmit to the Original Equipment Manufacturer (OEM). The OEM uses this data to train sophisticated models aimed, e.g., at predictive maintenance and remote diagnostics. This can significantly enhance operational efficiency, reduce downtime, and lead to cost savings, marking a substantial advantage over traditional production methods.

However, this continuous stream of data also brings substantial confidentiality concerns, as it may contain information about production processes or proprietary technologies. This makes the OEM a central trusted party, from which sensitive information may leak, e.g., to competitors, leading to direct economic harm.

Requirements. What is needed, therefore, are technologies that enable the federated training of advanced models, without a central entity that learns any information about the underlying data. Furthermore, in the optimal case, the different contributors to such a model do not need to know about each other, i.e., no direct communication among them is necessary.

Regarding efficiency requirements, no real-time needs exist for the training phase; however, to avoid damage to the production machines, real-time analysis of the data is required.

The results of this data analysis step may have immediate consequences on business continuity: in case of false negatives, unnecessary downtime may occur, increasing costs and reducing availability. Thus, especially if inference is done remotely, any decision leading to a machine shutdown needs to be auditable by third parties to avoid fraud.

Additionally, competitors may have incentives to introduce malicious data into their data sets (also known as data poisoning), aiming to increase the downtime of their competitors. As traditional countermeasures like data validation become harder in distributed settings without central authority, it would be desirable to receive cryptographic proofs that all data included in the training phase was indeed generated by a genuine sensor of the OEM.

2.1.3 Domain 3: Smart Buildings

Modern buildings are generating huge amounts of data, which can be used by property management, operators or manufacturers of certain devices (e.g., heating, cooling), or owners of the building. However, while data access is clearly managed for certain areas, it becomes more complicated when residents require information about the data of other apartments: for instance, it may be possible to optimize one's heating settings depending on the presence of neighbors, e.g., during winter holiday seasons.

Another illustrative example considers the following platform: residents provide their energy consumption, and receive back feedback comparing their use (e.g., in form of a quantile) to that of their peers in the same building with comparable external conditions (e.g., weather, building insulation). On the one hand such gamification could increase residents' willingness to reduce their energy consumption; on the

other hand this information could also be used to detect outliers, which might indicate needs for repair.

Requirements. Residents may be reluctant to disclose detailed time series to a central authority, and especially to their peers, but they should rather remain in full control over their data. On the other hand, however, usability is of utmost importance in such a setting, such that residents should not be required, e.g., to host compute nodes for federated computations themselves. Thus, the computation should be offered as-a-service while guaranteeing privacy.

Assuming that feedback is provided, e.g., on a weekly or monthly basis, no real-time requirements exist. However, given the computation is to be carried out repeatedly, pre-computations may be acceptable to increase efficiency. Also, given the gamification approach, no immediate auditability or public verifiability requirements exist.

To minimize costs, no powerful IT infrastructure should be required to carry out the computations.

2.2 Requirement Overview

Based on the scenarios described above, we can now derive a set of generic requirement categories that describe the unique needs of each application case. Following our goal of providing guidelines also to non-experts in the fields in the specification of their applications and subsequently identifying the potential cryptographic approaches, these categories are described on a qualitative level to ease an initial assessment.

Trust Model. This category describes the acceptable assumptions that may be made in an instantiation of the use case, thereby also describing data sovereignty needs. For instance, can data be processed (even in encrypted form) by a cloud provider, or must the computation happen in-house? Is there only a need to protect against accidental data leaks, or do active attacks by malicious parties need to be taken into consideration?

Scalability. This category subsumes in particular the frequency of computations to be performed, the data sizes to be processed in each computation, or the number of users providing data.

Efficiency. This category includes actual efficiency requirements such as bandwidth and computational limitations, but also maximum execution times for each computation including real-time requirements.

Flexibility. This category focuses on the computations to be performed, i.e., whether a single, static computation is computed periodically on different input data, or whether highly dynamic computations are to be supported. It also includes the function to be evaluated (e.g., Boolean or arithmetic circuit, multiplicative depth of circuits, etc.).

Data Integrity. This category is concerned with the quality of the processed data. In particular, it specifies whether input data needs to be checked for integrity and authenticity, which depends on the decisions to be taken based on the results, and on whether data providers are mutually trusting each other regarding the quality of the their data. Additionally, this category covers completeness, ensuring that no data is maliciously ignored during computation.

Verifiability. Closely related to the previous category, this covers all aspects related to the quality of the computation result. That is, is it necessary that the receiver of a computation result obtains formal proofs that the computation was done correctly, or is the data processor trustworthy in the sense that is follows the protocol specification? Also, specific use cases may require public verifiability (or auditability), where also a third party is able to check the validity of the result.

Ease of Integration. This category covers all aspects related to available hardware and software systems, acceptable costs, or the availability of skilled personnel for design and maintenance of a system.

Regulatory Constraints. Finally, this class captures all known legal requirements, e.g., related to data protection or critical infrastructures that need to be fulfilled by any deployed system.

A qualitative mapping of these requirements to our use cases can also be found in Table 1.

3 TECHNOLOGIES

In Section 3.1 we will now briefly introduce the different approaches found in the literature for federated computing in general and for time series in particular. In Section 3.2 we discuss additional PPTs that in combination with these paradigms can be used to overcome existing limitations.

3.1 Secure Computing Paradigms

For each of the following technologies, we provide a short description of their properties, as well as indications regarding its maturity and practical applicability.

Table 1: Comparison of challenges for our selected use cases.

	UC1 Medical Research	UC2 Industry 4.0	UC3 Smart Buildings
Trust model	Data should not leave own facility	Central data processor for protected data acceptable	
Scalability	Specific computations per study	Medium to high frequency of same computation	
	small data sizes	large data sizes	medium data size
Efficiency	No specific requirements	Real-time needs for analysis	No specific requirements
Flexibility	Computations might not be known when generating data	Computations clear upon data creation	
Data integrity	Only authentic data should be included in computations; source privacy desirable		No specific needs
Verifiability	Verifiability of computation results desirable		No auditing needs
Ease of integration	No inherent hardware limitations, skilled personnel may be available		Strong limitations regarding deployment, integration, costs
Regulatory constraints	Strict	medium to strict (e.g. for critical infrastructure)	low to medium

3.1.1 Secure Multi-Party Computation

Secure multi-party computation (MPC) was first introduced by (Yao, 1982), subsequently generalized by (Goldreich et al., 1987) and further researched in a large body of work, cf., e.g., (Evans et al., 2018). It allows parties to collaboratively perform any joint computation on their respective private inputs, without revealing any other information to the other parties than what is revealed by the predefined partial outputs to the individual parties.

Properties. The privacy and integrity guarantees of MPC are based on a non-collusion assumption. That is, these properties are maintained as long as no more than a predefined fraction of nodes collude, while no guarantees can be given if too many nodes collude maliciously. In particular, distinctions are made depending on whether an honest majority is assumed or not. Furthermore, different security notions exist, depending on whether malicious nodes still follow the protocol specification ("passive security") or whether they may arbitrarily deviate ("active security"). As a rule of thumb, the efficiency of MPC protocols decreases as the attacker model becomes stronger, leading to a delicate balance between achieving the desired levels of security and efficiency.

However, despite its versatility, MPC also comes with several limitations. Specifically, while any computation on sensitive inputs can be performed using MPC, the structure of the computation to be performed strongly influences the efficiency of the resulting protocol. For instance, for protocols based on Shamir's secret sharing scheme (Shamir, 1979), additions are basically for free, while multiplications require complex, interactive operations among the nodes, potentially causing significant delays depending on network latency and bandwidth. Therefore, circuits with low multiplicative depth are preferable in this approach. Furthermore, the bandwidth requirements often heavily depend on the size of the secret inputs held by the different parties, making applications with small(er) input sizes more suitable for MPC than those with larger input sizes.

Finally, depending on the concrete setup, MPC can be a useful technical control to (partially) overcome the challenges imposed by data protection regulations (Helminger and Rechberger, 2022).

Maturity Level. MPC is a highly versatile tool that has been used to demonstrate a variety of practical application domains, ranging from secure auctions (Bogetoft et al., 2009) over industry 4.0 (Lorünser et al., 2022a) to finance (da Gama et al., 2022).

Usable frameworks supporting the development of MPC protocols exist, e.g., MP-SPDZ (Keller, 2020). Furthermore, several standards developing organizations (SDOs) have already published standards on MPC, including ISO/IEC (within its ISO/IEC 4922 series) and IEEE 2842-2021.

Guidelines and open challenges for the integration of MPCs into data infrastructures such as Data Spaces have recently been investigated by (Siska et al., 2024).

3.1.2 Fully Homomorphic Encryption

While in traditional encryption schemes basically the only thing that can be done with a ciphertext is to decrypt it, fully homomorphic encryption (FHE) allows one to perform arbitrary computations on ciphertexts in the encrypted domain. Thus, FHE ensures that the operations on the ciphertext correspond

to well-defined operations on the underlying plaintext data. This enables data owners, e.g., to encrypt their data before sending them to a powerful cloud service. The cloud provider may then perform computations (e.g., data analysis on sensitive data) on the ciphertexts without ever having access to the plaintext data. Eventually, the cloud provider will return the encrypted result to the holder of the secret key who only needs to decrypt the final result.

While first envisioned much earlier, the concept was first instantiated only in 2009 by (Gentry, 2009). It has been an active area of research ever since, resulting in significant efficiency improvements.

Properties. Evaluating functions using FHE is several orders of magnitude slower compared to a plaintext evaluation (Sidorov et al., 2022). In particular, after a certain number of operations (e.g., multiplications), highly expensive so-called "bootstrapping" steps need to be performed to guarantee valid decryptions. This is partially overcome by somewhat homomorphic encryption schemes (Brakerski, 2012), which support only a small number (e.g., one) of multiplications but arbitrarily many additions. They thereby avoid the need for bootstrapping steps while limiting the expressiveness of the schemes. Thus, such schemes can significantly improve efficiency in specific applications.

If used to analyze data coming from different data sources, special attention needs to be paid to the management of the master secret key *msk*, as this could be used not only to decrypt the computation results but also the encrypted inputs. Thus, in scenarios where the intended receiver of the computation result (owning *msk*) must not get access to the individual encrypted inputs, it needs to be ensured that the computing server, e.g., does not leak encrypted input data to the receiver. This can be achieved on an organization level, and by enforcing strict access policies. Alternatively, multikey-FHE schemes, first introduced by (López-Alt et al., 2012), are capable of computing on ciphertexts encrypted under multiple unrelated keys can be deployed, yet causing higher computational costs and additional complexity as all individual secret keys are required for decryption.

Finally, verifiability of the performed computation can be achieved by deploying specific schemes, e.g., (Viand et al., 2023).

Maturity Level. FHE is a very flexible technology that can be used to perform arbitrary computations on sensitive data, which has been proved in a variety of scenarios including federated learning (Zhao et al., 2024) or healthcare (Munjal and Bhatia, 2022).

Over the last years, FHE has made significant advancements regarding practicability, and a range of open-source frameworks are available to researchers and developers[5,6,7]. Finally, fully homomorphic encryption is subject to ongoing standardization efforts, e.g., by ISO/IEC (within the ISO/IEC 28033 series), ITU-T[8], and community standardization efforts[9]. Finally, in order to overcome efficiency limitations, substantial efforts are currently taken to provide hardware acceleration for FHE in the near future.[10][11]

3.1.3 Functional Encryption

Similar to FHE, functional encryption (FE) (Boneh et al., 2011) offers a more versatile approach than all-or-nothing access to encrypted data, by allowing the holder of the master secret key *msk* to generate partial decryption keys sk_f. Now, upon receiving a ciphertext that encrypts a message m, the holder of such a partial decryption key cannot recover the message itself, but only $f(m)$, where $f(\cdot)$ is a function defined by the holder of the master secret key. If the key is controlled, e.g., by the data subject, fine-grained access to computations on sensitive data can be provided.

Several extensions to increase functionality or lower trust assumptions are available in the literature.

Properties. While requiring proper protection of the master secret key *msk* to ensure confidentiality of all data, functional encryption provides an interesting trust model, as the decryption – and thus the computation – is directly carried out by the receiving party, reducing the need for verifiability.

On the downside, FE is only meaningful in scenarios where the function to be computed is known in advance, as the schemes need to support the right class of functions. In particular, most efficient FE schemes are focusing on the computation of inner products (including, e.g., weighted sums of data items).

Maturity Level. Functional encryption has been proven useful in different contexts such as, e.g., machine learning (Panzade et al., 2024). Libraries for FE are available in different languages through, e.g., the FENTEC project[12]. To the best of our knowledge, there are no ongoing efforts for standardizing functional encryption.

[5] https://www.zama.ai/

[6] https://github.com/homenc/HElib

[7] https://www.openfhe.org/

[8] https://www.itu.int/ITU-T/workprog/wp_item.aspx?isn=17999

[9] https://homomorphicencryption.org/

[10] https://www.darpa.mil//news-events/2021-03-08

[11] https://dualitytech.com/partners/intel/

[12] https://github.com/fentec-project

3.1.4 Trusted Execution Environments

Trusted Execution Environments (TEEs) are secure areas within a processor designed to protect sensitive code and data from unauthorized access or modification by the operating system or other applications (Sabt et al., 2015). TEEs function by providing isolated execution spaces that operate alongside the main operating system, ensuring an additional layer of security for sensitive processes. In practice, TEEs work by offering a restricted processing environment where only authorized code can execute, and memory is encrypted to prevent external inspection or tampering. This is critical for handling operations such as cryptographic key management, biometric data processing, and secure financial transactions (Lind et al., 2016). When an application needs to perform a secure operation, it makes a call to the TEE, which processes the request in isolation, safeguarding the confidentiality and integrity of the data being handled. This makes TEEs vital for enforcing privacy and security in a broad range of digital applications.

Properties. This is also essential for compliance with stringent data protection regulations in industries like finance and healthcare. Additionally, TEEs help in mitigating a range of security threats, including malware and software vulnerabilities. Their ability to handle secure transactions and authenticate user interactions in a protected manner makes them invaluable for mobile banking, DRM (Digital Rights Management), and personal identification applications. Also the integrity and authenticity of code execution can be reliably verified (Chen and Zhang, 2023). Overall, TEEs strengthen the trustworthiness of devices and systems, increasing user confidence in digital platforms. TEEs face several challenges that can impact their broader application and effectiveness. One of the primary hurdles is hardware dependency, as TEEs require specific processor features to function, which can limit their use to certain devices and platforms, thus restricting scalability (Geppert et al., 2022). Integrating TEEs into existing systems also poses significant challenges due to the need for specialized knowledge to handle secure communication and operation management between the TEE and the rest of the system. This process is further complicated, as any code loaded into secure enclaves needs to be signed using tools provided by the TEE manufacturer to ensure security, making quick adaptations of the code complicated. Moreover, the complexity of developing and maintaining secure code that runs within TEEs increases the risk of vulnerabilities if not properly managed. Another main point to consider are strong trust assumptions that need to be put into the trustworthiness of the hardware manufacturer.

Additionally, cross-platform compatibility issues can arise, complicating the development of applications that can operate across different types of devices and operating systems with TEEs.

Finally, the evolving landscape of cybersecurity threats continuously tests the resilience of TEEs, necessitating ongoing updates and patches to maintain a robust defense. While this is true for any security-related implementation – e.g., considering side-channel attacks –, this issue requires further attention due to the required trust in hardware components: in case that an attack exploits hardware vulnerabilities, patches may be more complex and cost intensive than in the case of pure software applications.

Maturity Level. TEEs are provided by several companies, often integrated into their hardware or software products, including, e.g., Intel's Software Guard Extensions (SGX), AMD Secure Encrypted Virtualization (SEV), ARM TrustZone, or IBM Secure Service Container, just to name a few.

The standardization of TEEs has been primarily driven by the GlobalPlatform organization[13].

3.1.5 Federated Learning

Federated Learning (FL) is a collaborative machine learning approach, first introduced by (McMahan et al., 2017). It enables model training without centralizing data, thus preserving the privacy and security (Konečný et al., 2016) of training data. Therefore, it allows organizations to jointly develop models without sharing sensitive data, fostering collaboration and building more generalizable models (Kiss, 2022). Instead of pooling data centrally, each participant locally trains a model using their private data and then sends only the model updates (e.g., gradients) to a central server. This server aggregates the updates to improve a global model, which is then sent back to the participants for further training. This cycle repeats, enhancing the global model with every step.

To avoid, e.g., model inversion or model inference attacks, typically a trusted server is assumed for aggregation, but also secure aggregation protocols exist (Bonawitz et al., 2017).

Properties. Federated learning offers significant advantages, chiefly in privacy preservation, as it allows data to remain on local devices, reducing the risk of data breaches (Li et al., 2022). Furthermore, the data security of FL is usually supplemented with differential privacy to further protect potentially sensitive training data (Wei et al., 2020). This decentralized approach also enhances scalability by distributing com-

[13]https://globalplatform.org/

putation across numerous devices, thus avoiding centralized processing bottlenecks. Additionally, it supports data diversity, reflecting real-world variations, and improves model robustness. By leveraging multiple data sources without direct access, it complies with data protection regulations, making it suitable for industries like healthcare and finance where data sensitivity is paramount. Overall, federated learning enables more secure and scalable machine learning deployments (Zhang et al., 2021).

Despite its advantages, FL also faces several limitations. In the case of non-iid data distributions (data points are not independently and identically distributed, but may, e.g., be correlated), the 'client-drift' problem may cause clients to converge to local optima, leading to slower overall convergence and poor model performance (Moshawrab et al., 2023). Another limitation lies in the difficulty of determining hyperparameter values, hindering the global model's performance in each update; a proposed solution involves dynamically adjusting hyperparameters using a log function to address this issue.

Integration of FL into complex systems comes with a variety of challenges. Varying computational capabilities of participant devices, the complexity of establishing reliable communication channels, or the need to adapt existing systems to handle decentralized data processing need to be addressed. At the same time, the consistent and secure participation of joining and exiting nodes in the network must be addressed (Moshawrab et al., 2023).

Maturity Level. Federated learning is a rapidly advancing technique, which has found practical applications in diverse domains such as healthcare (Muazu et al., 2024) or finance (Shi et al., 2023).

Usable frameworks and platforms, such as TensorFlow Federated[14] and PySyft[15] (Ziller et al., 2021), support the development and deployment of federated learning protocols. Finally, several SDOs are actively working on federated learning standards, such as IEEE P3652.1.

3.2 Extensions

In the following, we give a concise overview of possible extensions that may be integrated to achieve all the requirements identified in Table 1.

3.2.1 Advanced Signature Schemes

As described in Section 2, data integrity and authenticity are crucial, especially for sensitive decisions. In the medical domain, when data is patient-generated, it is important to ensure only authentic data from genuine devices is used, reducing the risk of manipulation. For example, a diabetes study should only rely on data from verified glucometers.

However, signing data directly poses privacy risks, as it could link measurements to individuals. This can be mitigated with group signatures (Chaum and van Heyst, 1991), where a group manager (e.g., a device manufacturer) issues unique signing keys to users. A user signs data (e.g., encrypted blood sugar readings), and the verifier (e.g., a hospital) checks authenticity without identifying the specific user. If necessary, a third party (e.g., an ethics board) can reveal identities in cases of abuse.

For more ad-hoc scenarios, ring signatures (Rivest et al., 2001) provide similar privacy protections and are widely used in cryptocurrencies.

Maturity Level. All the aforementioned technologies are highly mature and have been included into relevant standards, or are currently subject to standardization, e.g., ISO/IEC 20008. Practical applications of group signatures include, e.g., direct anonymous attestation (DAA) implemented in Trusted Platform Modules (TPMs) as specified by the Trusted Computing Group.

3.2.2 Zero-Knowledge Proofs

Most computing paradigms in Section 3.1 lack built-in verifiability. This can be achieved using zero-knowledge proofs (ZKPs) (Goldwasser et al., 1985), including compact variants like so-called zk-SNARKs (Groth, 2016). These protocols allow a prover to demonstrate knowledge of information without revealing more than the claim itself.

In MPC, compute nodes could generate a zero-knowledge proof verifying that input data was (i) signed by providers, (ii) used in the computation, and (iii) the output is correct. This has been applied in smart manufacturing (Lorünser et al., 2022a) and air traffic management (Lorünser et al., 2022b).

A key application of ZKPs is authentic neural network inference. For instance, a service provider may want to protect a proprietary and certified model while proving to customers it was used for inference. ZKPs enable this without exposing the model. A step toward practical zero-knowledge inference is zkCNN (Liu et al., 2021).

Maturity Level. ZKPs have reached a relatively mature stage, with several practical implementations and real-world applications, including, e.g., private transactions in blockchains[16]. Companies like StarkWare,

[14] https://www.tensorflow.org/federated

[15] https://github.com/OpenMined/PySyft

[16] https://z.cash/

zkSync, and Aztec are leading in practical implementations of ZKPs. Standardization of zero-knowledge proofs is advancing, driven, e.g., by community standardization efforts such as ZKProof.

3.2.3 Advanced Encryption Mechanisms

Consider a system where computations are performed periodically, and users subscribe to the results. This creates asynchronous scenarios where eligible users may not be online at computation time, requiring dynamic access control. Traditional methods rely on a central policy-enforcement point, storing results in plaintext, which raises data protection concerns. Encrypting results separately for each user using public-key encryption is also impractical for large user bases.

A more scalable approach is attribute-based encryption (ABE) (Sahai and Waters, 2005), where users receive secret keys tied to attributes (e.g., subscription validity). Instead of directly outputting results, compute nodes encrypt under a central key with an access policy (e.g., creation date). While ciphertexts are publicly available, only users with matching keys can decrypt them.

If recipients are known but offline, or if the data receiver in an FHE scenario differs from the holder of *msk*, proxy re-encryption (Blaze et al., 1998) can convert ciphertexts for the intended recipient without exposing plaintexts.

These methods enforce access control cryptographically, ensuring plaintext results remain inaccessible to unauthorized users.

Maturity Level. The maturity of ABE is increasing, with growing interest and several practical implementations, but not yet widespread adoption. Yet, several companies like Virtru[17] implement ABE in their security solutions. Furthermore, several open-source implementations are available (Mosteiro-Sanchez et al., 2022). Standardization efforts are ongoing, including, e.g., ETSI's TS 103 532.

4 DISCUSSION

Based on the descriptions in Section 3.1, it can be seen that the available computing paradigms address the requirements set out in Table 1 in highly orthogonal ways, and that some of those requirements are already effectively addressed by existing technologies. We provide an overview of the properties of the computing paradigms in Table 2.

For instance, for the underlying trust model, a broad range of options is available, ranging from non-collusion assumptions in keyless and information-theoretically secure settings (MPC) over computational assumptions provided by encryption schemes with central key material (FHE, FE), up to hardware-based trust anchors (TEE). Regarding public verifiability and integrity of the computation result, TEEs and MPC can already provide certain guarantees by ensuring that computations on sensitive data are executed securely and correctly. For scalability, especially FL and TEEs provide efficient solutions that maintain reasonable performance levels, with FL excelling at distributing the computational load to address scalability concerns. In addition, FL and TEEs offer considerable flexibility and ease of integration, making them suitable for diverse applications.

Some requirements cannot yet be fully met by existing solutions. For example, while FHE and MPC offer strong privacy guarantees, they struggle with efficiently scaling large datasets due to high computational and communication overheads. These technologies need further efficiency improvements to be practical for real-time use. Ensuring the verifiability of computations without compromising privacy remains a challenge, requiring stronger, more practical mechanisms for approaches like FE or FL. Although flexible, FL needs more development to handle heterogeneous data sources and complex time series structures. Most technologies also require skilled personnel for initial setup. Infrastructure needs are low when deployed as a service, but can rise significantly if computations are done locally, increasing data sovereignty.

Finally, certain requirements are not immediately addressed by any of the technologies. For instance, none of the computing paradigms inherently supports data integrity validation, but they all assume valid input data. For all solutions, a legal analysis must be carried out on a case-by-case basis, considering factors such as data criticality or additional safeguards.

Finally, for many of the paradigm-specific or general limitations, additional compatible PPTs exist, as also discussed in Section 3.2.

Based on the findings presented in the previous sections, Figure 1 finally provides a decision diagram supporting developers in efficiently identifying suitable candidates for PPTs in the context of federated time series analysis. While it is not possible to give conclusive recommendations for every scenario, we believe that this diagram can provide valuable support in identifying a starting point for further investigations and analysis.

[17] https://www.virtru.com/

Table 2: Comparison of guarantees offered by different computing paradigms.

	MPC	FHE	FE	FL	TEE
Trust model	Non-collusion assumptions; support of active and passive adversary models	Computations performed by untrusted data processor; confidentiality depends on proper management of secret key	Computations performed by data receiver; requires proper protection of master secret key	Trust in data owners to use righteous data; requires aggregator to protect raw data	Hardware-based trust anchor; TEE provider needs to be trusted
Scalability	Well suited for small to medium data sizes			Well suited also for large data sets	
Efficiency	Communicational costs increase with the complexity of the circuit to be evaluated	Non-interactive. Computational costs increase with complexity of the circuit	Non-interactive. Costs depend on computation to be computed and scenario (input from one or multiple parties, etc.)	Computational costs decreased through local lightweight models but increased communicational costs	Moderate overhead compared to plaintext computation
Flexibility	Support of arbitrary functions; complexity increases with multiplicative depth of circuit		Limited, as type/class of computations need to be known upfront	Support of a broad range of FL methods; increased communicational costs with increased complexity of functions	Support of arbitrary functions; complex administrative processes by TEE provider
Data integrity	None by default, cf. Section 3.2				
Verifiability	Resilience against a certain number of malicious nodes can be achieved. Publicy verifiable protocols exist.	Not by default, but specific verifiable FHE schemes exist	None by default, yet computation can be carried out be data receiver, reducing the need for verifiability	None by default, but impact of data quality traceable	Integrity based on the trust assumptions into the secure enclave
Ease of integration	If hosted on-premise, computing infrastructure and trained personnel required.	Low integration costs if used as-a-service, initial setup may require trained personnel. Key management may require organizational processes.	Low to moderate. Key management may require organizational processes.	Complex Integration due to setting up infrastructure between nodes	Trained personnel needed in development phase; complex organizational processes to certify code
Regulatory constraints	Requires analysis on a case-by-case basis				

5 CONCLUSION

This research highlights the critical need for robust PPTs in the federated analysis of sensitive time-series data across different domains. Each considered domain presents unique challenges and requirements. Our evaluation of advanced PPTs and selected extensions reveals their respective strengths and limitations in meeting these requirements. Furthermore, given the complex state of affairs for PPTs for time series, we provide developers with guidance to identify candidate PPTs for specific scenarios. We consider this an important step towards making those tools accessible also to engineers. At the same time we acknowledge the need for further evaluation and higher granularity, which may be achieved in close collaboration with stakeholders from different domains.

We recognize several research gaps, including the need for more efficient algorithms for FHE and MPC to reduce computational and communication overheads. Improving the verifiability of computations in privacy-preserving settings is crucial to ensure that results can be independently verified without compro-

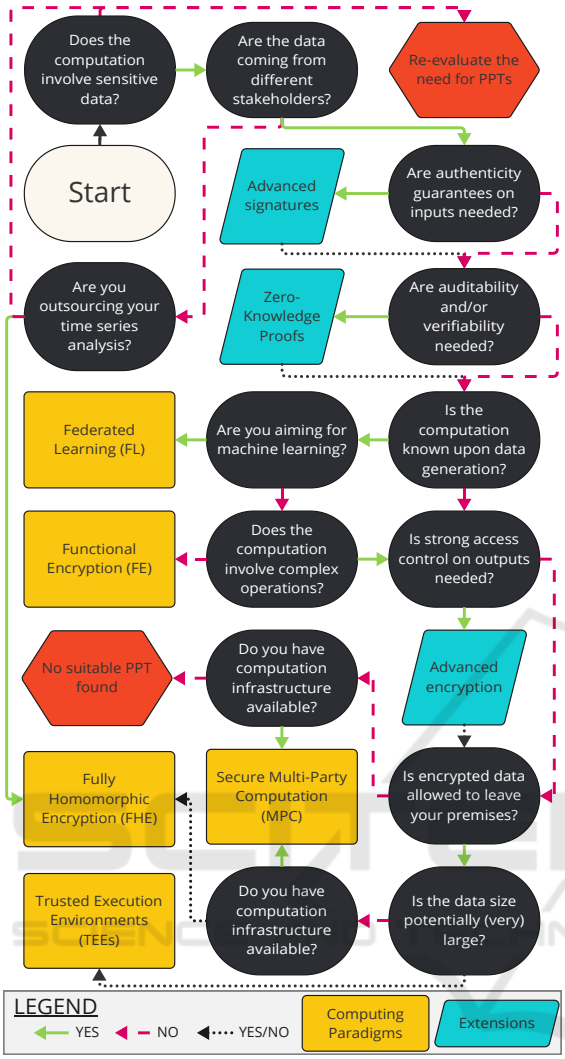

Figure 1: Decision tree supporting the selection of appropriate PPTs for federated time series analysis.

mising privacy. Improving the interoperability of different PPTs is essential to facilitate seamless integration and transition between methods as required by specific use cases. In addition, the development of optimized solutions tailored to the specific needs of each domain, especially when dealing with time series data, is of paramount importance.

Future research should focus on algorithmic innovation, efficiency benchmarking, cross-technology integration, domain-specific solutions, and regulatory alignment. Investment in the development of new algorithms and protocols to improve FHE and MPC efficiency and scalability is crucial. Benchmarking frameworks to evaluate PPT performance in real-world scenarios will provide additional insights. Integrating multiple PPTs to leverage strengths and mitigate weaknesses will enhance their practical applicability. Tailored solutions for specific requirements will be needed to address domain-specific needs. Engaging with regulators to ensure alignment with evolving data protection regulations will be essential for adoption.

Addressing these research gaps and pursuing the proposed agenda will lead to significant progress in privacy-preserving time series analysis. Ultimately, these advances will enable the secure and efficient analysis of sensitive data across critical domains.

ACKNOWLEDGEMENTS

This work was funded by the Austrian Research Promotion Agency FFG within the PRESENT project (grant no. 899544), and the CHIST-ERA project REMINDER through the Austrian Science Fund (FWF): I 6650-N. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the funding agency. Neither the FFG nor the granting authority can be held responsible for them.

REFERENCES

Bachlechner, D., La Fors, K., and Sears, A. M. (2018). The role of privacy-preserving technologies in the age of big data. In *WISP 2018*.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In Nyberg, K., editor, *EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127–144. Springer.

Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T. P., Krøigaard, M., Nielsen, J. D., Nielsen, J. B., Nielsen, K., Pagter, J., Schwartzbach, M. I., and Toft, T. (2009). Secure multiparty computation goes live. In Dingledine, R. and Golle, P., editors, *FC 2009*, volume 5628 of *LNCS*, pages 325–343. Springer.

Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In Thuraisingham, B., Evans, D., Malkin, T., and Xu, D., editors, *ACM CCS 2017*, pages 1175–1191. ACM.

Boneh, D., Sahai, A., and Waters, B. (2011). Functional encryption: Definitions and challenges. In Ishai, Y., editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer.

Brakerski, Z. (2012). Fully homomorphic encryption without modulus switching from classical gapsvp. In Safavi-Naini, R. and Canetti, R., editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer.

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer.

Chaum, D. and van Heyst, E. (1991). Group signatures. In Davies, D. W., editor, *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer.

Chen, G. and Zhang, Y. (2023). Securing tees with verifiable execution contracts. *IEEE TDSC*, 20(4):3222–3237.

Chiang, H.-L., Chia, Y.-Y., Linand, H.-S., and Chen, C.-H. (2016). The Implications of Tobacco Smoking on Acute Postoperative Pain: A Prospective Observational Study. *Pain Research and Management*, 2016:9432493.

da Gama, M. B., Cartlidge, J., Polychroniadou, A., Smart, N. P., and Alaoui, Y. T. (2022). Kicking-the-bucket: Fast privacy-preserving trading using buckets. In Eyal, I. and Garay, J. A., editors, *FC 2022*, volume 13411 of *LNCS*, pages 20–37. Springer.

Evans, D., Kolesnikov, V., and Rosulek, M. (2018). A pragmatic introduction to secure multi-party computation. *Found. Trends Priv. Secur.*, 2(2-3):70–246.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Mitzenmacher, M., editor, *STOC 2009*, pages 169–178. ACM.

Geppert, T., Deml, S., Sturzenegger, D., and Ebert, N. (2022). Trusted execution environments: Applications and organizational challenges. *Frontiers Comput. Sci.*, 4.

Goldreich, O., Micali, S., and Wigderson, A. (1987). How to play any mental game or A completeness theorem for protocols with honest majority. In Aho, A. V., editor, *STOC 1987*, pages 218–229. ACM.

Goldwasser, S., Micali, S., and Rackoff, C. (1985). The knowledge complexity of interactive proof-systems (extended abstract). In Sedgewick, R., editor, *STOC 1985*, pages 291–304. ACM.

Groth, J. (2016). On the size of pairing-based non-interactive arguments. In Fischlin, M. and Coron, J., editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer.

Helminger, L. and Rechberger, C. (2022). Multi-party computation in the gdpr. In Schiffner, S., Ziegler, S., and Quesada Rodriguez, A., editors, *Privacy Symposium 2022*, pages 21–39. Springer.

Jordan, S., Fontaine, C., and Hendricks-Sturrup, R. (2022). Selecting privacy-enhancing technologies for managing health data use. *Front. Public Health*, 10:814163.

Keller, M. (2020). MP-SPDZ: A versatile framework for multi-party computation. In Ligatti, J., Ou, X., Katz, J., and Vigna, G., editors, *ACM CCS 2020*, pages 1575–1590. ACM.

Kiss, P. (2022). *Federated Learning of Artificial Neural Networks*.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492.

Li, J., Li, X., and Zhang, C. (2022). Analysis on security and privacy-preserving in federated learning. *Highlights in Science, Engineering and Technology*, 4:349–358.

Lind, J., Eyal, I., Pietzuch, P. R., and Sirer, E. G. (2016). Teechan: Payment channels using trusted execution environments. *CoRR*, abs/1612.07766.

Liu, T., Xie, X., and Zhang, Y. (2021). zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In Kim, Y., Kim, J., Vigna, G., and Shi, E., editors, *ACM CCS 2021*, pages 2968–2985. ACM.

López-Alt, A., Tromer, E., and Vaikuntanathan, V. (2012). On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Karloff, H. J. and Pitassi, T., editors, *STOC 2012*, pages 1219–1234. ACM.

Lorünser, T., Wohner, F., and Krenn, S. (2022a). A privacy-preserving auction platform with public verifiability for smart manufacturing. In Mori, P., Lenzini, G., and Furnell, S., editors, *ICISSP 2022*, pages 637–647. SCITEPRESS.

Lorünser, T., Wohner, F., and Krenn, S. (2022b). A verifiable multiparty computation solver for the linear assignment problem: And applications to air traffic management. In Regazzoni, F. and van Dijk, M., editors, *CCSW 2022*, pages 41–51. ACM.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In Singh, A. and Zhu, X. J., editors, *AISTATS 2017*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

Moshawrab, M., Adda, M., Bouzouane, A., Ibrahim, H., and Raad, A. (2023). Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives. *Electronics*, 12(10):2287.

Mosteiro-Sanchez, A., Barcelo, M., Astorga, J., and Urbieta, A. (2022). Too many options: A survey of ABE libraries for developers. *CoRR*, abs/2209.12742.

Muazu, T., Mao, Y., Muhammad, A. U., Ibrahim, M., Kumshe, U. M. M., and Samuel, O. (2024). A federated learning system with data fusion for healthcare using multi-party computation and additive secret sharing. *Comput. Commun.*, 216:168–182.

Munjal, K. and Bhatia, R. (2022). A systematic review of homomorphic encryption and its contributions in healthcare industry. *Complex Intell Systems*, 3:1–28.

Panzade, P., Takabi, D., and Cai, Z. (2024). Privacy-preserving machine learning using functional encryption: Opportunities and challenges. *IEEE Internet Things J.*, 11(5):7436–7446.

Rivest, R. L., Shamir, A., and Tauman, Y. (2001). How to leak a secret. In Boyd, C., editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer.

Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, pages 57–64. IEEE.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In Cramer, R., editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22(11):612–613.

Shi, Y., Song, H., and Xu, J. (2023). Responsible and effective federated learning in financial services: A comprehensive survey. In *CDC 2023*, pages 4229–4236. IEEE.

Sidorov, V., Wei, E. Y. F., and Ng, W. K. (2022). Comprehensive performance analysis of homomorphic cryptosystems for practical data processing. *CoRR*, abs/2202.02960.

Sielaff, L., Hetfleisch, R., and Rader, M. (2023). Evaluation framework for the use of privacy preserving technologies for production data. In *ICAT 2023*, volume 11, pages 157–163.

Siska, V., Lorünser, T., Krenn, S., and Fabianek, C. (2024). Integrating secure multiparty computation into data spaces. In van Steen, M. and Pahl, C., editors, *CLOSER 2024*, pages 346–357. SCITEPRESS.

United Nations (2023). United nations guide on privacy-enhancing technologies for official statistics. United Nations Committee of Experts on Big Data and Data Science for Official Statistics, New York.

Viand, A., Knabenhans, C., and Hithnawi, A. (2023). Verifiable fully homomorphic encryption. *CoRR*, abs/2301.07041.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Vincent Poor, H. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE TIFS*, 15:3454–3469.

Yao, A. C. (1982). Protocols for secure computations (extended abstract). In *FOCS 1982*, pages 160–164. IEEE Computer Society.

Zhang, M., Wei, E., and Berry, R. (2021). Faithful edge federated learning: Scalability and privacy. *IEEE Journal on Selected Areas in Communications*, 39(12):3790–3804.

Zhao, Y., Zhou, H., and Wan, Z. (2024). Superfl: Privacy-preserving federated learning with efficiency and robustness. *IACR Cryptol. ePrint Arch.*, page 81.

Ziller, A., Trask, A., Lopardo, A., Szymkow, B., Wagner, B., Bluemke, E., Nounahon, J.-M., Passerat-Palmbach, J., Prakash, K., Rose, N., Ryffel, T., Reza, Z. N., and Kaissis, G. (2021). *PySyft: A Library for Easy Federated Learning*, pages 111–139. Springer.