TrustFLow: A traceable federated learning framework to enable trustworthy digital twins

Highlights

TrustFLow: a Traceable Federated Learning Framework to Enable Trustworthy Digital Twins

Nicolò Romandini,Andrea Roberta Costagliola,Armir Bujari,Rebecca Montanari

- Comprehensive tracking system using Decentralized Identifiers and Verifiable Credentials to ensure Federated Learning (FL) process governance.
- Robust influence estimation to quantify the trustworthiness of clients and data contributions.
- Automated policy-driven revocation mechanism for models and data based on established open standards.
- Seamless integration into FL–based architectures, enabling the development of transparent and trustworthy models supporting Digital Twin functions.
- Extensive experiments conducted on a real prototype across diverse settings demonstrate the effectiveness and efficiency of the solution.

TrustFLow: a Traceable Federated Learning Framework to Enable Trustworthy Digital Twins

Nicolò Romandini*, Andrea Roberta Costagliola, Armir Bujari and Rebecca Montanari

Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

ARTICLE INFO

Keywords:
Trustworthy Digital Twins
Federated Learning
Decentralized Identifiers
Distributed Ledger
Traceability
Influence Estimation

ABSTRACT

As Digital Twin (DT) ecosystems increasingly rely on distributed and collaborative intelligence, ensuring trust and reliability of the underlying Machine Learning (ML) models becomes critical, especially when raw data cannot be centrally aggregated due to privacy or security concerns. Federated Learning (FL) has emerged as a promising paradigm for collaborative model training across distributed data sources while preserving data privacy. However, the inherent opacity of the FL process introduces several challenges; individual data contributions, as well as local node updates, remain inaccessible to centralized oversight, hindering the overall trustworthiness of the global model training process. In addition, data and updates of the local model can be biased, or even maliciously modified, negatively affecting the global model. In the context of DTs, such vulnerabilities can directly compromise decision-making, leading to operational safety risks and affecting the system's reliability. To this end, we propose TrustFLow, a comprehensive framework that integrates Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) deployed over a distributed ledger infrastructure to securely trace the provenance of data and evolution of models in FL. TrustFLow tracking capabilities enable linking data to sources, tamper-proof monitoring, and process traceability. In addition, TrustFLow provides crucial functionalities for estimating the influence of individual producers (datasets) on the global model and for revoking both biased data and the global model(s) influenced by them. These features contribute to a broader vision of federated governance, where accountability, transparency, and trust are enforced between all participants. In addition, our framework offers the benefit of seamless integration into any FL-driven DT architecture. To evaluate our proposal, we have conducted an extensive set of experiments that measure the efficiency and effectiveness of the framework under different settings.

1. Introduction

The growing availability of data generated at the edge is driving the development of intelligent and adaptive digital infrastructures (1). This trend has accelerated the adoption of Digital Twins (DTs), which leverage continuous data flows for real-time analysis and decision-making to optimize system performance. As the volume of data and their sensitivity increase, concerns related to privacy, bandwidth, and regulatory compliance have become more pressing. In this context, Federated Learning (FL) (2) has emerged as an effective approach to train collaborative models between distributed data sources, avoiding the need for raw data sharing. The adoption of FL within the broader context of Industrial Internet of Things (IIoT), as illustrated in Figure 1, allows for scalable, privacy-preserving intelligence while preserving the fidelity and adaptability required by these systems, provided that certain requirements are met (3; 4; 5). In fact, the shift from centralized to distributed learning introduces new challenges in trust, accountability, and system robustness (6). Faulty, biased, or malicious updates can compromise FL models, leading to inaccurate or unsafe decisions. Therefore, addressing these risks is essential to ensure the trustworthiness, integrity, and reliability of FL-driven DTs and to fully realize their potential. However, current FL frameworks lack robust mechanisms to track training data, validate updates, or manage trust between entities. Essential metadata, such as the origin of the dataset or its influence on the global model, is typically missing. Therefore, it is challenging to identify which local updates contributed to a biased model or to revoke global models affected by flawed data. Addressing these gaps is key to enabling reliable and trustworthy FL deployments, especially in sensitive or safety-critical DTs deployments.

To clarify the target scenario and better illustrate our work, consider a DT use case (7; 8) where different manufacturing sites or machines may produce sensor data with proprietary and privacy-sensitive information. These sensors collect process data to train a FL model to predict equipment failures, detect anomalies, and optimize maintenance schedules without sharing sensitive raw data. In this scenario, each factory acts as a data producer, training the model locally or delegating the task to a third-party technology provider. A central server, operated by an industrial consortium or a shared research platform, aggregates model updates to create a global model. However, data used to train local models might be flawed or biased due to sensor malfunctions, incorrect calibration, or even deliberate tampering. If such flawed data influence the global model, it may lead to inaccurate maintenance predictions, causing unnecessary downtime and financial losses, or undetected failures that result in costly breakdowns (9). In such scenarios, adopting

*Corresponding author
nicolo.romandini@unibo.it (N. Romandini); andrea.costagliola@unibo.it (A.R. Costagliola); armir.bujari@unibo.it (A. Bujari); rebecca.montanari@unibo.it (R. Montanari)
ORCID(s): 0000-0002-2820-5978 (N. Romandini); 0009-0009-0119-2597 (A.R. Costagliola); 0000-0002-1955-7699 (A. Bujari); 0000-0003-1940-6804 (R. Montanari)

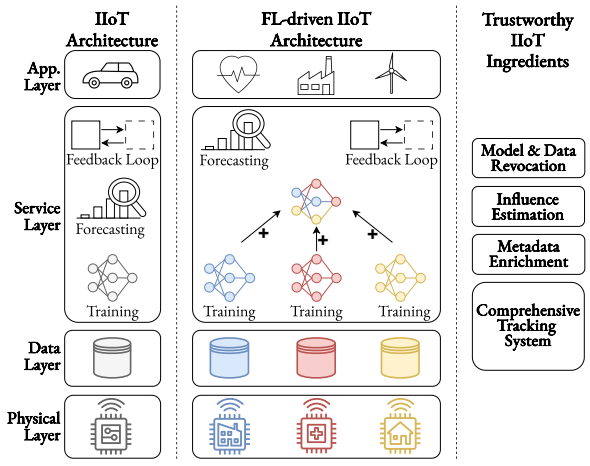

Figure 1: Bird's eye view of the FL-driven service architecture, showing both a (simplified) IIoT architecture and the key ingredients for building a trustworthy DT. At the top lies the Application Layer, which includes various verticals (e.g., industry, smart cities, healthcare, etc.) that leverage the functionalities provided by the Service Layer to obtain relevant insights about the monitored entity and potentially trigger adaptations to the process or system. The Service Layer, in turn, depends on the Data Layer to access real-time data streams that mirror the behavior of the process or system. These data streams are used to train data-driven models and pipelines.

a framework that incorporates robust tracking mechanisms, enabling the identification and isolation of flawed contributions, becomes paramount. This ensures that biased updates can be traced and the validity of derived models can be revoked. Such solutions are essential not only for industrial predictive maintenance but also for other domains, such as healthcare and smart cities (10). In the latter, subsystems such as traffic control, energy distribution, and environmental monitoring act as data producers, contributing local models to a city-wide DT. Faulty or adversarial updates may compromise planning and resource allocation, making robust tracking and revocation mechanisms necessary to safeguard the reliability of urban management and to incentivize participation (11).

To address these critical challenges, we propose TrustFLow, an innovative framework that provides end-to-end traceability of data-driven processes, a key enabler for reliable and transparent DTs. As part of its support for federated process governance, our framework integrates advanced tracking capabilities based on Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) enabled by Distributed Ledger Technologies (DLTs) to link data, datasets, or models to sources. As a key feature, TrustFLow also provides crucial functionalities to estimate the influence of individual datasets on the global model and biased global model revocation. These mechanisms ensure that problematic datasets can be identified and their impact on the global model mitigated, enhancing trust, transparency, and accountability in FL systems. An important aspect of our framework is that it can be seamlessly integrated into any FL-driven DT. Its design ensures compatibility with existing architectures, enabling large-scale and effective adoption without significant modifications to the underlying process.

The contributions of this work are as follows:

- **Comprehensive Tracking System**: We propose a DLT integrated system to trace all the data, updates, models, and actors involved in the FL process using DIDs, saved on the DLT, and VCs. DIDs uniquely identify not only the entities in the FL process, such as data producers, clients performing local training, and the server, but also the artifacts of the process, including the datasets, the client updates, and global models.
- **Metadata Enrichment**: The system enriches the information generated with crucial metadata. For example, data producers can specify the steps or policies followed during data collection or generation as metadata. Similarly, we monitor *influence*, indicating how local client data impact the global model. The metadata are essential for understanding the overall quality and trustworthiness of the FL process.
- **Influence Estimation**: We adapt a state-of-the-art algorithm (12), originally developed to calculate the influence of data in a centralized setting, for effective use within a FL framework. Depending on the degree of influence of the data, our approach can be employed to determine whether to revoke both the data that adversely affect the global models and the models themselves.
- **Data and Model Revocation**: We present an automated mechanism for revoking data and global models. This revocation process is customizable through specific policies and leverages associated metadata. The process is automatically managed through a parametric smart contract deployed on the DLT.
- **Experimental Evaluation:** We implement our solution and rigorously evaluate its effectiveness and efficiency through extensive experiments, showing that the introduced overhead is minimal and the revocation mechanism executes rapidly.

The remainder of the article is organized as follows: Section 2 provides the technical background and key concepts necessary to understand the problem domain. Section 3 highlights the current state of the art, explaining how our solution differs from and improves upon existing approaches. Section 4 presents our proposed framework in detail, outlining its architecture and functionalities. Section 5 describes the experiments conducted to validate our approach, including the setup, results, and a discussion of the findings. Finally, Section 6 draws conclusions and highlights directions for future work.

2. Background

This section provides the necessary background to support the core argument of this work, which focuses on how FL and decentralized identity mechanisms address the challenges of trust and traceability within DT ecosystems.

2.1. Federated Learning

FL is a distributed Machine Learning (ML) framework that enables multiple clients to collaboratively train a shared ML model without directly exchanging their data (13). The paradigm ensures that the raw data remain local on each client, and only model updates or gradients are transmitted to a central server for aggregation. This approach addresses privacy concerns, reduces communication costs, and enables collaborative learning across geographically distributed data sources (14). A simple but effective aggregation algorithm in FL is Federated Stochastic Gradient Descent (FedSGD) (2). In this approach, the server initializes the global model parameters θ^t at the start of round t. Each participating client k computes the gradient of its local loss function, $\nabla L(\mathcal{D}_k, \theta^t)$, based on its local dataset $\mathcal{D}_k$. The server aggregates the gradients from clients using a weighted average based on their dataset sizes, and the result is used to update the global model's weights using the standard stochastic gradient descent algorithm. This iterative process ensures that the global model updates reflect the contributions of all clients. However, a problematic update could negatively impact the global model, leading to poor performance or even degradation of the model's accuracy. Since updates from multiple clients are aggregated, a single biased or erroneous update can skew the global model parameters, especially when the update is from a client with faulty or malicious data. The misbehavior of the potential client highlights the importance of improving the traceability and observability of the FL training process. Tracking individual clients, estimating the influence of both clients in providing their local update models and of data exploited to train models, is crucial to understanding the contributions to the global model, thus increasing the quality, reliability, and explainability of the process. Despite various solutions that have been proposed to address the specific issue of client influence, many existing FL frameworks overlook and do not adequately incorporate influence estimation as a building block.

In the context of DT ecosystems, where accurate and reliable models are critical for real-time decision making, we consider that the estimation of the influence of clients on the process is paramount. For this reason, our framework integrates this capability as a core part of the training process, supporting the estimation of client-level and data-level influence.

2.2. Decentralized Identifiers, Verifiable Credentials, and VC Revocation Mechanisms

DIDs (15) are a new type of identifier proposed by the W3C to enable verifiable, decentralized digital identities. A DID can represent any subject, such as a person, organization, or device, as defined by its controller. Unlike federated identifiers, DIDs operate independently of centralized authorities, including identity providers and certificate authorities, making them well-suited for distributed environments. DIDs are typically stored in distributed ledgers such as blockchains and, therefore, are not controlled by a single entity. Technically, DIDs are Uniform Resource Identifiers (URIs) that associate a DID subject to a DID document, facilitating trusted interactions. These URIs comprise three parts: the DID URI scheme identifier, a DID method identifier, and a method-specific identifier. When resolved, the URI links to a specific DID document containing cryptographic materials, verification methods, and services that enable a DID controller to prove control over the DID. They can be linked to various digital credentials, including VCs and Verifiable Presentations (VPs) (16), which can be easily shared and verified without relying on a third party. VCs are digital certificates that assert specific attributes about an individual or organization, while VPs provide a way to present these credentials securely, allowing users to disclose information based on the context of their interactions selectively. It is crucial to invalidate compromised or outdated VCs, which is why numerous revocation mechanisms have been suggested in the existing literature (17). A widely used mechanism is the *Bitstring Status List* (18), recommended by the W3C Verifiable Credentials Working Group. This specification uses a bitstring-based mechanism, where each bit corresponds to a VC. When a revocation is necessary, the associated bit is set, effectively changing the credential's validity. The default status list can contain up to 131,072 entries (equivalent to 16 KB), making it efficient to manage even for many credentials. Since in most cases only a few credentials are revoked, the list can be compressed down to a few hundred bytes using common techniques.

Building on this, we integrate DID and VC technologies into our framework to provide complete traceability of all data, model updates, and actors. This capability is particularly valuable in DT scenarios, where the auditability and trustworthiness of FL models are required. Our approach allows monitoring and tracking of the entire process, ensuring that erroneous data or models can be promptly identified and revoked. For VCs revocation, given the requirements of FL systems, such as interoperability, scalability, and adaptability, we have chosen to adopt the *Bitstring* approach among the various proposed revocation mechanisms. This compressible privacy-preserving mechanism is ideal for FL-based DT ecosystems.

2.3. Distributed Ledger Technologies

DLTs are innovative systems designed to improve security and trust between untrusted parties by decentralizing data management. Unlike traditional centralized systems, DLTs operate through a Peer-to-Peer (P2P) network of nodes, each maintaining an up-to-date copy of the ledger. This distributed structure ensures transparency and reduces the dependence on intermediaries. A defining feature of

DLTs is their append-only model, which makes the recorded data immutable and resistant to unauthorized changes. By eliminating centralized control, DLTs also mitigates the risk of single points of failure, a common vulnerability in traditional systems. To achieve consensus and maintain data integrity, DLTs use cryptographic protocols, allowing participants to collectively validate and agree on transactions. This decentralized validation process fosters trust and strengthens the security and reliability of the system without the need for a central authority. Moreover, most DLTs support the execution of smart contracts, which are computer programs written in general purpose programming languages that run on DLT. Each node executes the program, and correct execution requires the consensus of all nodes on the result. By eliminating the need for intermediaries, smart contracts provide a framework for decentralized and trustless computing, ensuring transparency and reliability in execution.

We leverage DLT to securely store the DIDs and key metadata. In addition, we use a smart contract to record and manage the influence calculations, providing transparency, immutability, and traceability over time of each client's contribution to the global model.

3. Related Work

The integration of FL into DT technology has been examined in various studies (19; 20; 21; 22; 23), highlighting the growing interest in leveraging this synergy to enhance system intelligence while preserving privacy. In addition, different FL solutions have integrated individual components, such as DLT, DIDs, or influence estimation techniques, which target specific problems in the FL process. However, to the best of our knowledge, none of the existing approaches combines all these technologies in a unified framework specifically designed to improve the trustworthiness of FL-driven DTs, as advocated by TrustFlow. This section, therefore, adopts a vertical approach, reviewing related works that address each of these aspects individually.

3.1. DTL-based FL-driven DTs

In the context of DTs, the integration of FL and DLT has been explored by several recent works (24). The authors in (25) introduce the concept of DT Wireless Networks (DTWN), advocating the integration DTs into wireless network infrastructures to enable real-time data processing and computation at the edge. They propose a blockchain-enhanced FL framework operating within the DTWN to support collaborative intelligence. The blockchain serves as a tamper-proof ledger to record training models and to enforce the permission control of participants in the process. Similarly, in (26), the authors present a permissioned blockchain-based FL scheme for DT Edge Networks (DITENs), incorporating asynchronous aggregation and spectrum-aware scheduling to improve communication efficiency and data privacy. In this framework, the blockchain functions as a coordination layer, facilitating a secure and transparent FL process. The work in (27) proposes an approach for model update verification in decentralized DT edge networks using a Directed Acyclic Graph (DAG)-based DLT and a double auction mechanism to incentivize participation. Moreover, the work in (28) introduces FedTwin, an adaptive asynchronous FL paradigm that employs a custom Proof-of-Federalism (PoF) consensus protocol, differential privacy, and falsified model detection to enable robust DT network collaboration. Finally, the authors in (29) explore a blockchain-assisted hierarchical FL platform for Industry 4.0 applications, integrating DTs into Cyber-Physical Systems (CPS) and employing a two-stage aggregation mechanism to improve scalability and accuracy. The blockchain is used to verify and validate trained models by leveraging designated validation nodes.

The works presented primarily leverage blockchain to track model updates, ensure data immutability, or facilitate secure model aggregation. However, they fail to provide a comprehensive solution for enhancing trustworthiness in FL processes within DT systems. Specifically, they lack mechanisms to track the entire lifecycle of contributions by linking them to unique identifiers, such as DIDs, similar to our proposal. Furthermore, they do not support the inclusion of detailed metadata such as data provenance and collection context, which is essential for auditing and trust. In addition, they fail to estimate the influence of individual client updates or data on the global model, which is critical for assessing contribution value and potential risk. Most notably, these approaches lack revocation mechanisms enabling the system to revoke and reduce the impact of faulty or malicious updates or models. In contrast, TrustFLow integrates all these features into a unified and traceable architecture.

3.2. DIDs for Trustworthy FL

Regarding the integration of FL with DIDs, various solutions take advantage of technologies to improve privacy, security, or decentralization. DID-eFed (30) proposes a FL as a Service system using DIDs and smart contracts for flexible access management. Similarly, Goh *et al*. (31) developed a blockchain-enabled FL architecture that incorporates DIDs for access control. The authors in (32) propose a decentralized privacy-preserving workflow for trusted FL, leveraging decentralized identity technologies from Hyperledger. The work focuses on a medical use case, requiring that only participants with VCs issued by authorized entities can securely and verifiably participate in processes FL. Zeydan *et al*. (33) propose an identity management solution for vehicle users in FL. Using DIDs, this method ensures the confidentiality, authenticity, and integrity of user identities and data during FL processes. These approaches aim to improve data privacy, security, and traceability in FL while addressing key challenges such as participant identity verification and data accountability. However, this body of work neglects important system features for model influence estimation and revocation. These features are essential to preserve the integrity of the global model and mitigate the impact of compromised updates.

3.3. Influence Estimation

Regarding the estimation of influence in FL, different methods have been proposed for use in centralized or federated contexts. *Leave-One-Out* (LOO) and *Influence Functions* (IFs) (34) have been conceived for use in centralized settings. LOO evaluates the contribution of clients by retraining the model while excluding specific datasets and observing performance changes, but it is computationally impractical in FL (34). IFs approximate the effect of removing data points without retraining, using model parameters and second-order derivatives of the loss function. *TracIn* (12), another centralized approach, estimates the influence of a data point z (having input x and label y) by evaluating its impact on model loss throughout training. Unlike LOO, which requires retraining, or IFs, which depend on second-order derivatives computations, TracIn leverages gradient information from multiple model checkpoints to provide an efficient and scalable estimate. There is an ongoing research effort to estimate the influence of the client in a FL setting, and various alternatives have been proposed. Xue *et al*. (35) introduce the concept of *Fed-Influence*, a metric based on a leave-one-client-out approach to quantify the influence of individual clients. They propose an efficient and effective algorithm to estimate this metric. However, the algorithm relies on the Hessian approximation, which may still be impractical for devices with limited computational resources. More recent approaches like Wang *et al*. (36) use Shapley values (37), a well-known concept from cooperative game theory, to calculate the average marginal contribution of a client to the model, considering all possible combinations of clients. However, the high computational cost and the additional steps required make the approach impractical for large-scale or real-time applications in FL. Gill *et al*. (38) propose TraceFL, a method to improve transparency in FL by tracking individual neurons' contributions to the global model. This helps identify errors and biases, providing valuable insights for debugging. However, the approach introduces significant computational overhead as tracking neuron contributions requires additional resources and steps.

Although numerous studies focus on estimating a client's contribution, many adopt a coarse-grained approach, assessing the influence of a client as a whole rather than evaluating the influence of individual datasets, limiting the ability to pinpoint problematic data. Conversely, TrustFLow integrates support for both client and individual data influence estimation and enables the revocation of both data or models, therefore helping to maintain the integrity of the global model and preventing the spread of faulty information.

4. TrustFLow

In this section, we present our framework, which incorporates a comprehensive tracking system based on DIDs and VCs to uniquely identify process entities, datasets, updates and global models within FL-driven DTs. TrusFLow enriches these elements with specific metadata, such as data collection procedures, to ensure the quality and trustworthiness of the process. In addition, it incorporates an automated, policy-driven revocation mechanism for data and global models, enabling proactive retraining management in the case of flaws or bias in data contributions or local model updates. TrustFLow can be seamlessly integrated into any FL-driven architecture as a governance component, placed primarily within the Service Layer as evidenced in Figure 1.

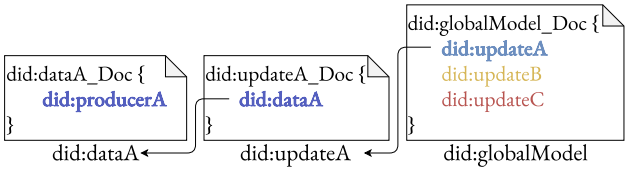

Figure 2: DID Documents and their associated metadata enable tracing back to the sources.

Without loss of generality, we assume that adversaries act as malicious clients within the federated learning system, attempting to poison the global model by submitting manipulated updates. These adversaries have access to their own local data and are aware of the model architecture and training protocol. However, they do not have access to the data or updates of other clients. Although adversaries can participate in training and submit arbitrary or malicious updates, the central server is assumed to be honest and secure. Attackers cannot alter, compromise, or manipulate the server, which reliably aggregates updates without leakage or alteration. These assumptions are widely accepted and grounded in standard security models commonly used in FL research (39; 40; 41).

4.1. Enabling Traceability in Federated Learning

A key feature of our framework is the comprehensive traceability of every element of the system, including data, actors, and models. This is achieved through the use of DID technology, with each component of the system having a valid DID stored on the DLT. We assume either a permissionless autonomous registration phase or a trusted authority managing participant selection in a permissioned manner. Each DID is also associated with a corresponding DID Document, which contains important information about the entity referenced by the DID. Figure 2 illustrates how we use DID Documents to link the various entities involved in the process.

Data Producers: Data producers generate data. They can be users of smartphones, smart city sensors, hospitals, industrial machinery, or other IoT devices. Producers can choose to remain completely anonymous, disclosing no information, or, conversely, provide crucial details to ensure clear identification. Any information about a producer is saved in its DID Document. For example, a public entity, like a hospital, might prefer to be easily recognizable to guarantee that users can trust the origin of the data.

Data: The data generated by a producer is linked to a unique DID, allowing unambiguous reference. The corresponding DID Document includes information about the producer, using the DID controller field, and other details regarding how the data was generated. As mentioned above, one of the most critical aspects of creating accurate models FL is to ensure that the data are produced and handled accordingly to minimize potential biases or errors. In the datasets DID Document, the producer can outline the procedures followed to ensure data integrity, such as those recommended by the EU Agency for Fundamental Rights (42). Trainers use this information to guide their data selection. For example, a trainer might select only data from recognized or globally trusted sources. Additionally, a trainer might opt for data produced according to specific guidelines or data that has gone through particular quality checks performed by the producers. To be valid, each data must have an associated valid VC signed by the producer specified in its DID Document. The server retains part of the data, comprising the test dataset, used by the server to calculate the influence of individual clients on the global model. This dataset also has DID and VC; all used in the event of disputes.

Model Trainers: Model trainers use data to train a ML model. As described in our example in Section 1, we assume that a trainer can be either the producer of the data it uses or a third-party entity that utilizes data from one or more producers. In the latter case, selecting the appropriate data for training is a crucial step to ensure that the resulting FL models are as accurate and reliable as possible. In the first case, the trainer and the producer coincide, so the DID and the DID Document of the trainer are the same as that of the producer. However, in the second case, trainers should have a unique DID stored on a DLT to be identifiable. The DID Document may also include information to identify the actual entity behind the DID.

Clients Updates: Local models are trained by the trainers participating in the FL process using selected data, producing an update. The latter is identified by a DID on the DLT published by the corresponding trainer. The associated DID Document contains information related to the training, such as the optimization algorithm used, the number of local epochs, and other metadata. In addition, the DIDs of the data used to train that model is also specified. This ensures that it is always possible to trace which data were used to train a local model and produce the specific update.

Server: The server performs the standard operations of a traditional FL server, along with additional specific tasks to manage the influences of the data and the revocation mechanism. It has a DID on the DLT and the corresponding DID Document. Typically, the server is a trusted third party that enables the FL process.

Global Model: The global model is generated by aggregating updates from various trainers. The server generates a DID and publishes it in the DLT along with its corresponding DID Document. This document includes metadata about the global model, such as the aggregation algorithm used and the DIDs of the local models involved in the aggregation process. This ensures full traceability of which models and, by extension, which data were used to create the global model and influence its development. This process is carried out during each round, providing a complete view of how the global model evolves and which data have played the most significant role in shaping it. To be valid, each global model must have an associated VC issued by the server that created it.

Table 1
Influence table stored on the smart contract.

Data	Global Models		
	did:globalModel^1	...	did:globalModel^N
did:dataA	-2	-1	-3
did:dataB	+2		
did:dataC	-1	+4	+2

DLT: The DLT stores the process information, guaranteeing its immutability and traceability during its lifecycle. It stores the DIDs and their respective DID Documents to provide full transparency of the FL process and accountability for each entity. In addition, a smart contract maintains a detailed record of how each dataset influences the global model. Table 1 illustrates an example of how influences can be recorded. For each global model generated at the end of a training round, the DIDs of the data used by the local models are listed, along with their corresponding influence values. Since not all trainers participate in every round, influence data may be missing for trainers who did not contribute during a specific round. By systematically tracking these influences, it becomes possible to enforce policies for revoking data or models when necessary.

4.2. Influence Estimation

Our framework uses the TraceIn (12) algorithm to assess the influence of a client, adapting it to a federated distributed context. This algorithm, originally designed for centralized learning, is efficient and allows fine-grained analysis of specific datasets, enabling the identification of problematic data when clients send multiple gradients. In a centralized context, the algorithm works as follows. Given a training dataset $\mathcal{D} = \{z_i = (x_i, y_i)\}_{i=1}^{N}\}$, let $\theta^{(t)}$ denote the model parameters in the training iteration t. The loss function for a data point z_i is represented as $L(z_i, \theta) = \ell(f_\theta(x_i), y_i)$, where f_θ is the model, and ℓ is the loss function.

The influence score for a data point z_i on a reference data point z' is given by:

$$Influence(z_i, z') = \sum_{t \in C} \eta_t \nabla_\theta L(z_i, \theta^{(t)}) \nabla_\theta L(z', \theta^{(t)}) \quad (1)$$

where:

- C is the set of selected checkpoints during training,
- η_t is the learning rate at iteration t,

- $\nabla_\theta L(z, \theta^{(t)})$ is the gradient of the loss function with respect to the model parameters θ at checkpoint t.

The influence score aggregates the dot products of the gradients at each checkpoint, capturing the cumulative effect of z_i on z'. A positive influence score indicates that z_i contributes to reducing the loss for z', meaning z_i is helpful in correctly learning z'. A negative influence score implies that z_i increases the loss for z', suggesting that z_i may introduce noise or conflict. To adapt TracIn to our framework, the server calculates the influence at the end of each round, using the global models produced at the end of the previous round as checkpoints. As in FedSGD, each client sends its update to the server after local training. This update is the gradient of the loss function with respect to the global model $\theta^{(t-1)}$ from the previous round, computed using the client's local data $\mathcal{D}_{Client}$, i.e., $\nabla_\theta L(\mathcal{D}_{Client}, \theta^{(t-1)})$. The server uses this value as the first term of Equation 1. A client with multiple datasets can send a single update or N updates, one for each dataset. In the first scenario, the influence is calculated collectively for all datasets and is equally attributed to each. In the second scenario, the server calculates the influence of each dataset.

Regarding the comparison data z', the server has a small dataset to compute the second term of Equation 1, i.e., $\nabla_\theta L(\mathcal{D}_{Test}, \theta^{(t-1)})$. This ensures consistency between all clients, since they are all evaluated using the same data, providing a more uniform measure of how each client's data influences the global model. The practice of retaining such a dataset is well established in the FL literature (43; 44; 45; 46), where it is commonly used for evaluation, debugging, or supporting specific computations.

Next, the server applies the TracIn formula to calculate the influence of each client's update. The influence score I_{Client} for a given client is:

$$I_{Client} = \nabla_\theta L(\mathcal{D}_{Client}, \theta^{(t-1)}) \nabla_\theta L(\mathcal{D}_{Test}, \theta^{(t-1)}) \quad (2)$$

This process is done for each client, and the server then publishes the influences on the DLT through the deployed smart contract. This process introduces minimal overhead, as client updates are already part of the FL process. The only extra step is for the server to compute the gradients on a smaller test dataset, resulting in negligible additional cost.

4.3. Revocation Mechanism

The TrustFLow decentralized revocation mechanism is tightly integrated with the tracking system, using DIDs, VCs, and a smart contract to ensure complete traceability and effective model management. This mechanism is governed by a smart contract that periodically monitors the recorded influences in the system to detect potential issues, applying predefined policies to identify problematic data and models. Algorithm 1 provides the pseudocode for an example smart contract. The revocation mechanism in TrustFLow is based on an influence-aware trust assessment that operates throughout the federated learning rounds. During each round, the system updates an *influence matrix* $T[r][g]$, where each entry $T[r_i][g]$ quantifies the contribution (positive or negative) of a data resource r_i to a global model g. This matrix is computed continuously during the learning process and serves as a foundational structure to identify underperforming or malicious contributions, allowing fine-grained control over the trustworthiness of each component in the system. Once the model training round is complete, the smart contract executes the `Revoke` function (lines 5-10). This function takes as input the global model g and the set of associated resource DIDs $\{r_1, \dots, r_n\}$, and performs a two-stage evaluation.

Algorithm 1 Revocation Mechanism for VCs

1: **Global:**
2: $T[r][g]$: influence score matrix
3: B_r: bitstring for resource VC revocation
4: B_g: bitstring for global VC revocation
5: **Function** Revoke(g, $\{r_1, ..., r_n\}$)
6: **Input:**
7: g: global model
8: $\{r_1, \dots, r_n\}$: DIDs of resources linked to g
9: **Output:**
10: Revocation status of VC_g
11: **for each** r_i in $\{r_1, ..., r_n\}$ **do**
12: **if** $T[r_i][g] < 0$ **then**
13: Extract CID in DID Document of r_i
14: Compute bit index idx_r from CID
15: Set $B_r[idx_r] \leftarrow 1$
16: Revoke VC_{r_i}
17: **end if**
18: **end for**
19: **return** RevokeGlobalVC(g, $\{r_1, ..., r_n\}$)
20: **Function** RevokeGlobalVC(g, $\{r_1, ..., r_n\}$)
21: $T_g \leftarrow$ number of DIDs associated with g
22: $R_g \leftarrow$ number of resource bits set to 1 in B_r for $\{r_1, ..., r_n\}$
23: **if** $R_g / T_g > \phi$ **then**
24: Extract CID from DID Document of g
25: Compute bit index idx_g from CID
26: Set $B_g[idx_g] \leftarrow 1$
27: Revoke VC_g
28: **end if**
29: **return** revocation status of VC_g

In the first stage, for each resource r_i, the system checks whether its influence score $T[r_i][g]$ is strictly negative (lines 11-12). If so, the system considers the resource untrustworthy and proceeds to revoke its associated VC_{r_i}. Technically, this is done by setting to 1 the corresponding bit in the global bitstring B_r, which maintains the revocation state of all resource credentials (lines 13-16). The position of the bit is computed by extracting the CID stored in the `serviceEndpoint` field of the DID Document associated with r_i. This approach ensures that revocation is transparent, tamper-proof, and efficient, as it avoids the need to remove the VC entirely and instead marks its invalidity verifiably.

In the second stage, the algorithm assesses whether the global model g should also be revoked. It counts how many of the associated resource VCs have been revoked (i.e., how

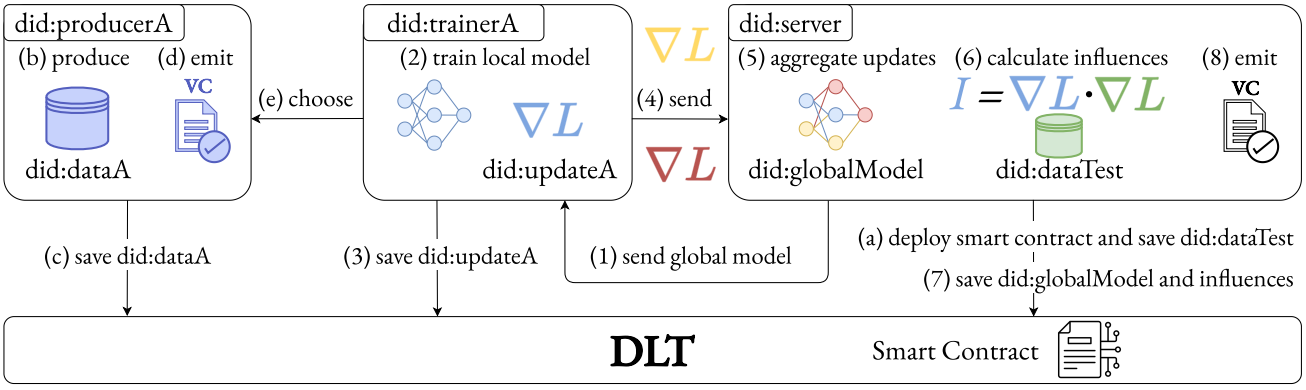

Figure 3: TrustFLow during the Preparation phase, (a)-(e) steps, and the traditional FL Process, (1)-(8) steps.

many corresponding bits in B_r are set to 1). If the proportion of revoked resources exceeds the threshold ϕ (line 23), the system considers the global model compromised. Currently, the threshold is empirically set to 0.5, as higher values can prevent the model from converging (47). The corresponding bit in the global model bitstring B_g is also set to 1 (lines 24-27). At this point, the bit index is computed from the CID in the `serviceEndpoint` field of the DID Document associated with g, which points to the VC of the global model saved on IPFS.

To ensure interoperability and verifiability, bitstrings B_r and B_g are managed on-chain and follow the W3C Status List 2021 specification (18). This design enables efficient revocation proofs and ensures consistency across decentralized components. By combining continuous influence tracking, structured trust evaluation, and decentralized bitstring-based revocation, TrustFLow enables granular, transparent, and scalable trust management of the underlying DT model. It allows for selective deactivation of harmful resources or outputs while preserving valid contributions, ensuring both security and continuity in decentralized AI workflows.

4.4. TrustFLow at Work

Building on the example introduced in Section 1, we consider a predictive maintenance scenario in which sensors, machines, and servers within a manufacturing environment collaboratively participate in creating a FL-driven DT capable of predicting failures and optimizing operations. The main execution flow can be divided into three main phases: (I) Preparation, (II) FL Process, (III) Revocation. Figure 3 depicts phases (I) and (II).

4.4.1. Preparation

The first phase, steps (a)-(e) of Figure 3, focuses on preparing all the essential elements required for the process. In step (a), the server, which could be a smart manufacturing server or an industrial Internet of Things (IoT) platform, deploys a smart contract on the DLT that allows recording the influence of the data on global models. The contract also implements the policies for the revocation of the data and the model. For a more detailed discussion of the policies, refer to Section 4.3. In addition, the server saves the DID related to its testing data. A producer, such as a smart sensor or an industrial machine, collects the data and makes it available (b). The producer publishes a DID to identify the dataset (c) and issues a unique VC (d) certifying the validity of the dataset. In step (e), a trainer, which could be an edge device or a local server, selects one or more datasets based on specific requirements. For example, a trainer might prefer datasets from specific sensors or machines. These steps could also be performed before or simultaneously with the steps executed by the server.

4.4.2. FL Process

The second phase, steps (1)-(8) of Figure 3, integrates the standard FL workflow with additional operations to track models and their associated metadata. The FL process begins with the server sending the initial global model (1). The trainer uses the selected datasets to train a local model (2) and produce an update. The trainer then publishes (3) a DID for the newly generated update and sends (4) it to the server. The server collects and aggregates all client updates (5) to create the global model and uses them to calculate (6) the influences on the global model. The server accepts only updates produced by models trained using data with valid VCs. It then publishes (7) a DID to identify the global model, records the influences using the smart contract, and issues (8) a corresponding VC to certify the validity of the global model. The server subsequently sends the global model back to the clients to continue the FL process.

4.4.3. Revocation

The revocation flow, described in Section 4.3, is executed when it is discovered that the data used in the training negatively impact the model. Every time the server publishes new influences on the smart contract, the contract applies a policy, like the one presented in Algorithm 1, to determine whether any data or models should be revoked. If necessary, the server can either reinitialize the training from scratch, rejecting updates from revoked sources, or restore the model weights to a state before the faulty data was aggregated if it had saved them. Overall, our framework introduces

Table 2
Influence score estimation in different settings.

Setting	Clients	IID				non-IID			
		Last Checkpoints			Total Sum	Last Checkpoints			Total Sum
5 Clients	4 Correct	1292.83	867.68	622.05	2782.57	1198.90	770.56	890.24	18381.10
	1 Faulty	-233.70	-863.92	14.39	-1083.23	118.07	-912.61	-738.47	3518.47
10 Clients	9 Correct	4144.13	213.64	67.34	4867.81	835.50	1142.98	1088.56	15551.82
	1 Faulty	2016.46	0.55	-164.02	2142.62	-981.03	-397.65	-721.70	-6206.53
	8 Correct	2074.12	284.41	88.34	2639.06	1737.32	1568.03	1238.61	13201.61
	2 Faulty	-359.04	-144.63	-354.06	-2114.39	-951.57	-958.97	-418.34	-3467.97
20 Clients	19 Correct	3605.32	663.47	455.97	6024.43	374.74	398.19	402.11	7160.18
	1 Faulty	-370.24	-449.91	398.66	1429.65	-292.61	-605.11	-1242.62	-4698.83
	18 Correct	3726.20	915.30	818.87	5809.99	287.25	214.60	156.58	8600.07
	2 Faulty	-283.83	384.12	-243.95	1038.18	-1818.28	-1467.94	-1222.87	-8632.68
	16 Correct	6271.37	803.36	275.62	7350.36	411.20	232.23	365.07	15584.95
	4 Faulty	613.95	-1282.07	-726.26	-1394.39	-1473.78	-1002.70	-1661.37	-8467.47

minimal overhead to the FL process, as shown also in the experimental part.

5. Experimental Evaluation

This section presents the experiments conducted to validate our proposal. We begin by describing the implementation setup, followed by an overview of the conducted experiments. Finally, we analyze the performance indicators.

5.1. Experimental Strategy

Our analysis focuses on evaluating the accuracy of the influence estimation algorithm, the latency of key-tracking operations, and the overhead of the revocation mechanism. Since TrustFlow does not modify the FL protocol, the accuracy of the global model remains unaffected and is therefore not reported here. Instead, we focus on metrics directly relevant to the objectives of our framework. Detecting detrimental clients and their influence is crucial to ensuring the reliability of the global model while minimizing tracking latency. Consequently, accurate estimation of both the data and the influence of the client is paramount.

Moreover, enabling fast revocation of data and models when flawed contributions are detected is necessary to maintain the integrity of the training process. In particular, we conduct our experimental evaluation on the operations that include the creation of DIDs and DID Documents, the saving of these DIDs and DID Documents on the blockchain, and the issuance of VCs. Moreover, we evaluate the time taken to store VCs on the InterPlanetary File System (IPFS) decentralized storage. The aim is to evaluate how well our proposed method can detect detrimental contributions, quickly revoke problematic data and connected models, and ensure that the overhead introduced by influence calculations and interactions with DLT does not significantly impact overall FL system performance.

5.2. Experimental Setup

We assess the TrustFLow mechanisms in different settings and with a varying number of clients (5, 10, 20) participating in the FL process. Furthermore, we consider various percentages of faulty clients: 5% (with 20 clients), 10% (with 10 and 20 clients), and 20% (across all client numbers). We considered these configurations to reflect realistic scenarios, such as a smart industry ecosystem relying on a predictive maintenance model or hospitals adopting advanced smart healthcare solutions, where the chosen number of participating clients can be considered reasonable. To simulate the impact of biased or incorrect data, we modify client datasets by performing label flipping (48), a common method to introduce noise in classification tasks. For the ML model, we use MobileNetV2 (49), a relevant state-of-the-art neural network, and employ Adam optimizer with a learning rate of 1e-3. Local training is conducted over 5 epochs with a batch size of 32. The evaluation uses the CIFAR-10 (50) dataset, a well-established benchmark in the ML community, with both Independent and Identically Distributed (IID) and non-IID data distributions. The non-IID setting is simulated using a Dirichlet distribution with $\alpha = 0.5$. Regarding the DLT, we base our work on Ethereum due to its robust support for decentralized, programmable smart contracts through Solidity (51), as well as its maturity, ease of programmability, and strong ecosystem support. This choice was driven by the need for a secure, transparent, and immutable infrastructure capable of automating the complex interactions required for revocation while ensuring accountability. Solidity's flexibility allowed us to design custom logic to handle updates to credential status. For local development, we use the Truffle framework in conjunction with a simulated Ethereum blockchain, such as Ganache, which has also been adopted in similar works within the FL domain (52) (53). This setup enables us to test and deploy smart contracts in a controlled environment efficiently. Specifically, we used Web3.js to facilitate interaction with the blockchain, enabling both the deployment and the management of smart contracts. To create

DIDs, DID Documents, VCs, and bitstrings, we leverage the Digital Bazaar library. This library enables the generation of DIDs and their associated DID Documents, as well as the creation and management of VCs. Concluding, we use IPFS to store the VCs.

5.3. Performance Results

5.3.1. Influence Score Estimation

Table 2 shows the trend of the calculated influence in different settings and clients. For simplicity, the table shows only the averages computed for the correct clients, i.e., those without erroneous data, and faulty clients, i.e., those characterized by flipped labels. Since each setting required a different number of rounds and more clients needed additional rounds to converge, only the influences from the last three checkpoints are explicitly reported (*Last Checkpoints* columns). The *Total Sum* column shows the sum of the influences, including those from the non-displayed checkpoints. The data indicates that faulty clients tend to contribute negatively to the global model's quality of the global model in both IID and non-IID settings. Their influence is generally negative or lower than that of the correct clients. However, in some cases, the influence score of faulty clients can be positive. This occurs because these clients still possess a portion of correct data, i.e., data without label flipping, which contributes positively to the model's classification ability. For example, in a scenario with 20 clients, where only one is faulty, splitting CIFAR-10 into 20 parts results in the faulty client holding a relatively small amount of corrupted data. Consequently, the data remain insufficient to significantly degrade the model's overall performance. An interesting observation in the IID setting is that as the proportion of faulty clients increases, their influence score decreases and eventually becomes negative. This indicates that the faulty data contribute very little to improving the model's performance. If the global model is generated from several faulty updates, the gradient computed by the server on the correct test data attempts to compensate for the errors introduced by the faulty data during training. As a result, this gradient differs significantly from the ones provided by faulty clients, showing how their contributions are less useful and eventually harm the performance of the global model. The non-IID case exhibits greater variability, as some clients may have a smaller or even negative influence despite not containing faulty data. This is not surprising, as the contribution of each client depends on the proportion and distribution of their data. For example, clients with smaller datasets will contribute significantly less than those with larger datasets. In addition, the results consistently show that faulty clients have lower influence scores, often falling into negative values, compared to correct clients. In these experiments, each client is assigned a single dataset for simplicity, so the influence score is assigned to the entire client. If a client were to possess multiple datasets, it would send multiple gradients and the influence score would be computed separately for each dataset. In any case, the shown results remain valid as the influence trends remain the same.

Table 3
Average Latency Times for Different Operations and Client Groups in Milliseconds

Operation	5 Clients	10 Clients	20 Clients
DID Generation	1.43	1.44	1.37
Save DID on DLT	198.80	197.67	197.68
VC Generation	864.74	876.23	882.98
Upload VC to IPFS	22.03	24.28	21.29

5.3.2. Latency Analysis

Table 3 shows the latency distribution for the indicated operations, i.e., DID Generation, Save DID on DLT, VC Generation and Upload VC to IPFS. It reveals that all operations exhibit stable latency across different client groups. In general, system performance remains consistent regardless of the number of clients, indicating that operations are not significantly affected. This result suggests that the system can effectively handle an increasing number of clients without introducing significant delays in operations. Figure 4 compares the average execution time required to perform local training with the time taken for operations related to the creation and storage of DIDs and VCs in different settings. The data indicates that the time required to execute the DIDs and VCs operations is constant, as seen in Table 3, and negligible compared to the neural network training, demonstrating the minimal impact of our framework on client performance. It should be noted that training time decreases as the number of clients increases, as each client receives a smaller portion of the dataset, leading to faster training. However, this result is specific to our controlled setup with a fixed-size dataset. This trend may not be observed in real-world scenarios where client data is not partitioned artificially.

Figure 5 shows the latency in seconds for the various settings in a challenging scenario where all the VCs of data and global models must be revoked due to non-compliance. The results show that as the number of clients increases, the latency of the revocation process increases significantly. This increase in latency is due to the need to manage a growing number of data and global models to revoke, which implies greater computational complexity and more time required to complete the operation. Nevertheless, it is important to highlight that we have considered the worst case, which does not practically occur, and that revocation is typically triggered only in specific cases, such as when non-compliant or malicious behavior is detected. As a result, its infrequent execution ensures that the system remains efficient during standard operations.

6. Conclusive Remarks

In DT ecosystems, where data is continuously generated by distributed sources, FL plays a key role in enabling collaborative and privacy-preserving model training. This approach enables DTs to adapt and evolve in real-time without compromising data privacy. As a result, ensuring the

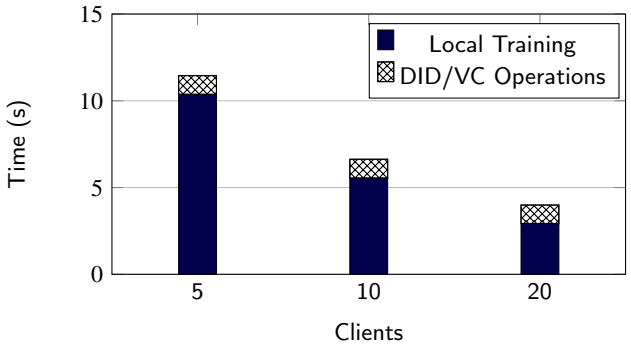

Figure 4: Average time (in seconds) required for local training and DID/VC operations, with varying numbers of clients.

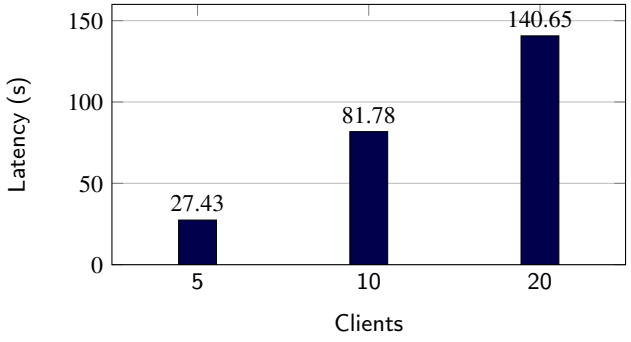

Figure 5: Latency (in seconds) of the revocation mechanism for 5, 10, and 20 clients.

trustworthiness, integrity, and reliability of FL processes becomes essential, demanding robust observability to monitor, analyze, and optimize system performance. In this direction, we presented TrustFLow, a robust tracking framework that leverages DLTs, DIDs, and VCs to achieve comprehensive traceability in FL processes. TrustFLow enables one to gain up-to-date information on client participation and model performance, fostering quality-aware, adaptive, and trustworthy learning processes. TrustFLow operates independently of the specific FL algorithm, ensuring broad applicability across various implementations. An original feature of TrustFLow is its integration of automated revocation mechanisms, which enable the invalidation of erroneous or malicious contributions, thereby safeguarding the integrity of the global model. In addition, it introduces minimal overhead to the FL process, preserving efficiency. Future work will focus on further testing TrustFLow in different settings, such as with various datasets, to assess its adaptability and effectiveness in diverse FL scenarios. Moreover, we plan to extend the framework to contemplate recent unlearning techniques (54), allowing the system to efficiently remove the influence of revoked data from the global model without compromising the learning process. Additionally, we aim to evolve TrustFLow towards a more decentralized architecture that reduces or eliminates reliance on central servers, further aligning with the principles of distributed FL. Another direction involves relaxing some of the assumptions made in the current threat model to better reflect real-world adversarial conditions and improve the robustness of the framework in more practical deployments. These advancements will further strengthen TrustFLow's role as a comprehensive and practical solution for trustworthy and transparent FL-driven DT systems.

References

[1] L. Rosa, A. Calvio, A. Garbugli, L. Foschini, A QoS-Aware Data Distribution Platform for Edge-Based Vehicular Digital Twins in Smart Cities, in: 2025 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2025, pp. 1–6.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient Learning of Deep Networks from Decentralized Data, in: Proc. of Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.

[3] S. P. Ramu, P. Boopalan, Q.-V. Pham, P. K. R. Maddikunta, T. Huynh-The, M. Alazab, T. T. Nguyen, T. R. Gadekallu, Federated learning enabled digital twins for smart cities: Concepts, recent advances, and future directions, Sustainable Cities and Society 79 (2022) 103663.

[4] S. Jamil, M. Rahman, Fawad, A comprehensive survey of digital twins and federated learning for industrial internet of things (iiot), internet of vehicles (iov) and internet of drones (iod), Applied System Innovation 5 (3) (2022) 56.

[5] M. M. Salim, D. Camacho, J. H. Park, Digital twin and federated learning enabled cyberthreat detection system for iot networks, Future Generation Computer Systems 161 (2024) 701–713.

[6] A. Mora, A. Bujari, P. Bellavista, Enhancing Generalization in Federated Learning with Heterogeneous Data: A Comparative Literature Review, Future Generation Computer Systems (2024).

[7] W. Sun, S. Lei, L. Wang, Z. Liu, Y. Zhang, Adaptive federated learning and digital twin for industrial internet of things, IEEE Transactions on Industrial Informatics 17 (8) (2020) 5605–5614.

[8] W. Yang, W. Xiang, Y. Yang, P. Cheng, Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial iot, IEEE Transactions on Industrial Informatics 19 (2) (2022) 1884–1893.

[9] Q.-V. Pham, K. Dev, P. K. R. Maddikunta, T. R. Gadekallu, T. Huynh-The, et al., Fusion of federated learning and industrial internet of things: A survey, arXiv preprint arXiv:2101.00798 (2021).

[10] L. U. Khan, W. Saad, Z. Han, E. Hossain, C. S. Hong, Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges, IEEE Communications Surveys & Tutorials 23 (3) (2021) 1759–1799.

[11] J. C. Jiang, B. Kantarci, S. Oktug, T. Soyata, Federated learning in smart city sensing: Challenges and opportunities, Sensors 20 (21) (2020) 6230.

[12] G. Pruthi, F. Liu, S. Kale, M. Sundararajan, Estimating Training Data Influence by Tracing Gradient Descent, Advances in Neural Information Processing Systems 33 (2020) 19920–19930.

[13] P. Bellavista, L. Foschini, A. Mora, Decentralised Learning in Federated Deployment Environments: A System-Level Survey, ACM Comput. Surv. 54 (1) (2021) 1–38.

[14] P. Kairouz, et al., Advances and Open Problems in Federated Learning, Foundations and trends in machine learning 14 (1–2) (2021) 1–210.

[15] W3 Recommendation, Decentralized Identifiers (DIDs) v1.0, [Online]. URL: https://www.w3.org/TR/did-core/, last Accessed: December 2024 (2022).

[16] W3 Recommendation, Verifiable Credentials Data Model v1.1, [Online]. URL: https://www.w3.org/TR/vc-data-model/, last Accessed: December 2024 (2022).

[17] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, M. Conti, A Survey on Decentralized Identifiers and Verifiable Credentials, arXiv preprint arXiv:2402.02455 (2024).

[18] W3 Consortium, Bitstring status list v1.0, [Online]. URL: https://www.w3.org/TR/vc-bitstring-status-list/, last Accessed: December 2024 (12 2024).

[19] L. U. Khan, E. Mustafa, J. Shuja, F. Rehman, K. Bilal, Z. Han, C. S. Hong, Federated Learning for Digital Twin-Based Vehicular Networks: Architecture and Challenges, IEEE Wireless Communications 31 (2) (2024) 156–162.

[20] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Communication-efficient federated learning for digital twin edge networks in industrial iot, IEEE Transactions on Industrial Informatics 17 (8) (2020) 5709–5718.

[21] W. Sun, N. Xu, L. Wang, H. Zhang, Y. Zhang, Dynamic digital twin and federated learning with incentives for air-ground networks, IEEE Transactions on Network Science and Engineering 9 (1) (2020) 321–333.

[22] S. D. Okegbile, J. Cai, H. Zheng, J. Chen, C. Yi, Differentially private federated multi-task learning framework for enhancing human-to-virtual connectivity in human digital twin, IEEE Journal on Selected Areas in Communications 41 (11) (2023) 3533–3547.

[23] S. D. Okegbile, H. Gao, O. Talabi, J. Cai, D. Niyato, X. Shen, Optimizing federated semantic learning in distributed aigc-enabled human digital twins: A multi-criteria and multi-shard user selection framework, IEEE Transactions on Mobile Computing (2025).

[24] K. Liu, Z. Yan, X. Liang, R. Kantola, C. Hu, A survey on blockchain-enabled federated learning and its prospects with digital twin, Digital Communications and Networks 10 (2) (2024) 248–264.

[25] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks, IEEE Transactions on Industrial Informatics 17 (7) (2020) 5098–5107.

[26] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Communication-efficient federated learning and permissioned blockchain for digital twin edge networks, IEEE Internet of Things Journal 8 (4) (2020) 2276–2288.

[27] L. Jiang, H. Zheng, H. Tian, S. Xie, Y. Zhang, Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks, IEEE Internet of Things Journal 9 (13) (2021) 11154–11167.

[28] Y. Qu, L. Gao, Y. Xiang, S. Shen, S. Yu, Fedtwin: Blockchain-enabled adaptive asynchronous federated learning for digital twin networks, IEEE Network 36 (6) (2022) 183–190.

[29] M. Aloqaily, I. Al Ridhawi, S. Kanhere, Reinforcing industry 4.0 with digital twins and blockchain-assisted federated learning, IEEE Journal on Selected Areas in Communications 41 (11) (2023) 3504–3516.

[30] J. Geng, N. Kanwal, M. G. Jaatun, C. Rong, DID-EFed: Facilitating Federated Learning as a Service with Decentralized Identities, in: Proc. of Evaluation and Assessment in Software Engineering, EASE 2021, Association for Computing Machinery, New York, NY, USA, 2021, p. 329–335.

[31] E. Goh, D. Kim, D.-Y. Kim, K. Lee, Blockchain-enabled Federated Learning: A Reference Architecture Incorporating a DiD Access System, arXiv preprint arXiv:2306.10841 (2023).

[32] P. Papadopoulos, W. Abramson, A. J. Hall, N. Pitropakis, W. J. Buchanan, Privacy and trust redefined in federated machine learning, Machine Learning and Knowledge Extraction 3 (2) (2021) 333–356.

[33] E. Zeydan *et al*., Blockchain-Based Self-Sovereign Identity: Taking Control of Identity in Federated Learning, IEEE Open Journal of the Communications Society (2024).

[34] P. W. Koh, P. Liang, Understanding Black-box Predictions via Influence Functions, in: Proc. of International conference on machine learning, PMLR, 2017, pp. 1885–1894.

[35] Y. Xue *et al*., Toward Understanding the Influence of Individual Clients in Federated Learning, in: Proc. of AAAI Conference on Artificial Intelligence, 2020.
URL https://api.semanticscholar.org/CorpusID:229340583

[36] T. Wang, J. Rausch, C. Zhang, R. Jia, D. Song, A Principled Approach to Data Valuation for Federated Learning, Federated Learning: Privacy and Incentive (2020) 153–167.

[37] E. Winter, The Shapley Value, Handbook of Game Theory with Economic Applications 3 (2002) 2025–2054.

[38] W. Gill, A. Anwar, M. A. Gulzar, TraceFL: Interpretability-Driven Debugging in Federated Learning via Neuron Provenance, arXiv preprint arXiv:2312.13632 (2024).

[39] A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: International conference on machine learning, PMLR, 2019, pp. 634–643.

[40] S. Li, Y. Cheng, W. Wang, Y. Liu, T. Chen, Learning to detect malicious clients for robust federated learning, arXiv preprint arXiv:2002.00211 (2020).

[41] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, arXiv preprint arXiv:2003.02133 (2020).

[42] European Union, Data Quality and Artificial Intelligence - Mitigating Bias and Error to Protect Fundamental Rights, https://fra.europa.eu/en/publication/2019/data-quality-and-artificial-intelligence-mitigating-bias-and-error-protect, last Accessed: December 2024 (2019).

[43] M. Yang, H. Qian, X. Wang, Y. Zhou, H. Zhu, Client selection for federated learning with label noise, IEEE Transactions on Vehicular Technology 71 (2) (2021) 2193–2197.

[44] D. Sivasubramanian, L. Nagalapatti, R. Iyer, G. Ramakrishnan, Gradient coreset for federated learning, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 2648–2657.

[45] V. S. Mai, R. J. La, T. Zhang, A study of enhancing federated learning on non-iid data with server learning, IEEE transactions on artificial intelligence (2024).

[46] X. Luo, B. Tang, Byzantine fault-tolerant federated learning based on trustworthy data and historical information, Electronics 13 (8) (2024) 1540.

[47] X. Cao, J. Jia, N. Z. Gong, Provably secure federated learning against malicious clients, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 6885–6893.

[48] Y. Jiang, W. Zhang, Y. Chen, Data Quality Detection Mechanism Against Label Flipping Attacks in Federated Learning, IEEE Transactions on Information Forensics and Security 18 (2023) 1625–1637.

[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510–4520.

[50] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Tech. rep., cIFAR-10 dataset, available at https://www.cs.toronto.edu/~kriz/cifar.html (2009).

[51] M. Wöhrer, U. Zdun, Smart Contracts: Security Patterns in the Ethereum Ecosystem and Solidity, Proc. of International Workshop on Blockchain Oriented Software Engineering (IWBOSE) (2018) 2–8.

[52] D. Commey, S. Hounsinou, G. V. Crosby, Securing Health Data on the Blockchain: A Differential Privacy and Federated Learning Framework, arXiv preprint arXiv:2405.11580 (2024).

[53] G. Shubham, V. Agarwal, S. Pal, IoT Data Security: An Integration of Blockchain and Federated Learning, in: Proc. of IEEE International Conference on Communications Workshops (ICC Workshops), 2023, pp. 434–439.

[54] N. Romandini, A. Mora, C. Mazzocca, R. Montanari, P. Bellavista, Federated unlearning: A survey on methods, design guidelines, and evaluation metrics, IEEE Transactions on Neural Networks and Learning Systems (2024) 1–21.