End-to-End Verifiable Decentralized Federated Learning

***Abstract*—Verifiable decentralized federated learning (FL) systems combining blockchains and zero-knowledge proofs (ZKP) make the computational integrity of local learning and global aggregation verifiable across workers. However, they are not end-to-end: data can still be corrupted prior to the learning. In this paper, we propose a verifiable decentralized FL system for end-to-end integrity and authenticity of data and computation extending verifiability to the data source. Addressing an inherent conflict of confidentiality and transparency, we introduce a two-step proving and verification (2PV) method that we apply to central system procedures: a registration workflow that enables non-disclosing verification of device certificates and a learning workflow that extends existing blockchain and ZKP-based FL systems through non-disclosing data authenticity proofs. Our evaluation on a prototypical implementation demonstrates the technical feasibility with only marginal overheads to state-of-the-art solutions.**

***Index Terms*—federated learning, blockchain, off-chain computations, verifiability, zero-knowledge proofs, zokrates**

I. Introduction

Federated learning (FL) [1] helps to overcome confidentiality and network challenges of machine learning in distributed settings like the Internet of Things (IoT) by separating the responsibilities of learning and aggregation [2]. The learning is executed on distributed workers to keep confidential learning data protected. In each learning cycle, only updated learning parameters are shared and aggregated into a global model that represents the collective learning progress and is returned to the workers for the next learning cycle. However, the splitting and outsourcing of responsibilities in the FL process introduces security issues in distributed environments where parties do not trust each other or a central aggregator, and hence, a strong need for the verification of FL tasks.

Decentralized FL through blockchains removes the need for a central aggregator that may fail or corrupt the result aggregation unnoticeably. Global tasks are executed through smart contracts in a transparent and tamper-resistant manner allowing participating nodes to verify aggregation correctness by participating in the consensus protocol. As such, blockchain-based FL has been applied in different domains and associated challenges addressed in a large body of literature [3]–[8]. While blockchains add a first notion of verification to decentralized FL systems, they are unsuitable for executing the local learning procedures due to confidentiality and scalability constraints.

To prevent model poisoning [9], verifiable decentralized FL systems have been proposed that advance blockchain-based FL through verifiable off-chain computation (VOC) where local model updates are executed using zero-knowledge proofs (ZKP) [10]–[13]. By verifying these ZKPs on the blockchain, the computational integrity of local learning becomes verifiable throughout the network without disclosing confidential learning inputs. However, VOC-based FL systems are not end-to-end: Learning data can still be corrupted by workers prior to the learning.

End-to-end verifiable decentralized FL systems add a third notion of verification of data sources and data authenticity, thereby protecting against Sybil attacks [14] and data poisoning [9]. This is especially relevant in IoT settings where the learning cannot be executed on constraint IoT devices but is outsourced to an intermediary node run by the same worker. For example, in smart energy applications [5], [6], learning is outsourced from smart meters to workstations or in smart healthcare applications [7], [8], from medical devices to hospital facilities.

The verification of authenticity proofs of learning data and certificates of device identities on the blockchain, however, encounters an inherent conflict of transparency and confidentiality. Signatures created on the learning data do not apply to the updated learning parameters after the learning and their verification on the blockchain discloses confidential learning data to the blockchain network. Furthermore, the verification of device certificates requires the disclosure of certificate attributes like the device public key which represent security leaks and must be protected, e.g., to prevent key search attacks.

Addressing this problem, in this paper, we suggest a first *end-to-end verifiable decentralized FL system* that makes the integrity and authenticity of data and computation verifiable, from certified edge devices to the blockchain's secure parameter storage. Our system builds upon state-of-the-art approaches leveraging blockchains for aggregation tasks and zkSNARKs for local learning tasks, e.g., [10], and extends them through integrated authenticity proofs of certified sensor devices. For that, we make three individual contributions:

1) First, we propose a system model that extends VOC-based FL systems through an integrated perspective of certified sensor devices acting as data sources. We derive verification objectives from a motivational healthcare use case and refine the problem of conflicting properties of confidentiality and transparency.
2) Second, we present a two-step proving verification (2PV) procedure realized in a registration and a learning workflow as the two central procedures for our system. The registration workflow onboards certified sensor devices by verifying their device certificates without disclosing confidential attributes. The learning workflow

provides local mode updates by verifying both, the integrity of the learning and the authenticity of the learning data without disclosing them.

3) Third, we evaluate our system proposal through a prototypical implementation using ZoKrates [15] and initial experimentation of both workflows. The results demonstrate that our solution only adds a marginal overhead to previous comparable implementations [10] but achives end-to-end verifiability.

In the remainder of this paper, we first provide background information in Section II, then discuss related work in Section III, and present our system model in Section IV-B. Based on that, we describe both workflows in more detail in Section V, present our implementation in Section VI, and the evaluation in Section VII. We discuss remaining issues in Section VIII and finally conclude in Section IX.

II. Background

We start describing the foundations of federated learning (FL) and zkSNARKs as central concepts of our system design.

A. Federated Learning

FL offers a confidentiality-preserving and network-efficient architectural framework for machine learning tasks in distributed settings. As depicted in Figure 1, a single global model GM is iteratively trained through a large set of distributed *workers* that locally execute learning tasks on confidential learning data LD in each cycle. The resulting local model updates LM consisting of learning parameters, such as weights w and biases b, are then integrated into a global model GM through an *aggregator* that, in decentralized FL systems, can be executed on blockchains. The global model represents the collective learning insights of the network and is returned to the workers for the next learning iteration. This mechanism enables each worker to benefit from the collective training insights garnered by other workers while safeguarding potentially sensitive inputs from disclosure and reducing network loads [16]. As an aggregation method, Federated Averaging is often applied which calculates a weighted average of the local weights w_k [17].

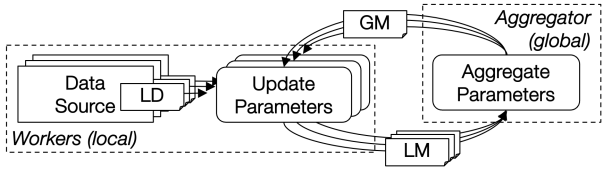

Fig. 1: Federated Learning Overview

B. ZkSNARKs and ZoKrates

Zero-knowledge succinct non-interactive arguments of knowledge (ZkSNARKs) represent a specific category of non-interactive zero-knowledge proofs (NIZK) known for their compact proof sizes and efficient verification times. The zkSNARKs procedure can be conceptualized through three main operations:

- The *setup* ($setup(ecs, srs) \rightarrow (ZK_{prov}, ZK_{verif})$) generates a public asymmetric key pair derived from the executable constraint systems (ecs). The ecs encodes the program logic in a provable representation, and a circuit-dependent structured reference string (srs) is used. Both proving and verification keys (ZK_{prov}, ZK_{verif}) are tied to the ecs. This process assumes a secure disposal of the srs to thwart any attempts at producing fake proofs.
- The *proving* ($P(ecs, x, x', w, ZK_{prov}) \rightarrow \pi$) occurs in two steps: Firstly, a witness w is created by executing the ecs on the public inputs x and the private inputs x'. The tuple (x, x') constitutes the proof arguments. The witness w signifies a valid variable assignment for the ecs inputs. Subsequently, the proof π is generated from the witness using the proving key.
- The *verification* ($V(\pi, x, ZK_{verif}) \rightarrow \{0, 1\}$) takes place on-chain, evaluating the proof π and the public inputs x using the verification key ZK_{verif}.

ZkSNARKs are particularly well-suited for applications in Verifiable Off-Chain Computing (VOC) where the verification is executed on the blockchain. VOC helps to overcome the privacy and scalability limitations of blockchains by offloading computation without compromising the blockchain's integrity. *ZoKrates* [15] is a language and toolbox that facilitates the development of zkSNARKs-based VOC for Ethereum [18]. The ZoKrates language provides a convenient interface for developers to define proving logic and the ZoKrates toolbox supports the compilation into ecs, the setup, the witness computation, and proof generation, as well as the creation of verifier contracts in Solidity.

III. Related Work

We position our contributions within related research for verifiable and decentralized federated learning (FL) [19] comprising blockchains for verifiable global tasks, zero-knowledge proofs (ZKP) for verifiable learning tasks, and authentication schemes against data poisoning.

Verifiable Aggregation with Blockchain: Blockchain-based FL has been proposed in different domains like smart energy grid [5], [6], smart healthcare [7], [8], or smart homes [20] and been researched under different objectives. Protection against *leakage of confidential information* from learning parameters is especially important for blockchain-based FL where local models become accessible by the blockchain network. Approaches leveraging differential privacy techniques applied to the learning data have been proposed in [11], [21], [22]. As another challenge, *fairness* refers to an equal and balanced contribution of the workers and has been addressed, for example, in [11], [23], [24] through accountability and incentive mechanisms. Furthermore, high transaction costs that incur through blockchain-caused overheads present a problem in constrained environments. To reduce such costs, approaches have been proposed advocating more efficient FL implementations [25], using permissioned blockchains implementing lightweight consensus protocols by design [26], or executing the aggregation off-chain using

ZKPs to preserve computational integrity [27]. While these approaches represent important lines of work, we set a different focus in this paper and consider such privacy, fairness, and performance features complementary to our system.

Verifiable Local Learning with ZKPs: Blockchain-based FL has been advanced by executing local learning as verifiable off-chain computations (VOC) with zkSNARKs, e.g., in [10]–[13] In [10], blockchain-based FL has been extended through zkSNARKs-based local learning. A similar system has been proposed in [11] that additionally leverages blockchains for worker incentives and applies a differential privacy scheme to prevent information leakage. In [12], another VOC-based FL system is proposed where inference from local models is prevented through the secure sum protocol. The authors of [13] propose a similar system for mobility environments where connected vehicles represent learning nodes and zkSNARKs create local models verifiable on a domain-specific blockchain system for aggregation. In this work, we conceptually extend these approaches through the verification of authenticity proofs and device identities using the prototype of [10] as our reference implementation.

Verifiable Data Sources and Data Authentication Authenticity proofs in blockchain-based FL systems have been proposed to establish accountability and mitigate poisoning and Sybil attacks. In [28], an anonymous authenticity scheme based on a Public Key Infrastructure (PKI) is proposed for workers in a blockchain-based FL system that leverages allows for holding workers accountable without revealing their identities on the blockchain. Similar approaches are proposed in [4] and [3] for vehicle-based workers. These approaches create authenticity proofs on computed learning parameters allowing for model and data poisoning despite the achieved accountability. Instead, in this paper, we prove the authenticity of learning data, however, without disclosure.

End-to-End Integrity Workflows in dApps: Outside the FL domain, end-to-end workflows have been proposed that leverage authentication schemes and zkSNARKs for data provisioning in blockchain-based decentralized applications (dApp) [29]–[31]. In [29] trustworthy pre-processing in sensor data provisioning workflows has been proposed for blockchain-based IoT applications using zkSNARKs and TEEs. Wan et al. [31] propose a zkSNARKs extension for signature-based data authentication applied to compute off-chain data in a manner that is verifiable on smart contracts. Similarly, Park et al. propose Ziraffe [30] as an approach for authenticating web-based data sources using zkSNARKs. Such end-to-end approaches have been applied to verify credentials of dApp users [32] where data sources are trustworthy identity issuers or to verify carbon footprints [33] where data sources are certified sensor devices. From a dApp perspective, we introduce data provisioning workflows to verify device certificates and local learning in FL dApps.

IV. Model

In this section, we first derive verifiability objectives from a motivation scenario, then we provide an overview of the system architecture and finally refine our problem statement.

A. Motivating Scenario and Verifiability Objectives

In smart healthcare applications, e.g., for remote patient tracking [7], [8], machine learning can help predict a patient's health state. The learning process takes health measurements, e.g., blood pressure, heartbeats, or temperature, as inputs and returns prediction classes like good, fair, critical. Such data is collected from medical devices distributed across hospitals, doctor's offices, and patient's homes, e.g., in cases of immobile patients. Medical devices undergo a strict certification procedure before they can be bought on the market.

As a problem for such smart healthcare applications, these measurements contain highly confidential information about a patient's health that fall under regulatory procetions like the Health Insurance Portability and Accountability Act (HIPAA) [34]. Through federated learning, this health data can be kept under the control of the learning nodes, i.e., the patient or the responsible hospital, and must not be shared externally.

Due to the criticality of the prediction, it is of utmost importance that the learning procedure is executed correctly. Verifiable FL [19] helps to prevent corruption from faulty processes or malicious behavior. Verifiability objectives must be taken from the perspective of the affected parties. Consequently, we propose the following design objective for an end-to-end verifiable decentralized FL system.

- Obj1 The workers should be able to verify that the aggregation of the local model updates into a global model has not been corrupted to exclude aggregation attacks.
- Obj2 The workers should be able to verify the computational integrity of the local model updates to exclude model poisoning.
- Obj3 The workers should be able to verify that the data applied for the local learning updates is of authenticity and originates only from the expected certified devices to exclude data poisoning through workers.

B. System Overview

Building upon the state-of-the-art in verifiable decentralized FL and aiming toward the previous objectives, we now introduce the general system model that consists of three layers.

1) Aggregation Layer: The aggregation layer contains system components for executing global FL tasks that are set up through the task initiator. To achieve *obj1*, these components are executed on a blockchain infrastructure as smart contracts. The *registration contract* manages the device identities. As a registration condition for a device, the smart contract verifies that they are certified under the same public key infrastructure (PKI). The *aggregation contract* contains the logic for aggregating the local learning parameters into a global model. As a condition for these local models to be aggregated, the integrity of the local learning procedure and the authenticity of the data need to be verified. The *verification contract* provides verification services to the aggregation and the registration contract. It is responsible for verifying the proofs provided through the learning and registration workflows.

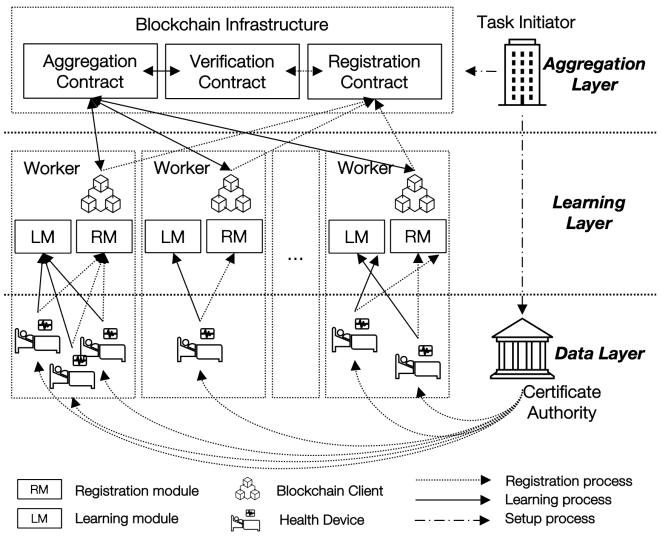

Fig. 2: Overall System Architecture

2) Learning Layer: The learning layer contains components for the local model updates that are executed on the workers' local infrastructures. To protect against model poisoning and fulfill *Obj2*, we assume that relevant local tasks are executed through zkSNARKs as described in Section II-B. Applying them as verifiable off-chain computations (VOC), the local model updates become verifiable on the blockchain without disclosing confidential inputs. Each worker hosts two zkSNARK-enabled system components: The *registration module* contains all logic to create proofs that allow for onboarding valid devices without disclosing certificate attributes. The *learning module* contains all the logic to create proofs of data authenticity and integrity of the learning procedure. Furthermore, workers connect to the smart contracts through a locally hosted blockchain client.

3) Data Layer: As a distinguishing feature of our system, we separate the data layer from the learning layer. The data layer contains data sources that are accessible to the workers and generate learning data for the local model updates. This data contains confidential information that must not be disclosed to parties other than the associated workers. We assume these sources to be *certified sensor devices* like smart meters, medical devices, or onboard devices that undergo a rigorous certification procedure including, for example, gauging, sealing, and securely installing the device. Certificates are managed in a public key infrastructure (PKI) and are issued through a *certificate authority* (CA) that attests to the well-functioning and security of the devices.

C. Problem Statement

Through zkSNARKs-based learning and blockchain-based aggregation, the integrity of computations in the FL process can be verified through the workers and respective attacks mitigated. However, achieving *obj3* is hindered by an inherent conflict of confidentiality and verifiability. We identify two specific challenges:

Ch1 To allow workers to verify that the expected data has been used for the local model updates, the data authenticity must be verified on the blockchain. Authenticity proofs, e.g., through cryptographic signatures, require the data to be revealed during verification. If done on the blockchain, confidential information contained in the learning data is revealed to the blockchain network.

Ch2 The verification of device certificates helps to build trust in the learning data. However, certificate attributes like the device public key must not become globally accessible to third parties as they may allow for insights into the internals of the device and with that may be used by external attackers.

V. System Design

Addressing the previously identified challenges, we present a *two-step proving and verification* (2PF) procedure that is applied to realize two workflows that integrate into the previous system model: a registration workflow for device identities addressing *ch2* and a learning workflow for locally updated learning parameters addressing *ch1*. The 2PF procedure involves three steps.

1) The *attestation* is executed on the data layer. A trusted party, i.e., the certificate authority or the certified sensor device, attests to confidential input data creating an authenticity proof verifiable through the worker nodes.
2) The *proving* is executed on the learning layer. The distrusted workers create proofs of registration or learning depending on the workflow, each verifiable on the blockchain. These proofs contain the verification of the authenticity proofs from the data layer, thereby preventing disclosure of confidential inputs on-chain.
3) The *verification* is executed on the aggregation layer. The verifier contract verifies the submitted proofs from the workers and, if successful, provides the registration or learning outcome to the respective registration or learning contract.

In the following, we first describe the system initialization and then each workflow in more detail.

A. System Initialization

To set up the system, the task initiator creates the necessary artifacts and deploys the smart contracts. The artifacts provisioning to the workers is described in Section VIII.

To initiate the *registration contract*, the task initiator obtains the PKI's root key (RK'_{pub}) from the CA and anchors it into the contract. The root key is required to validate that only the device certificates under the expected PKI are accepted and only corresponding device handles (DH) are registered.

For the *aggregation contract*, the task initiator implements the aggregation strategy, e.g., averaged-based [17], and creates the initial learning parameters that will be input to the first learning cycle and then updated in each iteration.

For the *verification contract*, the task initiator creates one proving and verification key pair (ZK_{prove}, ZK_{verif}) for the registration workflow and one for the learning workflow by

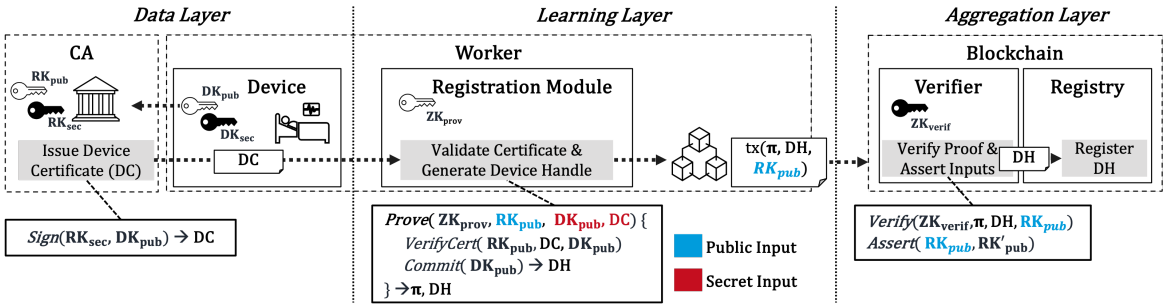

Fig. 3: One-time **Registration Workflow** for Non-disclosing Device Certificate Verification.

executing the zkSNARK setup respectively. It specifies the proving logic executed in the registration and learning module respectively, compiles it into a ecs, and then executes the setup as defined in Section II-B. The verification key ZK_{verif} is submitted to the verification contract and, with that, the registration logic bound to the proving key ZK_{prove} and ecs.

B. One-Time Registration Workflow

Successful device registration represents the prerequisite for participation in the federated learning system. Consequently, the registration workflow is executed *once* for each sensor device that contributes input data to the learning procedure. As depicted in Figure 3, it starts at the certificate authority (CA) and ends at the registration contract.

1) Attestation: The CA issues the device certificate (DC) by signing the device's public key (DK_{pub}) using the root secret key (RK_{sec}) such that $Sign(RK_{sec}, DK_{pub}) \rightarrow DC$. While DK_{pub} and DC are considered confidential to hide the device identity from potential attackers, RK_{pub} is considered public. The DC and the DK_{pub} are stored on the device and accessible to the worker.

2) Proving: The proving is executed using the proving key ZK_{prove} in the registration module of the worker that registers the device on the blockchain. The proving takes the device certificate DC and the device public key DK_{pub} as private input and the public root key RK_{pub} as public input and returns a non-disclosing device handle: $Prove(ZK_{prove}, DK_{pub}, RK_{pub}, DC) \rightarrow DH$. Proving comprises two steps:

- First, the device certificate is verified using the root public key: $VerifyCert(RK_{pub}, DC, RK_{pub})$.
- Second, a non-disclosing device handle is created as the commitment to the device's public key: $Commit(DK_{pub}) \rightarrow DH$.

On successful execution, the resulting proof π, the DH, and the public input RK_{pub} are wrapped into a blockchain transaction and submitted to the verifier contract.

3) Verification: The verification is executed by the verifier contract using the verification key ZK_{verif}. It takes the proof, the device handle, and the public root key as inputs: $Verify(ZK_{verif}, \pi, DH, RK_{pub})$. This assures that the computation has correctly been executed. In a second step, the contract asserts that the root key RK_{pub} has been used for proving by comparing it against the root key RK'_{pub} anchored by the task initiator: $Assert(RK_{pub}, RK'pub)$ If successful, the device handle DH is registered on the registration contract and can now be used to verify the data authenticity of the learning workflow.

C. Iterative Learning Workflow

The learning workflow is executed in each cycle of the federated learning system by each worker. It reaches from the certified device and to the aggregation contract and is initiated once the global model of the current cycle is available on-chain.

1) Attestation: The data source produces the learning data LD in a predetermined batch size and attests to it by signing the batch using the device secret key: $Sign(DK_{sec}, LD) \rightarrow \mu$ As will further be explained in Section VI, we assume that signature is created on a commitment to the learning data $comm(LD)$ for efficiency reasons and, as such, μ additionally contains $comm(LD)$. The learning data LD and the signature μ are then provided to the worker's learning module.

2) Proving: The proving is executed in the learning module using the proving key ZK_{prov} and, in addition to the local parameter update comprises authenticity-related operations. The worker obtains the learning data LD and the signature μ from the device, the current global model GM from the aggregator contract, the device handle DH from the registration contract, and executes the prove on these inputs: $Prove(ZK_{prove}, DH, DK_{pub}, \mu, DH)$. The proving logic consists of three steps and returns the updated parameters as local model lm and the proof π.

- First, the signature μ over the learning data is verified using the device public key: $VerifySign(DK_{pub}, \mu, LD)$. For that, the commitment of the learning data must be recreated and compared to the one contained in μ.
- Second, the mapping of the device handle and the device public key is asserted: $Assert(DK_{pub}, DH)$. For that, the commitment to the DK_{pub} is created and compared to the DH obtained from the blockchain.
- Third, the local model LM is computed using the learning data and the global parameters as inputs: $LocalLearn(GM, LD) \rightarrow LM$.

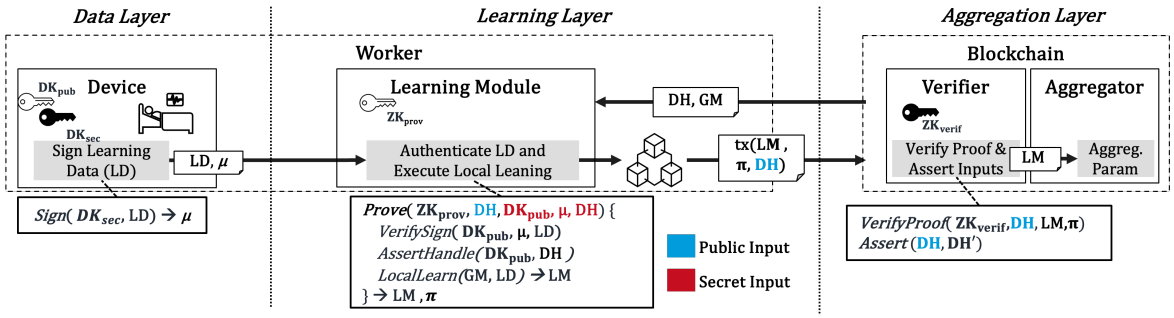

Fig. 4: Iterative **Learning Workflow** for Non-disclosing Verification of Model Integrity and Data Authenticity.

Upon successful proving, the newly computed local model LM, the proof π, and the public input DH are submitted as a blockchain transaction to the verifier contract.

3) Verification: The verification is executed through the verifier contract. First, the proof is verified using the verification key ZK_{verif} based on arguments contained in the blockchain transaction: $VerifyProof(ZK_{verif}, DH, LM, \pi)$. Then, the contract asserts that a valid device public key has been applied during proving by checking if the device handle is registered at the registration contract: $Assert(DH, DH')$. If successful, the verified local model is applied for aggregation by the aggregator contract.

VI. Implementation

We implemented the previous system as a proof-of-concept (PoC)[1]. To establish comparability of the prototype with the state-of-the-art, our implementation builds upon and extends the reference implementation of [10]. The PoC comprises software components for attestation, i.e., CA and devices, proving, i.e., learning and registration module, and smart contracts.

Attestation with CA and Devices: We implemented the CA and the device as web servers allowing for decoupled provisioning of data to the proving modules. For simulation purposes, we omitted the intermediate certificate storage on the device. As a signature algorithm, we decided on the EdDSA using the alt_bn128 (babyjubjub) curve that allows for verification in the zkSNARKs-based proving environment. For efficiency reasons, we decided to create the signature on hashed inputs. While not critical for signing small certificate attributes, hashing becomes relevant for signing large batches of learning data. As a hash algorithm, we decided on Poseidon [35], given its high performance for zkSNARKs-based usage. To create these signatures, we leveraged the PyCrypto[2] library that supports the creation of signatures in a zkSNARK-friendly format. Poseidon hashes were pre-computed with ZoKrates.

Proving Modules: Building upon previous work, we implemented the same simple feedforward neural network as in [10]. The matrix calculation is executed on the hidden layer based on the input arguments, the prediction is determined through an argmax function, and the loss function is implemented as the mean squared error. The local model consists of the weights and biases as learning parameters. To implement zkSNARKs-based proving in the learning and registration module, we used the ZoKrates toolbox and language (compare Section II-B) with Groth16 proving scheme and the alt_bn28 (babyjubjub) elliptic curve. The ZoKrates language is compiled into the ecs that is used for the one-time setup and the iterative witness computation and proof generation.

Smart Contracts: The verification contract for each zkSNARK program was generated using the ZoKrates toolbox which directly integrates the verification key into the verification logic of a Solidity contract. The aggregation contract was adopted from previous work given the same FL logic. It maintains two versions of the weight vector and the bias matrix, one used to implement incoming updates and one representing the latest learning state. After a learning cycle terminates (based on block time), the latest learning state changes to the updated one. The registration contract contains a key-value registry of identifiers and device handles that is filled after successful certificate verification and requested for each learning update. The contracts are deployed on a virtual Ethereum blockchain using Truffle and Ganache.

Challenges: By implementing the zkSNARKs-based logic we faced the following challenges: For one, the ZoKrates language only supports arithmetic operations on positive integer values, however, learning data often contains negative and floating point numbers. Data can be translated into positive integers by upscaling the input data which, however, can result in integer overflows. Taking this into account, we adopted the strategy of previous work [10] and reimplemented arithmetic operations in the ZoKrates program. Next, we were to decide if the authenticity proofs were heavy on the signature algorithm, i.e., signing large plain learning data, or on the hash function, i.e., hashing learning data prior to signing. We observed the best performance on large batch data in a hash-heavy approach using the Poseidon hash function in Merkle trees. A root hash was constructed from the batch data and signed during attestation. The same root hash was then recreated inside ZoKrates to verify the signature.

[1]https://anonymous.4open.science/r/End-to-End-Verifiable-Decentralized-Federated-Learning-8167/

[2]https://github.com/Zokrates/pycrypto

TABLE I: Average execution time and memory consumption for varying batch sizes

	Registration workflow	Batch size: 10		Batch size: 20		Batch size: 30		Batch size: 40	
		[12]	Proposed	[12]	Proposed	[12]	Proposed	[12]	Proposed
Compile Time [sec]	5.288	61.816	77.099	169.564	193.236	323.723	351.811	518.96	545.239
Constraints [count]	150,696	2,236,596	2,398,897	4,518,156	4,690,549	6,799,716	6,981,385	9,081,276	9,273,517
Compile Output Size [GB]	0.307	0.960	1.292	1.938	2.286	2.917	3.279	3.895	4.273
Setup Time [sec]	4.135	43.784	47.336	84.857	88.099	119.889	122.897	170.191	174.361
Proving Key Size [GB]	0.064	0.981	1.032	1.976	2.031	2.703	2.761	3.967	4.028
Verification Key Size [Byte]	5689	31437	35701	31437	35701	31437	35701	31437	35701
Witness Computation Time [sec]	1.975	7.888	9.991	15.973	18.161	24.289	26.328	32.802	34.565
Computed Witness Size [GB]	0.00594	0.089	0.095	0.179	0.186	0.270	0.278	0.361	0.369
Proof Generation Time [sec]	3.565	28.484	31.362	56.417	59.656	73.256	76.77	113.769	117.69
Generated Proof Size [Byte]	2699	14317	16241	14317	16241	14317	16241	14317	16241
Proving Time [sec]	5.540	36.518	42.573	72.434	79.616	97.548	106.115	146.574	155.294
Total Execution Time [sec]	11.215	38.780	45.032	74.759	82.511	99.923	108.77	149.029	159.01
Verification Cost [GAS]	559376	2315527	2597842	2363812	2687877	2303386	2638482	2306273	2614425

VII. Evaluation

To evaluate our system, we conducted experiments on our implementation for both workflows. While the registration workflow is evaluated on its own, the learning workflow is evaluated in comparison with experiments of [10]. Therefore, we aligned our experiments with the one of the reference.

A. Experimental Data and Setup

Datasets for registration and learning were selected in accordance with the system model. As inputs for the registration workflow, we used an EdDSA public key as a certificate attribute and input to the registration workflow. For learning workflow, instead, we used the data, obtained from the UC Irvine Machine Learning Repository[3] that consists of sensor data generated by wearables. It was collected at 5-minute intervals from 8 subjects performing 19 pre-defined activities. As in [10], we condensed the original dataset containing 45 features and 19 prediction classes to a dataset of 9 features and 6 classes to match the input vector of the learning model and the size of the hidden layer's weights and biases.

Iterations of the experiments depend on the workflow type. Given the one-time registration workflow, there was no need for iterative executions. Instead, the learning was repeated 300 times for each batch size using different batch sizes of 10, 20, 30, and 40 elements. We measured the system performance with a single worker on a single machine.

B. Results

Constraint numbers of the ecs used in the proving module demonstrate the complexity of the proving logic. As depicted in Table I, the proving registration is considerably cheaper than the proving the parameter update. However, our solution added only a relatively small number of constraints compared to the reference implementation without authenticity proofs.

Execution Times of the proving may impact the duration of the learning cycles if it takes too long. For the registration workflow, the total proving took 5.5403 seconds. Added to a negligible certificate creation of 0.0816 seconds and a verification time of 2.822 seconds, the overall execution time for the device registration process is around 10 seconds which we deem acceptable for a one-time execution. Compilation and setup took 5.288 sec and 4.135 sec respectively. For the learning workflow, there was no significant difference in the proving. The witness and proof creation time took up to 3.921 (Batch size: 40) longer. Similar proportional differences can be observed for the compilation and setup which, however, is non-critical given that they are only executed for system initialization. Overall, the operational learning workflow takes about 10 seconds longer compared to the reference.

Transaction Costs are critical given redundant blockchain transaction processing and expensive consensus protocols. Given a constant proof size of zkSNARKs, the verification costs vary depending on the size of the public inputs and the computational results. The costs for verifying the registration proof account to 559376 Gas. The transaction costs for learning workflow are considerably higher due to the model aggregation. However, our implementation only adds a small cost overhead compared to the ones of the reference which may be caused by the additional device handle operation.

Accuracy of the prediction as the learning outcome is specific to the learning experiments. The trend of the scores is shown in Figure 5 after repeating 300 epochs while varying the batch size. As the batch size increases, the model achieves higher scores more rapidly and with greater accuracy. Since both systems utilize the same training algorithm, the difference in learning performance is not prominent. A relatively low accuracy score (a maximum accuracy of 0.62 when the batch size was set to 40) is derived since there is only one node participating in federated learning to update the global model. However, as the number of workers increases, better scores can be obtained.

VIII. Discussion

The evaluation demonstrated the practicality of the proposed system workflows. In this section, we revisit aspects of attack resistance and discuss opportunities for managing computational overheads and system artifacts.

A. Security and Trust

We discuss security and trust assumptions for each role in our system.

Workers are verifiers and potential attackers at the same time. Recalling the verifiability objectives, our system enables

[3]https://archive.ics.uci.edu/dataset/256/daily+and+sports+activities

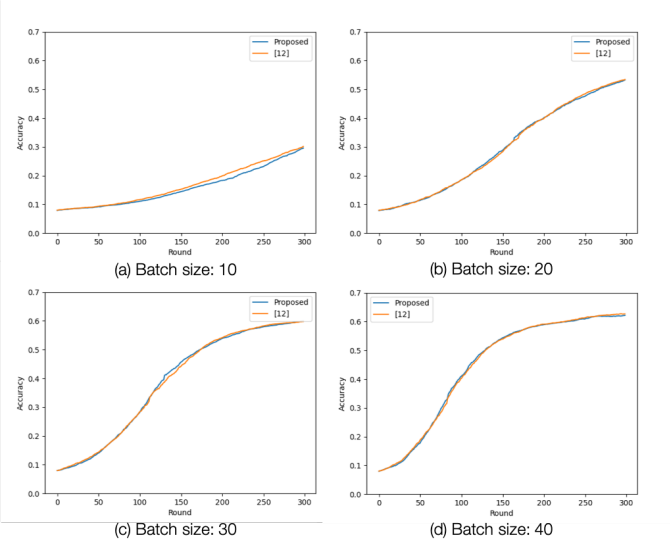

Fig. 5: Accuracy for various batch sizes

workers to verify and protect against attacks on global tasks through blockchains, local tasks through zkSNARKs, and device certificates and learning data through integrated non-disclosing authenticity proofs. Beyond that, lazy workers may resubmit already accepted model updates that would pass the 2PV procedure again. Such replay attacks can be prevented also from external attackers through batch counters that are signed by the devices together with the learning data and checked during proving and verification on-chain.

Task Initiator: By executing the setup, a malicious task initiator may corrupt the zkSNARKs setup by keeping the structured reference string (SRS) to create fake proofs. To remove trust assumptions from the setup procedure, secure multiparty computation (sMPC) has been established to make the setup verifiable across involved parties [36]. By applying sMPC to our system, workers can verify the setup procedure and trust assumptions can be removed from the task initiator.

Certificate Authority: With the responsibility of issuing device certificates that confirm the well-functioning of the devices, the CA bears trust assumptions. For one, the CA could execute a Sybil attack by issuing fake certificates to faulty or non-existent devices. Furthermore, the CA could hinder well-functioning devices from participating by refusing certificate issuance. Similar effects could be achieved through man-in-the-middle attacks where the communication channel between CA and device is intermitted through an attacker faking messages in both directions. Addressing such security risks research has been conducted on Decentralized Public Key Infrastructure [37], [38]. We consider a system extension towards DPKI as a possible future work.

B. Managing Computational Overheads

ZkSNARKs-based proving and blockchain-based aggregation introduce verifiability but also severe computational overheads that can represent obstacles for practical deployments.

To reduce transaction costs of the blockchain, further global tasks, like the aggregation, could be outsourced to an off-chain node as a verifiable off-chain computation, similar to [27]. While preserving verifiability, such an extension, however, would introduce novel challenges, e.g., regarding the system's liveness. Furthermore, the choice of the blockchain consensus protocol has a significant impact on the transaction costs but strongly depends on the requirements of the application scenario, e.g., permissioned versus permissionless settings.

A promising approach for optimizing the proving is the usage of a recursive proof system like Nova [39]. Such systems help to execute larger computational tasks on machines with constrained memory. Instead of executing a large batch at once, the proving is executed recursively on smaller chunks resulting in a single proof that can be verified on-chain. Furthermore, proving costs can be reduced through circuit optimization resulting in smaller ecs or by leveraging specialized hardware that implements the circuits closer to the processor.

C. Distributing Proving Artifacts

The proving key ZK_{prove} and the ecs are required by the workers to execute the proving. However, as can be seen in Table I, they are large files that can and should not be stored on a storage-constrained blockchain. To make them accessible to the workers, we propose that the ZK_{prove} and ecs of the registration and learning modules are provided through IPFS to preserve integrity, thereby unburdening the blockchain storage while preserving integrity. To further guarantee availability, IPFS can be extended through a protocol like Filecoin that provides storage incentives to decentralized nodes. Using the content addressable storage pattern [40], the artifacts can be linked from the verification contract while keeping them off-chain and can be retrieved faster. Additionally, we propose to provide the module applications in a human-readable format, e.g., the ZoKrates high-level language, to enable workers to understand the proving logic and to reproduce the ecs.

IX. Conclusion

We proposed a first system for end-to-end verifiable decentralized FL that extends state-of-the-art systems through non-disclosing authenticity proofs, thereby adding verifiability to the data source. As the core of the system, we introduced a two-step proving and verification method that, first, verifies authenticity proofs from the data layer on the learning layer, and second, verifies zkSNARKs from the learning layer on the blockchain-based aggregation layer. We applied this procedure to a registration and learning workflow enabling end-to-end verifiability between trusted data sources and the blockchain without revealing confidential information. Experimental results demonstrate the feasibility of the proposed system with only marginal overhead compared to the reference implementation.

In future work, we aim to incorporate aspects discussed in the VIII, especially enhancements of the system performance through recursive proof systems and further outsourcing of global FL tasks.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[3] P. Zhao, Y. Huang, J. Gao, L. Xing, H. Wu, and H. Ma, "Federated learning-based collaborative authentication protocol for shared data in social iov," *IEEE Sensors Journal*, vol. 22, no. 7, pp. 7385–7398, 2022.

[4] W. Wang, M. H. Fida, Z. Lian, Z. Yin, Q.-V. Pham, T. R. Gadekallu, K. Dev, and C. Su, "Secure-enhanced federated learning for ai-empowered electric vehicle energy prediction," *IEEE Consumer Electronics Magazine*, 2021.

[5] S. Otoum, I. A. Ridhawi, and H. Mouftah, "A federated learning and blockchain-enabled sustainable energy trade at the edge: A framework for industry 4.0," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3018–3026, 2023.

[6] O. Bouachir, M. Aloqaily, Özkasap, and F. Ali, "Federatedgrids: Federated learning and blockchain-assisted p2p energy sharing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 424–436, 2022.

[7] D. Gupta, O. Kayode, S. Bhatt, M. Gupta, and A. Tosun, "Hierarchical Federated Learning based Anomaly Detection using Digital Twins for Smart Healthcare," in *2021 IEEE 7th International Conference on Collaboration and Internet Computing*. IEEE, 2021, pp. 16–25.

[8] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018.

[9] G. Xia, J. Chen, C. Yu, and J. Ma, "Poisoning attacks in federated learning: A survey," *IEEE Access*, vol. 11, pp. 10 708–10 722, 2023.

[10] J. Heiss, E. Grünewald, S. Tai, N. Haimerl, and S. Schulte, "Advancing blockchain-based federated learning through verifiable off-chain computations," in *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2022, pp. 194–201.

[11] T. Rückel, J. Sedlmeir, and P. Hofmann, "Fairness, integrity, and privacy in a scalable blockchain-based federated learning system," *Computer Networks*, vol. 202, p. 108621, 2022.

[12] Z. Xing, Z. Zhang, M. Li, J. Liu, L. Zhu, G. Russello, and M. R. Asghar, "Zero-knowledge proof-based practical federated learning on blockchain," *arXiv preprint arXiv:2304.05590*, 2023.

[13] A. Smahi, H. Li, Y. Yang, X. Yang, P. Lu, Y. Zhong, and C. Liu, "Bv-icvs: A privacy-preserving and verifiable federated learning framework for v2x environments using blockchain and zksnarks," *Journal of King Saud University-Computer and Information Sciences*, p. 101542, 2023.

[14] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[15] J. Eberhardt and S. Tai, "ZoKrates – scalable privacy-preserving off-chain computations," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1084–1091, 2018.

[16] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[17] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, ser. DIDL '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–8. [Online]. Available: https://doi.org/10.1145/3286490.3286559

[18] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[19] Y. Zhang and H. Yu, "Towards verifiable federated learning," *arXiv preprint arXiv:2202.08310*, 2022.

[20] U. M. Aïvodji, S. Gambs, and A. Martin, "Iotfla : A secured and privacy-preserving smart home architecture implementing federated learning," in *2019 IEEE Security and Privacy Workshops (SPW)*, 2019, pp. 175–180.

[21] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[22] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2021.

[23] S. K. Lo, Y. Liu, Q. Lu, C. Wang, X. Xu, H. Paik, and L. Zhu, "Blockchain-based trustworthy federated learning architecture," *CoRR*, vol. abs/2108.06912, 2021. [Online]. Available: https://arxiv.org/abs/2108.06912

[24] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, and J. Jin, "Towards fair and decentralized privacy-preserving deep learning," *arXiv: 1906.01167 v2 [cs. CR]*, 2019.

[25] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized learning framework with committee consensus," *IEEE Network*, vol. 35, pp. 234–241, 2020.

[26] V. Mothukuri, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and K.-K. R. Choo, "FabricFL: Blockchain-in-the-loop federated learning for trusted decentralized systems," *IEEE Systems Journal*, 2021.

[27] Z. Wang, N. Dong, J. Sun, and W. Knottenbelt, "zkfl: Zero-knowledge proof-based gradient aggregation for federated learning," *arXiv preprint arXiv:2310.02554*, 2023.

[28] M. Fan, Z. Zhang, Z. Li, G. Sun, H. Yu, and M. Guizani, "Blockchain-based decentralized and lightweight anonymous authentication for federated learning," *IEEE Transactions on Vehicular Technology*, 2023.

[29] J. Heiss, A. Busse, and S. Tai, "Trustworthy Pre-Processing of Sensor Data in Data On-chaining Workflows for Blockchain-based IoT Applications," in *19th International Conference on Service-Oriented Computing*, ser. LNCS, vol. 13121. Springer, 2021, pp. 627–640.

[30] J. Park, H. Kim, G. Kim, and J. Ryou, "Smart contract data feed framework for privacy-preserving oracle system on blockchain," *Computers*, vol. 10, no. 1, p. 7, 2020.

[31] Z. Wan, Y. Zhou, and K. Ren, "zk-authfeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1335–1347, 2023.

[32] J. Heiss, R. Muth, F. Pallas, and S. Tai, "Non-disclosing credential on-chaining for blockchain-based decentralized applications," in *International Conference on Service-Oriented Computing*. Springer, 2022, pp. 351–368.

[33] J. Heiss, T. Oegel, M. Shakeri, and S. Tai, "Verifiable carbon accounting in supply chains," *IEEE Transactions on Services Computing*, no. 01, pp. 1–14, nov 5555.

[34] E. B. S. A. U.S. Dept. of Labor, "The health insurance portability and accountability act (hipaa)," vol. 112, 2004.

[35] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for {Zero-Knowledge} proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 519–535.

[36] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza, "Secure sampling of public parameters for succinct zero knowledge proofs," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 287–304.

[37] A. Papageorgiou, A. Mygiakis, K. Loupos, and T. Krousarlis, "Dpki: A blockchain-based decentralized public key infrastructure system," in *2020 Global Internet of Things Summit (GIoTS)*, 2020, pp. 1–5.

[38] C. Patsonakis, K. Samari, M. Roussopoulos, and A. Kiayias, "Towards a smart contract-based, decentralized, public-key infrastructure," in *Cryptology and Network Security*, S. Capkun and S. S. M. Chow, Eds. Cham: Springer International Publishing, 2018, pp. 299–321.

[39] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Advances in Cryptology – CRYPTO 2022*, Y. Dodis and T. Shrimpton, Eds. Cham: Springer Nature Switzerland, 2022, pp. 359–388.

[40] J. Eberhardt and S. Tai, "On or off the blockchain? insights on off-chaining computation and data," in *Service-Oriented and Cloud Computing*, F. De Paoli, S. Schulte, and E. Broch Johnsen, Eds. Cham: Springer International Publishing, 2017, pp. 3–15.