TruFLaaS: Trustworthy Federated Learning as a Service

Carlo Mazzocca, *Graduate Student Member, IEEE*, Nicolò Romandini, *Graduate Student Member, IEEE*, Matteo Mendula, *Graduate Student Member, IEEE*, Rebecca Montanari, *Member, IEEE*, and Paolo Bellavista, *Senior Member, IEEE*

***Abstract*—The increasing availability of data generated by Internet of Things (IoT) and Industrial IoT (IIoT) devices, as well as privacy and law regulations, have significantly boosted the interest in collaborative machine learning (ML) approaches. In this direction, we claim federated learning (FL) as a promising ML paradigm where participants collaboratively train a global model without outsourcing on-premises data. However, setting up and using FL can be extremely costly and time consuming. To effectively promote the adoption of FL in real-world scenarios, while limiting the overhead and knowledge of the underlying technology, service providers should offer FL as a Service (FLaaS). One of the major concerns while designing an architecture that provides FLaaS is achieving trustworthiness among involved typically unknown participants. This article presents a blockchain-based architecture that achieves trustworthy FLaaS (TruFLaaS). Our solution provides trustworthiness among third-party organizations by leveraging blockchain, smart contracts, and a decentralized oracle network. Specifically, during each FL round, the service provider supplies a sample, without overlapping, of its validation set to validate all partial models submitted by clients. By doing so, poor models, which tend to degrade performance or introduce malicious backdoors, are identified and discarded. Due to the transparency of the blockchain, not changing the validation set would enable participants to forge a malicious partial model that passes the validation phase. We evaluate our approach over two well-known IIoT data sets: the reported experimental results show that TruFLaaS outperforms the state-of-the-art literature solutions in the field.**

***Index Terms*—Blockchain, federated learning (FL), federated learning as a service (FLaaS), security, trust, trustworthiness.**

I. Introduction

THE INCREASINGLY widespread adoption of Internet of Things (IoT) and Industrial IoT (IIoT) devices is notably contributing to the design and development of next-generation services [1]. As reported in recent statistics, the number of such devices will surpass 125 billion by 2030 [2], generating an unprecedented amount of data that paves the way for new applications based on artificial intelligence (AI) [3]. However, traditional machine learning (ML) techniques, which require data centralization, are not feasible when a remarkable amount of information comes from multiple locations; both in terms of privacy awareness [4] and energy consumption [5]. Furthermore, law and privacy regulations, such as the European General Data Protection Regulation (GDPR) [6], hinder centralized ML approaches that may lead to potential data leakages.

These reasons are pushing industrial and academic communities toward more decentralized and collaborative ML approaches. In this direction, federated learning (FL) is envisioned as a promising ML paradigm in which the parties involved, which share common goals, collaboratively train a global model. Unlike centralized ML, which typically relies on cloud-based resources, data are no longer sent to a central entity, as training is performed directly on remote clients using on-premises data. Each client trains a local ML model with its own data and, subsequently, sends it to a server that combines all the partial models retrieved according to an aggregation strategy [7]. Nowadays, different services and companies, which could be also competitors, have to face similar problems that can be effectively solved through the use of distributed ML. Adopting a collaborative approach to train ML models can be extremely beneficial, especially for small/medium enterprises that may not have enough on-premises data to build useful models on their own. For example, in smart manufacturing environments, the various equipment and uneven load distribution may lead to unbalanced data regarding faults. In such a context, implementing a diagnostic model requires gathering a large amount of high-quality fault data, which is a hard task. Therefore, the lack and imbalances in fault samples represent two main factors that negatively affect the performance of fault diagnosis models [8]. These limits can be overcome through the training of a shared model, bringing advantages to every participant. Edge nodes deployed in multiple sites and the use of FL allow the exploitation of unbalanced samples to train models with excellent accuracy, generalizability, and efficiency [9].

Although FL is gaining much popularity in various fields [10], [11], [12], [13], there are just a few works that propose to provide FL as a Service (FLaaS) [14] to interested third parties. In the last decade, cloud providers have offered many cloud-based ML as a Service (MLaaS) [15] that comprise computing resources, APIs, open-sourced libraries, and tools for data analytics. However, despite increasing interest, current commercial solutions do not support collaborative training.

Manuscript received 15 November 2022; revised 12 April 2023; accepted 25 May 2023. Date of publication 5 June 2023; date of current version 7 December 2023. This work was supported in part by the SERICS Project through the NRRP MUR Program, which is funded by the EU-NGEU under Grant PE00000014. *(Corresponding author: Carlo Mazzocca.)*

The authors are with the Department of Computer Science and Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: carlo.mazzocca@unibo.it; nicolo.romandini@unibo.it; matteo.mendula@unibo.it; rebecca.4montanari@unibo.it; paolo.bellavista@unibo.it).

Digital Object Identifier 10.1109/JIOT.2023.3282899

Providing FLaaS with minimum overhead and knowledge of the underlying technology is a key factor in promoting the successful use of FL solutions. An effective FLaaS should be designed to: 1) offload developers from collecting data allowing them to only focus on the algorithm to implement; 2) preserve data privacy by avoiding data transfers from the owner to external entities; and 3) provide trustworthiness among unknown participants, which is the main focus of this work. For example, regarding the fault diagnosis use case reported above, smart manufacturing enterprises may want to be sure that the employed model can effectively predict a certain failure.

Although the FL paradigm enables tackling some AI-based challenges, such as preserving privacy, the global model can still be the target of different attacks (i.e., model poisoning and inference attack) [16]. Regarding trustworthiness, the main concern that hampers FL adoption in third-party applications is the presence of malicious clients and servers that negatively impact the performance of FL training and introduce malicious backdoors [17]. Furthermore, the traditional FL architecture based on the client–server model suffers from a single point of failure, low scalability, and tampering of the global model, including possible biases that induce some partial models over others [18].

To effectively improve the trustworthiness of the whole FL process, this article proposes a blockchain-based trustworthy FLaaS (TruFLaaS). Our solution provides trustworthiness among third-party contributors to the FL process by leveraging blockchain, smart contracts, and decentralized oracle networks (DONs). TruFLaaS proposes a novel validation strategy to aggregate partial models, resulting in an improved quality of the global model. Clients' models are validated by a smart contract through a sample of the validation data set given by the service provider through a DON. By evaluating the partial models on defined quality metrics (e.g., accuracy), we can generate high-quality global models. We associate a level of trust, updated during each round, with each client in order to properly weigh their contributions. To the best of our knowledge, we are the first to propose a validation protocol that leverages a smart contract to directly validate partial models. The experiments demonstrate that TruFLaaS outperforms conventional baselines and the state-of-the-art literature under different circumstances that are particularly relevant to FL scenarios. The following summarizes the major contributions of this article.

1) We present a novel blockchain-based architecture for enabling TruFLaaS. Our solution combines blockchain, smart contracts, and a DON to build a collaborative trustworthy AI model training system that can resist attacks from the server and malicious participants.
2) We design a novel validation protocol based on smart contracts and a DON. The DON is needed to dynamically feed the smart contract with a sample of the validation data set.
3) We propose a weighted aggregation strategy that takes into account the level of trust of each participant. To properly consider contributions, each client has a level of trust that is given by its performance achieved during all previous rounds.

The remainder of this article is structured as follows. Section II motivates the need for the FLaaS and discusses the main guidelines to design a TruFLaaS. Section III presents the blockchain-based architecture for enabling trustworthy FL, while Section IV discusses in detail the validation protocol as well as the level of trust of participants. Section V evaluates the proposed approach and presents experimental results. Section VI analyzes related work on trustworthiness and FL. Finally, Section VII draws our conclusions.

II. Motivation and Design Guidelines

FL is emerging as a valuable solution for creating ML models in a distributed manner without sacrificing data privacy. Despite the benefits, setting up and using FL can be extremely expensive and time consuming, especially in some sectors, such as industry or healthcare, where the necessary infrastructure and expertise are often lacking. FLaaS provides clients with an easy way to use FL with limited overhead and technological knowledge, allowing them to eliminate the heavy burden task of developing and tuning algorithms and tools. Furthermore, FLaaS is flexible to meet different participants' requirements while implementing FL training.

To facilitate the understanding of our proposal and to practically clarify the motivations behind the primary TruFLaaS design choices, we introduce an example that will be used as a reference use case throughout this article. Let us consider a company that sells industrial machines and offers a predictive maintenance service. All the customers who use a specific machine are interested in joining such FLaaS since predicting the breakdown of an industrial component brings significant advantages [19], such as reducing maintenance costs and increasing production capacity. Since all the machines of the same model share the same characteristics, they are prone to the same performance degradation trend over time. For this reason, the environmental condition experienced by each machine can be beneficial to others to understand the reasons behind, and so prevent, a component fault. In this context, when a smart manufacturing company buys a machine, it obtains the infrastructure needed to run FL training, too. Each local training resulting parameters are then sent to the vendor that aggregates them into a more generalized model able to predict the behavior of the machine under a wide spectrum of circumstances.

Therefore, due to the above consideration, FLaaS is designed to address the following scenarios.

1) FL training for a single client on an existing ML problem without the need of developing and tuning algorithms. For example, a smart manufacturing enterprise may want to model the temperature of a given machine to avoid overheating.
2) FL training between two or more clients to solve an existing task, which is the same for all the involved parties (e.g., predictive maintenance) giving access to a wider knowledge spectrum.
3) FL training two or more clients to solve a novel task not explored yet. For example, a smart manufacturing enterprise may want to model the temperature of a given machine during a specific month.

4) Allowing clients to specify the requirements to address while implementing FL training. Clients' requirements comprise quality metrics, aggregation strategy, and the number of nodes involved.

Although FLaaS can bring many advantages to third-party organizations, there are several challenges arising that need to be properly addressed.

A. Trustworthiness

Clients require transparency in the FL process, especially if they do not fully trust each other and/or the service provider. For example, the node in charge of aggregating models might have biases and prefer one update over another. In addition, it is important to check the models sent by clients, as they may be Byzantine. Aggregation of a malicious model could generate a global model with poor performance and/or backdoors [20], Therefore, to effectively allow unknown clients to collaborate, service providers have to guarantee that: 1) all partial models are equally treated without possible biases inducing to prefer some partial models over others and 2) malicious attempts to arbitrarily alter the global model are properly mitigated through a validation process of partial models.

Concerning the smart manufacturing example reported above, there are two potential targets for an attack: 1) the service provider (i.e., vendor) and 2) the customers (i.e., smart manufacturing industries). A malicious competitor of the service provider could be interested in joining the FL training as a client to negatively contribute to the global model or make the service unavailable, impacting the service provider's reputation and reliability. On the other hand, a contender of a third-party customer could buy that equipment only to participate in the FL process and degrade the performance of the global model, exposing the customer to potential machine failures.

These considerations lead us to claim that trustworthiness is one of the major requirements that service providers have to guarantee to offer an effective FLaaS. With this idea in mind, we propose the exploitation of blockchain and smart contracts to make more trustworthy the validation and aggregation processes of partial models. Many research works propose enriching FL with blockchain, but the use of blockchain is mainly devoted to avoiding a single point of failure, providing higher reliability, and tracking participants' contributions. This ensures accountability and maintains data providence [21], [22], [23], [24]. However, unlike our proposal, the validation and aggregation processes in these studies are usually performed off-chain, meaning they occur outside the blockchain network.

However, performing these operations off-chain significantly reduces the usefulness of the blockchain, while also reducing the benefits of its consequent overhead. Off-chain approaches, which do not rely on smart contracts, are neither public nor verifiable. Therefore, the results obtained do not represent solid evidence and could be challenged by clients. Furthermore, a participant may not have adequate or enough validation data to perform an accurate validation process. For example, in the case of predictive maintenance, a smart manufacturing enterprise may not have data on a particular fault and therefore could not assess whether the model can predict it.

Although keeping the validation on the blockchain avoids Byzantine contributions that could introduce backdoors in the global model, on-chain validation also brings challenges to address. In particular, validating a partial model through a smart contract implies that validation data are published on the blockchain. Hence, malicious contributors could intentionally craft an evil model that achieves satisfying performances on the validation data. Therefore, to offer a reliable FLaaS, service providers must implement security mechanisms that allow partial models to be validated using smart contracts without making validation data publicly available. Moreover, due to the transparency by-design nature of blockchain environments, service providers have to offer the possibility to verify how validation was performed.

B. Privacy-Preserving Techniques

Although one of the main advantages of FL is to avoid data exchange between the server and clients, personal information can still be inferred by analyzing the partial models submitted by the clients [25]. Therefore, to further improve privacy, privacy-preserving techniques are generally employed [26], [27]. Differential privacy (DP) is one of the most representative approaches [28]. It consists of adding artificial noise to the partial model before sending it for aggregation. However, while higher noise will result in higher privacy, the global model performance would be worst and a longer convergence time of the training process is likely to be required. Hence, an adequate tradeoff is needed to preserve privacy while guaranteeing satisfying performance. Another approach that is emerging in FL consists of using homomorphic encryption [29]. It is an encryption technique that allows operations to be performed on the encrypted data without having to decrypt them first. The result is provided in encrypted form and is equivalent, when decrypted, to that obtained by performing the same operation on plaintext data. In this way, clients encrypt partial models before sending them. The result of the aggregation is an encrypted global model that can be used by clients once decrypted.

C. Authentication and Authorization

Participants of FLaaS need to interact with each other transparently and securely. However, the highly distributed operating environment of FL hinders the adoption of centralized identities to identify clients and regulate their participation in FL training. Centralized approaches own and control clients' data that could be also shared with other services without their awareness. In addition, storing sensitive information in a unique server increases the risk of data leakage. Due to these considerations, a framework that offers FLaaS has to implement decentralized authentication and authorization mechanisms. Decentralized identities are only under the control of the data owner that decides with whom to share its information. The authorized access to FLaaS can be regulated through decentralized identifiers (DIDs) and verifiable credentials (VCs) [30]. A DID is a new type of identifier that enables verifiable, decentralized digital identity [31]. VCs are claims made by an issuer that states something about a

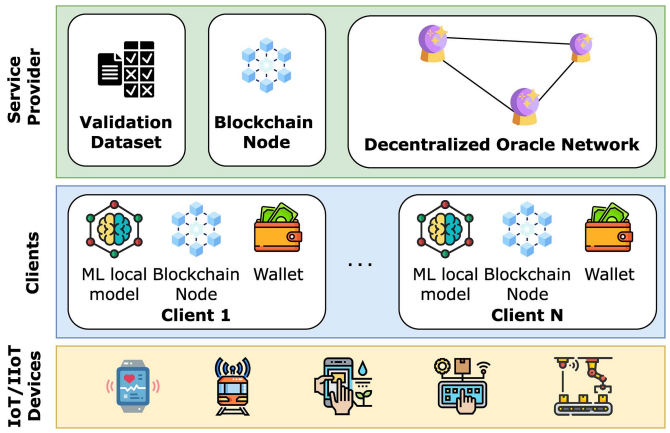

Fig. 1. TruFLaaS architecture.

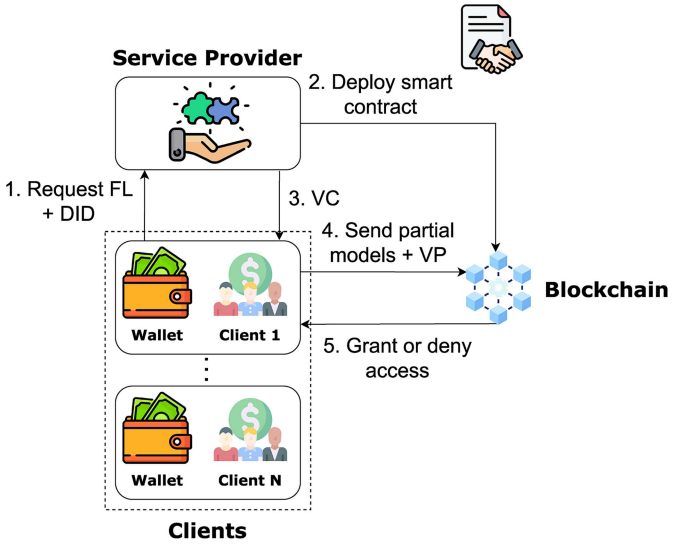

Fig. 2. Authorization workflow.

subject [32]. DIDs and VCs enable claim-based identity, a method of authenticating entities in other systems.

Specifically, to use FLaaS, clients need VCs with the necessary permissions to join the desired FL processes. Moreover, since participants may have different requirements, the service provider has to guarantee that the issued VCs allow clients to only join the FL training that satisfies its demands.

D. Incentive and Penalization

Clients are always reluctant to share their data, hence, incentives are needed to attract sufficient distributed training data and computation power. Therefore, to effectively involve as many positive participants as possible, resulting in a high-quality global model, service providers have to implement mechanisms to reward clients according to their contributions [33]. It sharpen that to correctively reward participant, avoiding low participation rate or financial loss, contributions have to be accurately evaluated. However, implementing incentives is not enough, because Byzantine participants may participate only to attempt to gain a reward. Thus, service providers must also determine penalization mechanisms to discourage spamming and incorrect computations, which could impact the quality of the global model.

III. TruFLaaS Architecture and Primary Design Choices

This section describes the architecture of TruFLaaS whose main components are highlighted in Fig. 1, and their interactions in Figs. 2 and 3. Clients interact with the service that offers FLaaS. The blockchain, smart contracts, validation set, and DON are the architectural entities that enable achieving a TruFLaaS. A DON is a middleware layer that enables to deliver off-chain validation data to the blockchain in a secure and reliable manner. The use of blockchain and smart contracts improves the trust of the participants in the FL process. Specifically, a smart contract validates partial models on a validation set provided by the DON. Then, it aggregates clients' contributions weighting them according to their level of trustworthiness. Honest participants are promoted to participate through incentives, while malicious adversaries are discouraged by penalization mechanisms. TruFLaaS provides the flexibility needed to meet the demands of different clients. To the best of our knowledge, TruFLaaS is the first designed and implemented framework that provides trustworthiness in the FLaaS paradigm and performs validation of partial models in FL by leveraging smart contracts and a DON.

A. Service Provider

The service provider is the entity that offers FLaaS to its clients for tasks that are usually worthy for all participants. For instance, it is noteworthy that all the smart manufacturing enterprises that use certain machinery are willing to prevent its breakdown. However, although the goal may be common, clients still may have different requirements in terms of metrics, number of involved nodes, and aggregation strategy. For example, a client may want to obtain the global model as soon as possible, hence, it can determine a threshold of participants that must be satisfied to aggregate collected partial models. On the other hand, another client may not be interested in retrieving the global model in a short time window since it may prefer to wait longer in order to collect a higher number of contributions. Therefore, the service provider has to implement a flexible service capable of meeting different demands. Clients provide such information to the service provider which in turn uses them to implement a smart contract that realizes an FL process compliant with them. The service provider is also responsible for registering clients and providing them with a valid identifier to interact with the blockchain. This way guarantees that only an identifiable and authorized client participates in the proper FL training.

Furthermore, since the service is directly offered by the service provider on a task that is under its control (e.g., predictive maintenance of its machines), we can assume that it has a validation data set large enough to validate partial models [8]. After each round of FL training, the service provider feeds a sample of this set into the blockchain. Such data are used to validate all the partial models before aggregating them. Given two distinct rounds, the validation sample has to be different

to avoid possible model forging attacks. Otherwise, malicious participants could exploit it to build a crafted partial model that passes validation checks. This could result in the introduction of backdoors in the global model that can compromise its integrity and effectiveness. Thus, to securely and reliably inject data into the blockchain, the service provider has to count on a DON that allows it to accurately fetch data off-chain and deliver it to the blockchain.

1) Decentralized Oracle Network: Oracles are trusted third entities that serve as bridges between blockchains and external systems. They enable smart contracts to make computations leveraging inputs and outputs from the real world. An oracle is a software component that queries, verifies, and authenticates external data sources and then relays that information to the blockchain. As previously mentioned, TruFLaaS relies on oracles to provide the validation data set to the smart contract. These oracles are responsible for feeding the smart contract with the validation data set managed by the service provider. This approach helps to ensure the integrity and security of the validation process, as the oracles act as trusted intermediaries between the participants and the smart contract. However, the usage of a single oracle leads to a central point of failure, which could contradict the decentralization principles of blockchain technology and lead to security vulnerabilities. The issue is known in the literature as the "blockchain oracle problem" [34]. To address this challenge, DONs have emerged as a solution [35]. A DON employs a combination of multiple independent oracle node operators and multiple sources of reliable data to provide decentralized and secure access to off-chain information. By leveraging the collective intelligence of multiple independent nodes, a DON helps to ensure the reliability and accuracy of data inputs into the blockchain network, while also maintaining the decentralization and security that blockchain promises.

In TruFLaaS, we use a DON as a middleware layer between the service provider and the blockchain. Without a DON, validation data have to be published on the blockchain. As a negative side effect, all participants may exploit published validation data for forging a partial model that, even if malicious, passes the validation phase. To ensure the integrity and security of the FL process, validation data should only be provided after the necessary requirements for performing aggregation have been met. For example, concerning the predictive maintenance use case, a malicious client may attempt to construct a partial model that achieves satisfying performance on the validation data although its model fails while estimating the remaining useful lifetime (RUL) when it falls under a certain threshold.

B. Client

Clients collect data provided by IoT and IIoT devices and use them to train local models, independently from the ML algorithm employed. Once the training is completed, the partial model is forwarded to a smart contract deployed on the blockchain. Thus, a client has a module to perform tasks related to the FL and hosts a node of the blockchain to join the service. Before joining an FL, the client has to express its willingness to join an FL. In case existing processes do not meet its demands, the client can provide new conditions to the service provider that meet them while setting up a novel FL training.

C. Blockchain Node

As discussed above, we leverage blockchain and smart contracts to improve the trustworthiness among unknown participants in the FL process. Therefore, the service provider has to deploy blockchain nodes. Clients may either run locally a blockchain node, as shown in Fig. 1, or connect to one of those deployed by the service provider. On the one hand, running a blockchain node can provide clients with direct visibility of FL processes. On the other hand, it comes with the potential downside of consuming a nonnegligible amount of client resources, which can be a challenge for clients with limited computing power or storage capacity. Therefore, it is important to carefully consider the tradeoffs between direct monitoring and resource consumption when deciding whether to run a blockchain node or connect to one of the proxy nodes and submit its partial model. This flexible configuration increases the ease of use of FL since each participant is not involved in the operations to manage a blockchain node, while it has to only train partial models and send them to the corresponding smart contract.

1) Validation and Aggregation: The validation and aggregation of the global model are performed through a smart contract, which is implemented according to the client's requirements. The smart contract first verifies whether the client is authorized or not to join the desired FL process. Then, before validating and aggregating partial models, it waits until the training requirements are satisfied. For example, if the aggregation strategy foresees that all participants have to provide their contributions, the smart contract waits until all the clients have submitted their partial models and then it sends a request for the validation set for that round. As anticipated above, the validation set is provided by the service through a DON. All the partial models are validated against the collected validation set and, if they achieve satisfying performance on the selected threshold, are considered in the aggregation phase.

2) DID-Based Access Control System: We regulate access to the FLaaS through DIDs and VCs. Each client has only one digital identity, which is a DID issued by the service provider, but has multiple claims (i.e., VCs) that prevent misuse of services and Sybil attacks [36]. Such identity information is not stored or controlled by other parties, rather they are kept in a wallet under the surveillance of the user, thus, improving both the control over the client's data and the degree of trust and security for external entities (e.g., apps or service providers) [37]. Our DID-based access control system comprises the following actors.

1) *Claim Holder:* To access the FLaaS, clients need VCs issued by the claim issuer and associated with their DID. A VC represents proof of membership for a specific FL training.
2) *Claim Issuer:* The service provider attests to the proof provided by the claim holder and generates a VC and signs it with its DID. Such a VC, which includes the

TABLE I
CONFIGURATION PARAMETERS

Parameter	Value
Minimum rounds	Positive integer
Minimum participants	Positive integer
Budget	Positive integer
Aggregation algorithm	JSON
Structure of the model	JSON
Privacy techniques	JSON
Metrics	JSON

claim holder's DID as the subject DID, is returned to the claim holder that will use it to join the corresponding FL process.

3) *Claim Verifier:* Verifying claims is implemented through a smart contract. A client signs a verifiable presentation (VP), which embeds a VC, with its DID and sends it to the claim verifier that checks if the client owns a valid VC to participate in the FL training for which it is applying.

IV. TruFLaaS Trustworthiness Protocol

This section discusses the main phases that enable achieving trustworthiness in an FLaaS environment. Let us consider a service provider s that offers FL training $f_l \in \mathcal{F}$ to its client set $\mathcal{C}$ to collaboratively train, according to given requirements, a global model mg_l on a given task. To join the FLaaS offered by s, a client $c_i \in \mathcal{C}$ must be already registered with that s. Once c_i is registered with s, it owns a DID issued by s that enables it to join the FLaaS.

A. Starting and Joining FL Training

A client c_i can either join an existing f_l or start a new one if the requirements implemented by existing processes do not satisfy its demands. TruFLaaS employs a robust authorization workflow, illustrated in Fig. 2, that leverages DIDs and VCs to regulate all interactions. In the following, we detail the steps involved in initiating a new f_l or becoming a member of an existing one.

1) c_i presents its DID and provides s with the requirements that f_l has to address. Table I summarizes the parameters that can be customized.
2) In case there are no preexisting smart contracts sc_l that meet the client's needs, s creates a novel sc_l that verifies and aggregates partial models according to the specified criteria. However, if such sc_l does exist, refer to step 3).
3) s returns to c_i a VC $vc_{i,l}$ signed with its DID that enables c_i to interact with the deployed sc_l.
4) once the local training is completed, c_i signs with its DID the previously obtained $vc_{i,l}$ generating a VP $vp_{i,l}$. Then, it provides such $vp_{i,l}$ and the partial model $mp_{i,l}$ to sc_l.
5) sc_l verifies the validity of $vp_{i,l}$ through the DID of s, which has released the $vc_{i,l}$, and subsequently grants or denies the participation to f_l.

It is worth noting that starting a new f_l is an expensive operation that should be avoided if there is existing training that already satisfies the client's demands.

B. Trust Level

Each client $c_i \in \mathcal{C}$ is assigned a trust level $t_{i,l} \in [0, 1]$. As pointed out in [38], most trust models in P2P networks distinguish trust toward a peer into direct and indirect. Direct trust is based on previous interactions with that peer, while indirect trust is based on that peer's global reputation. We denote with $TAR_{i,l}$ the *Transaction Acceptance Rate*, which is defined as

$$TAR_{i,l} = \frac{TA_{i,l}}{T_{i,l}} \tag{1}$$

where $TA_{i,l}$ is the number of accepted transactions and $T_{i,l}$ is the total number of transactions made, both referred to the f_l process. The *Global Trust Value*, which is denoted with GT_i, is defined as

$$GT_i = \frac{\sum_{j=1}^{N} t_{i,j}}{N} \tag{2}$$

where the trust levels are obtained by the clients in past or current FL processes. Thus, leveraging the direct and indirect trust, we calculate the trust level as follows:

$$t_{i,l} = \frac{GT_i + TAR_{i,l}}{2}. \tag{3}$$

These values are updated at each round, through the specific smart contract (Fig. 3, step 11). At this point, we also consider whether to revoke the $vc_{i,l}$ from client c_i in case its $TAR_{i,l}$ does not meet minimum requirements (i.e., pass a threshold). At each round, the average TAR μ_{TAR}, among all participants, and the corresponding standard deviation σ_{TAR} are calculated. Then, we estimate whether or not a client can, considering the number of remaining rounds, exceed the threshold value set to $\mu_{TAR} - \sigma_{TAR}$. In case it fails, the corresponding $vc_{i,l}$ is revoked and the client is excluded from f_l.

C. Validation and Aggregation

After having started a novel f_l or joined an existing one, a client is provided with the necessary $vc_{i,l}$ to contribute to that training. In the following, we detail the validation and aggregation workflow depicted in Fig. 3. These steps are repeated for each round k.

1) Each $c_{i,l} \in \mathcal{C}_l$ collects local data from the deployed IoT/IIoT devices.
2) Data are used to locally train a partial model $mp^k_{i,l}$.
3) Each $c_{i,l}$ provides $mp^k_{i,l}$ and $vp_{i,l}$, which is obtained by signing $vc_{i,l}$ through its DID, to sc_l that validate and aggregate all partial models MP^k_l. Algorithm 1 shows the algorithm implemented by the smart contract to validate and aggregate partial models.
4) sc_l forwards $vp_{i,l}$ to a smart contract responsible for authorizing participants.
5) This smart contract will grant or deny access to f_l. Specifically, it jointly verifies the validity of $vp_{i,l}$ and ensures that the embedded $vc_{i,l}$ has not been revoked.
6) Before aggregating all partial models MP^k_l, sc_l waits until the aggregation requirements are met and validates MP^k_l against a validation set $V^k_l \subset \mathcal{V}_l$ provided by a DON d.

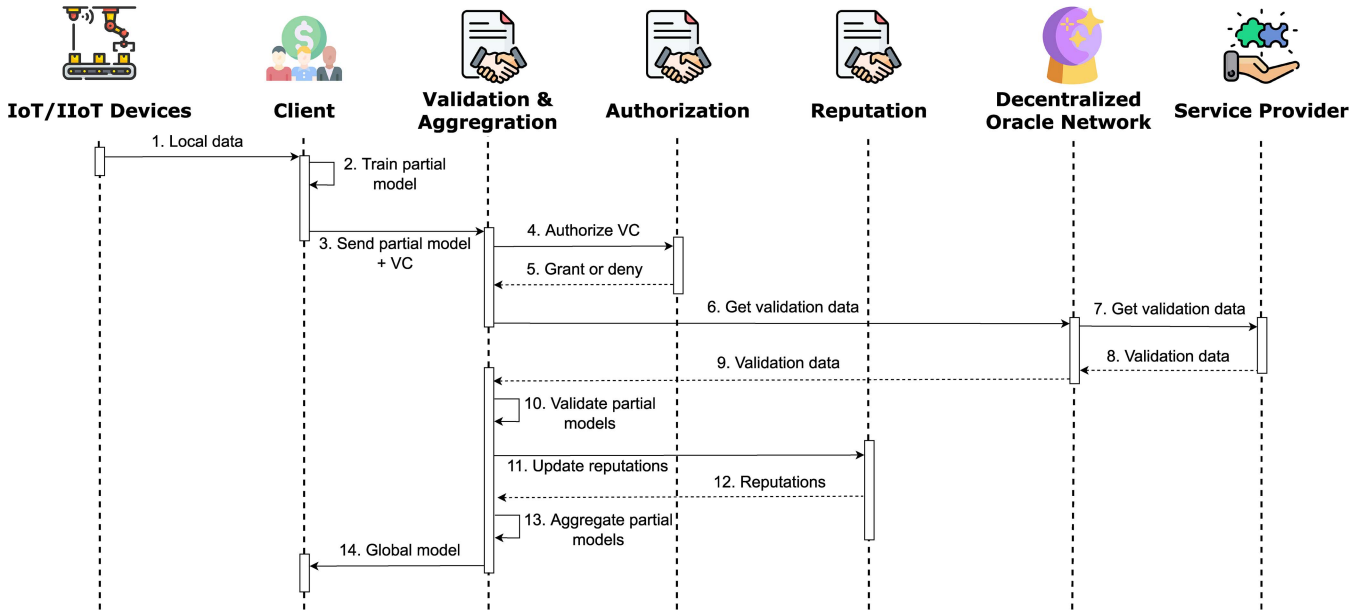

Fig. 3. Validation and aggregation workflow.

7) d requests V_l^k to s;
8) s provides V_l^k to d. Given two round j, z, where $j < z$, V_l^z must be $\neq V_l^j$; otherwise, $c_{i,l}$ could craft mp_i^z to achieve satisfying performance on a known V_l [].
9) d returns V_l^k to sc_l.
10) sc_l validates MP_l^k against V_l^k. To be accepted, a partial model must achieve performance equal to or better than a specific threshold. We employ the interquartile range (IQR) method for detecting outliers. This method does not use the median, mean, and standard deviation, being more robust to extremely large or small values. The IQR is calculated as $Q3-Q1$, where $Q1$ is the first quartile of the data, and $Q3$ is the third quartile. To detect outliers, we calculate the threshold as $Q1 - 1.5 * IQR$. Any data that falls below this value is considered an outlier and consequently discarded.
11) According to the collected metrics, sc_l sends the updated reputations to a smart contract employed to trace the t_i of each c_i.
12) The smart contracts calculate all the trust levels T^k of each $c_{i,l}$ and returns them to sc_l.
13) sc_l aggregate all validated all $mp_i^k \in MP_l^k$ weighting them according to the corresponding $t_i^k \in T^k$.
14) The global model mg_l^k is provided to all $c_{i,l}$.

It is worth outlining that transparent collaboration among smart contracts plays a key role in achieving a TruFLaaS. In particular, such a design choice is justified by the following considerations.

1) All MP_l^k are validated and aggregated without any biases, guaranteeing the correctness of gm_l^k.
2) Only $c_{i,l}$ having satisfying the process can join f_l.
3) Reputations of $c_{i,l}$ are calculated by a smart contract using as input the output of sc_l. Thus, we ensure the correctness of t_i for each participant.

Algorithm 1 Smart Contract—Validation and Aggregation

Input: $dids^k$, $partialModels^k$, $valSet^k$, $metric$
Output: $globalModel^k$
$acceptedPartialModels^k \leftarrow []$
$acceptedDids^k \leftarrow []$
$performances^k \leftarrow []$
for $i \leftarrow 1, partialModels^k.size()$ **do**
  $mp \leftarrow partialModels(i)^k$
  $performance \leftarrow evaluate(mp, valSet^k, metric)$
  $performances^k.push(performance)$
end for
$threshold \leftarrow generatedThreshold(performances)$
for $i \leftarrow 1, dids^k.size()$ **do**
  $did \leftarrow dids(i)^k$
  $mp \leftarrow partialModels(i)^k$
  $accepted \leftarrow false$
  if $performances^k(i) \geq threshold$ **then**
    $acceptedPartialModels^k.push(mp)$
    $acceptedDids^k.push(did)$
    $accepted \leftarrow true$
  end if
  $updateReputation(did, accepted)$
end for
$t^k \leftarrow getTrustLevels(acceptedDids^k)$
$globalModel^k \leftarrow \frac{\sum_{i=1}^{M} t(i)^k acceptedPartialModels(i)^k}{\sum_{i=1}^{M} t(i)^k}$

D. Incentives and Penalization

To start a novel f_l or join an existing one, clients use tokens that are by design the natural incentive mechanism for blockchain-based platforms. Tokens are purchased from s and earned by clients through positive participation. Moreover,

in order to participate in an already settled f_l, tokens are also required to discourage malicious behavior. Participants who provide incorrect contributions are penalized by having a portion of their tokens withdrawn in proportion to their contribution quality score (GT_i). This ensures that all participants have a vested interest in contributing high-quality work and helps maintain the integrity of the f_l. More in detail, at the time of the creation of a new f_l, the budget b_l (i.e., tokens) is locked up into the corresponding sc_l by the $c_{i,l}$ that initialized that f_l. This budget represents an incentive to promote participation to freshly started f_l. Indeed, at the end of f_l, it will be distributed among the participants $\mathcal{C}_l$ according to the corresponding $TAR_{i,l}$. The reward $r_{i,l}$ assigned to each c_i is calculated as follows:

$$r_{i,l} = b_l \frac{TAR_{i,l}}{\sum_{j=1}^{N} TAR_{j,l}}. \tag{4}$$

Such an incentive scheme fairly distributes b_l according to the contributions of all the $c_{i,l} \in \mathcal{C}_l$. For each $c_{i,l}$, the contribution corresponds to its $TAR_{i,l}$. It is clear that, given $c_{i,l}, c_{j,l} \in \mathcal{C}_l$, and $TAR_{i,l} > TAR_{j,l}$, it follows that $r_{i,l} > r_{j,l}$.

Furthermore, to deter malicious behavior, before joining f_l, each $c_{i,l}$ has to deposit an amount of tokens $d_{i,l}$ bounded by to $b_l(1/GT_i)$. Thus, the higher a client's reputation, the less it will have to deposit, and vice versa. This amount will be fully returned to the participant $c_{i,l}$ at the end of f_l only if the $TAR_{i,l}$ is greater than a threshold. Otherwise, the amount returned will be equal to $d_{i,l}TAR_{i,l}$. Such a mechanism is a strong deterrent to voluntarily submitting malicious or inaccurate models, as it would result in an economic loss of tokens.

V. Evaluation Results

To validate our proposal and compare it with the existing literature, we consider predictive maintenance and botnet attack detection use cases, which are of high interest for industrial deployment environments and call for data collection from multiple distributed sources. We first describe the implementation setup for our experiments and the employed data sets, then we present the details of the performed experiments, and, finally, we discuss the performance indicators that we have experimentally measured, by drawing some related considerations.

A. Implementation Setup

TruFLaaS can be integrated into any blockchain infrastructure that supports smart contracts and DONs. For instance, for the following assessment and evaluation, we have made TruFLaaS work with Hyperledger Fabric,[1] an open-source, modular, and extensible framework for deploying permissioned blockchains. Fabric-based applications are enterprise-grade and offer a high level of security, scalability, and performance [39]; in particular, Fabric smart contracts are written in general-purpose languages, such as Java, Go, and NodeJS. To implement the proposed validation protocol, we have implemented our smart contracts in NodeJS by using TensorFlow libraries. This choice is motivated by the need to recreate an ML model directly in the smart contract: TensorFlow is one of the few frameworks that implement ML also in JavaScript [40]. Concerning the DON, we have used Provable,[2] whose only requirement is to deploy a specific smart contract that acts as a connector between the blockchain and the outside world. Our experiments were run on a Python-simulated FL framework.[3]

B. Data Set

For the predictive maintenance use case, we selected the NASA Turbofan Jet Engine data set [41], which is a widely accepted and well-known baseline data set from NASA for engine degradation modeling. It enables estimating the RUL of the considered engine; the data set was generated through the simulation of the commercial modular aero-propulsion system. Specifically, it comprises four subdatasets, with temporal signals from 21 sensors (e.g., temperature and fuel flow ratio); each of the subdatasets consider different combinations of operational conditions and fault modes. To employ the data set effectively, first, we performed a data preprocessing step to remove features with nonconsistent values. In addition, since the training set does not present RUL values but only the number of time cycles of engine usage, we were forced to calculate them manually. For the purpose of the following evaluation, We assume that RUL decreases linearly over time so that it would have a value of 0 at the last time cycle of the engine: for each engine, RUL is calculated as $max\_time\_cycle - time\_cycle$; moreover, as usual for regression problems, we have normalized the input values. Finally, we have split the data set into training and testing subsets for 100 engines to replicate the behavior of an FL network during the training phase.

Concerning the botnet attack detection use case, we employed the N-BaIoT data set [42], which contains real traffic data gathered from nine commercial IoT devices authentically infected by Mirai and BASHLITE. Malicious traffic is divided into multiple different attacks (e.g., network scanning and firmware), thus, enabling us to use it for multiclass classification: 10 classes of attacks, plus 1 class of *benign*. To prepare the data for training, we first used a Label Encoder to convert the target value for each sample into a numerical value. The target value indicates the type of network traffic, either benign or belonging to one of the ten possible attacks. Next, we applied one-hot encoding to these values, resulting in a vector for each sample. Additionally, we normalize each feature by using a MinMaxScaler, which scales the data in the range of $[0, 1]$. To minimize the number of features, we implemented a feature selection mechanism based on an ExtraTreesClassifier. Tree estimators are utilized to compute feature importance through impurity calculations, which can subsequently be used in combination with the SelectFromModel meta-transformer to eliminate irrelevant features.

[1] https://www.hyperledger.org/use/fabric

[2] https://provable.xyz

[3] https://github.com/MMw-Unibo/TruFLaaS

TABLE II
BOTNET ATTACK DETECTION—TRUFLAAS EXPERIMENTS RESULTS

Exp.	Nodes %	Cross Entropy	Accuracy %	Precision %	Recall %	F1 %
#1	0	0.549	77.51	79.31	75.77	73.92
	10	0.461	75.83	74.15	76.42	70.51
	25	0.419	77.5	75.86	79.11	72.20
#2	0	0.390	77.55	81.13	79.47	73.38
	10	0.409	75.83	78.12	76.81	70.7
	25	0.387	74.16	71.43	77.53	69.39
#3	0	0.41	77.39	70.31	77.39	71.95
	10	0.41	77.38	70.29	77.38	71.94
	25	0.42	76.43	82.04	76.43	70.74

C. Experiments

To show the effectiveness of our solution, we conducted several experiments considering the application domains of predictive maintenance and botnet attack detection. In each of them, we considered both honest clients with a limited data set and malicious nodes, which aim to either disrupt the training process or introduce backdoors within the global model. We compare TruFLaaS against both the conventional baseline (i.e., no validation mechanisms) and TrustFed [21], i.e., a framework for fair and trustworthy FL. For the sake of fairness, in the predictive maintenance use case, we use the same FL model, configured as follows. The input layer takes `input_size` $\times$ 24 input neurons for each window. Two middle dense layers represent 24 $\times$ 24 neurons, while the output dense layer is mapped on 24 $\times$ 1 neurons. The ReLU activation function is used for all the layers and weights are adjusted through stochastic gradient descent (SGD) optimizer. Mean absolute percentage error (MAPE) is used to evaluate the accuracy of each model at the end of the final aggregation, while we calculated mean absolute error (MAE) to validate partial models and identify the most beneficial ones for training. However, TrustFed was not thoroughly validated against multiple data sets and models. Therefore, for the botnet attack detection scenario, we developed a novel FL model consisting of four layers. The first two layers are Dense layers with ReLU activation functions, comprising 64 and 32 neurons, respectively, followed by a Dropout layer with a rate of 0.2 to minimize overfitting. The final layer is a Dense layer with 11 neurons, one for each class to be predicted, utilizing a softmax activation function. Due to the nature of the problem being a multiclass classification, we used categorical cross-entropy as the loss function. Moreover, since this use case does not aim to solve a regression problem, in Table II, we report the final evaluation metrics for the experiments conducted.

1) Heterogeneous Data Distribution: First, we consider a scenario where the distribution of data, among honest clients, is heterogeneous. Hence, some nodes have more or fewer data samples than others. Having clients with heterogeneous data distribution is one of the major cases that justifies the adoption of FL: this situation is quite common in industrial environments since the size of enterprises directly affects the amount of data generated. We run this type of experiment while varying the number of participants and with different percentages of nodes having augmented data.

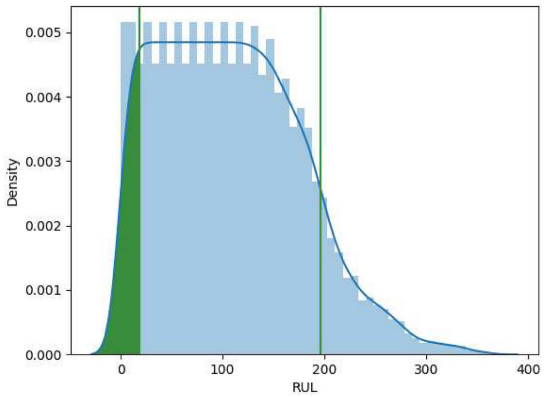

Fig. 4. NASA Turbofan Jan Engine—10th percentile distribution of training data.

2) Heterogeneous Data Distribution on Rare Cases: This experiment is a special case of the previous set. In FLaaS, clients may be interested only in a specific subtask (e.g., RUL under a given threshold or a specific botnet attack). We focus on a deployment environment where some nodes have no data on a particular class of events, which we define as *rare cases*. For example, there may be smart manufacturing enterprises that have not experienced the breakdown of specific machinery yet or have never been affected by a certain attack. In this experiment, for the predictive maintenance use case, we discriminate the data records accordingly to their RUL values. In particular, we tag as rare records all the samples inside a low percentile of a pseudo-normal distribution, by separating the low RUL values from the others. We identify, through statistical analysis, a subset of the data containing the data records with low RUL. To do that, we calculate the 10th percentile values on both the training and the validation sets. Since NASA data do not follow a normal distribution, we operate a standardization process to use *z-score* table to calculate exact areas for any given normally distributed populations. Mathematically, the standardization operation is described by the following formula:

$$z = \frac{x - \mu}{\sigma} \tag{5}$$

where z is the *z-score* value, x is the observation value, μ is the mean of the distribution, and σ is the standard deviation of the distribution. Figs. 4 and 5 illustrate the obtained percentiles on training and validation data sets, respectively. For the botnet attack detection use case, a multiclass classification approach is used. Thus, we consider the two attack classes with the lowest occurrence as rare cases. Specifically, as shown in Fig. 6, such classes are represented by junk and scanning attacks. Our experiments involve varying the number of nodes without rare cases and the strategy for discarding nodes. We test four strategies using two validation sets: one with only rare cases (Rares) and another with the same data distribution as the global test set (Overall). The first strategy entails discarding nodes that perform poorly on the validation set that contains only rare cases. The second strategy involves discarding nodes that perform poorly on the second validation set. The third strategy requires discarding nodes that perform poorly on both the first and second validation sets. Finally, the

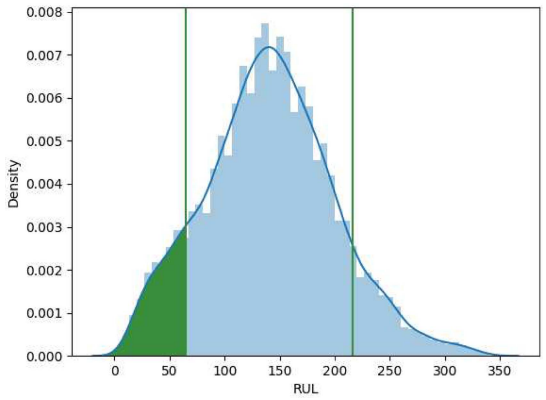

Fig. 5. NASA Turbofan Jan Engine—10th percentile distribution of testing data.

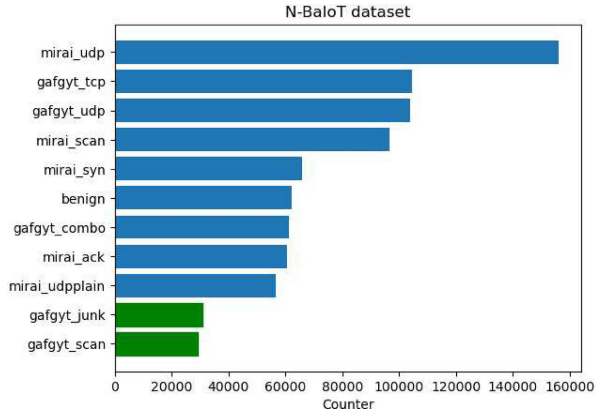

Fig. 6. N-BaIoT—Types of botnet attacks and their occurrences.

fourth strategy involves discarding nodes that perform poorly on either the first or second validation set.

3) Model Forging Attack: Ensuring the security and integrity of FL platforms is a major concern due to their widespread adoption in various domains. In model forging attacks, a malicious participant could craft a partial model to introduce backdoors into the global model or simply disrupt the training process. To assess the resilience of TruFLaaS against this class of attacks, we conducted several experiments with different percentages of malicious nodes. In these experiments, as done in TrustFed, we simulated malicious nodes' behavior by performing training on data containing random noise to corrupt the model.

D. Results and Associated Considerations

The experimental results reported in this section show that our solution outperforms conventional baselines and TrustFed under all the considered circumstances.

1) Heterogeneous Data Distribution: TrustFed only aggregates partial models whose accuracy falls in the interval identified by the neighborhood of the medium and standard deviation. However, the TrustFed approach neglects clients with a heterogeneous data distribution that results in high-quality partial models (which can achieve performance results that overcome the bound of the interval). Also, TruFLaaS discards partial models whose performance is below its threshold; on the opposite, in the aggregation phase, TruFLaaS involves all the partial models whose accuracy is more significant than the lower neighborhood of the medium and the standard deviation. Figs. 7 and 8 show how TruFLaaS outperforms TrustFed by better identifying clients' contributions with significantly larger data sets. TruFLaaS not only reaches the target accuracy faster than the others but also gains a greater advantage with the increase in the number of nodes with augmented data.

2) Heterogeneous Data Distribution on Rare Cases: In this type of experiment, for the sake of fairness, we have not compared TruFLaaS with TrustFed because the latter does not make any distinctions on the rare cases and our approach would certainly perform better. Figs. 9 and 10 highlight the experiment results by varying the number of nodes with no rare data and the strategy used to discard nodes with poor performance. We can observe that in the predictive maintenance scenario the first two strategies, i.e., discarding nodes that perform poorly on the validation set comprising only rare cases and discarding nodes that perform poorly on the validation set with the same distribution as the test set, performing better than the other aggregation strategies considered. In the botnet attack scenario, we can see the resilience of our solution to an increasing amount of nodes without rare data. The accuracy is not affected negatively by the higher amount of heterogeneous nodes.

3) Model Forging Attack: In the predictive maintenance use case, since the FL configuration employed by TrustFed was not leading to acceptable results in comparison with our solution, we increased the number of FL rounds to 100. Figs. 11 and 12 sharply outline how TruFLaaS is more robust than TrustFed against model forging attacks by varying the number of malicious nodes. This is mainly motivated by the fact that TrustFed, by using the mean and standard deviation to detect outliers, is less accurate in the presence of excessively large or small values. For example, there might be an outlier with such poor performance that would significantly shift the total mean. In this case, TrustFed ends up accepting outliers with performance that should not be accepted under nominal conditions. Moreover, by not weighing the partial models, aggregating an outlier will completely ruin the training performed up to that point. It is also interesting to observe that, in the predictive maintenance experiment, TrustFed performs worse than the baseline with 0 malicious nodes. This could be caused by the fact that TrustFed also removes nodes that reach performances much greater than the average.

On the contrary, TruFLaaS employs a more robust outlier detection algorithm, achieving in all cases very similar performance. Moreover, due to the use of weights based on the number of accepted transactions (i.e., level of trust), sporadic errors during the validation process would result in very little influence on the global model, without disrupting the whole training process. These results provide valuable insights into the robustness of our proposal and help ensure that it can effectively protect from model forging attacks.

VI. Related Work

One of the biggest challenges facing the widespread adoption of FL in real-world scenarios is the lack of trust among

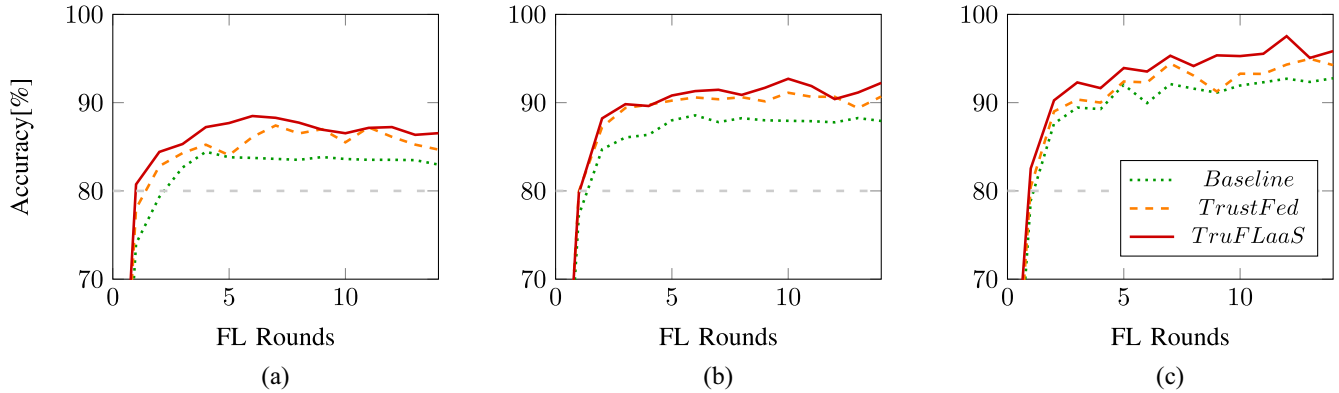

Fig. 7. Predictive maintenance: heterogeneous data distribution—Accuracy comparison of different node selection strategies with (a) 0 nodes having augmented data, (b) 10 nodes having augmented data, and (c) 25 nodes having augmented data.

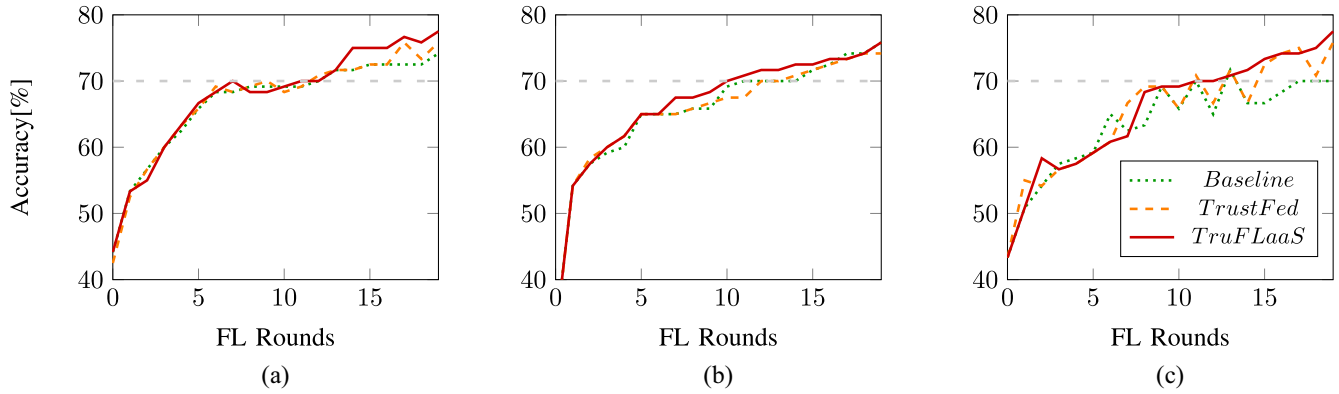

Fig. 8. Botnet attack detection: heterogeneous data distribution—Accuracy comparison of different node selection strategies with (a) 0 nodes having augmented data, (b) 10 nodes having augmented data, and (c) 25 nodes having augmented data.

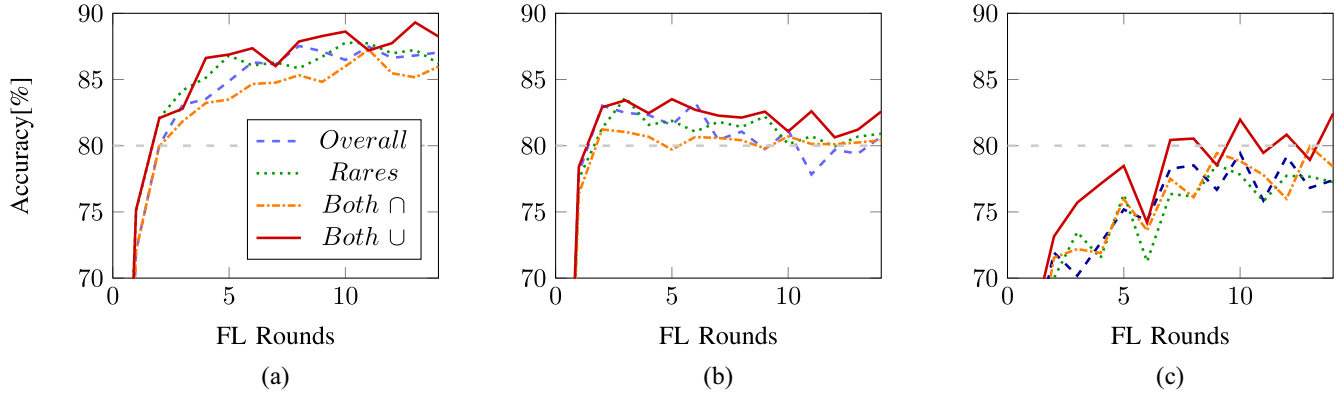

Fig. 9. Predictive maintenance: heterogeneous data distribution on rare cases—Accuracy comparison of different node selection strategies with (a) 0 nodes without rare data, (b) 10 nodes without rare data, and (c) 25 nodes without rare data.

unknown participants. However, in recent years, there has been a growing interest to design novel solutions that can increase the trustworthiness and fairness of FL environments. Many of these proposals are made possible by blockchain technology, which, in some cases, only ensures the correctness of the generated global model by replacing the centralized server (as in [47] and [48]). In this section, we review some of the most relevant works that aim to enhance the trustworthiness of FL. Table III provides an overview of such approaches and their main limitations, while Table IV summarizes their key features.

A. Accountability and Fairness

Blockchain technology can be utilized as a reliable data source that offers all participants a consistent and transparent view of the stored data. For this purpose, Lo et al. [23] employ blockchain to enable accountability and improve fairness in FL systems. Data-model provenance is granted through the blockchain that stores the hashed value of data, local, and global model versions. To increase fairness, the authors present an algorithm that dynamically samples training data from classes poorly represented according to the inverse of

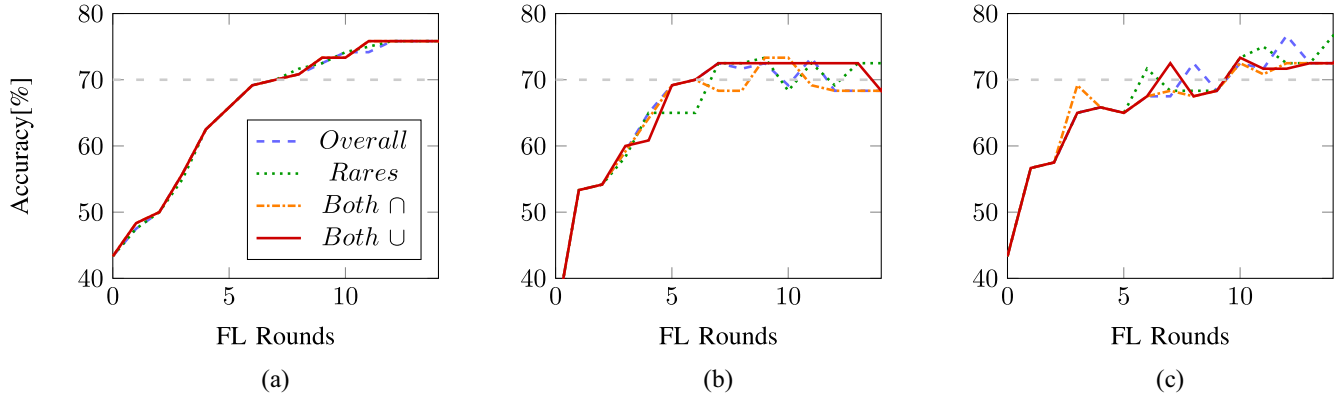

Fig. 10. Botnet attack detection: heterogeneous data distribution on rare cases—Accuracy comparison of different node selection strategies with (a) 0 nodes without rare data, (b) 10 nodes without rare data, and (c) 25 nodes without rare data.

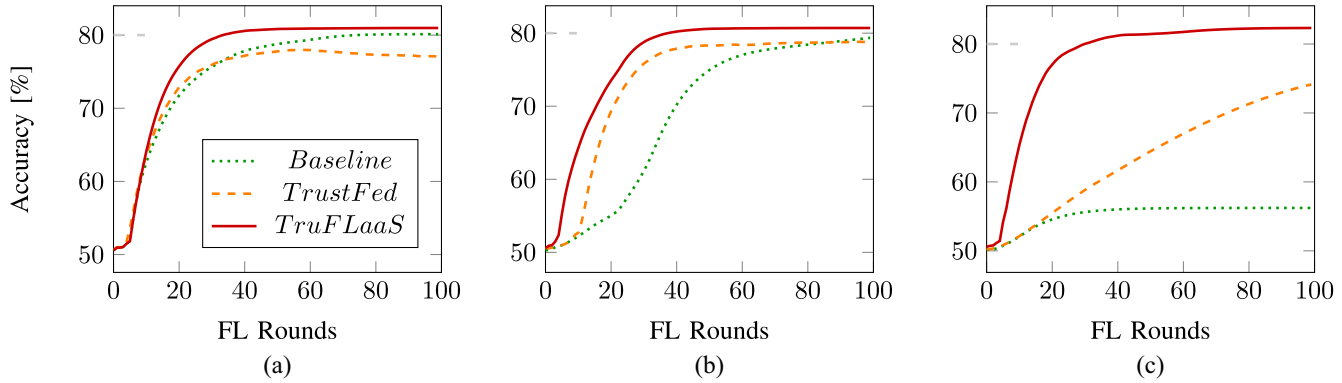

Fig. 11. Predictive maintenance: model forging attack—Accuracy comparison of the compared approaches with (a) 0 malicious nodes, (b) 10 malicious nodes, and (c) 25 malicious nodes.

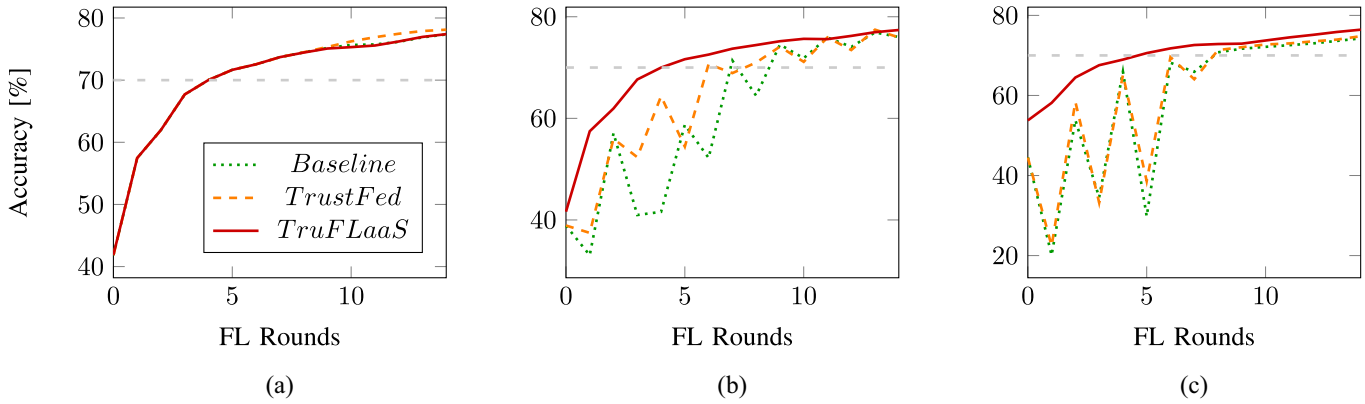

Fig. 12. Botnet attack detection: model forging attack—Accuracy comparison of the compared approaches with (a) 0 malicious nodes, (b) 10 malicious nodes, and (c) 25 malicious nodes.

the weight distribution of the data set used for testing. Their approach can contribute to building fairer models in a scenario where each client trusts the other. However, they do not prevent malicious participants are excluded from the model aggregation which is, as discussed above, one of the major concerns in an FLaaS context. Abdel-Basset et al. [24] presented FedTrust, a blockchain-orchestrated edge intelligence framework for trustworthy cyberattack detection in IIoT. However, their approach does not bring remarkable novelties in terms of validation of partial models since the verification phase consists in allowing fog nodes to collect the block comprising the partial models from all contributors to calculate the global model. In this work, trustworthiness is intended as one of the main targets of cyberattacks for the IIoT that can be protected through a distributed temporal convolution generative network.

B. Validation Mechanisms

The absence of adequate validation mechanisms led to the aggregation of any submitted model, leaving room for the introduction of malicious backdoors. Consequently, many papers in the literature have been dedicated to introducing novel validation methods. TrustFed [21] is a blockchain-based

TABLE III
COMPARISON OF RELATED WORK APPROACHES AND LIMITATIONS

Work	Approach	Limitations
Li et al. [8]	Partial models are validated against a secondary server-side dataset and included in the aggregation only if they achieve satisfying performance	Depend on a central server for the aggregation of partial models, exposing the system to vulnerabilities associated with the centralized aggregation
Remhan et al. [21]	Remove malicious participants through statistical outlier detection techniques	Honest clients that may have performed significantly better are excluded from the aggregation
Chen et al. [22]	A set of devices validates partial models through a local dataset and casts a vote on the legitimacy of the model. Devices with negative scores are considered malicious and removed	The validation is affected by the quality of the local dataset and by the number of compromised participants
Lo et al. [23]	Hashed value of data, local, and global model versions are stored on a blockchain. To build fairer models, training data are sampled from classes poorly represented according to the inverse of the weight distribution of the dataset used for testing	Failure to consider the possibility of malicious participants in the training process can result in security vulnerabilities that compromise the integrity of the entire system
Basset et al. [24]	Fog nodes collect the block comprising the partial models and calculate the global model	Assume that all participants will behave correctly, overlooking the possibility of malicious actors
Keng et al. [43]	Clients are selected through their reputations that are stored on a consortium blockchain, which are measured according to previous task completion	In some cases, discarding nodes in advances can be unfeasible and the lack of validation for partial models can compromise the overall reliability
Cao et al. [44]	Partial models are combined based on their weights, which are determined by their trust scores. These trust scores are computed by measuring the degree of similarity between the clients' updates and that of the server	Rely on a centralized server and the quality of the dataset used by the server. Honest clients that achieve better performance could be assigned with a low trust score
Wang et al. [45]	Leverage a local proof to ensure the authenticity of the partial models	Partial models are aggregated using a centralized server. Malicious contributions could be aggregated if their proofs are correct
Gao et al. [46]	Verify the integrity of partial models and the correctness of their aggregation through BLS signature and multi-party security	Do not validate the quality of the partial models. Their approach involves a centralized trusted authority

TABLE IV
COMPARISON OF RELATED WORK BASED ON THEIR FEATURES

Work	Blockchain	Smart Contracts	Validation Mechanism	Validation Dataset	Statistical Analysis	Weight Contributions	Reputations	Incentive
Li et al. [8]	-	-	X	X	-	-	-	-
Remhan et al. [21]	X	X	X	-	X	-	X	X
Chen et al. [22]	X	-	X	X	X	-	X	X
Lo et al. [23]	X	-	-	-	-	-	-	-
Basset et al. [24]	X	-	-	-	-	-	-	X
Keng et al. [43]	X	-	-	-	X	-	X	X
Cao et al. [44]	-	-	-	-	X	X	-	-
Wang et al. [45]	-	-	X	-	X	-	-	-
Gao et al. [46]	-	-	X	-	-	-	-	-
TruFLaaS	X	X	X	X	X	X	X	X

framework for fully decentralized cross-device FL systems. It provides fairness by removing malicious participants from the training distribution through statistical outlier detection techniques. While it employs blockchain and smart contracts to maintain participating devices' reputations. However, their approach removes outliers including contributions of clients that may have performed significantly better by having a local training set bigger than all the other participants. Experimental results demonstrate that TruFLaaS outperforms TrustFed under different circumstances. Chen et al. [22] presented a blockchain-based decentralized FL framework that validates partial models through a decentralized validation mechanism. During each round of the FL training, a set of devices is selected to act as validators. All the local updates are validated by all of them using their local data set. After having observed the experimental results, a validator casts its vote on the legitimacy of each model. Collecting votes from multiple validators enables removing malicious devices that are associated with a negative model. Such an approach improves robustness since validating operations can be properly performed despite several compromised validators.

Li et al. [8] proposed a dynamic verification strategy to decrease the influence of abnormal customers on the global model. Similarly to our work, the authors use a secondary server-side data set to validate the contribution of each client. Only the partial models that achieve satisfying accuracy are involved in the model aggregation process. However, their approach still suffers from all the weaknesses related to using a centralized server for model aggregation. Recently, there has been a rise in novel approaches that ensure the correctness of partial model aggregation without being dependent on blockchain and smart contracts. However, these new techniques still have a vulnerability to a single point of failure, which can compromise the entire system. Wang et al. [45]

proposed PTDFL, a decentralized FL scheme that prioritizes privacy and trustworthiness. Their method employs a local proof mechanism to verify that the partial model submitted by the client is the genuine output of their training. However, it is worth noting that a malicious model could still be aggregated if the corresponding proof is correct. The previous two approaches do not rely on blockchain technology, which means they are vulnerable to the drawbacks associated with utilizing a centralized server for model aggregation. Gao et al. [46] developed SVeriFL a novel protocol based on BLS and multiparty security that enables verifying the integrity of partial models provided by clients and the correctness of their aggregation. However, like previous work, the authors do not prioritize the quality of the submitted partial models. Furthermore, their protocol relies on a trusted authority, which introduces an additional element of centralization and potential vulnerability.

C. Reputations and Weighted Contributions

To improve trustworthiness among participants, clients can be selected according to their reputations and partial models can be weighted based on a trust score associated with each participant. Kang et al. [43] proposed to evaluate the reputation, stored on a consortium blockchain while selecting the participants of an FL training. According to the authors, reputation is measured from its training task completion history with the past behaviors of good or unreliable activities. Their approach is specifically for mobile devices. Indeed, it may not apply to scenarios, with a restricted number of clients that impede discarding nodes in advance. In these cases, only a small number of participants may satisfy the threshold making FL pointless. When the number of clients is not huge, all the contributions can be relevant to improve the quality of the global model.

Cao et al. [44] introduced FLTrust, a Byzantine-robust FL method that protects against malicious attacks by training a server model using a small, manually collected clean training data set as if it were a client. FLTrust assigns trust scores to each local model update based on its similarity with the server model update. Such trust scores are then used for weighting local model updates and generating the global model. Beyond the weaknesses of the centralized server, FLTrust heavily depends on the training data set provided to the server. In addition, honest clients that perform much better could be assigned with a low trust score.

VII. Conclusion

Service providers can simplify and promote the use of FL by offering it as a service. FLaaS significantly reduces the overhead and technological knowledge to develop and tune algorithms and tools for collaboratively training a global model on a shared task. However, despite the discussed advantages, designing and developing effective FLaaS still arise several technical challenges that have to be properly addressed.

In particular, the lack of trustworthiness among unknown participants is one of the major factors that hinders the adoption of FL in real-world scenarios. To overcome this concern, this article proposes a novel blockchain-based architecture and approach, which transparently validate partial models by leveraging blockchain, smart contracts, and a DON. In particular, before being aggregated, partial models are validated against a sample of the validation set, different for each round, provided by the service provider through a DON. TruFLaaS uses smart contracts to keep the level of trust of each participant, which is used to weigh the contributions of each partial model during the aggregation phase. The extensive experimentation work described in this article shows that TruFLaaS outperforms conventional baselines and the state-of-the-art literature for the detection of malicious nodes under different relevant families of use cases, i.e., when forging an ad-hoc model to pass the validation process, or discarding low-quality models on rare data, or making a global model converge by varying the number of malicious nodes.

References

[1] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of Things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," *IEEE Access*, vol. 8, pp. 23022–23040, 2020.

[2] "The Internet of Things: A movement, not a market." IHS Markit. Accessed: Oct. 6, 2022. [Online]. Available: https://cdn.ihs.com/www/pdf/IoT_ebook.pdf

[3] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for Internet of Things: Recent advances, taxonomy, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1759–1799, 3rd Quart., 2021.

[4] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[5] M. Mendula and P. Bellavista, "Energy-aware edge federated learning for enhanced reliability and sustainability," in *Proc. TEC Workshop Trustworthy Edge Comput.*, 2022, pp. 349–354.

[6] "Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation)." EU. 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng

[7] P. Bellavista, L. Foschini, and A. Mora, "Decentralised learning in federated deployment environments: A system-level survey," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–38, Feb. 2021.

[8] Y. Li, Y. Chen, K. Zhu, C. Bai, and J. Zhang, "An effective federated learning verification strategy and its applications for fault diagnosis in industrial IoT systems," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16835–16849, Sep. 2022.

[9] J. Zhang, Y. Wang, K. Zhu, Y. Zhang, and Y. Li, "Diagnosis of interturn short-circuit faults in permanent magnet synchronous motors based on few-shot learning under a federated learning framework," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8495–8504, Dec. 2021.

[10] U. Ahmed, J. C.-W. Lin, and G. Srivastava, "5G-empowered drone networks in federated and deep reinforcement learning environments," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 55–61, Dec. 2021.

[11] U. Ahmed, G. Srivastava, and J. C.-W. Lin, "Reliable customer analysis using federated learning and exploring deep-attention edge intelligence," *Future Gener. Comput. Syst.*, vol. 127, pp. 70–79, Feb. 2022.

[12] C. Mazzocca, N. Romandini, M. Colajanni, and R. Montanari, "FRAMH: A federated learning risk-based authorization middleware for healthcare," *IEEE Trans. Computat. Social Syst.*, early access, Oct. 10, 2022, doi: 10.1109/TCSS.2022.3210372.

[13] P. Boobalan et al., "Fusion of federated learning and industrial Internet of Things: A survey," *Comput. Netw.*, vol. 212, Jul. 2022, Art. no. 109048.

[14] N. Kourtellis, K. Katevas, and D. Perino, "FLaaS: Federated learning as a service," in *Proc. 1st Workshop Distrib. Mach. Learn.*, 2020, pp. 7–13.
[15] R. Philipp, A. Mladenow, C. Strauss, and A. Völz, "Machine learning as a service: Challenges in research and applications," in *Proc. 22nd Int. Conf. Inf. Integr. Web-Based Appl. Serv.*, 2020, pp. 396–406.
[16] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.
[17] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.
[18] D. C. Nguyen et al., "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.
[19] R. K. Mobley, *An Introduction to Predictive Maintenance*. Amsterdam, The Netherlands: Elsevier, 2002.
[20] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Stat.*, Aug. 2020, pp. 2938–2948.
[21] M. H. U. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, "TrustFed: A framework for fair and trustworthy cross-device federated learning in IIoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8485–8494, Dec. 2021.
[22] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," 2021, *arXiv:2101.03300*.
[23] S. K. Lo et al., "Toward trustworthy AI: Blockchain-based architecture design for accountability and fairness of federated learning systems," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3276–3284, Feb. 2023.
[24] M. Abdel-Basset, N. Moustafa, and H. Hawash, "Privacy-preserved cyberattack detection in industrial edge of things (IEoT): A blockchain-orchestrated federated learning approach," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 7920–7934, Nov. 2022.
[25] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 2512–2520.
[26] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, "Privacy-preserving multiobjective sanitization model in 6G IoT environments," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5340–5349, Apr. 2021.
[27] C.-W. Lin, T.-P. Hong, and H.-C. Hsu, "Reducing side effects of hiding sensitive itemsets in privacy preserving data mining," *Sci. World J.*, vol. 2014, Apr. 2014, Art. no. 235837.
[28] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.
[29] J. Park and H. Lim, "Privacy-preserving federated learning using homomorphic encryption," *Appl. Sci.*, vol. 12, no. 2, p. 734, 2022.
[30] J. Geng, N. Kanwal, M. G. Jaatun, and C. Rong, "DID-eFed: Facilitating federated learning as a service with decentralized identities," in *Proc. Int. Conf. Eval. Assess. Softw. Eng.*, 2021, pp. 329–335.
[31] "Decentralized identifiers (DIDs) v1.0." W3 Recommendation. 2022. [Online]. Available: https://www.w3.org/TR/did-core/
[32] "Verifiable credentials data model v1.1." W3 Recommendation. 2022. [Online]. Available: https://www.w3.org/TR/vc-data-model/
[33] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.
[34] G. Caldarelli, "Understanding the blockchain oracle problem: A call for action," *Information*, vol. 11, no. 11, p. 509, 2020.
[35] L. Breidenbach et al., *Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks*, Chainlink Labs, San Francisco, CA, USA, 2021.
[36] D. Maram et al., "CanDID: Can-do decentralized identity with legacy compatibility, Sybil-resistance, and accountability," in *Proc. IEEE Symp. Security Privacy (SP)*, 2021, pp. 1348–1366.
[37] O. Avellaneda et al., "Decentralized identity: Where did it come from and where is it going?" *IEEE Commun. Standards Mag.*, vol. 3, no. 4, pp. 10–13, Dec. 2019.
[38] D. D. S. Braga, M. Niemann, B. Hellingrath, and F. B. D. L. Neto, "Survey on computational trust and reputation models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–40, Nov. 2018.
[39] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, 2019, pp. 536–540.
[40] D. Smilkov et al., "TensorFlow.js: Machine learning for the Web and beyond," in *Proc. Int. Conf. Mach. Learn. Syst.*, vol. 1, 2019, pp. 309–321.
[41] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics Health Manag.*, 2008, pp. 1–9.
[42] Y. Meidan et al., "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.
[43] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 72–80, Apr. 2020.
[44] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," 2022, *arXiv:2012.13995*.
[45] L. Wang, X. Zhao, Z. Lu, L. Wang, and S. Zhang, "Enhancing privacy preservation and trustworthiness for decentralized federated learning," *Inf. Sci.*, vol. 628, pp. 449–468, May 2023.
[46] H. Gao, N. He, and T. Gao, "SVeriFL: Successive verifiable federated learning with privacy-preserving," *Inf. Sci.*, vol. 622, pp. 98–114, Apr. 2023.
[47] Z. Yang, Y. Shi, Y. Zhou, Z. Wang, and K. Yang, "Trustworthy federated learning via blockchain," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 92–109, Jan. 2023.
[48] Y. Qu et al., "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.

Carlo Mazzocca (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Naples Federico II, Naples, Italy, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Bologna, Bologna, Italy.

His research interests mainly include security mechanisms based on distributed ledger technologies, and authentication and authorization solutions for the cloud-to-thing continuum.

Nicolò Romandini (Graduate Student Member, IEEE) received the M.Sc. degree in computer science engineering from the University of Bologna, Bologna, Italy, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering.

His research focuses mainly on blockchain, cybersecurity, and machine learning, and how to integrate them into IoT domains.

Matteo Mendula (Graduate Student Member, IEEE) received the M.Sc. degree in software engineering from the University of Bologna, Bologna, Italy, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering.

His main research interests include big data processing and distributed learning on the edges of the network. In particular, his research relates to architectural aspects and machine learning-enhanced techniques in fog computing scenarios.

Rebecca Montanari (Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Bologna, Italy, in 2001.

She has been a Full Professor with the University of Bologna, since 2020, where she carries out her research in the area of information security and the design/development of middleware solutions for the provision of services in mobile and IoT systems. Her research is currently focused on blockchain technologies to support various supply chains, including agrifood, manufacturing, and fashion and on security systems for Industry 4.0.

Paolo Bellavista (Senior Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Bologna, Italy, in 2001.

He is currently a Full Professor with the University of Bologna. His research interests include middleware for mobile computing, QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimization in wide-scale and latency-sensitive deployment environments.

Prof. Bellavista serves on the editorial boards of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON SERVICES COMPUTING, *ACM Computing Surveys*, *ACM Transactions on Internet of Things*, and *Pervasive and Mobile Computing* (Elsevier). He is the Scientific Coordinator of the H2020 IoTwins Project (https: www.iotwins.eu).

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement