The BRIGHT project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 957816

Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design

Deliverable D6.1 DLT/Smart contracts Data Governance for digital fingerprinting of energy data – first version

Author(s): Denisa Ziu (ENG), Vincenzo Croce (ENG), Claudia Antal (TUC), Liana Toderean (TUC), Tudor Cioara (TUC), Ionut Anghel (TUC), Marcel Antal (TUC)

Imprint

Title:	**DLT/Smart contracts Data Governance for digital fingerprinting of energy data – first version**
Contractual Date of Delivery to the EC:	31.12.2021
Actual Date of Delivery to the EC:	27.12.2021
Author(s):	Denisa Ziu (ENG), Vincenzo Croce (ENG), Claudia Antal (TUC), Liana Toderean (TUC), Tudor Cioara (TUC), Ionut Anghel (TUC), Marcel Antal (TUC)
Participant(s):	ENG, TUC
Project:	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design (BRIGHT)
Work Package:	WP6 –DLT Enablers for Decentralized VPP
Task:	T6.1 – DLT & Smart contracts for cross-stakeholder trusted hybrid interoperable data sharing
Confidentiality:	Public
Version:	1.0

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Table 1 List of Acronyms and Abbreviations

ABAC	Attribute-Based Access Control
ABE	Attribute-Based Encryption
API	Application Programming Interface
BRIGHT	Boosting DR through increased communIty-level consumer engaGement by combining Data-driven and blockcHain technology Tools with social science approaches and multi-value service design
DLT	Distributed Ledger Technology
DR	Demand-Response
ETH	Ether
EV	Electric Vehicle
EVM	Ethereum Virtual Machine
FT	Fungible Token
IoT	Internet of Things
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
M2M	Machine-to-Machine
NFT	Non Fungible Token
PN-tier	Protocol and Network tier
P2P	Peer-to-Peer
RBAC	Role-Based Access Control
S-tier	Scalability tier
VPP	Virtual Power Plant
WP	Work Package
XACML	eXtensible Access Control Markup Language

Executive Summary

This document presents BRIGHT's Deliverable D6.1 - DLT/Smart contracts Data Governance for digital fingerprinting of energy data – first version, developed under task T6.1 - *DLT & Smart contracts for cross-stakeholder trusted hybrid interoperable data sharing* of Work Package 6 - DLT Enablers for Decentralized VPP.

This deliverable describes the first version of scalable distributed ledger for tamper evident storage and sharing of smart energy cross-domain that is the expected output of the task. Also providing the view of underlying infrastructural support for decentralized management of the energy community.

The first part of the deliverable introduces to blockchain technology fundamental principles. Specific attention is addressed to energy sector and its use cases. In addition, the Ethereum blockchain and smart contracts are presented in more detail, as these technologies are the ones used in the BRIGHT project.

The second part of this document contains the design and implementation of the first version of the **BRIGHT Scalable Distributed Energy Ledger**. The distributed ledger is composed of two tiers: 1) the Protocol and Network tier represents the blockchain infrastructure which the energy community members are registered as peer nodes of the network and their monitored energy values are stored as energy transactions; 2) the Scalability tier represents the technological solutions that deal with the scalability of the blockchain in registering and managing the energy transactions inside the community.

Finally, the third part of the deliverable reports on **BRIGHT data access policies**. As an initial step, in the first version of the deliverable, existing approaches in the field of blockchain-driven access control mechanisms are analysed. In the final version of the deliverable, i.e. D6.4, we will implement the integrated solution to manage data access policies in BRIGHT.

1. Introduction

1.1. Purpose

This deliverable D6.1 – DLT/Smart contracts Data Governance for digital fingerprinting of energy data – first version – aims to describe the solution developed in the BRIGHT project to provide secure data storage supporting decentralised management of energy communities through the blockchain technology.

Although it is possible to store data directly on-chain; this is not always efficient in terms of costs associated with processing data on the chain and in terms of scalability. Therefore, an efficient, cost-effective and scalable approach has been developed, which allows energy monitoring transactions to be recorded less frequently on the chain (e.g. one transaction per hour), without losing any of the benefits offered by the blockchain technology.

1.2. Relation to Other Activities

This deliverable is the output of Task 6.1 that benefits from the activities of WP2, especially regarding the use cases defined in deliverable D2.1 *User group needs, requirement and advanced DR engagement scenarios* and the requirements defined in deliverable D2.3 *DR technologies and tools*. The output of this deliverable is particularly relevant for the rest of activities in WP6.

The tools developed in WP6 will be tested and validated in BRIGHT pilot sites in the context of WP7.

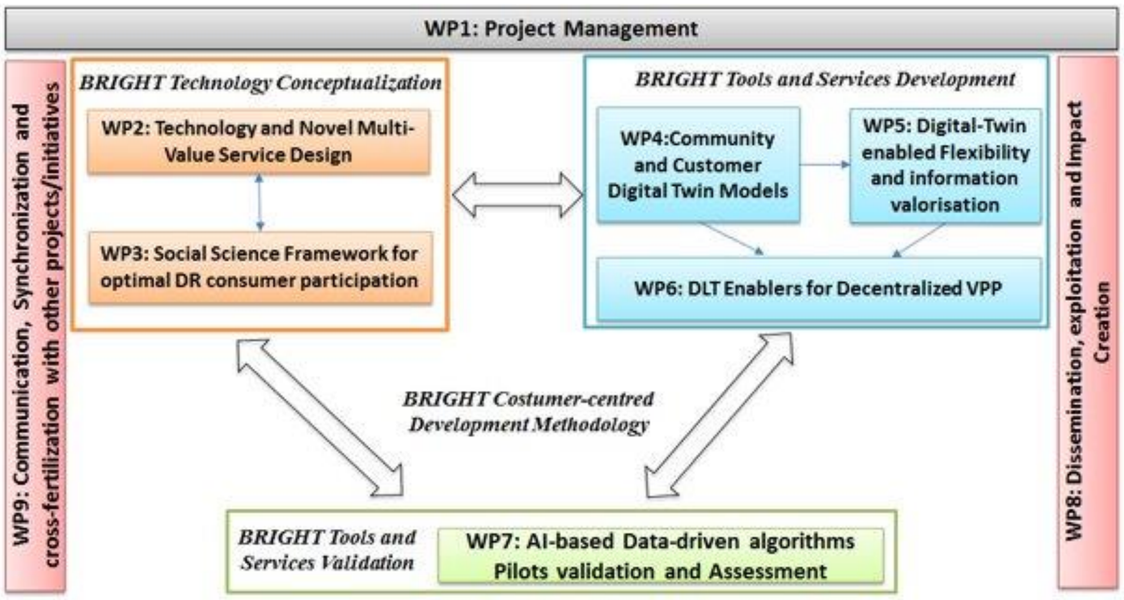

WP1: Project Management
WP9: Communication, Synchronization and cross-fertilization with other projects/initiatives
BRIGHT Technology Conceptualization
WP2: Technology and Novel Multi-Value Service Design
WP3: Social Science Framework for optimal DR consumer participation
BRIGHT Tools and Services Development
WP4:Community and Customer Digital Twin Models
WP5: Digital-Twin enabled Flexibility and information valorisation
WP6: DLT Enablers for Decentralized VPP
WP8: Dissemination, exploitation and Impact Creation
BRIGHT Costumer-centred Development Methodology
BRIGHT Tools and Services Validation
WP7: AI-based Data-driven algorithms Pilots validation and Assessment

Figure 1BRIGHT project relation among activities

1.3. Structure of the Document

Deliverable D6.1 is structured as follows:

- Chapter 2 is dedicated to the description of the blockchain technology and its application in the energy domain. It also presents the description of Ethereum blockchain and smart contracts, since these are the main technologies exploited in the BRIGHT project;
- Chapter 3 provides a detailed description of the design and implementation of the Energy Community Distributed Ledger. The implemented solution is based on the blockchain technology and it is able to provide the required secure and reliable means for peer-to-peer energy trading and for managing the communication between community-level energy stakeholders;
- Chapter 4 provides an introduction to the BRIGHT data access policies. In BRIGHT, the blockchain technology will be exploited in order to implement a mechanism for managing and enforcing data access policies. An overview of the blockchain-based data access control use cases in literature is also given;
- Chapter 5 concludes the deliverable.

2. Blockchain as a disruptive technological innovation for the energy sector

Blockchain has been considered an innovative technology, identified as to be as disruptive as Internet was considered when it was first introduced. Blockchain promises innovation in the commercial and financial area which is comparable to the impact the web has had on communication [1]. It is revolutionising the way we interact based on these main key advantages:

- **Traceability and data storage** - decentralised and distributed system that becomes a secure way to track changes in information over time;
- **Trust** - the creation of trust among untrusted participants;
- **Peer-to-peer transactions** - the absence of intermediaries promotes democracy.

In the energy sector, decentralization, decarbonization, and digitalization are the three key pillars of the global energy transition. Blockchain technology could help to address the challenges faced by future energy systems. Many use cases can be applied in the field of IoT and P2P trading, and thanks to the advanced cryptography that characterises this technology, it can solve the problems of cybersecurity and privacy preserving management. Coupled with smart contracts, blockchain can enable novel business solutions. The concept of smart contracts, first described by Nick Szabo long before the emergence of blockchain technology, denotes "a set of promises, specified in digital form, including protocols within which the parties perform on these promises". In their simplest form, for example, they are used to transfer assets to a buyer once the payment is made — comparable to a vending machine. When combined with blockchain, the rules set out in smart contracts must be fulfilled for a change of state to be triggered within the blockchain. This is the idea behind Ethereum, a platform that provides a blockchain with an integrated 'Turing-complete' programming language, meaning that it is suitable for any smart contract or transaction that can be defined mathematically.

2.1. Ethereum

The Blockchain is a distributed ledger, based on a shared and distributed database, containing a log of transactions in chronological order. Transactions are grouped into blocks and chained through cryptographic hashes into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work [2] (Figure 2). The main purpose of blockchain technology is to remove the need for intermediaries and replace them with a distributed network of digital users, who work in partnership to verify transactions and safeguard the integrity of the ledger. Differently to centralised systems, every member of the blockchain network holds his copy of the ledger or can access it in the open cloud. As a result, anyone in the network can have access to the historic log of the system transactions and verify their validity, enabling a high level of transparency.

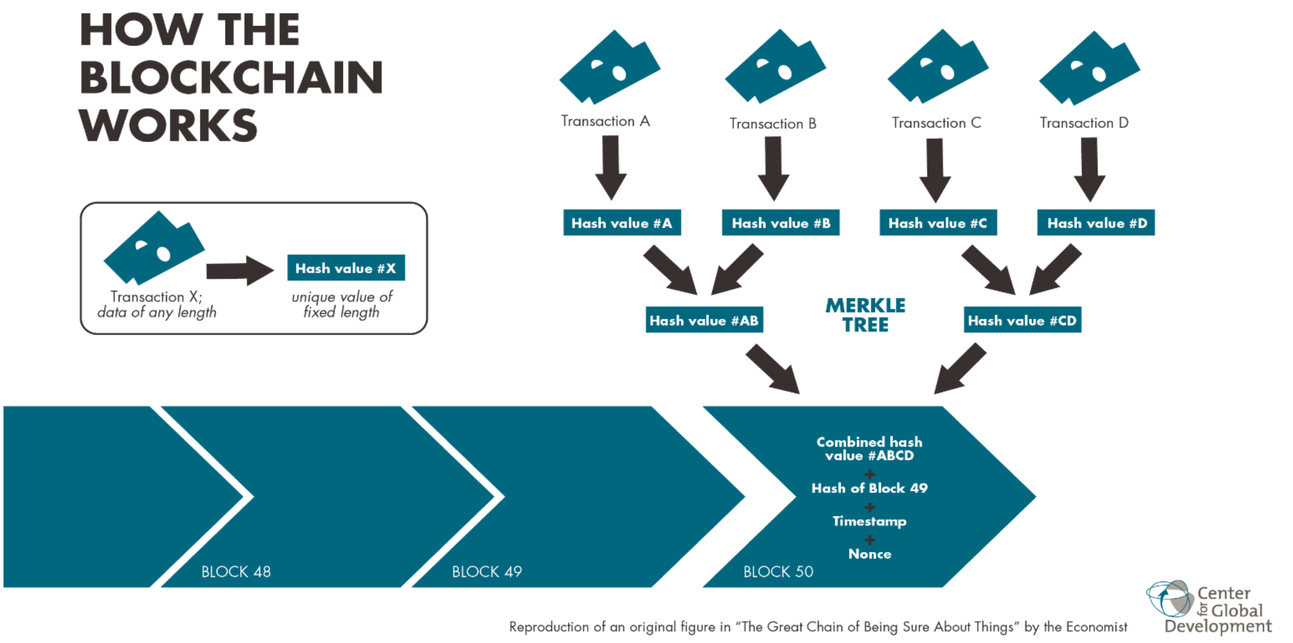


Figure 2 Blockchain technology (Reproduction of original figure in the "The Great Chain of Being Sure about Things" by the Economist)

The two most popular blockchains are Bitcoin and Ethereum. Bitcoin is the world's first cryptocurrency, established in 2009 following the public release of a paper by Nakamoto [2], an author whose real identity remains unknown. He proposed a distributed electronic cash payment system that uses P2P communication of anonymous and unknown Internet users. Digital cash transacted between users is not issued or controlled by a central bank, but by a network of computers that operate in collaboration and use cryptography to assure security. Bitcoins are created as a reward for a process known as mining. Bitcoin mining is performed by high-powered computers that solve complex computational mathematical problems; these problems are so complex that they cannot be solved by hand and are complicated enough to overload even incredibly powerful computers. The result of bitcoin mining is twofold. First, when computers solve these complex mathematical problems on the bitcoin network, they produce new bitcoins. And second, by solving computational mathematical problems, bitcoin miners make the bitcoin payment network reliable and secure by verifying transaction information.

Blockchains can be distinguished according to the consensus mechanism and programming capabilities. Considering the consensus mechanism, blockchains differ in the definition of the participation of nodes in the distributed network and the roles that they can perform. In particular, open blockchains and permissioned (or private) blockchains can be distinguished. Considering the programming capabilities, it is possible to differentiate between blockchains programmable via simple scripting and blockchains providing Turing-complete computational capabilities, enabling the creation of "smart contracts". Ethereum was the first blockchain supporting smart contracts and it is still the most notable example of Turing-complete programmable blockchain.

Ethereum was proposed by Vitalik Buterin in 2013 [3] and a further detailed analysis was provided by Gavin Wood in the 'yellow paper' (Ethereum: A Secure Decentralised Generalised Transaction

Ledger [4]). As the Ethereum website [5] reports, “Ethereum is a decentralized platform that runs smart contracts.” These contracts run on the “Ethereum Virtual Machine” (EVM), a distributed computing network made up of all the devices running Ethereum nodes. Like other blockchains, Ethereum has a native cryptocurrency called Ether (ETH) and it has a double use: it is used as an incentive for the network “validators”, but also to regulate the use of the blockchain computational resources.

The smart contracts are written in a low level bytecode language interpreted by the EVM. High level languages whose programs can be compiled in EVM bytecode (producing a .bin file containing the binary of the compiled contract and an .abi file containing the contract interface specification) have also been developed to ease human smart contract coding. The most widespread of such languages is a JavaScript style language called Solidity [6].

It is important to remark that every transaction has to pay a fee proportional to its complexity to repay the miners for their effort of maintaining the EVM. To every single operation of the EVM is assigned (by the protocol) a price proportional to its burden to the users (i.e., the number of computational steps needed for its execution and its storage weight), this is called *gas* and the total gas of a transaction is the summation of all the gas of every single instruction it contains. This is the gas that is consumed by the transaction upon validation. The entity (either a user or a contract) creating the transaction needs to decide two parameters, the *gas limit* and *gas price*. The gas limit is the maximum amount of gas the transaction is allowed to consume, if it is exceeded all gas is spent but the execution effects on the state are eliminated. This is useful to avoid too long or even infinite computations that would stall the EVM. Furthermore each block has associated a block gas limit to guarantee a limit to the amount of computation executed by all the transactions in that single block. The gas price is instead set by the user as the amount of ether the user is willing to pay for each unit of gas. Miners are free to choose what transaction to mine and so they can refuse the ones with a gas price too low.

2.2. Smart contracts

The term smart contract was introduced in 1994 by Nick Szabo in [7], when he first described how the computer-based execution of contracts between two parties could be secured without requiring any third party: “A set of promises, including protocols within which the parties perform on the other promises. The protocols are usually implemented with programs on a computer network, or in other forms of digital electronics, thus these contracts are ‘smarter’ than their paper-based ancestors”. In the blockchain context, it is generally related to computer code that is stored on a blockchain and that can be accessed by one or more parties. These programs are often self-executing and make use of blockchain properties like tamper-resistance, decentralised processing, and so on. In this interpretation, used for example by the Ethereum Foundation, a smart contract is not necessarily related to the classical concept of a contract, but can be any kind of computer program. It is called a ‘contract’ because the code that runs on Ethereum can control valuable things like ETH, currency notes or other digital assets [8].

The possibilities are infinite for smart contracts. They can be used to code and automate business processes that can be shared and executed among multiple parties offering increased trust and reliability in the process, often with significant gains in efficiency and cost reduction. Smart contracts can also be used to hard-code agreements between parties involving value and other types of asset transfer and allow them to be very transparent and run automatically based on predetermined rules, making it impossible for a party to back out. Smart contracts are used for tokenisation. A token is the digital representation of an asset on the blockchain. This asset can be both digital or physical, as well as tangible or intangible. The most important platform for the generation of tokens today is the Ethereum blockchain.

Smart contracts can directly manage payments between two (or more) actors on a blockchain, being completely self-enforcing. It is possible to transact cryptocurrency or tokens. A cryptocurrency is a digital currency that uses encryption techniques to secure and verify its transactions, while the term token usually indicates a digital representation of a specific asset. Unlike a currency, a token is not native to a blockchain, but it is created on top of a blockchain, and it is governed by a smart contract. The value of the tokens depends on the value of the underlying assets and services that the tokens represent. Two main categories of tokens are defined on Ethereum: ERC-20 standard [9] for Fungible Tokens (FTs) that are inter-changeable and not-unique and so can be used to represent a value like a currency note, and ERC-721 standard [10] for Non Fungible Tokens (NFTs), representing a unique asset like a collectable good. One of the first NFT projects to gain significant traction was CryptoKitties [11], a game developed on Ethereum that allows players to collect, breed and trade virtual cats.

2.3. Blockchain applications in the energy domain

Both energy generation and blockchain technology are enabling more democratic and decentralized markets. Future power networks will host more decentralized and new components with increased autonomy. The increased use of electric vehicles is expected to increase the complexity of future power systems and markets. Blockchain technology is a perfect match for future power markets and grids which develops decentralization at the core.

The idea of using blockchain in the energy sector has taken an increasingly large interest in academic research, industry stakeholders, utility companies, and energy decision-makers. According to a study on 140 blockchain innovation projects and research initiatives [12], in the energy space it is possible to identify these use-cases: (1) Metering/billing and security; (2) decentralized energy trading; (3) green certificates and carbon trading; (4) grid management; (5) IoT, smart devices, automation and asset management; (6) electric e-mobility; and (7) general purpose initiatives and consortia. Figure 3 reports the amount, in per cent, of blockchain-based applications or pilot-projects in different areas of the energy sector. It is possible to observe that decentralized energy trading is the most investigated area, while the other areas have a similar number of applications.

As a matter of fact, the most intuitive and popular application of blockchain to the electric power sectors is to turn the electricity grid into a peer-to-peer network for prosumers who trade electricity with one another. The most famous blockchain-based system of P2P energy trading is the Brooklyn Microgrid, allowing owners of rooftop solar panels in Brooklyn, New York City, to sell any electricity produced in excess of their needs directly to their neighbours [13]. While smart meters track electricity produced and consumed, an Ethereum blockchain records smart contracts, enabling automated peer-to-peer transactions. Similar pilot projects have been launched all over the world, such as Power Ledger in Australia [14] or SOLshare in Bangladesh [15].

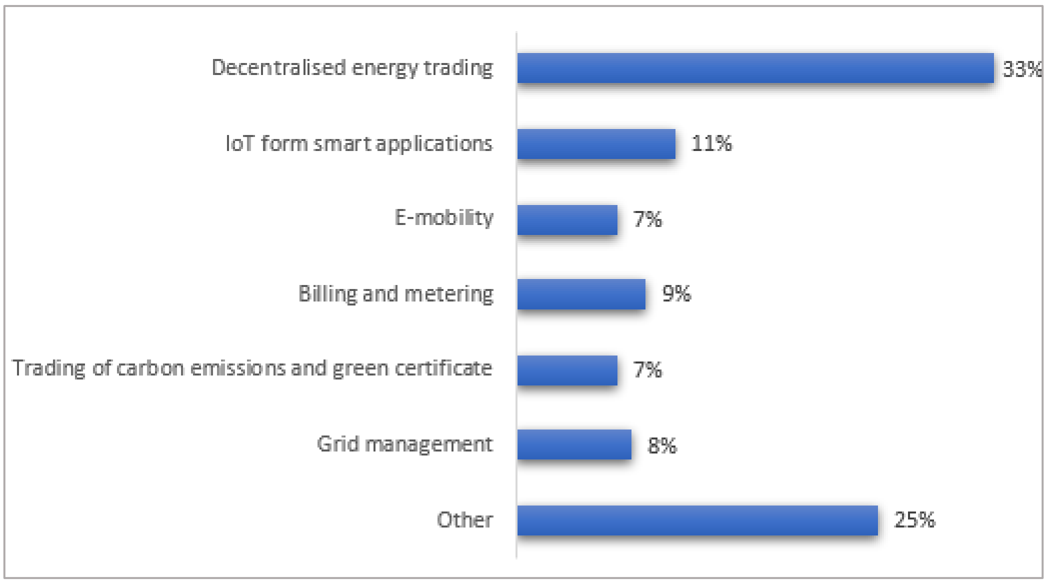


Figure 3 Blockchain based applications for different fields of the energy sector

The use of blockchain technology for the emerging field of IoT and, subsequently, energy efficiency in smart grid is another area of active research [16] [17]. Blockchain and self-enforcing smart contracts can facilitate data exchange and machine-to-machine (M2M) communication among energy metering devices. Oli SharEnergy [18] proposes a software platform to enable the optimization of smart grid components, such as single small power plants, prosumers, and storage system. The smart energy grid is formed by 'energy cells' (one cell is composed of at least one generation unit and single or multiple loads) which collaborate via blockchain. When the metering infrastructures are integrated with blockchain technology, it is possible to benefit from automated billing in energy services for consumers and distributed generators, which leads to the reduction of administrative costs. Similarly, blockchain technology can be used for collective self-consumption aiming to certify, validate, and automatically execute transactions between consumers and energy producers. Smart meters data is registered into a consortium blockchain and shared with DSO and energy suppliers to trace energy generation, having a more accurate billing [19].

The advent of Electric Vehicles (EVs) and e-mobility poses serious energy grid management challenges. In this context, blockchain technology brings several advantages, such as elimination of a centrally managed EV charging infrastructure, fault tolerance, elimination of price-setting, collusion between charging stations, or transport providers. The Ethereum-based platform Share&Charge [20], brings together electric car drivers and charging station owners and enables direct transactions between them. A mobile phone app enables electric car drivers to find the nearest charging station and to pay the owner of the charging station automatically with a digital currency based on a smart contract operated by a blockchain. The Share&Charge app was the first live blockchain-based mobility product in the world, allowing either the owners of charging stations or the drivers of electric vehicles to have an active share in the creation of the future of e-mobility. In a German pilot, the P2P app had over 1000 registered users and included approximately 1250 charging stations.

Most blockchain projects in the energy sector are still in the development phase, so the scale of their adoption is currently evolving, however, also according to senior consultancy and commercial reports by Deloitte [21] and PWC [22], blockchains have the potential of radically disrupting energy-related products and commodities. Therefore, these projects have the potential to radically change the established roles of energy companies, such as energy suppliers or grid operators, who are in most countries regulated monopolies and own the physical infrastructure. In addition, blockchain technology provides benefits to energy system operations, markets, and consumers. This technology offers disintermediation, transparency, reduction in transaction costs, increase in the efficiency of processes, but most of all, it offers novel solutions for authorizing consumers and small renewable generators to play a more active role in the energy market and monetise their assets.

2.4. Application of blockchain technology in BRIGHT project

The BRIGHT project will leverage on recent advancements on blockchain technologies, to deliver many-fold applications in order to support new community-enabled ways for engaging consumers in Demand Response. Specifically it will be provided four integrated software tools for the management of energy communities:

- Community level Blockchain based Flexibility Marketplace - application that allows community users to trade their energy flexibility in a P2P manner. Different types of flexibilities are considered such as electrical, thermal, comfort services, etc. The matching between the flexibility bids and offers is done using services integrated with the Blockchain via Oracles providing both cooperative and competitive (price driven) trading models for matching;
- Blockchain Management and Settlement of Flexibility driven DR - it allows the injection of energy goals in smart contracts, the tracking of flexibility delivery and energy and financial settlement. The implemented self-enforcing smart contracts will aim for tracking and checking the compliance of each prosumer enrolled in DR programs to the desired demand energy profiles, to calculate associated rewards and penalties, and to detect grid energy unbalances requiring the definition of new DR events;
- Community Self-governance to Deliver Flexibility Services Tool – application which allows the creation and decentralized management of coalitions in a community to deliver flexibility services on demand to the main grid. The smart contracts will be enabled to consider by means of direct injection the output of the hybrid optimization heuristics for cross sector combination of services. In this way, prosumers will be empowered to dynamically participate in coalitions by considering optimization targets energy profiles outputted by the optimization heuristics;
- Edge Metering Infrastructure and Interoperable Gateway – tool which allows monitoring and control of the home environment, integrating various home sensors, controllers, appliances belonging to different vendor ecosystems and making them interoperable. The tool will be integrated with smart contracts for flexibility actions automation.

3. Bright Scalable Distributed Energy Ledger

3.1. Overview of Design and Implementation

The distributed energy ledger developed in the BRIGHT project will handle energy and cross energy domain heterogeneous data with a view of providing the underlying infrastructural support for decentralized management of the energy community.

Blockchain can ensure trust between energy peers, secure data through cryptography and replication, ensure non-repudiation through immutable energy records and cryptography thus providing a secure distributed system of shared and replicated records that are tamper-proof and aggregated through consensus between all the nodes in the network. The increasing evolution of smart metering devices together with the prospect of renewable energy integration lead towards the adoption of decentralized energy networks where prosumers and energy assets can sell excess energy generation, or their energy flexibility being involved in a peer-to-peer sale/purchase process. More prosumers will be empowered to become active energy nodes of their communities increasing the amount of flexibility to be traded. In this scope, the blockchain technology can provide the required secure and reliable means for peer-to-peer trading and for managing the communication between community-level energy stakeholders.

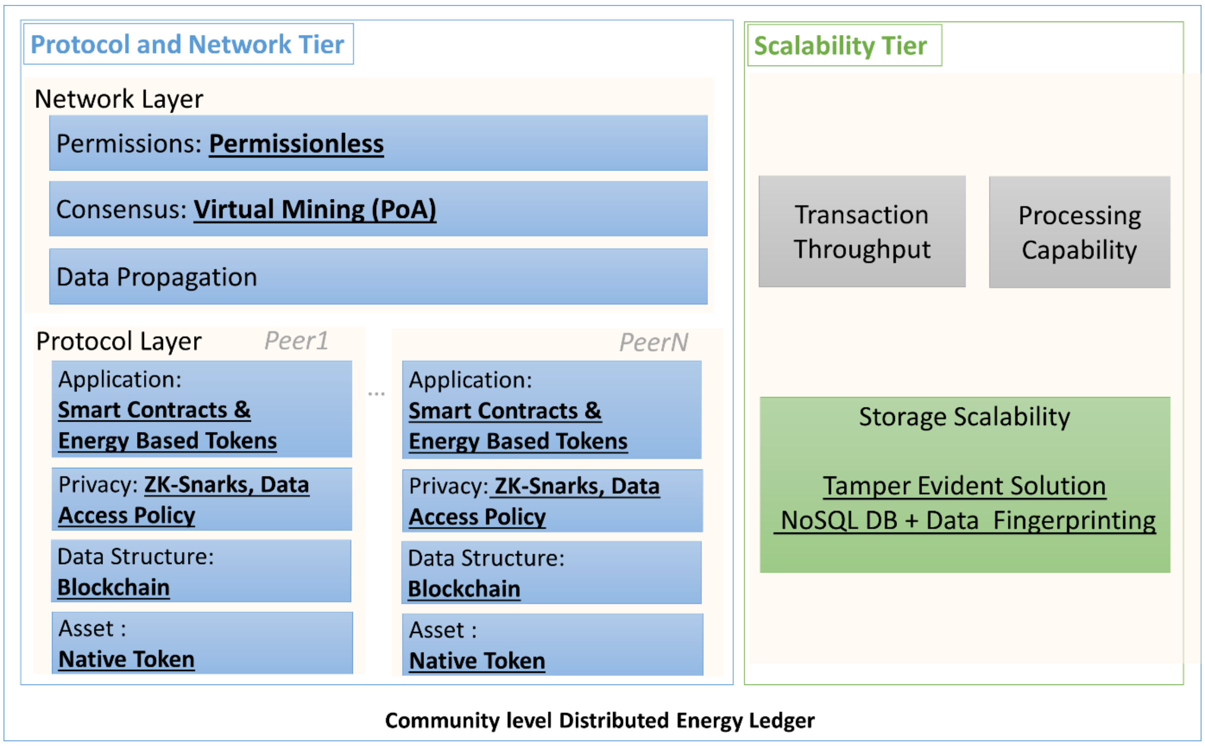

Protocol and Network Tier
Network Layer
Permissions: Permissionless
Consensus: Virtual Mining (PoA)
Data Propagation
Protocol Layer
Peer1
PeerN
Application: Smart Contracts & Energy Based Tokens
Privacy: ZK-Snarks, Data Access Policy
Data Structure: Blockchain
Asset : Native Token
...
Application: Smart Contracts & Energy Based Tokens
Privacy: ZK-Snarks, Data Access Policy
Data Structure: Blockchain
Asset : Native Token
Scalability Tier
Transaction Throughput
Processing Capability
Storage Scalability
Tamper Evident Solution
NoSQL DB + Data Fingerprinting
Community level Distributed Energy Ledger

Figure 4 Energy Ledger Architecture

The architecture of the distributed ledger for energy communities is presented in Figure 4. It is split into two tiers:

- Protocol and Network Tier (PN-tier) contains technological solutions for energy assets registration, energy transactions, data structure, and privacy and business rules implementation and the creation of peer-to-peer networks on top of the community physical level;

- Scalability Tier (S-tier) technological solutions that deal with the scalability of the blockchain in registering and managing the energy transactions inside the community.

3.1.1. Protocol and Network Tier

The PN-tier of the community level distributed energy ledger registers energy transactions generated on top of the monitored data of community members (also named prosumers). We have defined a blockchain-based solution in which the energy community members are registered as peer nodes of the network and their monitored energy values are stored as energy transactions (see Figure 5). The ledger stores in an immutable manner the energy transactions in blocks that are linked hashed back in the chain. Each prosumer is required to have an Ethereum node installed on-premises: either a full node deployed on a desktop computer or a light node deployed on a small single-board computer. The light nodes will only store headers of the blocks providing enough information to validate the consistency of the chain.

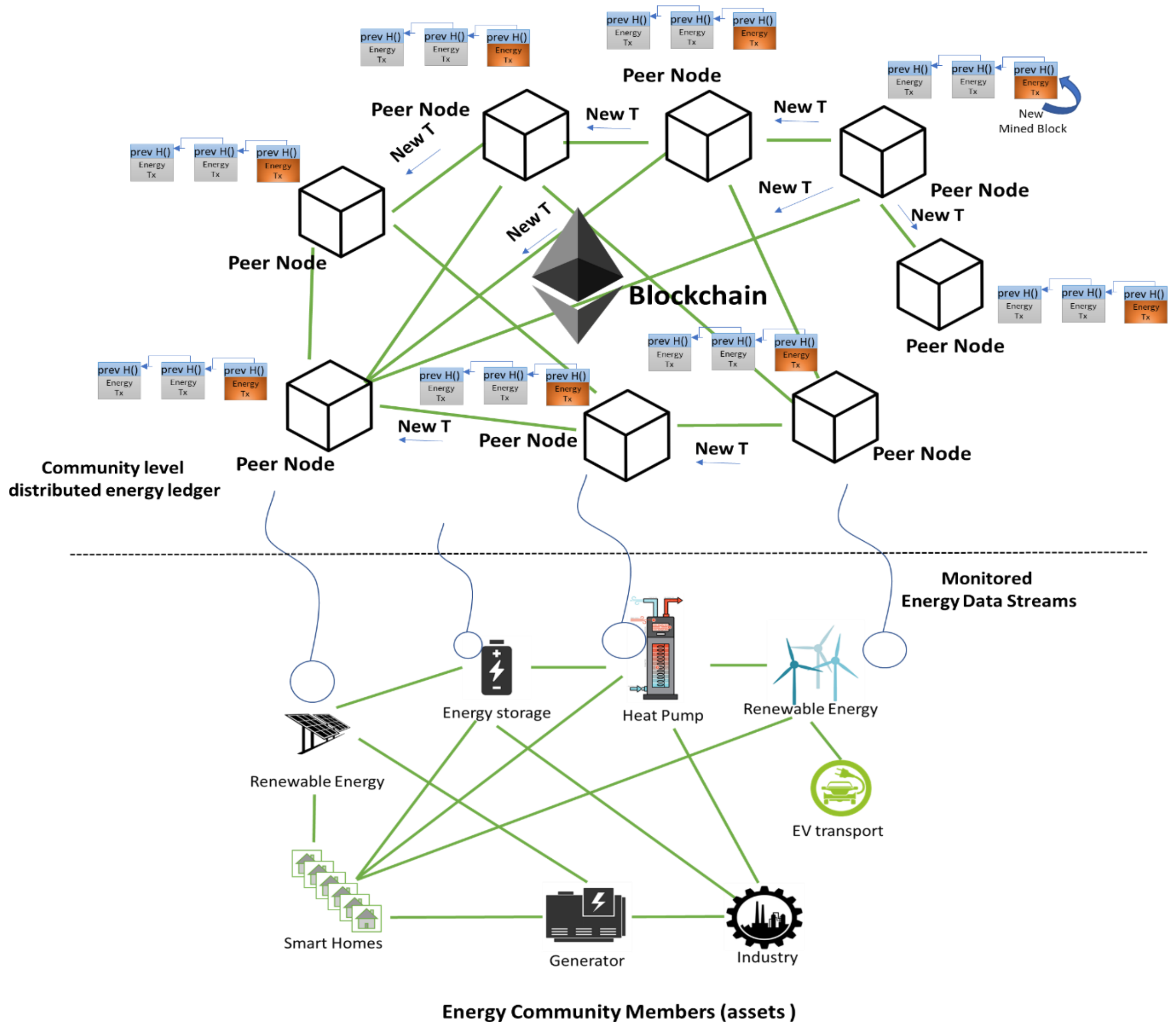


Figure 5 Energy Ledger and peer to peer network

Each community member has a smart contract that is associated with the smart meter and is used to manage in a decentralized manner its energy. The prosumer will register the energy transactions on the chain, by signing them and then broadcasting them across the entire network.

We consider that each smart energy meter associated with a prosumer collects monitored energy data at a finer grain t.; in an edge deployment case, the device running the Ethereum node will generate energy transactions by aggregating the monitored energy values over an interval T and signs the energy transaction with the private key. Consequently, the energy transaction will be signed with the prosumer's private key, while the recipient of the transaction is the smart contract associated with the smart energy meter.

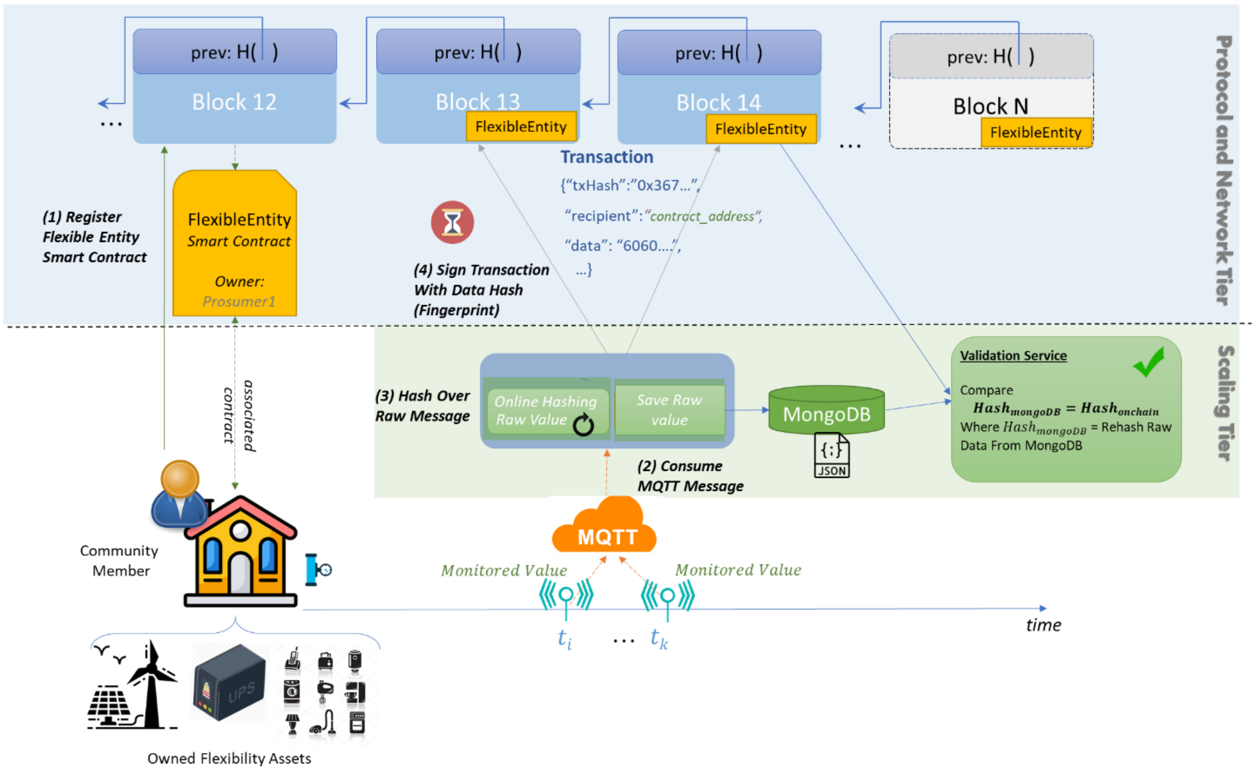


Figure 6 Energy Data Registration in the PN Tier

In more detail whenever a new community member joins the blockchain network as depicted in Figure 6 a new account is created by generating a pair of public-private keys-step. The public key is used to create the address that represents the community member into the distributed ledger network. For each of the monitored flexibility assets (or devices), an associated smart contract is created on-chain through a deploy transaction signed by the prosumer. Each contract will store immutable metadata about the device and mutable information regarding the device's monitored energy state. When the new aggregated monitored energy data is available to be registered on-chain, a transaction will be signed having as a payload the monitored data and as a receiver the address of the associated Flexible Entity Smart Contract. The energy transaction will be issued on-chain to be mined in the future blocks. To prove the ownership of data, the service provides the signature over the energy transaction, thus authenticating and validating the transaction. The monitored energy data provided in each transaction will update the state of the corresponding contract. The energy transaction and the associated state update will be stored in a block, benefiting from all the advantages brought by the technology in terms of availability, reliability, consensus, and immutability.

Considering that in the distributed energy ledger all transactions are duplicated and shared across the network peer nodes, it is imperative to provide solid ways for ensuring data and resources integrity. Each time a device is associated with a prosumer member of the community, and registered on-chain for energy data tracking, the prosumer will need to sign a transaction permitting the deployment of such a Flexible Entity Smart Contract. On deployment, the address of the user (generated based on the public key) will be immutably stored in the contract, thus becoming an Ownable contract where the owner is the user of the entity. From this point forward, any operation done upon the data stored in the contract will be authenticated and only updates triggered from transactions signed by the device's owner will be considered. Any attempt to trigger an update using an unauthorized transaction will lead to the transaction being dropped.

The private key of the user will be used to sign each energy transaction which will be addressable on the blockchain network only via the public key. Being based on mathematical functions that make it easy to compute the public keys, but infeasible to compute the private key given the public key, the cryptographic signature of transactions will ensure non-repudiation in the platform.

Although the transparency of a blockchain system is most of the time the desired feature, that brings openness and audit capabilities in rather centralized systems like the energy grid, in the energy domain, complete transparency over the monitored energy data is not desirable. The monitored devices can provide information that may allow inferring the inhabitant's behavior thus a privacy-preserving mechanism may be required for storing energy data. A zero-knowledge proof solution can be used to ensure data privacy in the distributed energy leger, while not interfering with network-wide validation of the energy transactions. The level of data disclosure can be set by each prosumer using Data Access Policies which may be integrated with the mechanism through which privacy is being assured in the ledger.

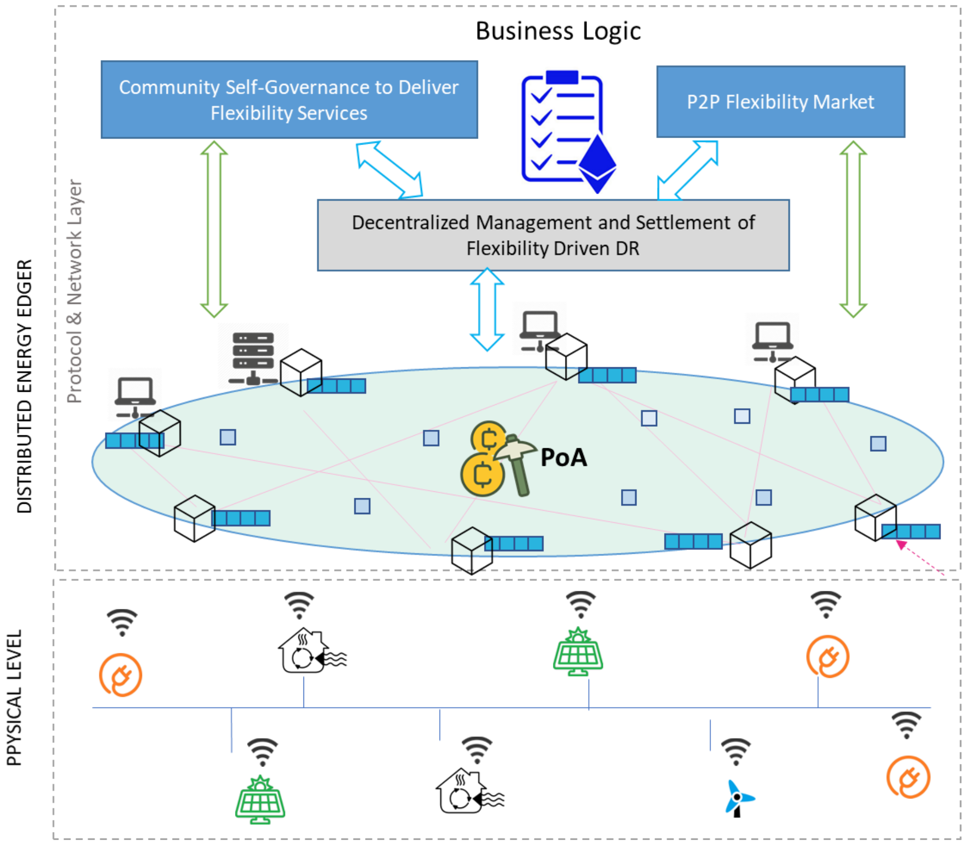

Business Logic
Community Self-Governance to Deliver Flexibility Services
P2P Flexibility Market
Decentralized Management and Settlement of Flexibility Driven DR
Protocol & Network Layer
DISTRIBUTED ENERGY EDGER
PoA
PPYSICAL LEVEL

Figure 7 Business rules implementation as smart contracts integrated with the energy ledger

The business rules are enforced on the energy chain using self-enforcing smart contracts to benefit from the distributed energy ledger network validation (see Figure 7). Any energy-related use case requires the monitored energy values to be used as input to implement the business logic such as data sharing and operational constraints; economical evaluation against different pre-arranged contracts (e.g., energy flexibility market); validation against a set of constraints (e.g., demand response settlement), aggregation of prosumers in Virtual Power Plants (e.g., virtual community self-governance), etc. Since the DLT can be used as support for both energy and economic settlement as well as for action enactment we provide the support for more complex business scenarios to be built on top of the monitoring layer provided by the smart contracts developed by BRIGHT solution. Smart contracts can be defined, each responsible to enforce business rules on the distributed energy ledger. Smart contracts are responsible for encoding the operational rules of the assets as well as local constraints for safe operation and data sharing. Community-level smart contracts are responsible for capturing and enforcing the rules for grid safe operation, community-level energy services delivery, and community sustainability goals.

Each node of the blockchain network keeps a copy of the community-level energy ledger which will be automatically updated when new blocks will be mined. Whenever a new community member connects to the network, the corresponding node will be connected to a predefined list of seed nodes. The seed nodes will provide the newly joined node with information about all the peer nodes they know about, the process being repeated with the newly discovered peers until the new node builds its list of peers. The process of peer discovery and association is performed regardless of the joined peers' location providing a completely decentralized geo-distribution making it suitable for modeling both bounded energy communities as well as virtual energy communities or communities on the move. The location information if needed can be associated with the *Flexible Entity* contract upon deployment. The access is considered public, so any new node can register, transact, and commit blocks to the chain, however, a higher-level control is enforced using smart contracts, by keeping registries of whitelisted community members that can participate in different business scenarios.
The proposed PN-Tier techniques for distributed energy ledger can be implemented using public chains. In terms of consensus algorithms, to avoid the high energy requirement of mining, a virtual mining algorithm (Proof of Authority [23]) is considered, which can ensure up to a certain level Byzantine Fault Tolerance, which is highly required in a network where a large number of devices, even malicious ones can register.

3.1.2. Scalability Tier

Due to the high costs associated with the on-chain data processing, it is not efficient (in terms of cost and scalability) to register a transaction each time new data is sampled by the smart metering devices. Thus, an efficient, cost-effective, and scalable S-tier approach has been developed, allowing for monitoring energy transactions to be less frequently registered on the chain (e.g., one transaction per hour) while not losing any of the benefits brought by the blockchain technology.

The energy data gathered from energy meters is not registered in the distributed energy ledger in the same form and rate that it was monitored, but a pre-processing step may be applied to aggregate and compress the data in a meaningful way for future processing at the business logic layers. Consequently, our solution to the storage and transaction scalability problems is a tamper-evident double-layered compressive system, where:

- the PN-Tier is ensuring the registration of energy transactions at a higher granularity and the execution of business rules on chain thus avoiding the congestion due to the high number of fine-grained energy transactions that cannot be sealed in blocks according to the rate and limitation set by the network, and
- the S-Tier is ensuring the data storage at a higher throughput and finer granularity of raw monitored energy values linking and hashing them prior to the registration in the PN-Tier.

The business rules implementation using self-enforcing smart contracts ensures the logic implementation for various community level scenarios as well as their energy and financial settlement does not require a higher transactions registration rate than once per hour.

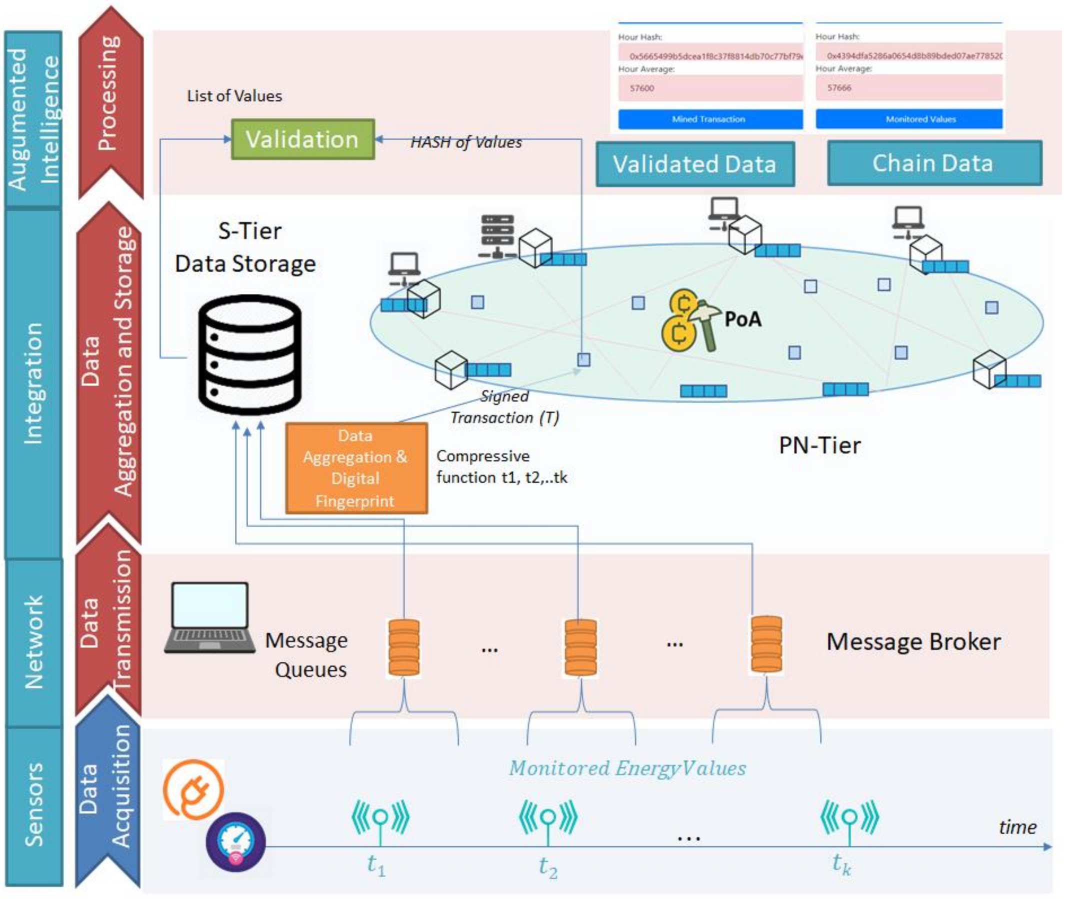


Figure 8 Scalability tier of the distributed energy ledger

As depicted in Figure 8 once monitored the energy values are received, they are stored in a NoSQL database [24] and passed through an aggregation function. The compressive function is used to aggregate the monitored values, link hash-back them and send energy transactions to the PN-Tier for storage and business evaluation. One major benefit brought over the classical architecture is the tamper-evident characteristic of the obtained system. The values registered on-chain can be used to ensure that no successful attack has been registered on the S-Tier solution, by validating the value registered off-chain with the ones registered on-chain.

As a result, the BRIGHT community level distributed energy level offers a hybrid solution consisting of (i) storing off-chain, the raw monitored energy in a database, the real-time data as collected from the metering devices while (ii) all the collected values are hashed-linked back the raw data and periodically stored on-chain energy transactions for business enforcement and (iii) validation services are built upon the off-chain stored data resulting in a tamper-evident solution.

3.2. Digital Fingerprinting and Energy Transactions

A digital fingerprint is a unique digital identifier that contains a set of data used to uniquely identify the monitored energy values sampled by an energy meter over an hour and used to generate the energy transaction which will be registered and shared on the blockchain. The digital fingerprint is generated using hashing algorithms and it is used for allowing to store in a tamper-evident manner the monitored energy values off-chain and at the same time to allow their validation at later stages.

All monitored energy values from sensors are collected from MQTT [25] queues and saved in raw format in a non-relational MongoDB [26]. MongoDB stores data in JSON [27] allowing for more flexibility in managing heterogenous data streams allowing the integration of other types of energy data besides electricity and metadata describing social of physical characteristics of the assets.

The fields from the JSON object structure retrieved from the queue may vary depending on the sensor and the type of monitoring. Besides inserting the raw monitoring received from each energy meter into the MongoDB, monitored values are aggregated at a time granularity of one hour, and a digital fingerprint is computed for that hour that ensures the tamper-evident registration of data. At the same time, an associated energy transaction for that hour (i.e., an average of the monitored energy values) is registered and shared in an immutable and tamper-proof manner on the blockchain. The digital fingerprint is obtained using a hashing algorithm over all information stored in the JSON object sampled and stored in MongoDB. Thus, the MongoDB database stores all monitored data in a raw format, while on-chain the aggregated monitored values are registered through a blockchain energy transaction at every hour, together with a digital fingerprint that assures the validation of the information stored in the database.

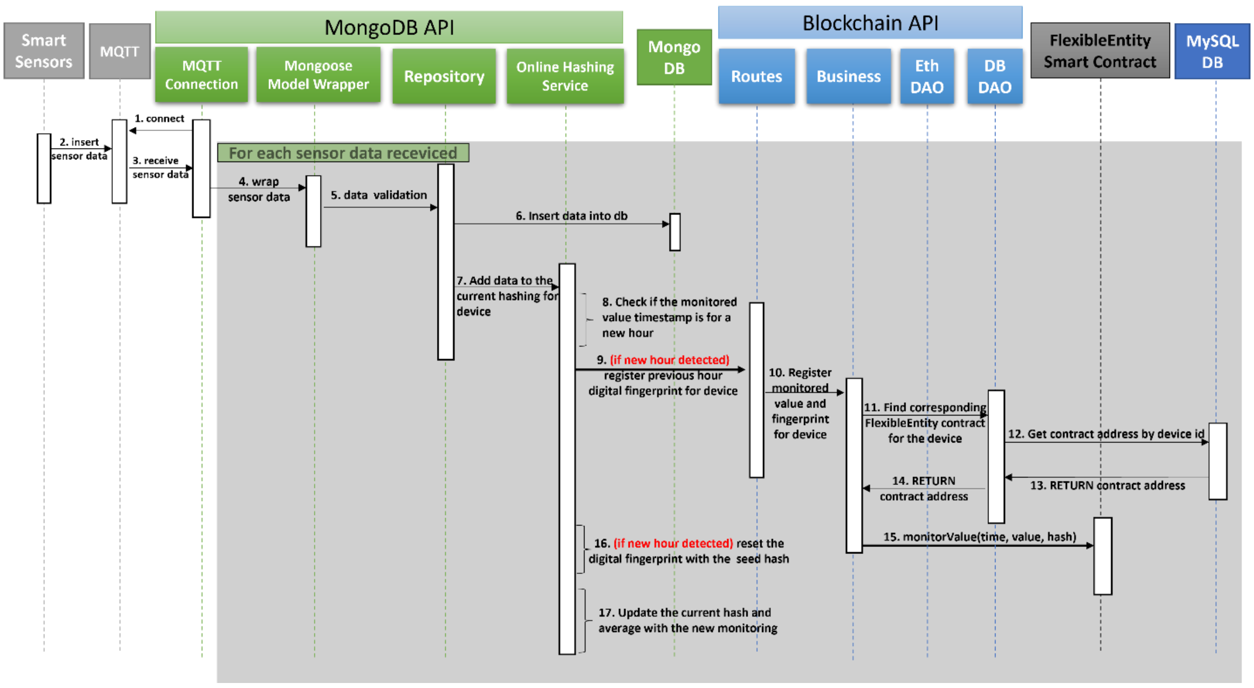

Smart Sensors
MQTT
MongoDB API
MQTT Connection
Mongoose Model Wrapper
Repository
Online Hashing Service
Mongo DB
Blockchain API
Routes
Business
Eth DAO
DB DAO
FlexibleEntity Smart Contract
MySQL DB
1. connect
2. insert sensor data
3. receive sensor data
For each sensor data receviced
4. wrap sensor data
5. data validation
6. Insert data into db
7. Add data to the current hashing for device
8. Check if the monitored value timestamp is for a new hour
9. (if new hour detected) register previous hour digital fingerprint for device
10. Register monitored value and fingerprint for device
11. Find corresponding FlexibleEntity contract for the device
12. Get contract address by device id
13. RETURN contract address
14. RETURN contract address
15. monitorValue(time, value, hash)
16. (if new hour detected) reset the digital fingerprint with the seed hash
17. Update the current hash and average with the new monitoring

Figure 9 Flow of data from energy meters to energy transactions registration in the distributed ledger

The data flow from the sensors to the MongoDB and blockchain can be seen in Figure 9. The MongoDB API module is connected to the MQTT queues to receive data from sensors (steps 1-3). For each message received, which corresponds to a monitored energy value the data is wrapped in a Mongoose object, validated, and then inserted into the database (steps 4-6). Then, a hashing algorithm (step 7) is used on raw monitored energy values, which are also aggregated on an hourly basis and the obtained digital fingerprint is saved on the blockchain. The process repeats until the new considered monitored energy value has a timestamp that indicates a new hour (step 8).

The digital fingerprint and the aggregated monitored value for the previous hour must be registered on-chain. The monitored data is stored on the blockchain through a call to the Blockchain API which finds the corresponding flexible entity contract for the energy asset (see next section) and makes an energy transaction call to register the data and the digital fingerprint (steps 9-15). The current hash is reset to the seed hash used at the beginning of each hour (step 16). The new monitored value sampled is used to update the current hourly hash (using the hash function) and is aggregated to the average monitored value for that hour (step 17).

The digital fingerprinting algorithm (see Figure 10) computes a hash from JSON objects stored in the MongoDB database objects which may contain monitored energy values as well as metadata representing physical or social characteristics of the energy assets. At the same time, the monitored energy values sampled during an hour are aggregated and averaged and an energy transaction is generated and stored in the blockchain together with the generated hash. Inputs of the algorithm are the JSON files stored in MongoDB containing information associated with a flexibility entity that belongs to a community member, a seed hash that is uniquely generated for each flexibility entity, providing the genesis hash value for the linked back data structure. The outputs of the digital fingerprinting algorithm are the hash generated for all the JSON files of a flexible entity during an hour, the average energy data, and the associated energy transaction.

```
Input: $t_{init}$ - initial timestamp, $seed$ - seed hash, $sensors$ - list with all energy metering sensors each
        sensor being associated with a flexible entity
Output: $h_{map}$ $< sensor, hash >$ – map between energy sensors and the hash value of the monitored
         energy data; $avg_{map}$ $< sensor, average >$ – map between sensors and averaged
         monitored energy values over an hour, $latest_{map}$ $< sensor, timestamp >$ – map between
         sensors and the timestamp of the latest monitored energy value
Begin
   1.  foreach sensor in $sensors$ do
   2.    $h_{map}$[sensor] ← seed
   3.    $avg_{map}$[sensor] ← 0
   4.    $latest_{map}$[sensor] ← $T_{init}$
   5.    count [sensor] ← 0
   6.    foreach new monitored $EnergyData$ sampled do
   7.      sensor ← getSensor ($EnergyData$)
   8.      if (getTimestamp ($EnergyData$) is newHour ($latest_{map}$[sensor]) then
   9.        $avg_{map}$[sensor] ← $avg_{map}$[sensor] / $count$[sensor]
   10.       registerNewTransaction(sensor, $h_{map}$[sensor], $avg_{map}$[sensor])
   11.       $h_{map}$[sensor] ← seed
   12.       $avg_{map}$[sensor] ← 0
   13.       count[sensor] ← 0
```

```
14.     else
15.        $h_{crt}$ ← HASH (getBytes($EnergyData$))
16.        update $h_{map}$[*sensor*] with $h_{crt}$
17.        $avg_{map}$[*sensor*] ← $avg_{map}$[*sensor*] + getMonitoredValue($EnergyData$)
18.        $latest_{map}$[*sensor*] ← getTimestamp($EnergyData$)
19.        *count*[*sensor*]++
20.      end if
21.    end foreach
22. end foreach
End
```

Figure 10 Digital fingerprinting algorithm

The first step of the algorithm is to initialise the hash of the period with the value of the seed for the specific energy sensors and also the other variables used in the computation. For each new energy monitored data at timestamp t. In the hourly interval (line 6), the hash will be computed (lines 15 and 16) by hashing the current energy value with the timestamp and storing it into the Hash Map in the corresponding position (sensor and next to the previous monitored energy data hash). At the same time, the sum of energy is calculated (line 17) so that at the end of the hour the average energy could be determined. At the end of the hour (see line 8) the average energy will be used to generate the energy transaction to be registered on blockchain while the hourly digital fingerprint hash will be hashed-linked back with all the digital fingerprints of previous hours (line 10).

The final hash obtained at the end of the interval of an hour is a digital fingerprint of all the values. The hashed-linked structure is chosen to accommodate the seriality of the data received from the sensors as well as any fluctuations in the sample rate (see Figure 11). Since at timestamp the only value required is the hash of the previous timestamp associated value the computational processing overhead is low. All the energy values sampled by the energy meters associated with flexibility entities are stored in the MongoDB database which can easily handle large amounts of document-oriented data. In this case, the immutability is assured by the off-chain linked hashing structure which will not allow the tampering of the MongoDB data. In the blockchain, we will save only the generated hash and energy transaction determined as an average of the monitored values during the interval thus in this case the immutability will be assured by the blockchain.

Another advantage of the proposed fingerprinting method is related to the scalability of the community-level distributed energy ledger when it comes to the transactional throughput. The scalability is increased by registering energy transactions rarer on the chain at the same time assuring the energy data immutability, traceability, and auditability. On the on-chain side, since during the entire period, there will be digital fingerprints specific for each hour, the storage space would increase proportionally with this value. Nevertheless, the number of data sampled in an hour by sensors is bounded by being successfully managed by MongoDB. For each hour one fixed-length hash will depict the digital fingerprint of energy values acquired and one energy transaction will be registered on the blockchain. Each hash corresponding to a time interval is mined in a block as depicted in Figure 11 and validated by the network nodes rendering it immutable once enough blocks are mined on top of it.

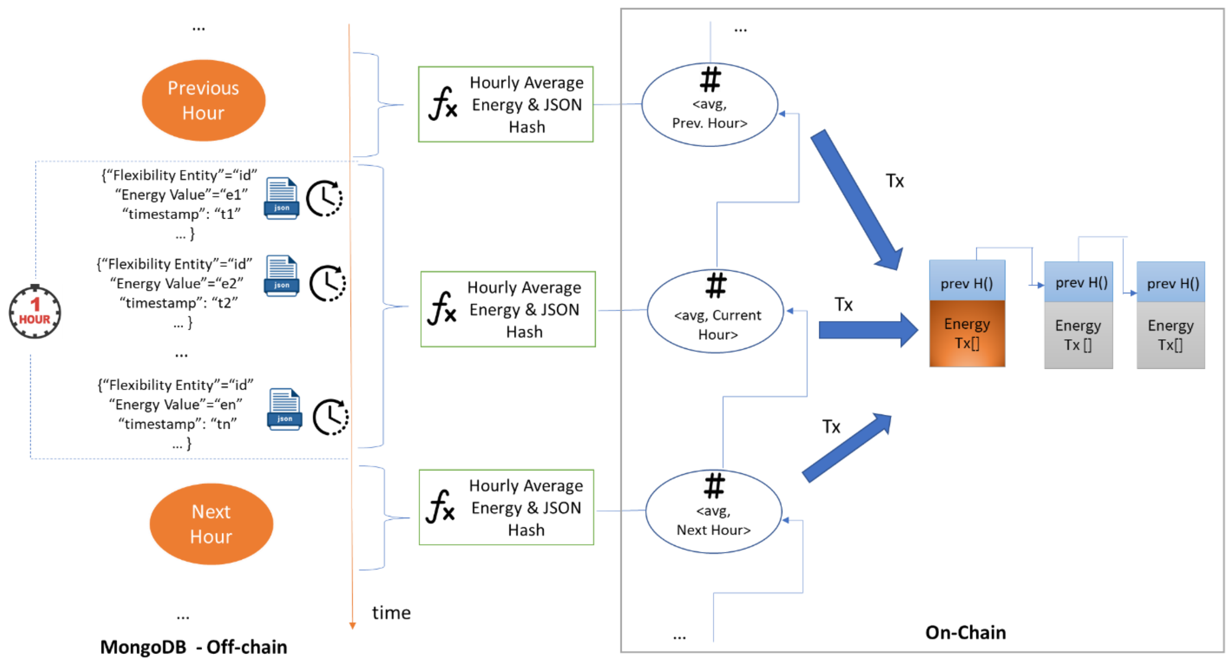

Previous Hour
{"Flexibility Entity"="id" "Energy Value"="e1" "timestamp": "t1" ... }
{"Flexibility Entity"="id" "Energy Value"="e2" "timestamp": "t2" ... }
{"Flexibility Entity"="id" "Energy Value"="en" "timestamp": "tn" ... }
1 HOUR
Next Hour
MongoDB - Off-chain
time
Hourly Average Energy & JSON Hash
<avg, Prev. Hour>
<avg, Current Hour>
<avg, Next Hour>
Tx
prev H()
Energy Tx[]
On-Chain

Figure 11 Generation and linking transactions on blockchain

Finally, the proposed distributed energy ledger allows for different types of assets to be monitored (thermal, gas, electrical, etc.) and their information integrated on the blockchain. It offers generic methods for integrating these monitored values by converting them to energy data (if this information is not available) off-chain and generating the associated energy transactions to be stored on the blockchain.

When historical monitored energy data values need to be retrieved for a specific hour all the off-chain stored energy values are validated against the data fingerprints registered on the blockchain (see Figure 12). All the values that are fetched from MongoDB are digitally fingerprinted again and the resulted hash is compared with the one already stored on the blockchain (lines 5-10). If the data stored off-chain was modified the obtained hash value will not be equal with the one stored in blockchain (lines 11-13). Any changes of monitored energy values or the associated timestamps will be detected since any modification will render a completely different hash of the data. In case an inequality between the hashes is detected, further inquiries are made to detect the exact interval or values that have been tampered with. A high sampling rate is beneficial to narrow as much as possible the subset of monitored energy data that contains the unreliable values, but at the same time could affect the ledger's scalability since the blockchain can handle only a limited throughput.

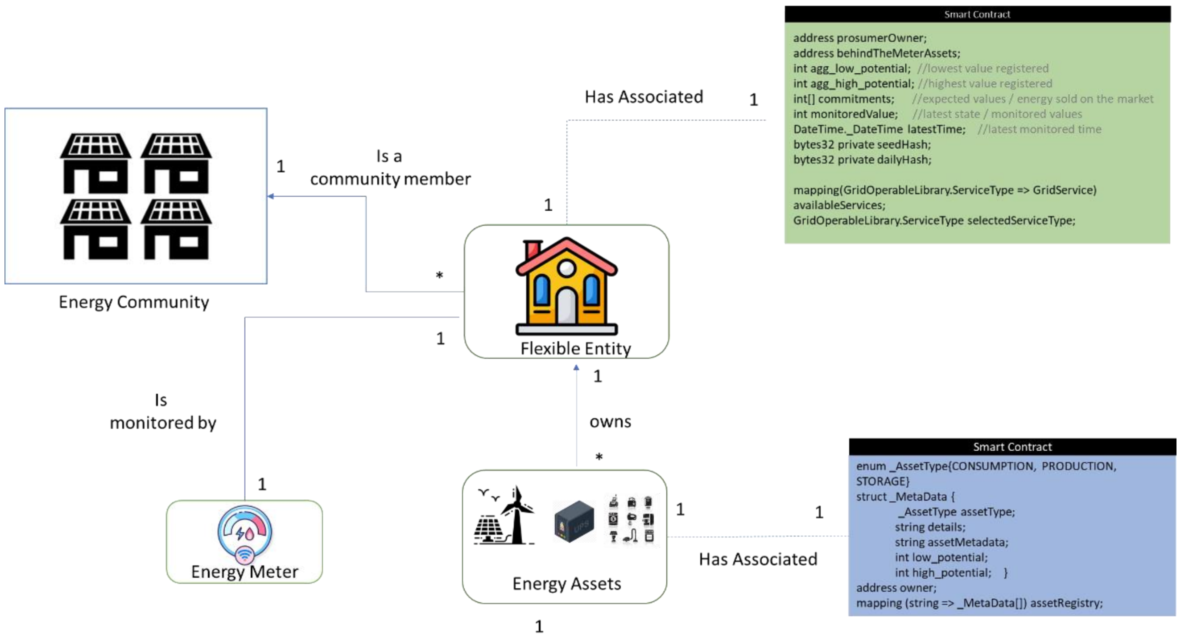
```
Input: $timestamp$ - the hour to be validated, $sensor$ - energy metering sensor associated with a flexible
       entity for which the validation will be conducted, $seed$ - seed hash
Output: $True$ – the monitored energy data is valid and was not tampered; $False$ – the monitored energy
        data is valid, and that data was modified
Begin
  1.  monitoredEnergyData ← getMonitoredValuesForHour (sensor, timestamp)
  2.  $offchain_{hash}$ ← seed
  3.  $offchain_{avg}$ ← 0
  4.  foreach m in monitoredEnergyData do
```

```
5.      $offchain_{avg} \leftarrow offchain_{avg}$ + getEnergyValue($m$)
6.      $h_{crt} \leftarrow$ HASH (getBytes(m))
7.      update $offchain_{hash}$ with $h_{crt}$
8.    **end foreach**
9.    $average_h \leftarrow offchain_{avg} / \mathbf{length}(monitorings)$
10.   $< chain_{hash}, chain_{avg} > \leftarrow$ **getBlockchainMonitoring** (sensor, timestamp)
11.   **if** ( $chain_{hash} = offchain_{hash}$) and ($chain_{avg} = offchain_{avg}$) **then**
12.     **Return** *True*
13.   **else**
14.     **Return** *False*
**End**
```

Figure 12 Validating the data stored off-chain in MongoDB

3.3. Smart Contracts for Registration and Data Sharing

Two categories of smart contracts are used to store and share energy assets data and metadata on-chain. A smart contract is associated with a community-level flexible entity which is monitored by an energy meter. It manages associated monitored energy data, regardless of the asset type. Each flexible entity can have its configuration of energy assets (see Figure 13) and information characters that may need to be stored on-chain. The data sharing of specific energy asset characteristics and metadata is being managed by an Energy Asset smart contract.



```
address prosumerOwner;
address behindTheMeterAssets;
int agg_low_potential;  //lowest value registered
int agg_high_potential; //highest value registered
int[] commitments;      //expected values / energy sold on the market
int monitoredValue;     //latest state / monitored values
DateTime._DateTime latestTime;   //latest monitored time
bytes32 private seedHash;
bytes32 private dailyHash;

mapping(GridOperableLibrary.ServiceType => GridService)
availableServices;
GridOperableLibrary.ServiceType selectedServiceType;
```

```
enum _AssetType{CONSUMPTION, PRODUCTION,
STORAGE}
struct _MetaData {
        _AssetType assetType;
        string details;
        string assetMetadata;
        int low_potential;
        int high_potential;   }
address owner;
mapping (string => _MetaData[]) assetRegistry;
```

Figure 13 Flexible Entity and Energy Assets smart contracts

The state variable of the smart contract describing a *Flexible Entity* is represented in Table 2.

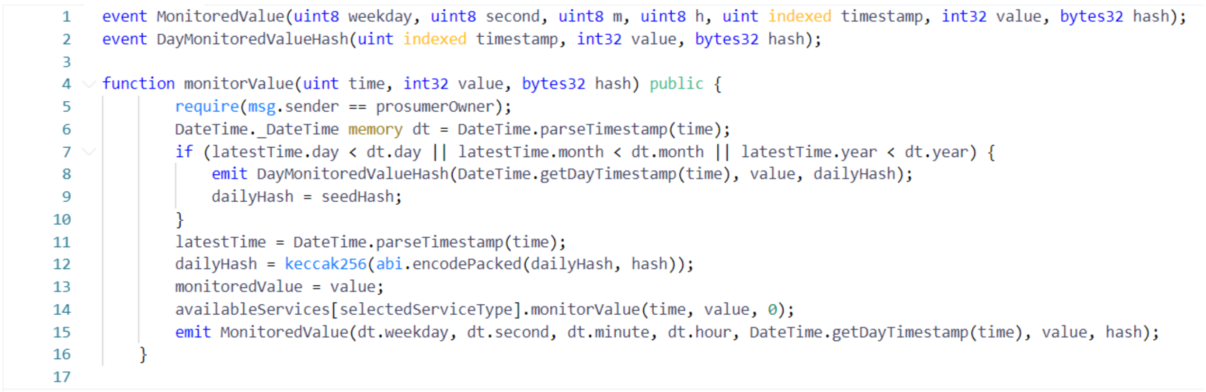
Table 2 State variables of the Flexible Entity smart contract

State Variable	Description
Owner	Community member account address who owns the flexible entity
Behind the meter asset	A contract address which stores information about the characteristics of the asset that is behind the smart meter

Commitments	Commitments made on the business logic level (e.g., energy flexibility to be traded on the flexibility market or committed in DR program)
Monitored Value	Mutable state representing the latest monitored energy value
Latest timestamp	Latest monitoring timestamp
Flexibility potentials	Potential limits (high and low potential) of the asset
Seed and Daily Hash	The seed hash is used to initialize the daily hash which stores the hashing of all the current day monitored values; at the end of the day the daily hash is saved through an event and the daily hash is reset

When an energy transaction for the flexible entity is sealed containing the time, value, and hash of a new monitored energy value, the smart contract will check the address of the message sender (see Figure 14). An energy transaction can be registered on-chain only by the community member which is the owner of the flexible entity using hourly aggregated energy data (see line 5). The next step is to compare the latest timestamp of the monitored energy value of the flexible entity with the timestamp of the new monitoring (see lines 6, 7). If the new timestamp is registered in a new hour the hash of the previous hour is registered and the hourly hash is initialized with the seed hash. Similarly, we manage the daily hash by emitting a MonitoredValueHash event and the mutable state daily hash is initialized with the seed hash.

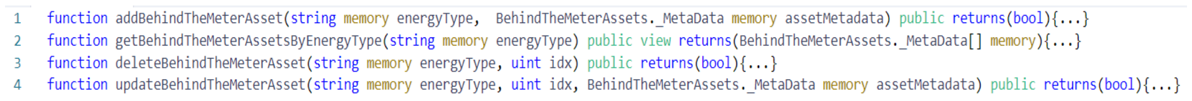
The latest time is set with the timestamp of the new monitoring (see line 11), the hourly/daily hash is updated with the hash obtained from the current monitored data (see line 12) and the value is stored in the MonitoredValue state (see line 13). The monitored energy value is also registered for the services in which the entity is active (see line 14). A history of all entity monitored values and their hash is kept off-chain for validation purposes using the MonitoredValue event (see line 15).



```
1   event MonitoredValue(uint8 weekday, uint8 second, uint8 m, uint8 h, uint indexed timestamp, int32 value, bytes32 hash);
2   event DayMonitoredValueHash(uint indexed timestamp, int32 value, bytes32 hash);
3
4   function monitorValue(uint time, int32 value, bytes32 hash) public {
5           require(msg.sender == prosumerOwner);
6           DateTime._DateTime memory dt = DateTime.parseTimestamp(time);
7           if (latestTime.day < dt.day || latestTime.month < dt.month || latestTime.year < dt.year) {
8               emit DayMonitoredValueHash(DateTime.getDayTimestamp(time), value, dailyHash);
9               dailyHash = seedHash;
10          }
11          latestTime = DateTime.parseTimestamp(time);
12          dailyHash = keccak256(abi.encodePacked(dailyHash, hash));
13          monitoredValue = value;
14          availableServices[selectedServiceType].monitorValue(time, value, 0);
15          emit MonitoredValue(dt.weekday, dt.second, dt.minute, dt.hour, DateTime.getDayTimestamp(time), value, hash);
16      }
17
```

Figure 14 FlexibleEntity smart contract method for energy data sharing

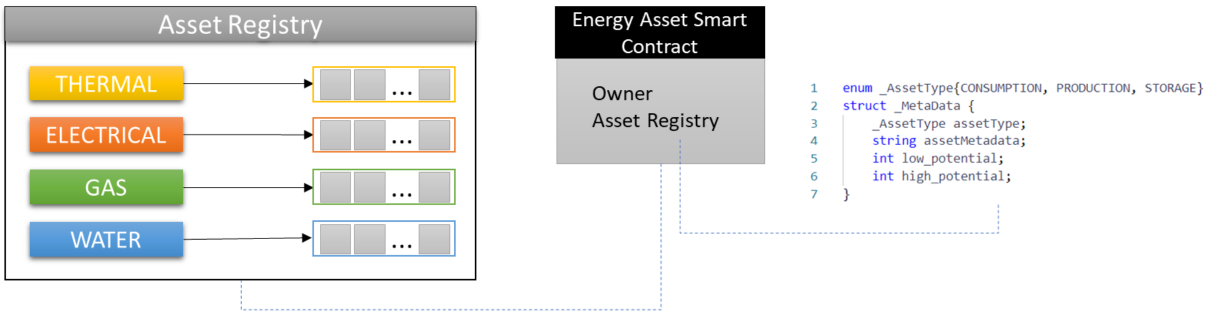
The *FlexibleEntity* contract has methods for the management of the associated energy assets (Figure 15). It allows for adding new assets and enables the visualization of the assets belonging to the entity by specifying the energy type, as well as deleting and updating information on existing assets. Thus, the address of the flexible entity contract has the right to add, delete and update assets from its registered *Energy Assets smart* contract by calling its methods.



```
function addBehindTheMeterAsset(string memory energyType,  BehindTheMeterAssets._MetaData memory assetMetadata) public returns(bool){...}
function getBehindTheMeterAssetsByEnergyType(string memory energyType) public view returns(BehindTheMeterAssets._MetaData[] memory){...}
function deleteBehindTheMeterAsset(string memory energyType, uint idx) public returns(bool){...}
function updateBehindTheMeterAsset(string memory energyType, uint idx, BehindTheMeterAssets._MetaData memory assetMetadata) public returns(bool){...}
```

Figure 15 Smart contract methods for energy assets management

Assets are organized in the registry by energy type (thermal, electrical, gas, water, etc.), and the type according to their role (consumption, production, or storage) is stored as information together with potential limits and metadata relevant for the prosumer (see Figure 16). The potential flexibility limits (high and low potential) provide additional information on the assets useful for business logic implementation.



```
enum _AssetType{CONSUMPTION, PRODUCTION, STORAGE}
struct _MetaData {
    _AssetType assetType;
    string assetMetadata;
    int low_potential;
    int high_potential;
}
```

Figure 16 Energy Assets Registry

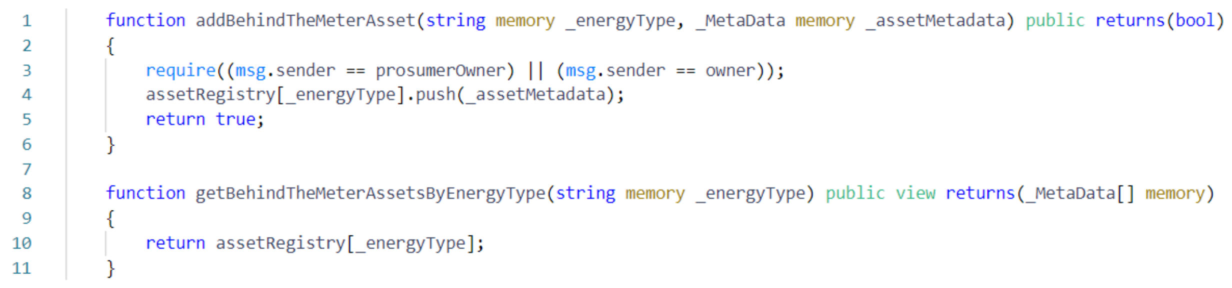
Table 3 presents the state variables defined by the energy assets smart contract. An asset is represented in the *smart* contract using a data structure and an enumeration for the asset types (see Figure 16). The data structure *_Metadata* stores information about the asset type (consumption, production, or storage), a string for metadata, and two integer values for the low and high potential of the asset.

Table 3 Energy Assets smart contract state variables

State Variable	Description
Owner	The corresponding flexible entity contract and the community member owner account
Asset Registry	Mapping that links the energy type (**_AssetType**) key with an array of assets represented using the ***_MetaData*** structure

In the constructor of the energy asset smart contract, the flexible entity address is set with the message sender address which is the address of the flexibility entity contract containing the asset. Also, the prosumer owner is set with the address received as the *_prosumerOwner* parameter which represents the community member who owns the flexible entity.

The implementation of the energy assets management operations is presented in Figure 17 and Figure 18.



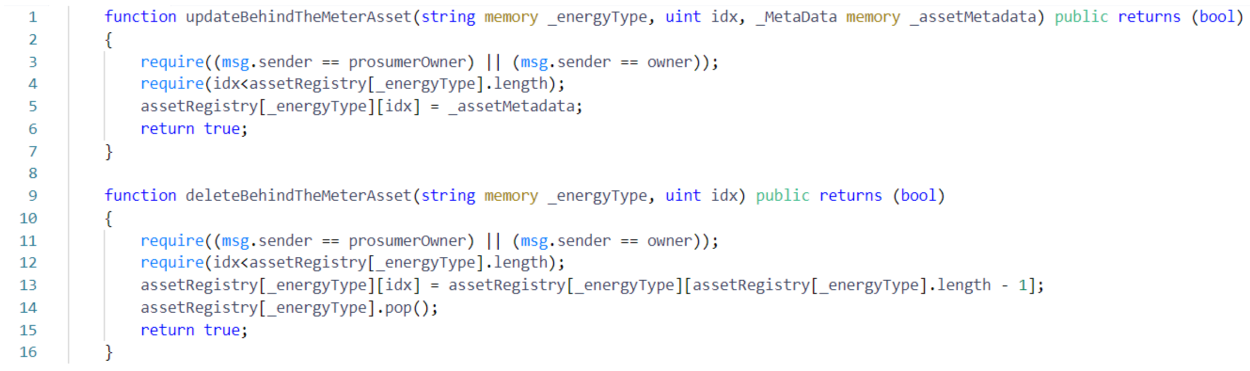
```
1   function addBehindTheMeterAsset(string memory _energyType, _MetaData memory _assetMetadata) public returns(bool)
2   {
3       require((msg.sender == prosumerOwner) || (msg.sender == owner));
4       assetRegistry[_energyType].push(_assetMetadata);
5       return true;
6   }
7
8   function getBehindTheMeterAssetsByEnergyType(string memory _energyType) public view returns(_MetaData[] memory)
9   {
10      return assetRegistry[_energyType];
11  }
```

Figure 17 Methods for viewing and adding a new energy asset

The operation of adding an asset is only allowed for the flexible entity or for the community member which owns the asset (see Figure 17 line 3). The add method inserts the asset at the end of the array that corresponds to the energy type and associates it with the corresponding owner.

To view information of energy assets they are selected from the mapping using the energy type, and the get method returns an array of *_Metadata* elements which have the indicated energy type (see Figure 17 line 10).

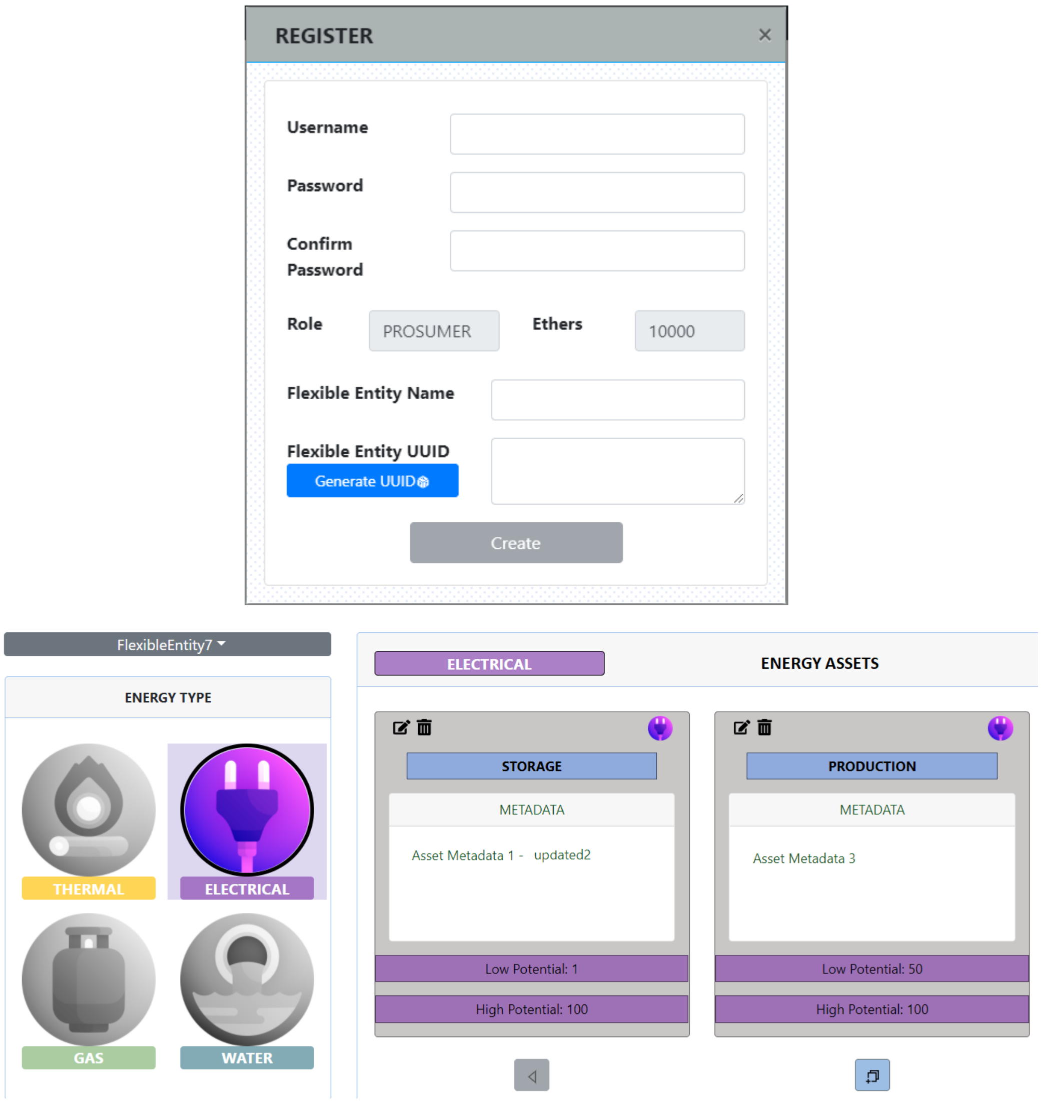
To update or delete an asset, its energy type and index must be provided (see Figure 18). These two operations are also permitted only to the owners (see lines 3 and 11) and the index must be valid (see lines 4 and 12). For the delete operation, the asset indicated by the index is replaced by the last asset in the array and the duplicated element from the last position of the vector is deleted using a pop operation (lines 13 and 14).



```
1   function updateBehindTheMeterAsset(string memory _energyType, uint idx, _MetaData memory _assetMetadata) public returns (bool)
2   {
3       require((msg.sender == prosumerOwner) || (msg.sender == owner));
4       require(idx<assetRegistry[_energyType].length);
5       assetRegistry[_energyType][idx] = _assetMetadata;
6       return true;
7   }
8
9   function deleteBehindTheMeterAsset(string memory _energyType, uint idx) public returns (bool)
10  {
11      require((msg.sender == prosumerOwner) || (msg.sender == owner));
12      require(idx<assetRegistry[_energyType].length);
13      assetRegistry[_energyType][idx] = assetRegistry[_energyType][assetRegistry[_energyType].length - 1];
14      assetRegistry[_energyType].pop();
15      return true;
16  }
```

Figure 18 Methods to update and delete an energy asset

Figure 19 shows the front-end interface used to manage the registration and participation of community members with their owned energy assets to the community-level distributed energy ledger. The entire process is being managed by the above-described smart contracts enabling the sharing of energy data and metadata on the chain.



REGISTER ×

Username			
Password			
Confirm Password			
Role	PROSUMER	Ethers	10000
Flexible Entity Name			
Flexible Entity UUID	Generate UUID		

Create

FlexibleEntity7

ENERGY TYPE

THERMAL
ELECTRICAL
GAS
WATER

ELECTRICAL

ENERGY ASSETS

STORAGE	PRODUCTION
METADATA	METADATA
Asset Metadata 1 - updated2	Asset Metadata 3
Low Potential: 1	Low Potential: 50
High Potential: 100	High Potential: 100

Figure 19 Flexible entities and energy assets registration and management front end

4. BRIGHT Data access policies

Data access control is a mechanism in computer security that regulates access to the system resources. The rights of subjects to access such resources are typically expressed through access control policies, which are evaluated at access request time against the current access context.

Among traditional approaches, the Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) approach is mostly used in modern IT organizations. RBAC is a security and authorization model for securing access to computer resources. Sometimes referred to as “Role-Based Security,” RBAC access is based on roles as defined by the business using them. In the RBAC model,roles are created and then sets of permissions for resources are assigned to the role. Users are then granted one or more roles to receive access to resources. ABAC, on the other hand, stands for Attribute-Based Access Control. As suggested by the name, ABAC relies on user attributes for authorization decisions. ABAC policies are rules that evaluate access based upon four sets of attributes. These include: Subject Attributes, which are attributes concerning the person or actor being evaluated; Resource Attributes, which are attributes of the target or object being affected; Action Attributes, which describe the action to be performed onthe Resource; and, Environment Attributes, which include attributes such as the time of the day, IP subnet, and others that do not relate to either the Subject or the Resource.

Each of these models has its weakness and benefits. The main benefits of RBAC are:

- RBAC is deterministic. An RBAC approach makes it easy to know who has access to what at any moment in time;
- RBAC is more direct and easier to visualize. Security admins can visualize the actors and resources they will affect when creating or modifying a policy;
- RBAC is inherently auditable. With RBAC assignments it is simple for business owners to certify or attest to access granted, as the consequences of that accessarevisible. This visibility contrasts with ABAC where a “before the fact audit" is not possible and the effects of a rule are not easy to grasp;
- RBAC can be simpler than ABAC. For example, with RBAC, bundles of access can be directly assigned to a user. To do this in ABAC requires the creation of a new rule.

RBAC’s primary weaknesses are:

- RBAC requires advance knowledge of the Subjects and Resources and typically does not support making on-the-fly contextual decisions;
- An RBAC-only approach can result in a huge number of roles to accomplish fine-grained authorization;
- Resource Owners must know something about the roles and their intended purpose to grant access to those roles accurately;
- Resources must be organizedinto collections to facilitate delegation;
- Given a substantial number of roles and collections of resources, a correspondingly large number of delegations would need to be created and managed.

On the other hand, also ABAC has advantages and disadvantages. The principal advantages of ABAC are:

- ABAC enforces centralized management of authorization policies;
- ABAC makes it easy to specify access rules as simple queries;
- ABAC rules can be extremely fine-grained and contextual;
- ABAC rules can evaluate attributes of Subjects and Resources that are not inventoried by the authorization system;
- ABAC rules need less maintenance and overhead because theydonotrequire the creation or maintenance of the structure on which an RBAC model depends (e.g., roles and resource locations.)

ABAC's principal weaknesses are:

- ABAC makes it extremely difficult, if not impossible, to perform a "before the fact audit" and determine the permissions available to a specific user. Potentially, a huge number of rules might need to be executed, and in the same order in which the system applies them, to successfully determine access. As a result, it could be impossible to determine risk exposure for any given employee position;
- ABAC can lead to a "Rule Explosion" (somewhat in the same way as RBAC can create a "Role Explosion) as a system with N number of attributes would have 2^Npossible rule combinations;
- ABAC systems (which don't pre-calculate the net result of access rights) can be unacceptably slow to answer authorization queries unless rules are kept extremely simple and do not access data from multiple source systems.

Hybrid approaches that combine the richness of the RBAC relational modeling system with the flexibility and contextual nature of ABAC are also possible and offer the best solutions. EmpowerID proposes in this white paper [28] a new hybrid model. EmpowerID's sophisticated role and relationship modeling allows security architects to model the organization and its structure and policies, including segregation of duties policies to prevent undesired combinations of access. A flexible ABAC-based system supports acting as a centralized real-time decision point for applications that can call the EmpowerID API for authorization decisions. The ABAC engine enhances or modifies the decisions calculated by the powerful RBAC engine, allowing their use only when greater flexibility or contextual information is required. ABAC policies are made much more powerful by the inclusion of the pre-calculated access results that the engine derives from complex RBAC policies that account for inheritance and even attribute-based queries.

In recent years, blockchain-based access control received tremendous attention due to the fundamental features of blockchain including auditability, distributed management, trust, and immutability. In BRIGHT the blockchain technology combined with traditional models will be exploited in order to implement a mechanism for managing and enforcing data access policies. As a first step, in the first version of deliverable, we are going to analyse the existing approaches in the field of blockchain driven access control mechanisms. In the final version of deliverable, D6.4 we will implement a solution to handle data access policies in BRIGHT.

4.1 Blockchain-based data access control: use cases in literature

Blockchain has desirable features that make it a trustable alternative infrastructure for access control systems. In literature there are several studies proposing to consider blockchain as an infrastructure for access control systems.

Maesa et al [29] proposes to use blockchain technology to represent the rights to access resources and to transfer them from one user to another. The study uses the attribute-based access control mechanism and eXtensible Access Control Markup Language (XACML) [30] to define policies. The policies and the rights exchanges are publicly visible on the blockchain, accordingly any user can know at any time the policy paired with a resource and the subjects who currently have the rights to access the resource. This solution allows distributed auditability, preventing a party from fraudulently denying the rights granted by an enforceable policy. The approach has been validated through a reference implementation based on Bitcoin. In their recent study [30] the same authors refined and extended the previous approach by considering smart contracts to enforce access control policies instead of simple transactions. They have implemented a proof of concept using XACML policies and Ethereum platform. In order to evaluate the feasibility and performance of the represented system, they have defined a new scenario where smart contracts are considered as resources that need to be protected and access to them is restricted. They have concluded that applying Ethereum to their implemented system has brought benefits in terms of flexibility and efficiency.

In the field of cloud storage [32] proposes a data storage and sharing scheme for decentralized storage systems combining a decentralized storage system, the Ethereum blockchain and the Attribute-Based Encryption (ABE) technology. The only one who has access to the secret key is the data owner. Ethereum blockchain has been applied for managing the private keys. Essentially there are two smart contracts: the Data Sharing contract that is deployed by the data owner and includes methods to register a user who need access to the specific data belong to the owner of the contract and DataUser contract that is deployed by data requester to invoke the search function defined in data sharing contract to view the search results. In a similar way, this study [32] proposes a Reputation Based Knowledge Sharing system to protect the copyright using fine-grained access control. The system includes three main roles: Questioner, Answerer, and Bystander. The Questioner is the one who designs a question. The answerer is one who is an expert to answer the question and receives rewards from Bystander. The Bystander is the one who is willing to pay a small fee in order to get access to the shared knowledge.

In the IoT field the management of attributes (e.g., location, date, time, etc.) is significant to provide a decentralized, flexible, and fine-grained authorisation for IoT devices. Attributes are significant as they are used to express specified access policies by a target to decide if the requested entity fulfills the required privileges that are necessary for access. Blockchain is utilized in such cases that allow authentic and reliable credentials. In different studies [34] [35] [36] [37] it is proposed an attribute-based access control mechanism for IoTs that provides local access, authorization of clients, privacy, and interoperability by using smart contract data sharing and user-controlled encoded policies. The user can own their data and have authority to share it with other users. The ABAC model is used for its high compatibility and expressiveness.

Blockchain has desirable features that make it a trustable alternative infrastructure for access control systems. The distributed nature of blockchain solves the problem of single point of failure and other centralized management problems. Also, by eliminating third parties, we do not need to be concern about privacy leakage from their side. In addition, we can have access to a trustable and unmodifiable history logs. Consensus mechanisms are applied, so only valid transactions are recorded on blockchain. Furthermore, by using smart contracts, we can monitor and enforce access permissions under complex conditions. All of these features have motivated researchers to consider blockchain as an infrastructure for access control systems. Current access control methods which are static might be inadequate for future systems and more dynamic access control methods, one in which resources define their own access control, might be required. Integrating blockchain with dynamic access control approaches could be an interesting area to investigate in the future.

5 Conclusions

In this deliverable it is reported the work done in task 6.1 in relation to the implementation of the BRIGHT Distributed Energy Ledger tool. The blockchain-based approach offers a hybrid solution consisting of (i) storing off-chain, the raw monitored energy in a database, the real-time data as collected from the metering devices while (ii) all the collected values are hashed-linked back the raw data and periodically stored on-chain energy transactions for business enforcement and (iii) validation services are built upon the off-chain stored data resulting in a tamper-evident solution. The proposed solution brings benefits in terms of costs, scalability, and performance of the system maintaining the advantages provided by the distributed ledgers of transparency, security and trust.

In the document the concept of data access policies is also introduced. As a first step, existing traditional approaches like RBAC or ABAC are analysed and more innovative approaches in the field of blockchain driven access control mechanisms are reported. For the next steps a solution to handle data access policies in BRIGHT will be implemented and described in deliverable D6.4 for M24.

References

[1] "Could Blockchain Have as Great an Impact as the Internet?," [Online]. Available: https://www.jpmorganchase.com/news-stories/could-blockchain-have-great-impact-as-internet.

[2] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[3] "Ethereum Whitepaper," [Online]. Available: https://ethereum.org/en/whitepaper/.

[4] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," [Online]. Available: http://gavwood.com/paper.pdf.

[5] "Ethereum website," [Online]. Available: https://ethereum.org/en/.

[6] "Solidity documentation," [Online]. Available: https://docs.soliditylang.org/en/develop/.

[7] N. Szabo, "The Idea of Smart Contracts," *Nick Szabo's Papers and Concise Tutorials,* vol. 6.

[8] "How Ethereum works," [Online]. Available: https://ethereum.org/en/learn/#how-ethereum-works.

[9] "ERC-20 standard," [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[10] "ERC-721 Non-Fungible Token Standard," [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md.

[11] "CryptoKitties," [Online]. Available: https://www.cryptokitties.co/.

[12] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum and A. Peacock, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and Sustainable Energy Reviews,* vol. 100, pp. 143-174, 2019.

[13] "Brooklyn Microgrid," [Online]. Available: https://www.brooklyn.energy/.

[14] "Power Ledger," [Online]. Available: https://www.powerledger.io.

[15] "SOLshare project," [Online]. Available: https://me-solshare.com/.

[16] B. L. Risteska Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *Journal of Cleaner Production,* vol. 140, pp. 1454-1464, 2017.

[17] A. Dorri, S. S. Kanhere, R. Jurdak and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),* pp. 618-623, 2017.

[18] "OLI Technology," [Online]. Available: https://www.my-oli.com/en/technology.html.

[19] "Sunchain," [Online]. Available: https://www.sunchain.fr.

[20] "Share&Charge application," [Online]. Available: https://shareandcharge.com/.

[21] "Blockchain enigma paradox opportunity," [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/Innovation/deloitte-uk-blockchain-full-report.pdf.

[22] "Blockchain – an opportunity fore energy producers and consumers?," [Online]. Available: https://www.pwc.com/gx/en/industries/assets/pwc-blockchain-opportunity-for-energy-producers-and-consumers.pdf.

[23] S. Joshi, "Feasibility of Proof of Authority as a Consensus Protocol Model," 2021. [Online]. Available: https://arxiv.org/abs/2109.02480.

[24] [Online]. Available: https://www.mongodb.com/nosql-explained .

[25] "MQTT protocol," [Online]. Available: https://mqtt.org/ .

[26] "MongoDB," [Online]. Available: https://www.mongodb.com/ .

[27] "Introducing JSON," [Online]. Available: https://www.json.org/json-en.html.

[28] EmpowerID, "Best Practices in EnterpriseAuthorization:The Next Generation Access Control with RBAC/ABAC Hybrid Model," 2019.

[29] D. D. F. Maesa, P. Mori and L. Ricci, "Blockchain Based Access Control," *IFIP International Conference on Distributed Applications and Interoperable Systems,* pp. 206-220, 2017.

[30] "OASIS: eXtensible Access Control Markup Language (XACML) version 3.0," January 2013. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[31] D. F. Maesa, P. Mori and L. Ricci, "A blockchain based approach for the definition of auditable Access Control systems," *Computers & Security,* vol. 84, pp. 93-119, 2019.

[32] S. Wang, Y. Zhang and Y. Zhang, "A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems," *IEEE Access,* vol. 6, 2018.

[33] S. Hu, L. Hou, G. Chen, J. Weng and J. Li, "Reputation-based Distributed Knowledge Sharing System in Blockchain," *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services,* pp. 476-481, 2018.

[34] S. Zaidi, M. Shah, H. Khattak, C. Maple, H. Rauf, A. El-Sherbeeny and M. El-Meligy, "An Attribute-Based Access Control for IoT Using Blockchain and Smart Contracts," *Sustainability,* 2021.

[35] L. Song, M. Li, Z. Zhu, P. Yuan and Y. He, "Attribute-Based Access Control Using Smart Contracts for the Internet of Things," *Procedia Computer Science,* vol. 174, pp. 231-242, 2020.

[36] H. Liu, D. Han and D. Li, "Fabric-iot: A Blockchain-Based Access Control System in IoT," *IEEE Access,* vol. 8, pp. 18207-18218, 2020.

[37] Y. Zhang, B. Li, B. Liu, J. Wu, Y. Wang and X. Yang, “An Attribute-Based Collaborative Access Control Scheme Using Blockchain for IoT Devices,” *Electronics,* 2020.