future internet

MDPI

Article

Proof of Flow: A Design Pattern for the Green Energy Market

Valerio Mandarino, Giuseppe Pappalardo and Emiliano Tramontana *

Dipartimento di Matematica e Informatica, University of Catania, 95125 Catania, Italy; valerio.mandarino@phd.unict.it (V.M.); pappalardo@dmi.unict.it (G.P.)
* Correspondence: tramontana@dmi.unict.it; Tel.: +39-095-7383008

Abstract: The increased penetration of Distributed Energy Resources (DERs) in electricity markets has given rise to a new category of energy players, called Aggregators, whose role is to ensure fair remuneration for energy supplied by DERs, and support the smooth feeding of the intermittent energy produced into the distribution network. This paper presents a software solution, described as a design pattern, that governs the interaction between an Aggregator and DERs, leveraging blockchain technology to achieve a higher degree of decentralization, data integrity and security, through a properly designed, blockchain-based, smart contract. Thus, the proposed solution reduces the reliance on intermediaries acting as authorities, while affording transparency, efficiency and trust to the energy exchange process. Thanks to the underlying blockchain properties, generated events are easily observable and cannot be forged or altered. However, blockchain technology has inherent drawbacks, i.e., mainly the cost of storage and execution, hence our solution provides additional strategies for limiting blockchain usage, without undermining its strengths. Moreover, the design of our smart contract takes care of orchestrating the players, and copes with their potential mutual disagreements, which could arise from different measures of energy, providing an automatic decision process to resolve such disputes. The overall approach results in lower fees for running smart contacts supporting energy players and in a greater degree of fairness assurance.

Keywords: blockchain; smart contracts; design patterns; energy market

Citation: Mandarino, V.; Pappalardo, G.; Tramontana E. Proof of Flow: A Design Pattern for the Green Energy Market. *Future Internet* **2023**, *15*, 313. https://doi.org/10.3390/fi15090313

Academic Editor: Qiang Qu

Received: 12 August 2023
Revised: 13 September 2023
Accepted: 14 September 2023
Published: 17 September 2023

1. Introduction

The growing adoption of Distributed Energy Resources (DERs), such as small-scale green energy producers that feed into the main grid energy gathered by photovoltaic power plants or small wind turbines, has given rise to Aggregators, a new category of energy players in electricity markets [1–3]. Aggregators play a crucial role by ensuring fair and reliable compensation for the energy supplied by DERs and facilitating the seamless integration of intermittent energy production from renewable sources into the distribution network [4]. An Aggregator is responsible for supplying an overall baseline power (aggregate power) to a higher-level entity (company), hereafter termed the *Operator*, which manages electric lines and other power grid devices to serve residential, commercial, and industrial consumers.

From the viewpoint of system software architecture, researchers are actively exploring various approaches to aggregating and coordinating DERs, with a view to addressing key challenges and enabling efficient energy management. While standards like OpenADR (Open Automated Demand Response, https://www.openadr.org, accessed on 25 July 2023) are available to address the goal of interoperability among different systems (by a common format for message exchange), several other noteworthy, high-level, concerns are still awaiting satisfactory solutions. Such concerns include, e.g., reliable orchestration and data storing, incentives to behave fairly, resolution of potential disagreements, etc.

Thanks to its decentralized nature, blockchain is immune from vulnerabilities caused by single points of failure. Also, blockchain-run code, or smart contracts, can be used to store on the blockchain permanently data that cannot be altered later on and are visible to any interested player, thus ensuring transparency and fairness, and reinforcing trust. For

these reasons, blockchain has been widely acknowledged as an effective framework for designing distributed applications and achieving decentralization and transparency [5–7].

Nevertheless, in this respect, there are several outstanding issues. To begin with, simply relying on efficient consensus algorithms to ensure the consistency of the blockchain state is, admittedly, not sufficient to prevent malicious behavior by participants. When employing a technology that is known to be vulnerable to attacks, this possibility must constantly be addressed [8–10]. In addition, while the blockchain has the potential to reduce disagreements due to its transparency feature, current approaches lack specific procedures for addressing and resolving disputes that may arise between a DER and the Aggregator [11–15]. Moreover, many approaches use the blockchain essentially to handle financial and energy balances [16–20], and, due to the large amount of interactions and data typical in the energy market, possibly incur in excessive cost for the use of blockchain computing and storage resources.

This paper proposes a design pattern, named Proof of Flow (PoF), which governs the interactions between DERs, providing electrical energy to the system, and an Aggregator that manages the DERs according to the energy flows it detects and the requests made by the Operator. Accordingly, the proposed framework, besides catering for the transparent and efficient coordination of DERs with a view to ensuring fair remuneration, addresses the case where a DER is unable to provide the amount of energy it committed to offer, and provides means for the resolution of any potential disputes. In addition, this pattern minimizes blockchain access and related costs in terms of fees incurring when running a smart contract. We consider a localized scale setting, such as a microgrid or a small energy community comprising a few dozen DERs, where the Aggregator is physically linked to the DERs it manages. This enables seamless energy transfer and measurement, while data exchange occurs via the blockchain when needed.

The PoF pattern leverages a smart contract we implemented in Solidity, called "ARIA" (Aggregator Renewable Integration Agreement), which oversees the interaction between energy players. It works in conjunction with a smart contract called "Vault" to maintain and manage a balance for rewarding DERs and the Aggregator. We first introduced the Vault smart contract in the context of the "Treasury Manager" design pattern [21].

Blockchain technology gives support to the verifiability of smart contract code and the permanence of storage, ensuring their immutability, regardless of potential tampering that could occur in a single host, thus ensuring security, transparency and ultimately trust. Specifically, our solution relies on blockchain to make sure that algorithms governing energy players are immutable, and generated events can be easily observed, cannot be questioned, and cannot be lost. However, the data storage we employ is outside the blockchain, this reduces the frequency of blockchain accesses, the storage space needed, hence the associated costs. Data recorded outside the blockchain in a set of external hosts are then ordered temporally, regardless of local clocks, by exploiting, as a surrogate timestamp, the *blockstamp*, i.e., the hash of the latest block stored in the blockchain. As data are stored on external hosts, labeling them with blockstamps ensures that all actors can assign them temporal identifiers, which depend on the blockchain state and maintain a pace consistent with the blockchain itself. This functionality plays a crucial role in our system's dispute-resolution measures. Disputes could arise when a DER has not received the expected amount for the energy production, then energy measures from different smart meters have to be compared for given timeframes.

In short, the approach outlined leverages the blockchain, and its decentralised nature, to address the trust-related issues that may arise in traditional cloud-based systems, while minimizing the vulnerabilities associated to solutions relying on a disputable central authority. Furthermore, the proposed PoF pattern, with some expedients, succeeds in greatly reducing the typical costs that would arise from a naive, indiscriminate usage of blockchain computation and storage resources, without sacrificing the level of trust such an usage would straightforwardly—albeit expensively—afford.

The remaining sections of the paper are structured as follows. Section 2 discusses the related work in the literature. Section 3 introduces blockchain technology, smart contracts, and describes the Treasury Manager design pattern. Section 4 describes and explains the proposed pattern. Section 5 highlights potential issues that may arise in implementing the pattern. Finally, Section 6 draws the authors' conclusions.

2. Related Work

Multiple papers addressing the interaction between DERs and Aggregators have been proposed. Technologies such as IoT, artificial intelligence, and blockchain were discussed in the context of the optimization of the energy industry to address challenges related to security, authorization, and data integrity [22]. Blockchain technology and smart contracts in the energy sector were employed to have a decentralized approach balancing energy demand and supply [23].

A blockchain architecture was used to share the aggregator's role across all devices on a microgrid network with the goal of minimizing the energy provision costs while considering the operational limitations of the distribution network and DERs, ensuring both operational feasibility and fair payments to all parties involved. In addition, to tackle the optimization problem, the Alternating Direction Method of Multipliers (ADMM) algorithm was used to obtain solvable subproblems assigned to each respective node across the network [24].

To manage the various actors involved in the generation, trade, and consumption of green energy, cloud storage and Energy Web blockchain technology (https://www.energyweb.org/, accessed on 25 July 2023) were used in combination to ensure the immutability and authenticity of data. Information about production and consumption was stored in the cloud, while their corresponding hash signatures (having a smaller footprint) were stored on-chain. The inherent trust feature of blockchain technology ensured secure and indisputable transactions [25].

A decentralized energy trading system based on consortium blockchain technology and credit-based payment schemes was proposed to ensure secure and decentralized energy trading, specifically in microgrid scenarios. To enable fast payments, peer nodes can request "energy coins" loans from credit banks which are participants in the network. Additionally, the Stackelberg game theory was used to determine an optimal loan pricing strategy that maximizes economic gains for credit banks [11].

A blockchain-based system was used to make smart contracts address transparency issues in the process of managing aggregated DERs, by storing on the blockchain operational data when a control event is triggered by the VPP. Stored data include supplied power, timestamps, and deviations. Then, a smart contract was used to publish such data and make payments to DERs [12].

Aggregators were proposed to implement state changes in devices by using local hardware and Hyperledger Fabric blockchain technology (https://www.hyperledger.org/, accessed on 25 July 2023) to create a decentralized energy trading platform, enabling DERs, that in this case are prosumers (entities that both produce and consume energy), to trade surplus energy. At the start of each market round, prosumers transmit the amount of extra energy they can provide or absorb from the microgrid. A smart contract was used to match these values and determine energy injection or withdrawal from the grid [13].

A smart contract executing on an Ethereum blockchain, with a proof-of-authority (PoA) consensus algorithm, was proposed to coordinate DERs and minimize energy costs, while enabling prosumers to update their energy trading decisions, optimize energy trading scheduling, and participate in demand response activities [14].

A sub-metering device was designed to locally collect energy flow data employed to handle demand response actions on the blockchain. Data science and machine learning algorithms were used to produce load-demand and electricity price forecasts using collected data and weather forecasts [16].

A framework, built on the Ethereum platform in a consortium environment using the PoA, was proposed to record energy and monetary transactions among the entities involved to improve DER visibility and the verification of transactions in the wholesale energy market. The blockchain verifies transactions, executes smart contracts, and records events [17].

A platform was designed to assess blockchain solutions' performance in various scenarios within modern grids, with the goal of accelerating the development of solutions for efficient energy exchange. The platform offers a collection of reusable services that connect existing grid tools with the blockchain, along with an environment that allows developers to abstract connection and performance measurement activities. The authors employed HyperLedger Fabric to implement a blockchain-based app that allows Aggregators to participate in wholesale markets under the oversight of a Distribution System Operator. Off-chain databases are also used to store operational data like measurements, configurations, and processing logs [18].

To address the vulnerability of single point of failure, a governance platform based on blockchain and smart contract was proposed to enhance secure and resilient control services for DERs. It operates during outages until normal functionality is restored [26].

A model was proposed to have the Aggregator coordinate users in the microgrid and provide data collected from consumers and prosumers to the blockchain. By using a smart contract, it matches supply and demand employing a double auction mechanism. Inter-planetary File System (IPFS https://ipfs.tech, accessed on 25 July 2023) was used as a decentralized data storage. By leveraging blockchain technology a more efficient energy use was obtained [19].

The integration of DERs, including a microgrid, utility-scale batteries and solar panels, was proposed, by means of smart contracts, allowing to adjust the desired output for each DER in response to grid needs, and establishing criteria for each participant and compensating them, using a token reflecting the quantity of energy generated, the type of energy resource, and the sort of ancillary service [20].

The above papers investigated the use of blockchain technology to develop frameworks enabling the efficient management of energy players. Blockchain has been proposed as a solution to overcome the single points of failure and enhance security, transparency, and automation, by using smart contracts. However, relying merely on the inherent characteristics of blockchain is not sufficient to effectively transition to a decentralized management of DERs. Blockchain technology has some limitations [21] that must be taken into account. None of the above approaches explicitly addresses the issues of unprocessed transactions, transactions cost, and on-chain storage costs. However, regarding the first two, the majority of the above proposals refer to private or consortium blockchains, implicitly assuming that transactions, whether they are payments or smart contract invocations, are always processed and that their associated fees are not subject to the fluctuations observed in public blockchains. These assumptions have allowed the authors to focus on the impact of consumer prices or producer revenues. Nevertheless, given the inherently slower nature of blockchain code execution, compared to a client-server architecture, and considering that data are stored permanently, further advances for optimizing blockchain access are needed. Regarding the storage, several proposals combined cloud and on-chain storage without providing specific strategies to prevent single points of failure. In addition, even smart contracts may represent a weakness: a bug or poor programming techniques could lead to the loss of their funds [21]. The above said approaches discussed the use of smart contracts in a general context, emphasizing their role in automation, however without providing details on how to mitigate such a risk.

From a broader perspective, not all the forces at play in governing the energy players were considered in previous works. The proposed solutions do neither incentivize or penalize the participants according to their behavior, nor do they address the issue of data veracity that is broadcast to the blockchain. Finally, none of them proposed a method to

automate the resolution of potential disagreements in the measurement of the amount of energy fed into the power grid.

In our proposal we put forward a solution that can adopt a public blockchain like Ethereum while considering not only the inherent constraints of blockchain technology in general, but also the specific limitations that Ethereum may have when compared to private or consortium blockchains. Our main results include: (i) a software architecture that governs the actors involved in a distributed green energy system, without resorting to a centralized authority to store data (Section 4); (ii) incentives for actors to behave fairly, thanks to the use of ranking, rewards and penalties (Sections 4.1 and 4.2); (iii) a way to check the truthfulness of data measuring energy production (Section 4.2); (iv) a solution to limit the interactions with the blockchain, hence lowering execution fees (Section 4.2); (v) an automatic procedure to manage disagreements (Section 4.4).

3. Background

This section highlights the core features of blockchain technology and smart contracts, discusses some of their limitations, and describes the Treasury Manager design pattern used to hold the balance of the contract [21].

3.1. Blockchain

A blockchain is a peer-to-peer distributed ledger that stores cryptographically signed transactions in an append-only list of linked blocks, namely the chain [27]. Blockchain technology combines different well-known concepts, such as digital signatures, cryptographic hashing functions, and decentralized consensus algorithms, to validate transactions without relying on a central authority [28–31]. Once written, the content of a blockchain is immutable and cannot be deleted, otherwise the entire data recorded become invalid. This feature ensures the provenance, integrity, and authenticity of data, preventing tampering. Consensus among all peer network nodes, called Validators, which reach a common agreement on the state of the distributed ledger, ensures the reliability of the blockchain. The consensus algorithm also discourages dishonest behavior by making it economically disadvantageous for a Validator to deviate from the expected protocol behavior [5].

Blockchains are classified as either public (permissionless) or private (permissioned). Public blockchains provide a decentralized system in which users can view transactions, write data, and run validator nodes. Private blockchains need permission to access and are governed by a limited number of nodes, which are often controlled by the company that created the network [32]. This approach allows for prioritizing scalability and security, while sacrificing decentralization. A consortium blockchain is a form of private blockchain in which equally powerful validators, from multiple organizations or entities, share the responsibility of maintaining the blockchain network and ensuring its integrity and security. Consortium blockchains provide a balance between decentralization and control.

3.2. Smart Contracts, Block Time, Gas

Ethereum [5] was the first blockchain that introduced the innovative possibility of executing user programs on a virtual machine, called the Ethereum Virtual Machine (EVM), that operates under each Validator node within its network. These programs, or "smart contracts", are compiled to bytecode and deployed to the blockchain. They can be executed requesting a transaciton towards their addresses and can independently maintain their own balance. Once stored on the blockchain, smart contracts' bytecode cannot be modified, reflecting the immutable feature of the blockchain. However, they can be removed by the owner, i.e., the entity owning the private key of the address that deployed the contract. The advent of smart contracts has turned the blockchain into a global computer accessible from a variety of different devices, making it a general-purpose technology that extends beyond finance and offers opportunities to leverage the blockchain as a backend for designing distributed applications, offering enhanced transparency, security, traceability, and trust across various sectors.

In the blockchain domain, time is marked by the block time, which refers to the average time it takes for the network to generate an additional block. e.g., in Ethereum, a new block is minted every 12 s, except in unexpected conditions. Consequently, a transaction that proposes a new state based on the prior state of the blockchain will face at least a 12-second delay [33,34]. Moreover, smart contracts' execution is limited by a maximum amount of "gas" consumption allowed. Gas is a metaphor that quantifies the computational effort required while executing a smart contract, which results in a financial cost that users have to pay to execute the code [35].

3.3. Treasury Manager Design Pattern

Generally, many smart contracts manage users' balances. When the code of a smart contract has to change to meet new requirements or to fix bugs, then a new version of the contract has to be deployed and the accumulated financial reserve has to be transferred to the latest version, then the previous version will be deleted. However, for such a balance transfer operation expensive gas fees could incur [36].

The Treasury Manager pattern has been devised to avoid the need to transfer the accumulated reserve from the old smart contract to the latest version, thereby avoiding the gas fees (see Figure 1). The following participants constitute the Treasury Manager.

- Registry smart contract that stores the address of the Service smart contract [37].
- Service smart contract that provides a useful service.
- Vault smart contract that retains the financial reserve and manages user balances.

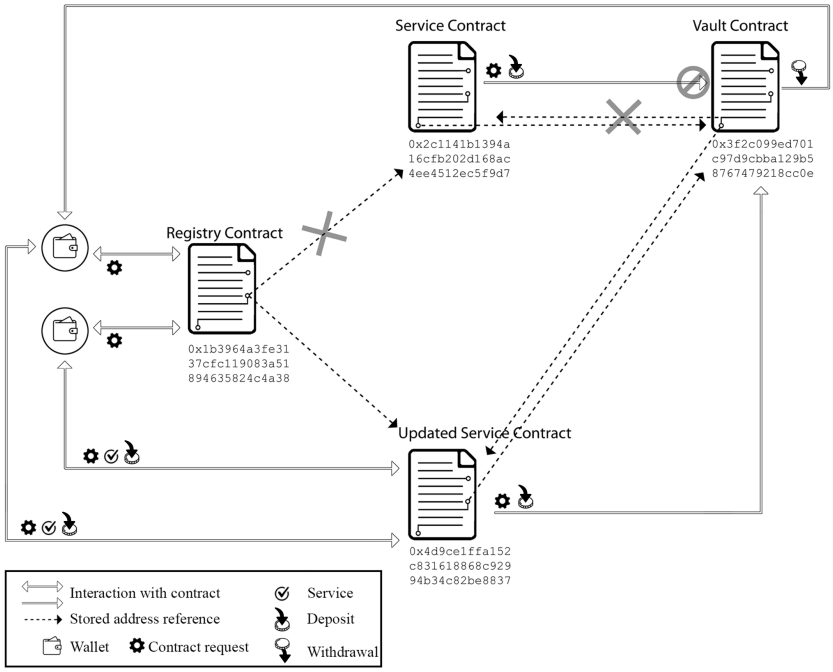

Figure 1. Treasury Manager pattern after the Service Contract has been updated.

Registry acts as a proxy and hands to client the requested Service address. When the Service is updated, its new address is stored in Registry, ensuring that clients always access the current version of the service. This solution enables users to interact with new versions of Service seamlessly, even though the new smart contracts have different addresses. Vault is responsible for managing users' accounts and balances, and can only be triggered by Service to authorize withdrawals and deposits. Service, which runs the business logic, can be updated independently of Vault, which remains unchanged and continues to hold all balances. The new Updated Service will refer to the same Vault contract, hence this approach avoids the need to transfer the accumulated reserve from one wallet to another, thereby saving expensive gas fees.

After deploying a new version of the Service contract (Updated Service Contract), the owner of the contracts, (see Sections 4.6 and 4.8), will store its address in the Vault contract, allowing the Updated Service contract to become the only entity authorized to interact with the Vault to unlock users' coins (hence, old Service Contract cannot deposit or withdraw any amount anymore).

Treasury Manager is used by the Proof of Flow pattern to hold, manage, and maintain the energy players' balances, as well as handle DERs' and Aggregator's deposits and rewards.

4. Proof of Flow Design Pattern

To provide a clear and unified framework that defines the roles, responsibilities, and interactions between DERs and Aggregators, we propose the Proof of Flow design pattern. We assume that DERs only operate as energy producers, i.e., they inject energy into the power grid. Furthermore, the Aggregator acts on behalf of the Operator, which is responsible for coordinating the overall energy grid. Proof of Flow design pattern aims at providing efficient and seamless coordination between the entities involved, enabling effective integration of DERs into the grid, and solving potential disputes. We describe the pattern according to the typical sections found in the GoF book [38].

Name

Proof of Flow.

Intent

Assist the actors responsible for the energy production and transmission to interact with each other and trace data to reward producers and solve potential disagreements on the amount of energy produced.

Motivation

Consider an Aggregator orchestrating DERs aiming at providing electrical energy to the system in a localized, small-scale setting. DERs use local smart meters to measure and locally store the energy production data which, typically, are sent to the Aggregator. This acknowledges the offers of energy from selected DERs whose production goes into the distribution network.

Data consisting of measured energy produced and delivered, as well as commands and agreements between several parties are crucial for all the actors to work together. If such data and messages are exchanged over the cloud, a trusted authority is needed to hold them, and ensure their veracity. However, when none of the involved entities are an undoubted authority, then it is difficult to make one emerge as trusted. By leveraging blockchain technology, it is possible to forgo the cloud-based trusted authority while assuring transparency, immutability, and decentralization in the exchange process.

Blockchain technology offers the possibility to record data in a permanent storage, in such a way that they cannot be modified, deleted or disputed. Blockchain nodes, running smart contracts and storing data, consume gas for such operations. The consumed gas is an operating cost, equivalent to real currency, that need to be paid. Such a cost has to be minimised for making the solution based on blockchain sustainable.

Moreover, the blockchain nodes cannot determine by themselves whether received data are accurate, hence a mechanism is needed to solve potential disagreements on the truthfulness of data.

Applicability

Use the Proof of Flow pattern to replace reliance on a trusted authority holding data in the cloud by recurring to blockchain technology, in a cost-effective and efficient manner, and when the need to solve disputes could arise.

Structure

Figure 2 shows the structure of the Proof of Flow design pattern, the energy players, the proposed smart contracts, and all the interactions among them.

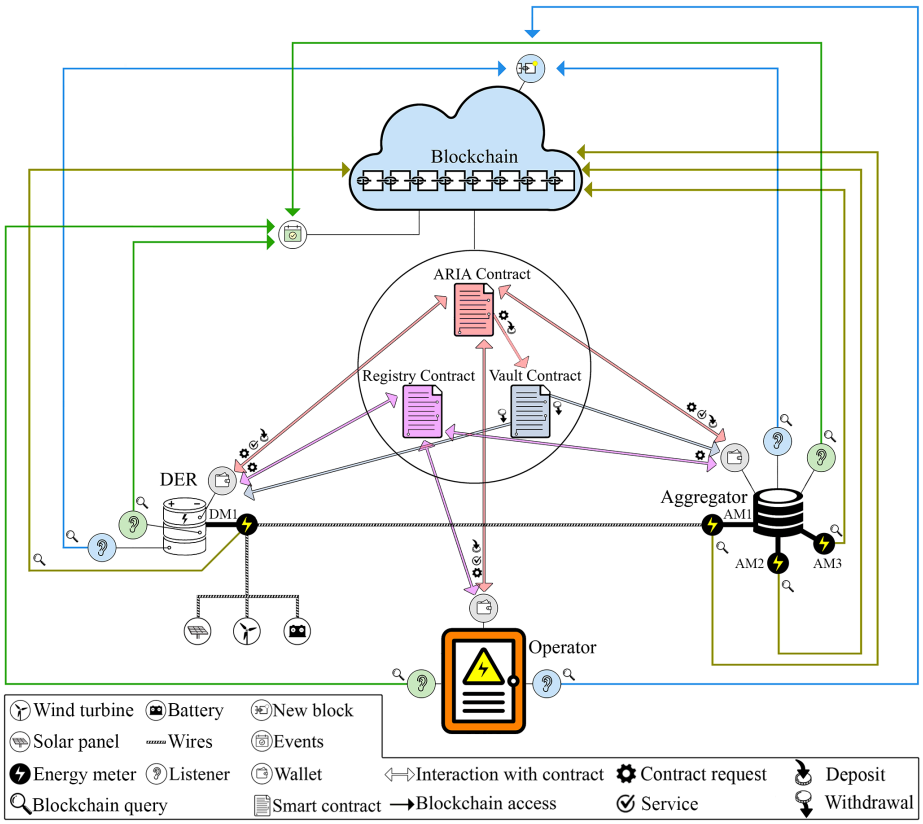

Figure 2. The structure of the Proof of Flow design pattern.

The DER that is depicted has an energy meter (DM1) physically linked, by a power line, to an energy meter of the Aggregator (AM1). The latter, in turn, has multiple energy meters, each associated to the power lines connecting to other DERs.

There are several interactions with and between smart contracts, according to the wallet address invoking a function of the smart contract. e.g., ARIA, as can be seen, only accepts deposits, which it routes to Vault, and does not allow withdrawals. Instead, it forwards withdrawal requests to Vault, which will approve or deny them according to the available unlocked balance associated with the specified wallet address.

Participants

The participants of the Proof of Flow design pattern are the following.

- ARIA (Aggregator Renewable Integration Agreement) smart contract, holding a list of active and prospective DERs, keeps energy balances for DERs and triggers payments to DERs that have provided energy and to the Aggregator, for its role in governing them.
- DERs, namely energy resources, such as solar panels, wind turbines, and batteries, participate as energy producers and transfer energy into the electrical grid.
- Aggregator coordinates DERs by monitoring their energy contributions, and ensures the compensation for the energy they supply.
- Smart meters measure and record energy flow in real-time, then cryptographically sign and locally store the measured energy flow, frequently.
- Operator manages the network transporting energy to consumers.
- Registry smart contract stores the address of the current version of ARIA.
- Vault smart contract holds coins or tokens and manages financial balances.

Data recording energy produced by DERs, and received by the Aggregator, are used to provide evidence of energy balance that is needed for resolving potential disagreements in detected and recorded energy flows.

Collaborations

The Aggregator is in charge of providing the aggregate energy, produced by DERs, to the Operator. Its responsibility is to govern DERs located within the same grid, by utilizing ARIA. It must guarantee that DERs with varying baselines (i.e. the standard level of energy production), comply with the established rules and criteria, by monitoring their energy contributions and adherence to the requests, and ensure that they are rewarded accordingly for their contributions.

Here we provide an overview of the interactions governed by the pattern, while details are in the following subsections.

A DER can join a community, managed by an Aggregator, by invoking the associated ARIA's Join() function. Together with the request, some parameters are given, notably the quantity of kilowatts that the DER can inject into the grid, the requested price of the energy, and the security deposit. If the DER meets the criteria and the number of maximum prospective-DERs has not been reached before, then it is accepted as prospective DER; otherwise, the transaction is reverted (see Section 4.1 for details on joining a community). A DER that has been accepted as an energy provider measures the produced energy, and stores the measures locally (by means of a smart meter provided by the Operator). The energy produced by the DER is transmitted to the Aggregator, which measures the energy received by each DER using its own smart meters, and stores measures locally (see Section 4.2 for details of interactions during energy production). The Aggregator checks whether the agreements with each DER have been satisfied and if so, it periodically invokes ARIA's SweepEpoch() function to compensate DERs (see Section 4.3 for details on rewards management). If a disagreement between energy provided and compensation arises, then DERs can ask the Operator to resolve a dispute (see Section 4.4 for details on resolving disputes). Additionally, Vault is used to hold financial balance and, while executing ARIA's SweepEpoch() function, Vault can be asked to transfer a compensation into the DERs' and the Aggregator's accounts, for their energy production or coordination work, respectively. Registry is used to store and provide ARIA's most up-to-date address, which can only be modified by the contract owner (the Operator). If there is a need to deploy a new version of ARIA, the Operator will update the new address in the Registry; the Registry will publish an event to notify all participants of the ARIA replacement; finally, the participants can request the Operator for the source code of this new ARIA version, to inspect it and compare its bytecode with the one stored on the blockchain. As for blockchain storage, ARIA emits events that carry payment receipts and messages to the players, which will be stored on-chain.

Collaboration between the said participants is described according to three phases: DERs joining an energy community; DERs producing energy; rewarding for produced energy. Such phases are detailed in the following subsections. Another subsection details how the energy players can requests the Operator to resolve disputes, if needed.

4.1. Phase 1: A DER Joins an Energy Community

To participate in the system and provide electric power, DERs need to register as energy providers by invoking the ARIA Join() function.

This registration process involves transferring a security deposit to Vault which, if the DER is accepted, creates an account associated with the DER's wallet address. Additionally, the DER communicates the following parameters that characterize it, that are stored in ARIA, and will be used in subsequent activities:

- The ID of the smart meter;
- Category: type of the DER, representing the specific source of energy used;
- Power output: namely the maximum capacity or amount of electric power it can generate and supply to the energy grid over a given time period t;
- Availability: the schedule of the DER, specifying the time periods during which it can actively participate in energy provision;

- Flexibility: the capability of the DER to modulate its energy production based on varying requirements;
- Compensation: the desired price expected by the DER for the supplied energy; the energy price is proposed by the DER and when ARIA analyses whether to include the DER among the energy suppliers, the proposed price can be deemed satisfactory;
- Wallet address: the unique identifier of the DER in the blockchain, allowing identification and tracking, that is the sender of the registration transaction and represents the DER univocally.

These parameters are stored by the Join() function in a list that holds the prospective-DERs, enabling ARIA to determine whether to authorize a DER to transmit energy. When a DER is allowed to feed the grid, it is notified via a blockchain event (see Sections 4.5 and 4.7) and its address is moved from the list of prospective-DERs to the active DERs' pool. The security deposit provided by the DER serves as a countermeasure against unreliability and is used in cases of non-compliance or failure to fulfill obligations.

During the acceptance waiting period, if the DER cancels its subscription before actively participating in the energy provision, the security deposit can be redeemed. Alternatively, it can be redeemed at the end of the contract period if it has not been used to pay fines in case the DER failed to meet its obligations or violated the terms and conditions set by ARIA. Similarly, the Aggregator is required to stake a deposit to ensure the proper functioning and adherence to the rules of the system, and may face penalties imposed by the Operator, as described in Section 4.4.

This mechanism incentivizes DERs and the Aggregator to comply with the system's requirements by holding them financially responsible for their actions.

To successfully implement the pattern, we make the following assumptions.

1. The pattern's time interval is synchronized with the block time of the blockchain. i.e., ARIA's SweepEpoch() function (see Section 4.7), cannot be executed more frequently than the block time.
2. When we refer to "time", we intend the validation of a certain number of blocks in the blockchain starting from a specific block.
3. We use the term "epoch", denoted as E, to define a specific duration of time, univocally identified by a progressive nonce, that is represented by a fixed number of blocks.
4. We use the term "blockstamp" to denote a reference of a specific block in the blockchain, represented by its hash, that provides a reliable mechanism for avoiding synchronization issues. This value is used to establish a temporal cadence that is derived from the blockchain, allowing to overcome some issues that may arise when adopting timestamps. Timestamps might not provide an accurate time reference, as clocks can be tampered with or subject to malfunctions. Furthermore, unlike a timestamp, a blockstamp establishes a definitive temporal order in relation to the blockchain state and events, and is synchronized with the blockchain's pace.
5. We use the term "smart meter" to refer to a customizable device that, under the hood, is composed of two elements: an energy meter and a device with storage and computation ability. The energy meter is capable of providing energy readings [39] as numerical values through Ethernet, using a standard protocol such as Modbus [40]. In our system, AMR functions [41] are not needed. The device is equipped with local data storage that includes a software component using libraries like secp256k1 [42] to implement cryptographic functionalities, such as the Elliptic Curve Digital Signature Algorithm (ECDSA) [42]. Additionally, this device accesses the blockchain either through online API services like Infura (https://www.infura.io, accessed on 25 July 2023) or Alchemy (https://www.alchemy.com, accessed on 25 July 2023) or by means of locally installed wallet software such as go-ethereum (https://geth.ethereum.org, accessed on 25 July 2023). Alternatively, it can run a blockchain node. Depending on the latter choice, its computational capabilities may differ.

From the perspective of the Aggregator, which needs to store data from multiple energy meters, there are two alternatives. The Operator could provide it with a smart meter for each DER. Alternatively, the Operator could supply it with an energy meter for each DER and a single server, that would run the same previously outlined software, and store data from all the DERs in the same way as the standalone smart meter.

To detect the end of an epoch, every energy player runs a listener that constantly monitors the blockchain and keeps track of the increment in the number of newly added blocks. Once the required number of blocks that constitutes an epoch is reached, the current epoch concludes and the listener resets its counter for the start of the next epoch.

4.2. Phase 2: Energy Flows from DERs to the Aggregator

The DER and the Aggregator are equipped by the Operator with a smart meter that measures the energy produced by the DER at each end of the transmission power line. During the normal functioning of the system, after a DER has been requested to transmit energy, as the smart meters are positioned downstream of both the DER and the Aggregator, the energy flow data are periodically measured on both sides. Such data are signed by the DER's smart meter, to prove its own identity, and saved locally. The blockstamp of the latest block in the blockchain is included in local data records to indicate the time of recording. Moreover, the Aggregator, on the other side of the transmission power line, measures the energy received by each DER, signs the data, and saves them locally in the same way. The Aggregator will use the said measures when having to reward DERs (see Section 4.3), whereas DERs use saved measures when asking for a revision of the reward amount (see Section 4.4). The data stored by the smart meters is used to generate proof that a specific amount of energy has been produced (in the case of the DER) or received (in the case of the Aggregator) within a given time frame. We dubbed this proof: "Proof of Flow." Table 1 shows the data of the Proof of Flow (PoF).

Table 1. Proof of Flow data types for a specific epoch.

Name	Type	Purpose
EpochNonce	uint16 [i]	time-frame to which the Proof of Flow refers
Blockstamp	uint256 [ii]	time of recording
kWs	uint16	kilowatts of energy flow
ID	uint256	unique ID of the smart meter
Address	address [iii]	address of the wallet of the DER or the Aggregator

[i] uint16 represents a sequence of two bytes of data. [ii] uint256 represents a sequence of 32 bytes of data. [iii] address represents a 20-byte Ethereum address.

Considering the significant volume of data and the potential cost when keeping them in a blockchain storage, it is not feasible to store the whole data recorded by smart meters on-chain or in online databases like IPFS. Instead, by separating data storage from the blockchain, a more scalable and cost-effective solution is achieved. Nevertheless, proofs of energy flow are needed when potential disagreements arise. In such cases, these proofs can be automatically used to resolve them and impose penalties to the malicious actor (see Section 4.4 for details). This mechanism is a countermeasure for potential malicious behavior; it provides a further protection and strengthen trust among parties.

The smart meter may take energy readings at regular intervals, even multiple times per second. However, it is advisable to record the average of measurements for the time span that corresponds to the creation of a block. i.e., a new record is recorded every time the value of the Blockstamp is updated. Given that the data recorded to generate Proofs of Flow (PoFs) consist of EpochNonce (uint16), Blockstamp (uint256), and kWs (uint16), and that the other fields are added only when generating the PoF, the total space for a single record takes exactly 36 bytes. If a record is saved every 12 s (Ethereum's blocktime), 36 bytes would be written every 12 s, for a total of 180 bytes each minute. That is 10,800 bytes in an hour and 259,200 bytes in 24 h. If an epoch lasted 24 h, the minimum storage size required

to archive a year of recordings would be about 90.5 MB. Since longer epochs summarize more data, the amount of storage space required decreases as the epoch duration increases. The Aggregator will store the same amount of data as DERs multiplied by the number of active DERs.

4.3. Phase 3: Rewarding DERs and the Aggregator

To reward DERs and the Aggregator, ARIA must have a balance that is fed by the Operator. Rewards can be provided in the form of fungible tokens that adhere to the ERC20 standard (https://ethereum.org/it/developers/docs/standards/tokens/erc-20, accessed on 25 July 2023) or in the native coin of the blockchain, such as ETH (Ether), in the case of Ethereum. In both cases we refer to rewards as "credits".

At the conclusion of every epoch, the Aggregator triggers the SweepEpoch() function of ARIA, providing for every active DER the number of kilowatts (kW) produced, as recorded by its smart meters. The function allocates credits to each DER according to their energy production and emits "Receipts" events which allow them to verify their compensation (see Section 4.7 for further details).

If a DER receives rewards lower than expected and the gap with the actual earnings exceeds the reward threshold set by the error tolerance of t, it has the option to raise a dispute within a specific number of epochs, denoted as D (see details in Section 4.4).

At the end of this phase, ARIA assigns DERs a rank to assess their performance and dependability. When a registered DER is first accepted as an energy provider, as it is still in the practice stage, it is given an initial rank value, which is initialized to a threshold constant. As discussed later, this threshold is a reference to distinguish DERs with ranks lower than the initial value. As DERs actively fulfill their responsibilities, their rank may increase over time, reflecting their performance and reliability within the system. If a DER is aware that at a certain time it will not be able to fulfill the contract by producing the claimed quantity of power, it can proactively notify the Aggregator by invoking the NotifyPowerShortage() function of ARIA. This allows the Aggregator to schedule a request for power modulation to other DERs to compensate for the anticipated future power shortfall. If DERs fail to provide the declared amount of energy or interrupt their energy supply, without notifying the Aggregator, in addition to being potentially fined, their rank will be reduced.

The ranking can be a valuable tool during the contractual renewal process or for engaging DERs in energy modulation when required by the Aggregator. To assess the impact of ranks, different criteria could be implemented into the CalculateRanking() function of ARIA. One possible selective criterion could be to remove DERs from the pool (and not re-admit them) as soon as their rank goes below the specified threshold. A more moderate criterion could be to allow DERs with ranks below the threshold to continue operating, albeit with a lower probability (compared to DERs with higher ranks) of being selected in situations where there is competition for power production. A democratic approach could be to ignore ranks and renew contracts in a deterministic random manner until the required aggregate power is achieved, thus preventing any DER from gaining a monopoly. We have implemented a first-come-first-served strategy. However, as mentioned before, the code of the CalculateRanking() function can be modified to adopt the most suitable criterion for the specific context.

4.4. Resolving Potential Disputes between DERs and Aggregator

When there are no discrepancies between the energy flows measured and recorded by the Aggregator's and DERs' smart meters, ARIA rewards DERs according to the amount of energy received by the Aggregator. This flow should align with the capacity declared by DERs in the Join() function with a tolerance error of t (set in ARIA by the Operator and to which DERs and Aggregators must adhere). However, in case of disputes or for verification purposes, the Operator may request Proofs of Flow from both the DERs and the Aggregator for a specific epoch. The Operator performs such requests by emitting events ProvidePoF or ProvidePoFs (see Section 4.5).

Proofs of Flow data must be signed by smart meters and passed to the ProveTheFlow() or the ProveTheFlows() functions that record the requested relevant data in a blockchain event. Data in the Proof of Flow can have one among several time resolutions. At block-time level, they present a detailed and granular view of the energy flow. Alternatively, they provide an overview delivering a summary over a longer time period. In this latter case, the involved data are shown in Table 2.

Table 2. Proof of Flow data types for a range of epochs.

Name	Type	Purpose
EpochStart	uint16	beginning of the time-frame to which the Proof of Flow refers
EpochEnd	uint16	end of the time-frame to which the Proof of Flow refers
BlockstampBegin	uint256	time at the beginning of the epoch
BlockstampEnd	uint256	time at the end of the epoch
kWs	uint16	total kilowatts of energy flow
ID	uint256	unique ID of the smart meter
Address	address	address of the wallet of the DER or the Aggregator

Disputes can be initiated by DERs which received a reward lower than expected. To trigger a dispute, the DER needs to execute the RaiseDisputeEvent() function of ARIA, that emits a Dispute event. The function's input requires the Proof of Flow data (see Tables 1 and 2) for the corresponding epoch, its nonce, and the blockstamps for the period that resulted in a smaller reward. Once the Operator, which is responsible for the smart meters of both the Aggregator and the DER, detects a Dispute event on the blockchain, it can start the dispute resolution process that involves the emission of a ProvidePoF, or ProvidePoFs, event addressed to the Aggregator, which, in turn, is requested to provide the Proof of Flow from its smart meter for the same period in question. If the Aggregator fails to respond, within a specific number of epochs, denoted as D, the Operator will impose a fine to penalize the Aggregator's behavior and to compensate the financially impacted DER.

When, otherwise, the Operator receives the Aggregator's Proof of Flow in time, it can analyze and compare the data to investigate the reasons for the discrepancy between the flows detected by the smart meters. This solution entails DERs and the Aggregator entrusting the smart meters provided by the Operator; however, the open-source nature of the smart meter's software ensures transparency and accountability. Furthermore, DERs and the Aggregator do not own the private keys used by the smart meters to cryptographically sign data, which prevents them from tampering with the recorded information. The smart meter is an external actor to the blockchain; nevertheless, it provides an equivalent degree of security by using the same cryptographic technology, i.e. encryption to prove its identity and cryptographically sign stored data. This is a key advantage since energy flow data can be published only when necessary, rather than storing them on the blockchain indefinitely, even when they are no longer needed.

However, there are scenarios that require the Operator's involvement in addressing disputes. Disputes could arise for the following reasons.

1. The Aggregator could provide to the SweepEpoch() function values for the energy flows lower than the ones detected by its smart meter.
2. A DER or the Aggregator could alter the smart meter's software, resulting in it detecting and recording a different energy flow than the actual one.
3. A DER or the Aggregator could tamper with the physical system which detects the energy flow without affecting the smart meter's software.
4. A malfunction could result in inaccurate recorded data.

Case 1 is a straightforward scenario since the Proof of Flow data provided by the Aggregator's smart meter would prove the real energy flow. This would result in penalties for the Aggregator. In cases 2, 3, and 4, the Proofs of Flow data allow the Operator to detect a discrepancy, however both Proofs of Flow data are legitimate from the point of view

of cryptographic signature. To determine the source of discrepancies, the Operator can examine the software of the DER's and the Aggregator's smart meters. If the software is compliant, a physical inspection, examining smart meters and transmission lines of both DER and Aggregator, can establish the cause of the discrepancy.

Considering that disputes can be initiated only by the DERs, as a temporary resolution from a software perspective, the Operator can pause the DER's participation in energy production, using the StopDER() function, without unlocking its deposit until the dispute is resolved. If the issue was caused by a malfunction, the Operator, using the ReAdmitDER() function, will re-admit and compensate the DER for the incurred economic losses. However, if the problem was caused by DER tampering, its deposit will be retained to discourage such behavior. On the contrary, if the discrepancy was due the Aggregator tampering, the Operator will re-admit the DER and compensate it for the incurred economic losses using the Aggregator's deposit. The Operator has also the authority to lock the contract, using the emergency stop pattern (see Section 4.8), while searching for a new reliable Aggregator. Once the Operator has resolved the dispute, it will emit the Resolution event, which will be published on the blockchain using the ResolveDispute() function.

There could be another scenario in which the provided Proofs of Flow data do not allow the Operator to identify a misbehavior. It occurs when the Aggregator and a DER collude against the Operator, manipulating either the software or the physical system, but resulting in compatible Proofs of Flow data. To keep this scenario from happening, it is the Operator's responsibility to monitor the total energy received from the Aggregator and check if there is a discrepancy with the aggregated amount detected, then the Operator can initiate the actions described above.

Consequences

Utilizing blockchain technology to synchronize DERs, the Aggregator and the Operator, provides a verifiable account of all their activities. All the actors involved can rest assured that no entity can take advantage of others, as their interactions are governed by the immutable and verifiable code of the smart contracts. The system of deposits, rewards and penalties incentivizes all participant to fulfill their obligations and discourage malicious behavior. Proofs of Flow represent the history of energy flows, and since they are cryptographically signed by smart meters and stored on the blockchain when requested, they provide reliable evidence to address discrepancies in energy supply and disagreements in payments. By avoiding unnecessary data storage on the blockchain, the Proof of Flow pattern not only mitigates potential privacy concerns, but also reduces the number of transactions, allowing to leverage a public blockchain such as Ethereum without being affected by fee fluctuations while avoiding expenses for on-chain storage.

4.5. Communication Details

Communication between DERs, the Aggregator, and the Operator takes place thanks to blockchain events emitted by ARIA EmitCommEvent() function. Each entity runs a listener that monitors events and when it identifies itself as the intended recipient of a request or a command, it activates the corresponding component to address the event prompt, taking necessary actions. Failure to respond within the specified timeframe of R epochs may result in penalties, except for the Operator. Table 3 lists the signals and their corresponding events, based on their sender.

Moreover, the Operator employs ARIA to publish other events that broadcast information, such as the alteration of the epoch duration E (EpochModified), the modification of the tolerance t (ToleranceModified), and the adjustment of penalties (PenaltyModified).

4.6. Checking Authorisations

All the above smart contracts (ARIA, Vault, Registry) retain the address of the owner (Operator) and the address of the authorized entity (Aggregator). This choice is a consequence of using the design patterns described in Section 4.8. e.g. due to the Authorization pattern, ARIA stores only one address for Aggregator in its _authorized variable, and it

stores only one owner in its _owner variable. The value of _authorized can be set by the owner only (which is the Operator).

Table 3. Signals categorization.

Sender	Signal/Event	Function	Effect
Aggregator	Activated	EmitCommEvent()	Notifies that a prospective DER has been moved to the active pool and must start providing energy
Aggregator	Deactivated	EmitCommEvent()	Notifies that an active DER has been removed from the active pool
Aggregator	Adjust	EmitCommEvent()	Asks DERs to adjust energy production levels [i]
Aggregator	Receipt	EmitCommEvent()	Writes payment receipts to wallets [ii]
Aggregator	Warning	EmitCommEvent()	Warns DERs about abnormal energy production
Aggregator	ProveTheFlow	ProveTheFlow()	Publishes a Proof of Flow for a blockstamp
Aggregator	ProveTheFlows	ProveTheFlow()	Publishes a Proofs of Flow for a period [iii]
Operator	Alert	EmitCommEvent()	Publishes alerts about system status or changes
Operator	ProvidePoF	EmitCommEvent()	Requests a Proof of Flow to DERs or the Aggregator for a blockstamp
Operator	ProvidePoFs	EmitCommEvent()	Requests a Proofs of Flow to DERs or the Aggregator for a period
Operator	Receipt	EmitCommEvent()	Writes payment receipts to wallets [iv]
Operator	Resolution	ResolveDispute()	Publishes the dispute resolution outcome
DER	Dispute	RaiseDisputeEvent()	Initiates a dispute addressed to the Operator
DER	Unavailability	PowerShortage()	Inform the Aggregator in advance about the inability to provide the requested energy
DER	ProveTheFlow	ProveTheFlow()	Publishes a Proof of Flow for a blockstamp
DER	ProveTheFlows	ProveTheFlow()	Publishes a Proof of Flow for a period

[i] In case of deviations or when power modulation is needed. [ii] Belonging to DERs that have been rewarded. [iii] Or a range of blockstamps [iv] Belonging to the Aggregator or DERs that have been rewarded.

When a function of ARIA has been called, to check whether the caller's address matches the value stored in the _owner or the _authorized variables, the Solidity's modifiers, custom-defined fragments of code attached to functions, come handy. If the condition specified in the modifier is not satisfied at the time the function has been invoked, the transaction requesting the execution of the code will be automatically reverted. i.e., msg.sender == _authorized requires that the address of the account that started the current function call, represented by the global variable msg.sender, matches the address stored in the authorized local variable. This enables restricting the invocation of a function to entities holding specific roles. e.g., SweepEpoch() can be executed by both the Aggregator and the Operator, while excluding other addresses; the function that triggers an emergency stop of the smart contract can be called by the Operator only; the Join() function can be invoked by DERs, namely any address distinct from the Aggregator or the Operator.

4.7. The SweepEpoch() Function in Detail

The Aggregator calls the SweepEpoch() function of ARIA at the beginning of each new epoch. This function carries out the following tasks.

1. Epoch conclusion and settlement: verify whether the energy quantities, as provided by the Aggregator, during the latest epoch, align with the amounts declared by active DERs. If the reported values match within the error tolerance t, then proceed to step 2, otherwise proceed to step 3.
2. Accredit: calculate the reward for each DER and proceed to step 4.
3. Penalty and deposit withholding: DERs are subject to a penalty when they have produced an average amount of energy below the requested amount without having notified the Aggregator. In this case a percentage of their deposit or credit is withheld as a penalty.
4. Reliability ranking: assess the reliability of each DER in the active pool and update their ranks accordingly.

5. DERs removal: remove from the active pool the DERs whose balance has been depleted due to fines and, according to the CalculateRanking() function, those whose rank falls below a specified threshold.
6. Contract expiration: move DERs whose contracts have expired from the active pool to the prospective-DERs list (i.e., they have provided energy for the specified number of epochs, d given in the contract).
7. Transaction fee allocation and payment: keep track of the gas consumed during its execution and distribute the cost between Aggregator and DERs, deducting it from their credits. This ensures that all parties contribute to covering the execution cost. Then, trigger the Vault to transfer the reward to the DERs' and Aggregator's wallets and emit Receipts events reporting the epoch nonce.
8. Activation: analyze the list of prospective-DERs to identify DERs that meet the criteria for providing energy. The commitment of eligible DERs is formalized by moving their wallet addresses from the prospective-DERs list to the active DERs' pool and publishing the Activated event that alerts DERs, specifying the number of epochs they are required to participate in supplying energy.

4.8. Lower-Level Design Patterns

The proposed pattern makes use of the following lower-level patterns.

- Owner Pattern [21,37]: this pattern involves storing the address of the contract deployer as the contract owner. It allows restricting to the owner some operations, ensuring privileged access, by checking whether the address of a user invoking a contract method matches the stored owner's address. In our solution, the Operator's address is the owner of all the smart contracts involved. As owner, the Operator can perform various activities, such as triggering an emergency stop of ARIA, setting the value for some variables (e.g., the address of ARIA stored in the Registry), or resolving disputes.
- Authorization pattern [21,43,44]: it can be considered as an evolution of the Owner pattern, allowing for more flexibility and granular control over smart contract actions. It involves storing a collection of addresses belonging to "super users" within an address mapping, (Solidity's hash table or dictionary). This allows different users to have different privilege levels and perform critical changes or operations that other users cannot. In our solution, this pattern enables both the Operator and the Aggregator to trigger functions in ARIA which then request Vault to transfer credits from one account to another.
- Migration Pattern [21,44]: this pattern employs a proxy contract that keeps a reference to the address of the actual contract. This strategy allows the contract owner to provide a new version of the contract by updating such a reference. Client code interacts with the proxy contract which either redirects execution to the new version of the actual contract or returns its address. This pattern addresses blockchain immutability, which prevents the direct modification of a contract's code, ensuring a smooth transition for users while enabling contract updates and improvements. This pattern is employed to enable the Operator to update new versions of ARIA to resolve bugs, add features, and improve functionality.
- Emergency Stop [21,37]: it enables the owner to trigger an emergency stop of the smart contract, locking its functions. This feature provides a mechanism to halt the contract's operations in case of emergencies. The Operator employs this pattern for emergency situations like system failures, security breaches, or any other critical event that requires an immediate halt to the operations or functionality of the system.
- Push-Based Outbound Oracle [21,45]: this pattern enables information exchange between the blockchain and the off-chain world. An off-chain Event Listener constantly monitors relevant changes on the blockchain and forwards event data to a Controller, which transfers the data to an off-chain component via a Transmitter. The Operator,

Aggregator, and DERs employ this pattern to detect requests issued by means of blockchain events and monitor block progression to identify the end of an epoch.

- Treasury Manager [21]: as described in Section 3.3, it is the adopted solution that consists in separating the logic from the balance, to avoid the transfer of the latter from the old version of a smart contract to a newer version.

Table 4 summarizes the above design patterns.

Table 4. Design patterns used by Proof of Flow design pattern.

Name	Main Characteristics
Owner [37]	Set the address of the sender of the deployment transaction as the owner of the contract
Authorization [44]	Perform an operation after authorization check
Migration [44]	Hide the real service
Emergency Stop [37]	Perform an operation after authorization check
Push-Based Outbound Oracle [45]	Perform the update of a state
Treasure manager [21]	Manage a financial reserve regardless of the main service

5. Discussion

The proposed system aims at facilitating seamless integration between Aggregators, Operators, and DERs that interact in a localized scale environment. This objective can be achieved by leveraging the blockchain's immutability, reliability, and consistency features. However, although blockchain technology ensures transparency, accountability, and integrity, it may also present limitations that are not always addressed by researchers who propose solutions that rely on this technology as a framework. e.g., one is the costs associated with transaction processing and on-chain storage. Our solution focuses on reducing the frequency of blockchain interactions to minimize the overhead and costs.

Regarding blockchain transactions and data stored on-chain, the immutability feature of blockchain technology assures that stored data cannot be altered or tampered with by anyone. However, outside economic transactions, handled by the consensus algorithm, there is no way to guarantee the veracity of data stored on-chain. Resolution of potential disagreements among parties about data stored in the blockchain is thus left to the application level. We have proposed a general solution, applied in the context of energy data, to address such kind of disagreements. Another crucial aspect to consider when using blockchain technology is the possibility of malicious actors who may attempt attacks. Our solution prevents malicious behavior and effectively coordinates the parties.

Below, we provide a list of limitations and illustrate how we have addressed them.

5.1. Blockchain Access

The first limitation consists in the requirement to query the blockchain to track events, detect the conclusion of an epoch, and acquire a blockstamp. Despite ARIA provides functions to generate a blockstamp, in the context of the Ethereum protocol, only the latest 256 blocks can be accessed through a smart contract [46]. If calls to these functions fail, it may become impossible to retrieve this information later. Such a limitation can be overcome by utilizing services like Etherscan (https://etherscan.io, accessed on 25 July 2023) just for reading data, or services Infura or Alchemy which allow the use of web3.js (https://web3js.org, accessed on 25 July 2023) functionalities. Web3.js is a JavaScript library that provides a collection of functions and utilities for interacting with the Ethereum blockchain. However, these services may not always be available without charge. As a result, it would be advisable to run a dedicated blockchain node to provide reliable and cost-effective access to blockchain data. The same principle applies when the system operates on a private or consortium blockchain like Energy Web.

5.2. Blockstamp Generation

A potential limitation we may come across in our use case is related to the blockstamp generation. As previously mentioned, blockstamps are similar to timestamps and are identifiers for specific moments, not in terms of time, but rather in the growth of the blockchain. To ensure reliability it is advisable to assign the blockstamp for the latest block to its k_{th} ancestor, as this block is deep enough to reduce the possibility of it being an orphan block or part of an accidental fork that could be eliminated by the protocol. According to Gervais et al. [47], setting k to 37 is safe when using Ethereum. Their research, while focusing on the previous Proof of Work (PoW) consensus algorithm, is nevertheless applicable to the Proof of Stake (PoS) algorithm that Ethereum adopted after "the merge" [33,34].

5.3. Synchronization

The third potential issue may arise, on public blockchains such as Ethereum, when heavy network traffic or too low gas price set for execution cause delays in transactions processing. This could eventually lead to wrong epoch synchronization and communication delays affecting the overall pace of the system.

We distinguish two major cases. The first one occurs when a DER fails to execute one or more transactions within the specified timeframe. These transactions could be in response to prompts from the Operator or the Aggregator, or they could involve the initiation of a dispute. Regardless of the mentioned scenario, the failure to complete the transactions within the designated time period results in a penalty for the DER, as it could be fined or experience a loss of credits if the dispute is not initiated correctly. The second major case occurs when the Aggregator's transaction, responsible for managing the current epoch and initiating the next one, is not executed within the specified timeframe. The SweepEpoch() function of ARIA must be executed at the end of each epoch before the subsequent epoch can take place. The nonce ensures the sequential consistency of epochs by requiring that epoch *i* cannot start until epoch *i-1* has been successfully processed. Therefore, if an epoch is not processed, the rewards for DERs will not be issued until the pending epoch is successfully processed. When this event occurs, the Operator may impose a penalty on the Aggregator to incentivize it to complete transactions on time and also to compensate DERs.

These two cases may prompt DERs and the Aggregator to increase the gas price to ensure the processing of their transactions, which can result in increased costs for transaction processing. However, not only our solution minimizes the interaction with the blockchain, but it also splits among the Aggregator and the active DERs the transaction fees when running the SweepEpoch() function, to lower the overall costs, as reported in the next section.

5.4. Gas Consumption

To evaluate the efficiency and operational costs of the designed smart contracts, we performed a series of simulations, under various conditions, with the aim of quantifying the gas consumption in different scenarios. Table 5 shows the gas consumption of the smart contracts' deployment in the first three rows, then the gas consumption for executing the main functions. Of course, unless ARIA's logic must be updated, deployment is performed just once for each smart contract, hence the related gas consumption is one-shot. The Join() function has a fixed cost due to the allocation of space for storing the data that characterize DERs, which must be saved.

We have analyzed the costs of SweepEpoch() considering three different cases and a varying total number of DERs. This quantity affects the cycles that access the lists of prospective and active DERs, while the scenarios "supply start", "ongoing supply", and "supply end", give different results due to some operations that are only carried out under specific circumstances. e.g., under normal conditions (i.e., a DER is compliant with the rules and the energy production is performed continuously), a DER is moved from one

list to another only at the beginning or at the end of its supply period. The selection of new DERs depends on the adopted criteria. In our experiments, DERs were selected in a first-come-first-served basis until the required aggregate power was reached.

Table 5. Measures of gas consumption when deploying smart contracts and when executing significant smart contract's functions.

Function	Scenario	Number of DERs	Gas Units
Deployment	Registry	-	231,024
Deployment	ARIA	-	3,620,829
Deployment	Vault	-	493,662
Join()	-	-	239,713
SweepEpoch()	supply start	1	212,391
SweepEpoch()	ongoing supply	1	164,662
SweepEpoch()	supply end	1	164,848
SweepEpoch()	supply start	5	444,786
SweepEpoch()	ongoing supply	5	551,031
SweepEpoch()	supply end	5	619,309
SweepEpoch()	supply start	10	735,426
SweepEpoch()	ongoing supply	10	1,076,641
SweepEpoch()	supply end	10	1,197,481

During each execution, the SweepEpoch() keeps track of the gas used and calculates the total cost by multiplying it with the tx.gasprice. The latter is set by the user when requesting a transaction and represents the price in wei per unit of gas (1 wei = 10^{-18} ETH, the smallest unit of Ether). The function then splits the cost equally among the active DERs and the Aggregator, and deducts it from their respective rewards. This approach ensures fairness in the allocation of costs.

In our analysis, the best-case scenario occurs with ten DERs in the "supply start" stage, consuming 73,543 gas units for each DER. The worst-case arises when a single DER is in the "supply start" phase, consuming 212,391 gas units.

According to Etherscan (https://etherscan.io/chart/gasprice, accessed on 1 September 2023) from August 2022 to August 2023, the minimum gas cost was just less than two Gwei, while the maximum was around 97,456 Gwei (1 Gwei == 10^9 wei). It should be noted that this maximum value was much higher than other observed max values during the said period, and much higher than the average. If SweepEpoch() had been invoked with the gas price set to 20 Gwei per gas unit, the cost would have been 14,708,520 Gwei (ETH 0.01470852). Taking into account the average price (according to https://www.coingecko.com, accessed on 1 September 2023) between the relative minimum and maximum values of ETH during the cited time range—USD 1606.89 (EUR 1506.32)—the final fees would amount to USD 23.63 (EUR 22.16) to be split among DERs and Aggregator, resulting in USD 2.15 (EUR 2.01) for each.

The significant gas price fluctuation demands monitoring it daily and setting an appropriate maximum gas price for the transaction that invokes the SweepEpoch(). This activity can be integrated into the Aggregator's listener that monitors the blockchain to generate blockstamps and detect the end of an epoch.

5.5. Block Time, Epoch Duration and Contract Expiration

As already mentioned, the block time, that in the case of Ethereum is twelve s, determines the pace of the blockchain, e.g., the speed at which a blockchain can add data or change its state. This characteristic affects the actors relying on the blockchain to exchange or store data, as well as perform transactions. In our case, while block time represents the minimum amount of time that actors have to wait to detect a change in the blockchain, the epoch E, (proportional to the block time), can vary. As noted earlier, an appropriate duration E for the epoch has to be chosen. Too short epochs may lead to higher costs as they involve more transactions, while long epochs can delay the updates to DERs' financial

balances for their produced energy. Conversely, when an entity needs to communicate with another, using the blockchain events, as when the Aggregator requires production modulation or notifies a DER that it has been chosen for energy production, its request can be submitted independently of the SweepEpoch() function; i.e., the request need not be synchronized with the function that governs epochs.

The value of E depends on the type of blockchain being used. In a private blockchain, which could be employed to aggregate DERs within a tightly knit area such as a neighborhood, islanded microgrids, or nanogrids, and where transaction costs are minimal, the value of E can be low, potentially as low as 1. Otherwise, when using a public blockchain like Ethereum, it may be impossible to keep up with the block time since transaction costs for the smart contract execution may be high as already said above. In this scenario, by setting the epoch to 72,000 blocks (running the SweepEpoch() function every ten days) the frequency of transactions is low. Then, during periods of network congestion, the value of E can be increased to mitigate the impact of high gas prices. This strategy, coupled with the division of fees among Aggregator and involved DERs, strikes a balance between regular updates and computational operations to minimize transactions costs.

In relation to the expiration of the contract, we propose a flexible contract duration, denoted by d epochs, during which a DER must provide energy. Once a DER's contract expires, the SweepEpoch() moves it back to the list of the prospective DERs, from where it can be selected again for the renewal of a new contract. This approach optimizes DER contributions by considering their capabilities, their ranking (when applicable), and the energy needs.

5.6. Wallet Generation and Ranking Criteria

Creating a wallet is a straightforward process that allows for the generation of a virtually unlimited number of wallets on the blockchain. This might result in a challenge related to an entity's identification when they are recognized by means of their wallet address. This issue could have an impact in our use case and arise under selective ranking criteria. A DER with a low rank, in an attempt to reset its status, might generate a fresh wallet address to subscribe again as a new participant. However, this action would lead to a financial loss, as the DER should provide a new deposit. In addition, since smart meters are provided by the Operator, their IDs cannot change nor can they be modified, and ARIA is designed to reject subscriptions from seemingly new DERs that provide a smart meter's ID that has already been used by a different wallet address.

5.7. Transactions Must Always Originate from the User (Or Software Wallet)

DERs or the Aggregator could cease their activity abruptly at the expense of the system; however, this behaviour would put them at a loss. For both, the abnormal cease of activity leads to financial losses, as they forfeit their deposits. Moreover, though the Aggregator triggers payments, by calling the SweepEpoch() function, it cannot keep the user funds, as it cannot have direct access to the balance held in Vault. Therefore, it could not benefit from such a malicious behavior. Nevertheless, in the unlikely event of this scenario, the Operator, in virtue of the Authorization pattern, has the authority to reward compliant DERs and unlock their balances to ensure that they can recover their funds.

Considering that the adopted deposit system could result in losses, it is crucial to carefully determine the security deposit amount. Opting for a lower deposit could improve accessibility and increase the participation opportunities for DERs. To ensure a balance between accessibility and risk mitigation, one potential approach could be to lock the earned credits in the Vault, instead of making them immediately redeemable, and gradually release them. This strategy raises the stakes and encourages proper behavior from all participants in the system. However, network malfunction or power outages could lead to such complications described in this scenario, thus, it is essential to allow for a certain degree of flexibility before considering a DER or the Aggregator as non-compliant with the system's rules.

5.8. Future Research

In future work, we plan to extend the Treasury Manager to be able to create payment channels [48], thus enhancing the functionality of the Proof of Flow pattern and enabling more granular and lower-fee payments to DERs. Another line of research is to modulate the epoch duration depending on the analysis of the fluctuating gas price and fees patterns, with the goal of minimizing the costs as much as possible.

6. Conclusions

In our research we aimed at finding a solution to govern Distributed Energy Resources (DERs) in localized settings, like microgrids or small energy communities, by means of an Aggregator that can provide maximum transparency on its activities and decisions, such as: choosing participating DERs, giving financial rewards or penalties, to DERs; while also minimising the costs associated to the use of resources for executing automatic procedures and storing data.

The proposed solution consists of a design pattern named Proof of Flow that leverages blockchain technology to build trust among potentially distrustful energy players, i.e., DERs and Aggregator, hence avoiding to rely on a centralized authority. A smart contract, named ARIA, is responsible for assessing whether DERs can be accepted as energy providers, and for giving them rewards (for their production) or penalties (when the agreements have not been satisfied). DERs, producing energy, measure the provided energy and locally store the amount of data and the time of recording, the Aggregator (being physically connected with each DER) measures the energy that has been transmitted by each DER. ARIA emits events to notify energy players when they can start transmitting energy, when they have been payed, or when they need to provide evidence of their energy production. Thanks to the use of blockchain technology, such a smart contract emits events that are permanently stored, visible to interested players, and that cannot be contested. To mitigate the known problem of the cost typically associated with Ethereum blockchain, due to the use of computation and storage resources, we have resorted to an external local storage for the big amounts of data holding the measured energy produced and minimized accesses to smart contracts. We have also proposed to use hashes of the blocks in the blockchain as timestamps to easily align with the pace of block generation and then order recorded data that have been stored in DERs' external hosts, hence overcoming possible tampering with local clocks or their desynchronization. Potential disagreements between a DER and an Aggregator on measured energy production and financial balance due have been characterised and dealt with by an automatic procedure.

As the proposed solution enhances trust among energy players due to transparency of activities, recorded events, and verifiable execution and data storage, it has tackled high-level issues that were not solved. This could effectively assist in the adoption of a fair energy production system. Moreover, by keeping the biggest parts of data needed to the governed system outside the blockchain itself, our solution minimizes blockchain access, storage, and associated costs, without compromising the trust reinforcement made available by blockchain mechanisms, while enhancing the privacy of the data related to energy production. Therefore, we have mitigated the inherent limitations of blockchain technology that hinder its adoption.

Author Contributions: Conceptualization, V.M., G.P. and E.T.; methodology, V.M. and E.T.; software, V.M.; validation, V.M. and E.T.; writing—original draft preparation, V.M.; writing—review and editing, V.M, G.P. and E.T.; supervision, G.P.and E.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge the support of University of Catania PIACERI 2020/22 Project "TEAMS".

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Di Somma, M.; Graditi, G.; Siano, P. Optimal bidding strategy for a DER aggregator in the day-ahead market in the presence of demand flexibility. *IEEE Trans. Ind. Electron.* **2018**, *66*, 1509–1519. [CrossRef]
2. O'Donovan, P.; O'Sullivan, D.T. A systematic analysis of real-world energy blockchain initiatives. *Future Internet* **2019**, *11*, 174. [CrossRef]
3. Marin, O.; Cioara, T.; Anghel, I. Blockchain Solution for Buildings' Multi-Energy Flexibility Trading Using Multi-Token Standards. *Future Internet* **2023**, *15*, 177. [CrossRef]
4. Burger, S.; Chaves-Ávila, J.P.; Batlle, C.; Pérez-Arriaga, I.J. A review of the value of aggregators in electricity systems. *Renew. Sustain. Energy Rev.* **2017**, *77*, 395–405. [CrossRef]
5. Antonopoulos, A.M.; Wood, G. *Mastering Ethereum: Building Smart Contracts and Dapps*; O'reilly Media: Sebastopol, CA, USA, 2018.
6. Sanka, A.I.; Irfan, M.; Huang, I.; Cheung, R.C. A survey of breakthrough in blockchain technology: Adoptions, applications, challenges and future research. *Comput. Commun.* **2021**, *169*, 179–201. [CrossRef]
7. Fornaia, A.; Marotta, G.; Pappalardo, G.; Tramontana, E. A Decentralized Solution for Epidemiological Surveillance in Campus Scenarios. *IEEE Access* **2022**, *10*, 103806–103818. [CrossRef]
8. Saad, M.; Spaulding, J.; Njilla, L.; Kamhoua, C.; Shetty, S.; Nyang, D.; Mohaisen, D. Exploring the attack surface of blockchain: A comprehensive survey. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 1977–2008. [CrossRef]
9. Aggarwal, S.; Kumar, N. Attacks on blockchain. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2021; Volume 121, pp. 399–410.
10. Guo, H.; Yu, X. A survey on blockchain technology and its security. *Blockchain Res. Appl.* **2022**, *3*, 100067. [CrossRef]
11. Li, Z.; Kang, J.; Yu, R.; Ye, D.; Deng, Q.; Zhang, Y. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3690–3700. [CrossRef]
12. Mnatsakanyan, A.; Albeshr, H.; Al Marzooqi, A.; Bilbao, E. Blockchain-integrated virtual power plant demonstration. In Proceedings of the 2nd International Conference on Smart Power & Internet Energy Systems (SPIES), IEEE, Bangkok, Thailand, 15–18 September 2020; pp. 172–175.
13. Gayo, M.; Santos, C.; Sánchez, F.J.R.; Martin, P.; Jiménez, J.A.; Tradacete, M. Addressing challenges in prosumer-based microgrids with blockchain and an IEC 61850-based communication scheme. *IEEE Access* **2020**, *8*, 201806–201822. [CrossRef]
14. Yang, Q.; Wang, H. Exploring blockchain for the coordination of distributed energy resources. In Proceedings of the 55th Annual Conference on Information Sciences and Systems (CISS), IEEE, Baltimore, MD, USA, 24–26 March 2021; pp. 1–6.
15. Gawusu, S.; Zhang, X.; Ahmed, A.; Jamatutu, S.A.; Miensah, E.D.; Amadu, A.A.; Osei, F.A.J. Renewable energy sources from the perspective of blockchain integration: From theory to application. *Sustain. Energy Technol. Assess.* **2022**, *52*, 102108. [CrossRef]
16. Mihaela, C.; Mircea, C.; Daniel, D. Smart Hub Electric Energy Data Aggregation Platform for Prosumers Grid Integration. In Proceedings of the 9th International Conference on Modern Power Systems (MPS), IEEE, Cluj-Napoca, Romania, 16–17 June 2021; pp. 1–7.
17. Khoshjahan, M.; Kezunovic, M. Blockchain implementation for DER visibility and transaction verification in wholesale market. In Proceedings of the Transmission and Distribution Conference and Exposition (T&D), IEEE, New Orleans, LA, USA, 25–28 April 2022; pp. 1–5.
18. Johnson, G.; Sebastian-Cardenas, D.J.; Balamurugan, S.P.; Harun, N.F.; Mukherjee, M.; Blonsky, M.; Markel, T.; Johnson, B. A Unified Testing Platform to mature Blockchain applications for Grid Emulation environments. In Proceedings of the Transactive Energy Systems Conference (TESC), IEEE, Portland, OR, USA, 2–6 May 2022; pp. 1–5.
19. Malik, S.; Thakur, S.; Breslin, J.; Duffy, M. Blockchain based Decentralized Home Energy Management System using Double Auction. In Proceedings of the The Fourteenth International Conference on Information, Process, and Knowledge Management (eKNOW), Porto, Portugal, 26–30 June 2022.
20. Mnatsakanyan, A.; Albeshr, H.; Almarzooqi, A.; Iraklis, C.; Bilbao, E. Blockchain mediated virtual power plant: From concept to demonstration. *J. Eng.* **2022**, *2022*, 732–738. [CrossRef]
21. Mandarino, V.; Pappalardo, G.; Tramontana, E. Some Blockchain Design Patterns for Overcoming Immutability, Chain-Boundedness, and Gas Fees. In Proceedings of the 3rd Asia Conference on Computers and Communications (ACCC), IEEE, Shanghai, China, 16–18 December 2022; pp. 65–71.
22. Gu, Q.; Qu, Q. Towards an Internet of Energy for smart and distributed generation: Applications, strategies, and challenges. *J. Comput. Des. Eng.* **2022**, *9*, 1789–1816. [CrossRef]
23. Papadopoulos, K. Using smart contracts in smart energy grid applications. In *Sinteza 2019-International Scientific Conference on Information Technology and Data Related Research*; Singidunum University: Belgrade, Serbia, 2019; pp. 597–602.
24. Münsing, E.; Mather, J.; Moura, S. Blockchains for decentralized optimization of energy resources in microgrid networks. In Proceedings of the Conference on Control Technology and Applications (CCTA), IEEE, Trieste, Italy, 22–25 August 2022; pp. 2164–2171.

25. Calvagna, A.; Casablanca, E.; Marotta, G.; Pappalardo, G.; Tramontana, E. Providing Trust in a Dynamic Distributed Energy Production Scenario by means of a Blockchain. In Proceedings of the Workshop on Blockchain for Renewables Integration (BLORIN), IEEE, Palermo, Italy, 2–3 September 2022; pp. 13–18.
26. Ahmad, S.; Ahn, B.; Kim, T.; Choi, J.; Chae, M.; Han, D.; Won, D. Blockchain-Integrated Resilient Distributed Energy Resources Management System. In Proceedings of the International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), IEEE, Singapore, 25–28 October 2022; pp. 59–64.
27. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Tech. Rep., 2008. Available: https://bitcoin.org/bitcoin.pdf (accessed on 1 September 2023).
28. Nguyen, C.T.; Hoang, D.T.; Nguyen, D.N.; Niyato, D.; Nguyen, H.T.; Dutkiewicz, E. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access* **2019**, *7*, 85727–85745. [CrossRef]
29. Li, A.; Wei, X.; He, Z. Robust proof of stake: A new consensus protocol for sustainable blockchain systems. *Sustainability* **2020**, *12*, 2824. [CrossRef]
30. Lashkari, B.; Musilek, P. A comprehensive review of blockchain consensus mechanisms. *IEEE Access* **2021**, *9*, 43620–43652. [CrossRef]
31. Xiong, H.; Chen, M.; Wu, C.; Zhao, Y.; Yi, W. Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms. *Future Internet* **2022**, *14*, 47. [CrossRef]
32. Hao, Y.; Li, Y.; Dong, X.; Fang, L.; Chen, P. Performance analysis of consensus algorithm in private blockchain. In Proceedings of the Intelligent Vehicles Symposium (IV), IEEE, Changshu, China, 26–30 June 2018; pp. 280–285.
33. Kapengut, E.; Mizrach, B. An Event Study of the Ethereum Transition to Proof-of-Stake. *Commodities* **2023**, *2*, 96–110. [CrossRef]
34. Grandjean, D.; Heimbach, L.; Wattenhofer, R. Ethereum Proof-of-Stake Consensus Layer: Participation and Decentralization. *arXiv* **2023**, arXiv:2306.10777.
35. Yang, R.; Murray, T.; Rimba, P.; Parampalli, U. Empirically analyzing ethereum's gas mechanism. In Proceedings of the European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, Stockholm, Sweden, 17–19 June 2019; pp. 310–319.
36. Donmez, A.; Karaivanov, A. Transaction fee economics in the Ethereum blockchain. *Econ. Inq.* **2022**, *60*, 265–292. [CrossRef]
37. Wöhrer, M.; Zdun, U. Design patterns for smart contracts in the ethereum ecosystem. In Proceedings of the International Conference on Internet of Things (iThings) and Green Computing and Communications (GreenCom) and Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData), IEEE, Halifax, NS, Canada, 30 July–3 August 2018; pp. 1513–1520.
38. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Pearson: London, UK, 1995.
39. Hindér, J.; Wijk, S. Controlled Charging of Electrical Vehicles on Residential Power Grid. Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2018.
40. Peng, D.g.; Zhang, H.; Yang, L.; Li, H. Design and realization of modbus protocol based on embedded linux system. In Proceedings of the International Conference on Embedded Software and Systems Symposia, IEEE, Zhejiang, China, 29–31 July 2008; pp. 275–280.
41. Löfström, E. Smart meters and people using the grid: Exploring the potential benefits of AMR-technology. *Energy Procedia* **2014**, *58*, 65–72. [CrossRef]
42. Mayer, H. ECDSA security in bitcoin and ethereum: A research survey. *CoinFaabrik* **2016**, *28*, 50.
43. Bartoletti, M.; Pompianu, L. An empirical analysis of smart contracts: Platforms, applications, and design patterns. In Proceedings of the Financial Cryptography and Data Security: FC International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, 7 April 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 494–509.
44. Worley, C.R.; Skjellum, A. Opportunities, challenges, and future extensions for smart-contract design patterns. In Proceedings of the Business Information Systems Workshops: BIS International Workshops, Seville, Spain, 26–28 June 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 264–276.
45. Mühlberger, R.; Bachhofner, S.; Castelló Ferrer, E.; Di Ciccio, C.; Weber, I.; Wöhrer, M.; Zdun, U. Foundational oracle patterns: Connecting blockchain to the off-chain world. In Proceedings of the Business Process Management: Blockchain and Robotic Process Automation Forum: BPM Blockchain and RPA Forum, Seville, Spain, 13–18 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 35–51.
46. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
47. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the ACM SIGSAC Conference On Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 3–16.
48. Erdin, E.; Mercan, S.; Akkaya, K. An evaluation of cryptocurrency payment channel networks and their privacy implications. *arXiv* **2021**, arXiv:2102.02659.