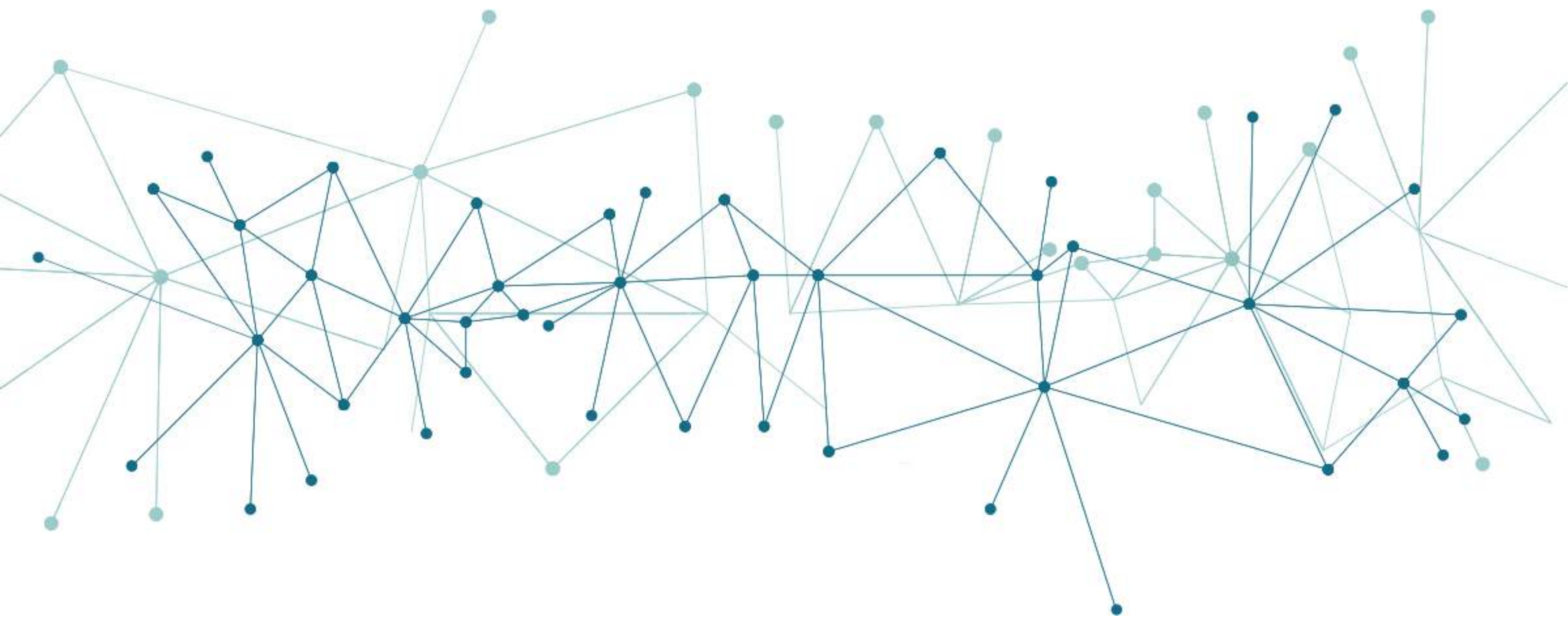
The eDREAM project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 774478

enabling new Demand REsponse Advanced, Market oriented and secure technologies, solutions and business models

DELIVERABLE: D5.1 Blockchain platform for secure and distributed management of DR programs V1

Authors: Giuseppe Raveduto, Vincenzo Croce



Imprint

D5.1. Blockchain platform for secure and distributed management of DR programs V1

Contractual Date of Delivery to the EC:	31.05.2019
Actual Date of Delivery to the EC:	31.05.2019
Author(s):	Giuseppe Raveduto (ENG), Vincenzo Croce (ENG),
Participant(s):	Tudor Cioara (TUC), Claudia Pop (TUC), Marcel Antal (TUC), Giuseppe Mastandrea (E@W), Alessio Cavadenti (ASM), Francesco Bellesini (EMOT)
Project:	enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models (eDREAM)
Work package:	Wp5 – Blockchain-enabled decentralized network control optimization and DR verification
Task:	T5.1
Confidentiality:	public
Version:	1.0

Legal Disclaimer

The project enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models (eDREAM) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 774478. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the Innovation and Networks Executive Agency (INEA) or the European Commission (EC). INEA or the EC are not responsible for any use that may be made of the information contained therein.

Copyright

Executive Summary

Deliverable 5.1, “*Blockchain platform for secure and distributed management of DR programs V1*”, is the document which reports on the activities of Task 5.1 “*secure energy data handling via blockchain/ledger*” carried out by the project consortium during the first half of the project. Work package 5 explores the application of blockchain solutions to the energy sector at different levels: this deliverable, together with D5.4 “Blockchain platform for secure and distributed management of DR programs V2” focus on secure storage capabilities, deliverables from Task 5.2 (D5.2 and D5.5) explore the execution of smart contracts and D5.3 focuses on the validation of the DR services and financial settlement.

The document starts with the description of the current status of blockchain technologies and the identification of use cases in the energy sector for the application of blockchains, then illustrates the benefits provided by blockchains and the potential issues, and finally presents two different approaches explored by the project to overcome scalability issues: the first one based on a third-party solution and the second one entirely developed on purpose as a second-tier energy data storage.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

eDREAM	enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models
BFT	Byzantine Fault Tolerance
DLT	Distributed Ledger Technology
DR	Demand Response
EVM	Ethereum Virtual Machine

1. Introduction

1.1. Purpose

This document describes the solution developed by the eDREAM project to provide a secure data storage layer for microgrid energy transactions and DR flexibility services through a blockchain.

Although blockchains could be used to directly store data *on-chain* their characteristics make this too expensive to be affordable, from the perspectives of both costs and performance. For this reason, the implemented blockchain solution is focused on scalability, reducing the footprint in terms of on-chain storage while preserving the security and trust provided by blockchain usage.

Recently, blockchain-based distributed databases such as BigchainDB are being engineered, trying to provide the best of the two worlds between distributed databases and blockchains: we evaluated how this kind of platform applies to the energy data storage context and how this compares with the novel hybrid solution proposed.

1.2. Relation to other activities

This document benefits from the activities carried out in WP2, in particular regarding the use cases defined in "*D2.2 Use case analysis and application scenarios description V1*", the requirements defined in "*D2.1 User group definitions, end-user needs, requirement analysis and deployment guidelines V1*" and the analysis of the regulatory framework regarding blockchain solutions presented in "*D2.3 eDREAM standardisation report and regulatory roadmap*", and provides a storage solution for the tools, techniques and services to be developed in WP3 (*Techniques for DR and energy flexibility assessment*) and WP4 (*Next generation DR services for aggregators and customers*).

Regarding the other activities part of WP5, "*D5.2 Self-enforcing smart contract for DR tracking and control V1*" describes the application of smart contracts to prosumers' flexibility aggregation and the peer-to-peer energy trading, while "*D5.3 Consensus based techniques for DR validation and financial settlement*" will describe the validation of the DR services and the definition of associated financial transactions incentivising or penalising the participants.

1.3. Structure of the document

Section 2, "Blockchain and Energy Data Handling" describes the current status of blockchain technologies and how this can be related to energy data handling. It also presents a description of the pilot equipment and the data to be stored and describes the different logical "layers" of the proposed solution. In Section 3 an overview of a blockchain distributed ledger at the micro-grid level is presented, describing how the properties are assured at ledger level with also a focus on privacy and a description of the proposed network, describing the different kind of nodes that could be involved. Next, Section 4 describes the scalability problem and how the tested solution based on BigchainDB and the hybrid *off-chain* storage solution aim to address it. Finally, Section 5 presents the conclusions from the work presented in this deliverable.

2. Blockchain and Energy Data Handling

A Distributed Ledger (or Distributed Ledger Technology, DLT) is a distributed, tamper-proof database, not controlled by a single institution. A blockchain is a distributed ledger in which transactions are grouped in *blocks* and *chained* through cryptographic hashes[1] into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work[2].

Different blockchain technologies may differ in the *consensus mechanism* and the *programming capabilities*.

Considering the consensus mechanism, blockchains differ in the definition of the nodes participation in the distributed network and the roles that they can perform. In particular we can distinguish between *open* blockchains and *permissioned (or private)* blockchains.

Considering the programming capabilities, we can 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.

2.1. Ethereum

Ethereum was proposed by Vitalik Buterin in 2013[3] and further detailed by Gavin Wood in the "yellow paper"[4] (Ethereum: A Secure Decentralised Generalised Transaction Ledger).

The Ethereum blockchain produces new blocks roughly every 15 seconds on average. The high block generation rate has the consequence of a high probability that more than one block is mined in parallel: this event is an actual fork and the winning block will be determined by the subsequent blocks. To mitigate the impact of large pools of nodes, which can take advantage of the time required to disseminate the block to the entire network disseminating new blocks among themselves before disseminating them to the rest of the network, Ethereum rewards also valid orphan blocks on the abandoned branch of a fork.

The reward is assigned when a new block (*nephew*) links a past block (*uncle*) in addition to its parent block. The reward is then distributed between the nephew (12.5%) and the uncle (87.5%) blocks.

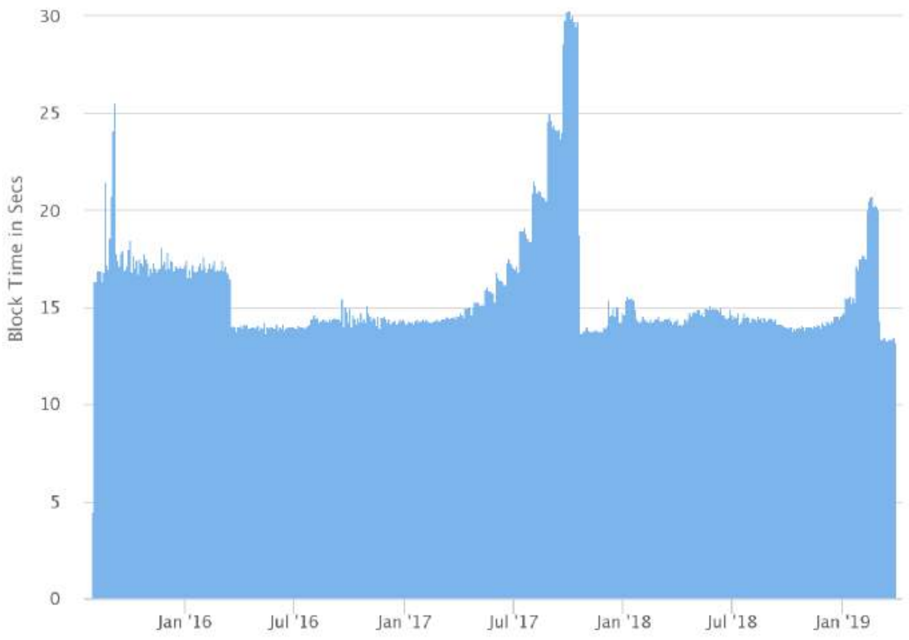


Figure 1: Ethereum average block time chart (https://etherscan.io/chart/blocktime)

The currency used in Ethereum is called *Ether* and it has a double use: it is used as incentive for the network "validators" but also to regulate the use of the blockchain computational resources. More in detail, each operation on the Ethereum network has its own cost, determined by the computation, storage and bandwidth required, and this cost is measured in a unit called *Gas,* so *gasLimit* (the maximum amount of gas that can be spent) and *gasPrice* (the price-per-gas) are standard transaction parameters in Ethereum and also invoking a smart-contract function both must be specified. Smart contracts are run on a virtual machine (*Ethereum Virtual Machine*, EVM) and the presence of the gas price and limit have the purpose of prevent denial-of-service attacks; specifically, is impossible to run infinite loops (due to the gas limit) and the non-optimised usage of system resources would be expensive due to the gas price.

2.2. Blockchain for the energy supply

Blockchains or distributed ledgers have drawn considerable interest from energy supply firms, start-ups, technology providers, financial institutions, national government and the academic community[5].

From March 2017 to March 2018, start-ups raised over $300 million to apply blockchain to the energy sector, while utility-sponsored initiatives represent the second-most numerous category of blockchain ventures[6].

Most of the actors currently sponsoring energy and blockchain initiatives are in Europe, followed by North America, but are not limited to these. For example, Chile's National Energy Commission (CNE) will be utilising the Ethereum blockchain to authenticate data such as average market prices and fuel prices, with the aim of improving the security of the country's energy data[7].

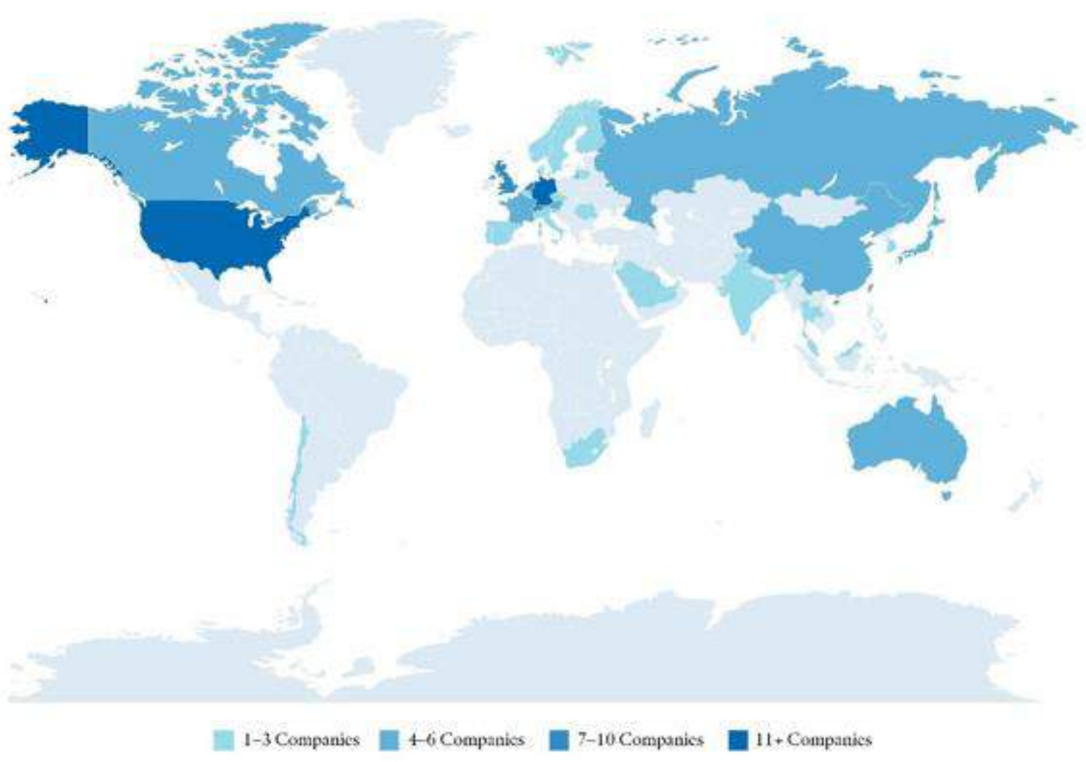


Figure 2: Geographic Distribution of blockchain Initiatives in the Electric Power Sector (http://www.ourenergypolicy.org/wp-content/uploads/2018/07/Discussion_Paper_Livingston_et_al_Blockchain_OR_0.pdf)

The global microgrid market size is growing and there is an estimated 15% compound annual growth rate between 2018 and 2022[8]. The market liberalisation and the renewable energy provide the basis for blockchain technology to drive the transformation towards a decentralised ecosystem of users, producers, retailers, traders and utilities.

2.2.1. Use cases for the application of blockchain in the electric power sector

The first application of the blockchain in the electric sector, is the facilitation of electricity trading. This field is directly correlated with the primary application of blockchain, the cryptocurrency trading.

The most intuitive approach is to enable *peer-to-peer transactions* between customers, for example trading excess renewable power. Another approach is the usage of blockchain to support an electric power system which keeps the current power grid with *grid transactions*.

Another use case considers the application of blockchain and cryptocurrencies to fund *energy financing*[9]*,* while other notable examples of application include the record of *sustainability attributes*, the usage of charging infrastructure to sell *charging services to EV owners*[10] and *asset management*.

More in general, the blockchain use cases in the energy sector can be classified into eight large groups, namely[11]:

1. Metering, billing and security

2. Cryptocurrencies, tokens and investment
3. Decentralised energy trading
4. Green certificates and carbon trading
5. Grid management
6. IoT, smart devices, automation and asset management
7. Electric e-mobility
8. General purpose initiatives

The eDREAM project, is exploring blockchain capabilities to enable the following high level use cases as defined in "*D2.2 Use Case Analysis and application scenarios description V1*":

- HL-UC01: Prosumers DR flexibility aggregation via smart contract
- HL-UC02: Peer-to-peer local energy trading

The two use cases can be mapped inside the groups "*grid management*" and "*decentralised energy trading*" respectively, but will also include elements related to the other groups as well, in particular "*IoT, smart devices, automation and asset management*", "*Cryptocurrencies, tokens and investment*", "*Electric e-mobility*" and "*Metering, billing and security".*

In particular, In scenario 1, Multiple decentralised active microgrids, we address a case where the DSO (Distributed System Operator) can access micro grid resources, especially stationary and mobile battery storage and loads, exploiting microgrid flexibility in order to guarantee smart-grid stability, providing flexibility-as-a-service through smart contracts and, on the other hand, the prosumers, directly or via enabling aggregators, are able to offer via smart contracts their flexibility resources, both production and loads modulation. In this context, blockchain technology is essential not only for real-time writing of micro-contracts between prosumers and their DSO, but also. and above all, to ensure that energy data collected in real-time are reliable and transparent. This paves the way for a new business for DSOs, directly involving end user to improve the electricity grid status, making the grid more stable, efficient and suitable for a massive integration of energy generated from renewable sources.

Electric mobility plays a fundamental role in this scenario, as it allows the end user to provide much more flexibility to the electricity grid than he could before; in fact, the loads that can be activated by a home user is about 3 kW (washing machine + lights + refrigerator) while the load related to charging an electric vehicle is about 22 kW. Involving electric mobility in Demand Response campaigns is possible thanks to smart charging stations: EMOT charging stations, the SpotLink EVO, exchange data through their single-board computer, a Raspberry Pi 3, with a CPU of quad-core ARM Cortex A53 1.2 GHz, a SD of 16 GB, a RAM of 1 GB and a Raspbian Stretch 4.14 S.O.; these features allow the charging station to be monitored in real time and managed remotely, enabling a charge that instantly meets the needs of the electricity grid, varying the charging power according to the DSO's request. The data collected by the charging station are made transparent and reliable by blockchain technology; in particular they are energy data (V, A, kW, kWh), number of sockets in use, current vehicle ID attached and charging station status (reports, alarms). Regarding electric-vehicle (EV) monitoring, EMOT will use an OBD (On-Board Diagnostic) device to retrieve data from the EV; OBD is an IoT component that utilises a TCP/IP communication to a TCP/IP server. The network connectivity of the OBD device is via data SIM (UMTS) and the server runs python software, which queries the EV each 5 seconds. The OBD connects to the diagnostic interface from which it is able to extract the information from the electric vehicle control unit using the CAN-bus protocol. The output data format of the OBD is an ASCII string; when the data is sent to the server, it is reorganised into a wrapper, thus obtaining a grouping of the data in JSON format. The following data will be retrieved and processed by blockchain: battery state-of-charge (SoC), geolocation, doors car status and engine car status; knowing the position and the SoC of the electric vehicle, it is possible to estimate its impact on the electricity network by providing flexibility.

The DSO of the city of Terni (ASM Terni S.p.A.) has planned to test for research purposes the use of blockchain technology and related smart contracts for managing the levels of energy demand flexibility between aggregators and enrolled prosumers and between aggregators and the DSO. In this regard, a possible framework to develop the blockchain can be the installation in the ASM server farm of one virtual machine, equipped with duplicated units (RAM, Operating System, CPU, Hard Disk). Basically, for reaching an operational structure and test a blockchain-based system, considering that at least 3f + 1 nodes are required to tolerate f faults[12] for Byzantine Fault Tolerance, 4-7 VMs or, even better, physical nodes may be installed to run the framework under a "production" perspective. ASM started using a basic configuration of a blockchain technology for research and this activity is currently ongoing.

Regarding the handling of energy data, at the moment almost all the electricity customers have smart meters installed in their premises which generate data every 15 minutes. In some critical points of the power network, ASM TERNI has installed new generation smart meters that are able to obtain real-time measurements. Basically, those devices are able to measure the following parameters: voltages and currents, active, reactive and apparent power, frequency and flicker. The time-based readings of the intelligent meters are managed, read and detected at a distance from an advanced measurement infrastructure (AMI). ASM is currently using two different channels to collect data from installed meters. The first is used exclusively for the point of delivery(PoD) for customers with systems up to 30 kW. In this case the data are temporarily stored in a concentrator of electricity meters (one for every 400 meters), extracted and transmitted via the GPRS network to the ASM Terni servers. The second is for consumption above 30 kW; the data are extracted and transmitted via GSM/GPRS to software for managing the meter data and analysed in order to check for any data transmission problems and, finally, temporarily stored on the server of the ASM for at least five years. The whole medium-voltage (MV) network is under the control of a SCADA system that communicates with the calculation platform, and receives and sends information from/to the main station/substation equipment. However, ASM's pilot site by means of the smart meter extension (SMX) devices (a result of Nobel GRID project), is able to communicate with different protocols (e.g. DLMS, OpenADR, IEC61850) and compliant with different interfaces (e.g. USB, RS-232, RS-485). SMXs enables measurement all the data gathered by the meter by means of digital signals. The data connections are available by means of 3G sim, Ethernet, Internet protocol, Supporting VPN and the data formats are only .txt or .json. The availability of the data in real time (5s delay) have a dimension of 1 MB/day and a transmission frequency every 5 seconds.

2.3. Blockchain stack for distributed applications

Solutions built on blockchain can be described using different logical layers, covering data storage, business logic and financial transactions. This can be represented by the concept of "fat protocol".

2.3.1. Thin vs Fat protocols

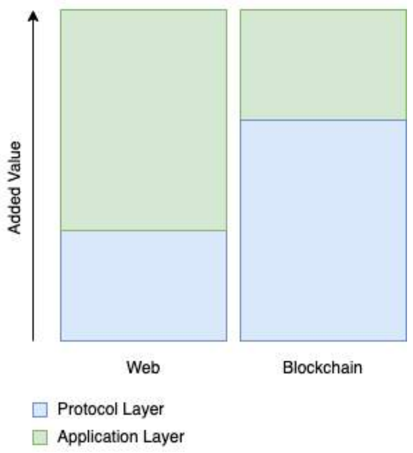


Figure 3: Representation of a "thin protocol" vs a "fat protocol" stacks

Differently from what happens with Internet protocols (TCP/IP, HTTP, etc.) in which the value concentrates largely on the applications layer built on top of different "thin" layers, in the blockchain application stack the relation is inverse: the majority of value lies in the protocol layer and not in the application layer. This is mostly caused by two major factors: the existence of a *shared data layer* and the introduction of crypto-graphic "*tokens*".

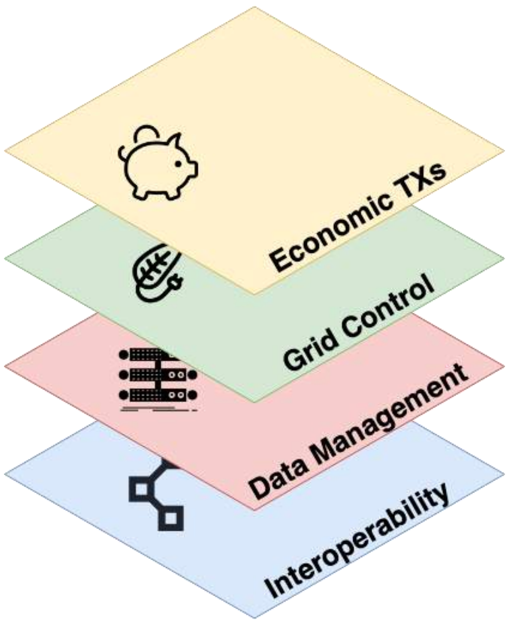


Figure 4: Representation of the added value provided by the blockchain as a protocol

2.3.2. Interoperability

If it is easy to understand how the speculative value of some of the tokens may incentivise protocol develop-ment, it is important also to underline how the shared data layer contributes to this.

In summary, the replication and storage of data in a decentralised network (opposed to individual applications accessing their own siloed information) reduces the entry barrier for new players, leading to a more competi-tive environment of products and services.

Moreover, protocols such as Interledger (ILP, InterLedger Protocol) are being developed, aiming to achieve the integration with any type of ledger with a variety of higher-level protocols, maintaining the advantages provided by the distributed ledgers of transparency, security and trust[13].

2.3.3. Data Management

Even if theoretically possible, the direct usage of a blockchain for data storage is impractical in terms of costs and performance.

According to the yellow paper, the fee to store a 256bit word is 20k gas, resulting in a price per kilobyte of 640k gas. Even if the storage cost is reduced by EIP 1283[14], the median gas price for the last 1500 blocks is – to date – 10 gwei (0.000000010 ETH)[15], so (not considering EIP 1283) a KB of storage costs 0.0064 ETH and a GB about 6710 ETH.

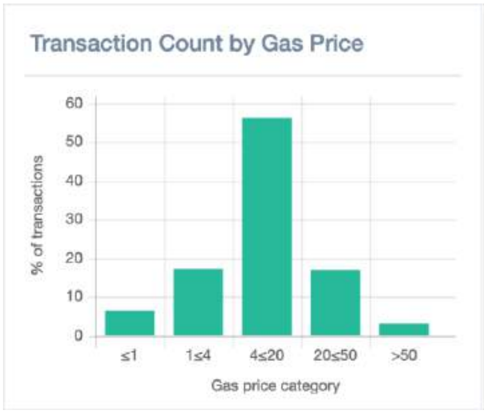


Figure 5: Transaction count by gas price (https://www.ethgasstation.info/)

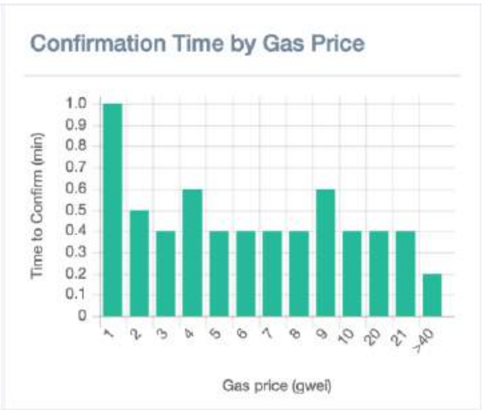


Figure 6: Confirmation time by gas price (https://www.ethgasstation.info/)

Given the current Ethereum price of 122.43 €, this will be 0.78 € for a KB and more than 820000 € for a GB. The final price is also highly volatile, depending on both the Ether/Euro and the Ether/Gas prices.

While for some kind of high-value transactions this cost may be affordable (e.g. the notarisation of a real-estate transfer), for frequent low-value transaction, such as fifteen minutes readings from a smart meter, this become a prohibitive limit.

To minimise the costs, and avoid fluctuations, it is crucial to optimise resources, minimising the usage of the *on-chain* storage capabilities. In this document we will describe two different solutions aimed at this without losing the advantages provided by a secure and distributed system.

2.3.4. Grid Control

The blockchain can also be used to control DR flexibility services and energy transactions.

The eDREAM project will demonstrate how smart contracts can be applied to prosumers' flexibility aggregation and local peer-to-peer energy trading, making the transactions trackable and tamper-proof. *"D5.2 Self-enforcing smart contract for DR tracking and control V1"* will cover this topic in depth.

2.3.5. Economic Transactions

The economic incentive provided to the *miners* or *validators* of a blockchain is the incentive for them to cooperate and actually support and contribute to the network's security on a public blockchain. This is why every blockchain primary use case involves some kind of economical transaction.

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 own transactions, while the term token usually indicates a digital representation of a specific asset, built on top of a blockchain.

Currently two main categories of tokens are defined on Ethereum: ERC-20[16] for *fungible* tokens that are interchangeable and not-unique and so can be used to represent a value like a currency note, and ERC-721[17] for *non-fungible* tokens, representing a unique asset like a collectible good.

The eDREAM project will explore the usage of tokens as a way to reward or penalise users involved in DR programs based on their behaviour in part of *"D5.3 Consensus based techniques for DR validation and financial settlement"*.

3. Distributed Ledger for Energy Transactions

In eDREAM we had proposed the implementation of a blockchain distributed ledger at the micro-grid level in which all energy-monitored data are registered at the level of an individual prosumer (or any other distributed energy resource) and then stored as immutable energy transactions. The individual energy transactions are aggregated in blocks which will be replicated in the ledger. The prosumer is modelled as a node of the peer-to-peer distributed energy network (i.e. a graph of peer nodes) and will maintain a copy of the ledger which will be automatically updated when new energy transactions are being registered. Other energy participants such as the energy aggregators or the DSO that are interested in micro-grid management, are also registered as peers.

Whenever a new prosumer joins the blockchain network, a new node is created and will be connected to a predefined list of seed nodes. The seed nodes will provide the new joint node with information about all the prosumer peers they know about, the process being repeated with the new discovered peers, until the new node builds its own list of peers. The nodes can be *light* or *full*, depending on their hardware capabilities and on the amount of information they can store. The full nodes keep the entire blockchain locally (i.e. replicated), can build blocks and make transaction validations on their own. They are nodes that actively participate to consistency and the integrity validations of the blockchain by participating in the consensus algorithms. The light nodes do not have large storage resources; thus, they do not hold the entire blocks (e.g. Node 5 in Figure 7), but only their headers thus reducing the storage space by 1000 times.

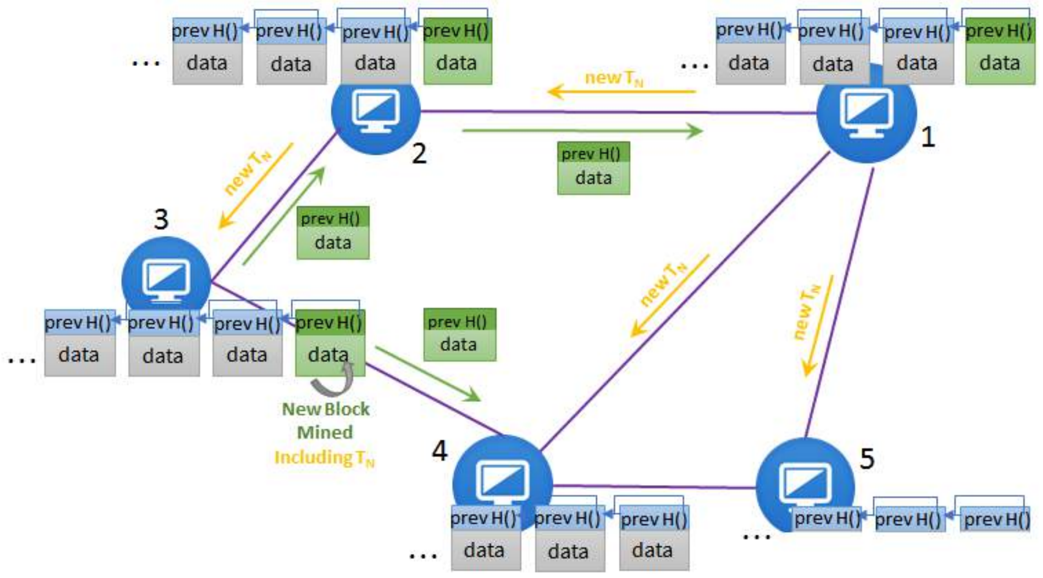


Figure 7: Blockchain blocks distribution among peers (Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie and Massimo Bertoncini, Blockchain Based Decentralized Management of Demand Response Programs in Smart Energy Grids, Sensors 2018, 18(1), 162.)

When a new energy transaction is registered, the issuer node (e.g. Node 3 in Figure 7) will propagate it to all its peers (Node 2, Node 4, and Node 5 in Figure 7). In turn, the nodes receiving the energy transaction will validate and forward it to their own peers, and so on until the transaction is known by all the nodes in the network. In a similar way, when a node manages to mine a block of energy transactions (e.g. Node 1) it will propagate it to its peers (e.g. Node 2, Node 4) and each receiving node will validate it, before sending it further to other nodes. To avoid loops in the network, a node will also decide not to forward an energy transaction if it was already previously registered. In the rest of the section we discuss how the important benefits of the blockchain technologies are assured at the level of energy transactions and we end with a topological description of the proposed network. Due to the high costs associated with blockchain-based processing of data, registering an energy transaction each time new energy data is being sampled by the smart metering devices is not scalable. Thus, we have developed a scalable approach, allowing for energy transactions to be registered more rarely on the chain (e.g. 1 every hour) while not losing any of the benefits brought by the blockchain technology. This approach is detailed in Section 4.2.

3.1. Data Provenance and Immutability

The data structures used to create the proposed energy transactions in the ledger assure the provenance property by enacting their tracking back until the moment of their registration in the blockchain. The distributed ledger is a collection of blocks, linked back using hash pointers, each block storing a set of valid transactions on the registered digital assets (see Figure 8). The linked list is an append-only data structure. Any changes that would appear in previous registered nodes would lead to inconsistencies, because the hash pointer of that block would change. If one needs to change the content of a previous block, all the following blocks will need to be rehashed and re-linked to obtain a consistent updated data structure. The advantage brought by this structure is the tamper proof log on all the transactional information contained in the blocks. Furthermore, because this is an append-only type of data structure (new blocks are always added at the head of the chain) it offers reliable historical information and also preserves the order in which the energy transactions are registered. The probability of changing the value of the transacted energy asset in a block by an attacker decreases with the number of blocks following that block in the append-only linked list.

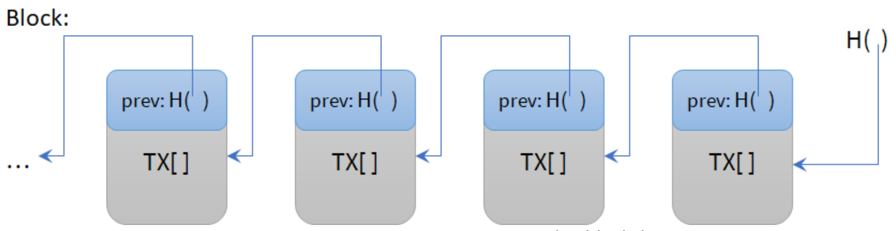


Figure 8: Energy transactions registered in blockchain

In our proposed approach each prosumer who generates energy can register energy assets in the distributed ledger, based on the information provided by the associated smart meter, by signing and registering an energy transaction having as a receiver its own contract account address[18]. The flow of energy between prosumers will be then represented in blockchain as energy transactions between two prosumers accounts. To prove the ownership of the energy, the prosumer provides pointers to previous energy transactions showing that the energy asset belongs to him and signs the current transactions, to validate the transfer.

Three main features that make the hash functions suitable to be used for registering the energy transactions securely and in a tamper-proof manner in the distributed ledger are the following.

- *Collision-free property* – for any two input values X and Y there is a very high probability to have two different hash values H(X)≠ H(Y). Collisions are possible, due to the fact that the size of the input data is longer than the size of the output data. However, a good hash function is designed such that, by knowing the hash code there are no better ways to find the input value, other than trying all the possibilities. Considering that an attacker has knowledge of the hash code of a value, H(X), the probability of determining the actual information X is extremely low due to the fact that it would take an infeasible long time to try all the possibilities.
- *Data-concealing property* – any entity is allowed to hide or conceal their data by providing a hash of these data (considering that it is improbable for an attacker to provide another piece of data to have exactly the same hash).
- *Data-binding property* – data-binding between the hash code and the input data an entity can prove the origin and the ownership of the data at any time in the future by applying the hash function on the original piece of data. Hash values can be further used to identify and verify the integrity of data. When retrieving the data based on a specified hash pointer one can check that the data have not

changed since their creation. The concept of hash pointers is the foundation for different hashed data structures commonly used in blockchain systems: linked list with hash pointers, binary trees with hash pointers (Merkle Tree), etc.

To provide better scalability and decrease the length for the chain multiple transactions are aggregated in a single block [18] (see Figure 9). Depending on the solution, different approaches have been used in order to keep track and encode the transactions mined in a block. In Bitcoin29 the transactions of the block are grouped and encoded using Merkle Trees. To support the protocol enhancement the Ethereum[19] solution, uses three different Modified Merkle Patricia Tries per block that store key value pairs: the State and Storage Trie, the Transaction Trie and the Receipt Trie. To build a Merkle Tree, all energy transactions in the block are paired two-by-two and the tree is built from the bottom to the top based on the hashes of these transactions. The tree leaves will contain the energy transactions while the upper levels will be incrementally constructed by pairing and combining the hashes of two elements from an inferior level until the root is reached. In this way, a binary hash tree is built up to the root. The result is a 32-byte string encoding an entire set of data. Due to its structure, any change that occurs at leaf level, due to tampering with data, will trigger modification up to the root of the tree.

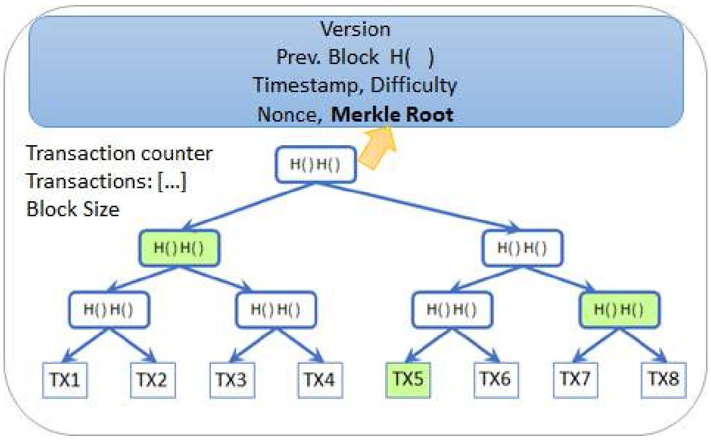


Figure 9: Merkle Tree for storing multiple energy transactions in a block and Merkle path example

The Merkle Tree Root hash encodes the entire collection of energy transactions that are aggregated in the current block. This hash value is of important significance because it is responsible to ensure the validity and the integrity of the recorded energy transactions in time. The root hash is added in the header of the block (see Figure 9) and, together with all the other fields in the header, is used to generate the hash of the block. The block hash is further used to identify a block in the entire blockchain. The hash of the block is not stored in the current's block structure or in the blockchain storage; instead, it is always computed by each node based on the information contained in the header and used as a hash pointer in the following block. The hash of the tree root provides a smaller footprint, an important advantage for the prosumer nodes that do not have enough storage capabilities (i.e. light nodes). This is extremely important for small scale prosumers which may not have large computing capabilities deployed on site. Thus, the prosumers associated light nodes will store only the header of the blocks while the actual energy transactions will be stored remotely. The Merkle tree root will provide enough information for light prosumer nodes to be able to check the consistency of the chain. At the same time, light prosumer nodes may interrogate other network full nodes for information to verify if an energy transaction was mined and to identify the block that stores the actual transaction.

One of the simplest way to prove that an energy transaction is stored in a block would be to obtain all the transactions of a block and rehash the entire tree to obtain the root hash. If the root hash obtained is the same with the one stored in the header of light prosumer node block, it would lead to the conclusion that the specific energy transaction was successfully mined. However, this process is not optimal. Since a large number of energy transactions can be included in the block, the verification process is significantly improved by providing only a path instead of the entire set of transactions. For example, consider that the light prosumer node needs to find if energy transaction TX6 was mined or not. It will request a path that proves the membership of this transaction in a specific block as presented in (19) for systems like Bitcoin and Ethereum. This path will contain the hash roots of the subtrees that are not influenced by the hash of the TX6 transaction, marked by green nodes in Figure 9. In this sense, the first hash of the path is the hash of the transaction that was paired with TX6. The hash of the two transactions will be then paired with the second hash from the path, and so on until reaching the root of the tree. The performance improvement of this approach is considerable, since the node will need to compute only log(N) hashes to prove the membership of an energy transaction.

Considering that in the blockchain all energy transactions are duplicated and shared across the network peer nodes, it is imperative to provide solid ways for ensuring security mechanisms in the systems that provide authorization and authentication of registered prosumers. Public-key cryptography plays a crucial role for assuring ownership of the data, the security of all energy transactions as well as authentication and authorization. Prosumers will use their private key to sign their own energy transactions which will be addressable on the blockchain network only via their 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 blockchain-based management platform.

To enforce energy asset ownership, each entry of an energy transaction is linked to the identity of a prosumer which must own at least one pair of public-private keys. By applying several hashing and encoding algorithms over the public key prosumers' identity is generated as a 34-character string. Since the energy transaction is guarded with cryptographically locking scripts, the only way to prove energy ownership, unlock the assets and perform a transaction is to own the paired private key.

Private-public keys help build an authentication and authorisation mechanism in a distributed ledger. The traded energy tokens part of a transaction is not sent directly to the address of the receiving prosumer but rather to a smart contract (or locking script in Bitcoin[20]) which contains the public key of the recipient. The smart contract contains a set of rules that must be enforced whenever the energy token is transacted again in the future. In this way the energy tokens are locked and next energy transactions involving them will need to provide the required signature generated using the private key. The transactions' energy tokens are locked in a smart contract by associating the actual tokens with the address (computed based on the public key) of the prosumer initiating the transaction. To unlock tokens, a cryptographic signature is required that can only be provided by the prosumer holding the paired private key. As a result, once the transaction is mined, the recipient prosumer is given ownership over the energy assets.

3.2. Distributed Energy Network

From the perspective of network topology, the simplest description of a blockchain solution for an electric grid, consists of a set of blockchain nodes connected to the same network and different actors connected to the electrical grid, connected to the blockchain nodes and exchanging data with them. To keep the description as most generic as possible, blockchain nodes may or may not coincide with the actors involved in the electrical grid. It is realistic to imagine a scenario in which large producers may afford the host locally their own full node, while small prosumers or consumers may host a light node (as described above in Section 3) or choose to *trust* a third party node.

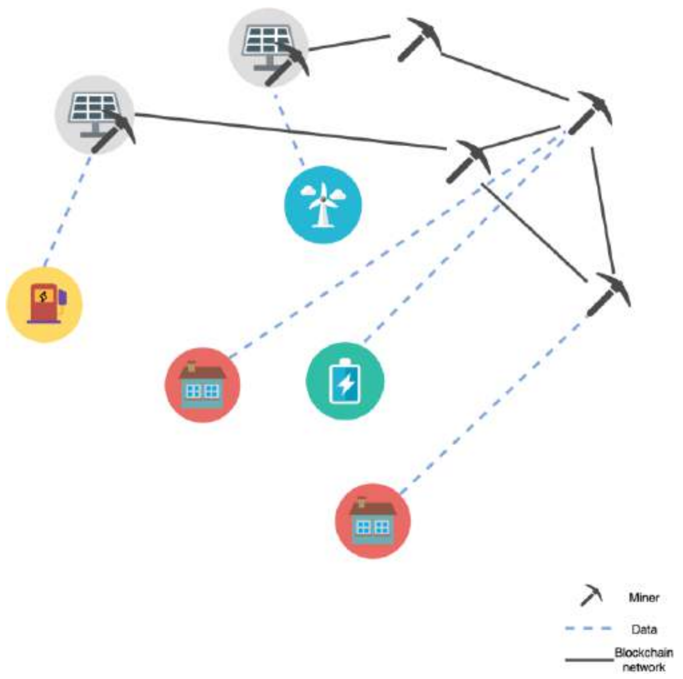


Figure 10: High level representation of the proposed network

In any case, smart metering devices must be interfaced with a blockchain node, in a secure manner. Figure 11, schematises the connection between a blockchain node and a smart meter, via an enabling embedded device. As a side note, the blockchain node depicted in Figure 11 can be a local one or a third party remote node. The replicated nature of the blockchain, the common shared data storage and the cryptography-based authentication enable seamless turnover.

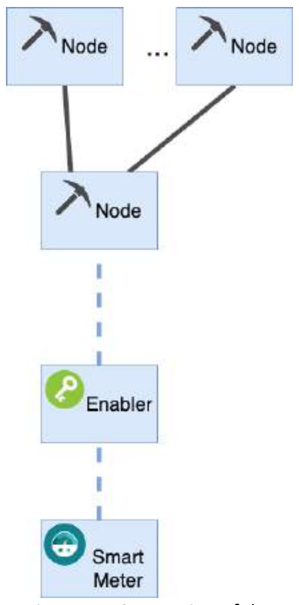


Figure 11: Connection of the smart meter to the blockchain via embedded enabling device

The *enabler* device, can be any embedded device (e.g. Raspberry Pi) capable of communicating with the smart meter via a standard protocol such as MQTT or Modbus, signing the received data with its own private key and initiating a transaction on the blockchain. A more detailed description of the proposed implementation is provided in Section 4.

4. Ledger Implementation for Improved Scalability

The increasing pace of the evolution of smart metering devices together with the prospect of renewable energy integration requires new designs to be considered for accommodating the storage and the validation of the large amount of data registered by smart grid systems. When addressing the problem of storage, the system should be able to handle both high requirements in terms of data size as well as data throughput. Furthermore, functionalities regarding the validation and evaluation of the monitored data, should be provided in near real time fashion, in order to ensure the financial settlement and system diagnosis as often as possible. Recently, it has been discussed that blockchain technology can provide the required secure and reliable means of ensuring the correct functionality of a smart grid system[21]. However, even if in terms of reliability and security, blockchain is a good solution, being able to decentralise and democratises the entire business process of the smart grid, the most pressing problem is scalability when integrating the business evaluation process with the real-time data provided by sensors. Storing data on blockchain is very expensive with respect to more traditional systems. Besides the extremely high cost, another drawback is that by increasing the size of the chain, a lot of resources will be required from the full nodes to store the entire history. High storage and processing hardware requirements would lead to a decrease in the number of full nodes in the system that would threaten the reliability of the entire distributed ledger. Another major restriction in terms of scalability is the transaction throughput supported by a blockchain system. Existing blockchain solutions like Bitcoin or Ethereum allows up to 7 transactions per second, or 15 transactions per second respectively. Thus, considering that thousands of energy sensors having a sampling rate at intervals of seconds should be able to register their values at the level of a microgrid, a blockchain solution could be rendered infeasible due to on chain congestion.

In terms of storing the monitored energy values there are two types of solutions (see Table 1).

- Firstly, to use existing distributed-database solutions that are well known for the high scalability and implement some level of decentralised control. However, their shortcomings in terms of Byzantine fault tolerance and immutability, rendering them unsuitable for specific use cases where Byzantine attacks can be expected;
- Secondly, to use the blockchain for storing data on-chain which is extremely costly and at the same time inadequate in use cases where high scalability is required.

Table 1: Comparison between Data Storage Solutions

	Distributed Database	**On-Chain Data**	**Blockchain & Distributed Database (2nd Tier Solution)**	**BigchainDB**
Immutability	no	Tamper proof	Tamper evident	Tamper evident
Decentralized Control	yes	yes	yes	yes
Byzantine Tolerant	no	yes	yes	yes
Storage Scalability	high	low	high	high
Costly (*public network)	no	yes	medium	yes
Smart Contract	no	yes	yes	no

The disadvantages of both types of solutions make them unsuitable in the context of eDREAM project. Thus, for constructing our distributed ledger we have proposed two alternatives.

- Firstly, use of the BigchainDB[22] solution that that relies on the Tendermint[23] blockchain for transaction broadcasting and consensus, and on MongoDB[24] for off-chain storage;
- Secondly, a novel hybrid solution based on a symbiosis between distributed databases and blockchain solutions that benefits from the advantages of each.

BigchainDB allows for prosumers defined energy assets to be stored, while ensuring high scalability and a tamper-evident solution (changes cannot be prevented but will be detected). However, the major disadvantage is that it does not provide smart functionality over the registered values. According to the official website of the BigchainDB solution[25] in order to apply any logic or validation upon the registered values, another chain providing smart contracts capabilities must be considered that can fetch data from BigchainDB through different mechanisms (e.g. Oracles[26]) and apply custom logic through the execution of smart contracts on the fetched values.

The proposed novel hybrid solution aims to improve the costs and the scalability of an energy monitoring system, while benefiting at the same time of the major advantages that the blockchain brings in terms of immutability and Byzantine Fault Tolerance. Furthermore, it offers a scalable solution to the major challenge of integrating an energy monitoring system with the blockchain, since it is infeasible from a technical perspective and a cost perspective to process and store data at the rate provided by the smart energy meters sensors (for example 1 energy transaction at every 5 seconds).

4.1. BigchainDB for Energy Data

Even if the primary purpose of a blockchain is to provide storage capabilities (at least for transaction data), blockchains in comparison with traditional databases results in terrible performance and costs.

Distributed databases, for example, can exceed 1 million transaction per second, capacity of Petabytes and latencies with a fraction of a second as magnitude order. On the other hand, a blockchain can support just a few transactions per second and has no querying abilities.

BigchainDB is defined as a "*blockchain database*": it was designed to include the scalability and query-ability typical of databases with the decentralisation, immutability and authentication of blockchains.

Record validation is demanded to a federation of voting nodes and can be deployed in both public and private configurations.

BigchainDB

From a high-level point of view, BigchainDB combines the advantages of distributed databases to blockchains, focusing in particular on scalability.

From traditional enterprise distributed databases, BigchainDB inherits high throughput, low latency, high capacity, permissioning and a full featured NoSQL query language. From the scalability point of view, the addition of nodes increase throughput and capacity (figure 12 [27]).

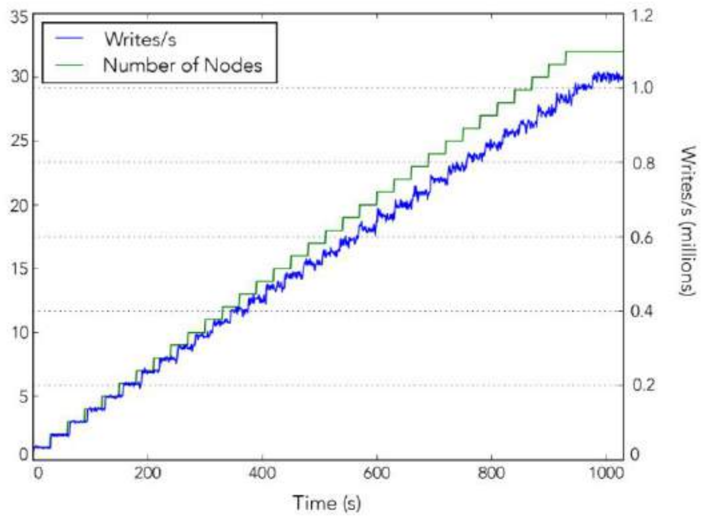


Figure 12: Time-series plot of throughput in terms of writes/s

BigchainDB also inherits from traditional blockchains the tamper-resistance, decentralised control and the creation and transfer of assets over network: any entity with asset-issuance permissions can issue an asset and any entity with asset-transfer permissions and the asset's private key may transfer the assets, avoiding compromised actors or malicious attackers to arbitrarily change data.

Use Cases

Thanks to its characteristics, BigchainDB supports different use cases. To name a few:

- Storage of *legally-binding contracts* in both human-readable and computable form;
- Secure high-volume *assets management;*

- Tamper-proof high-volume *physical assets tracking;*
- *Intellectual property licensing;*
- *Irrefutable evidence of electronic actions;*
- *Improved reliability database.*

In this project, we explore the time-stamping capabilities and asset-tracking.

Byzantine fault tolerance

BigchainDB 2.0, is developed to be Byzantine Fault Tolerant (BFT). Even if up to one third of the nodes "fail", the rest of the nodes can come to consensus on the next block. This means that there is no single point of failure and no single point of control.

To do so, BigchainDB integrates Tendermint[23] to manage the communication, replication, voting, and consensus logic (see Figure 13).

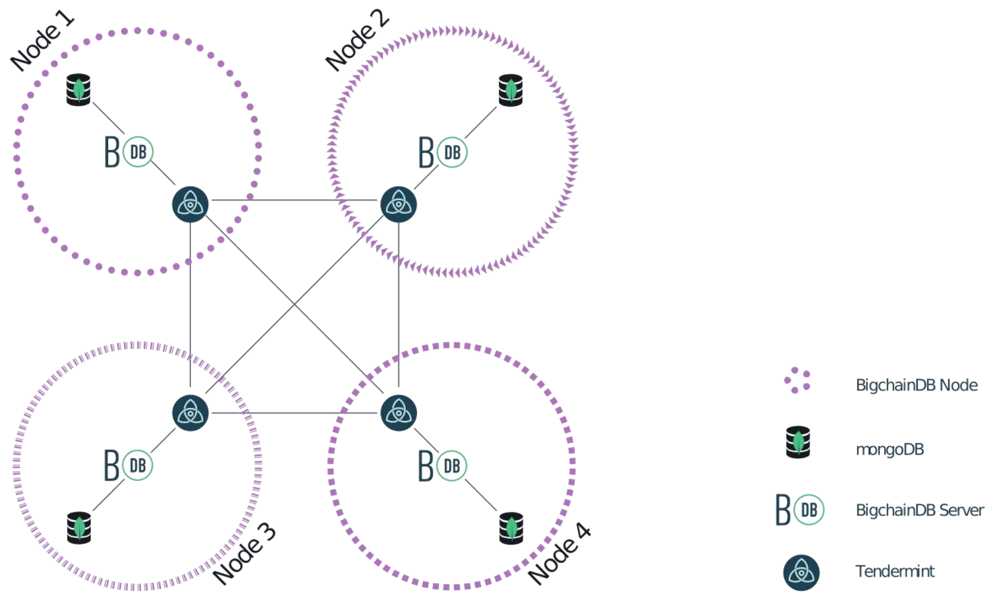


Figure 13: A four-node BigchainDB cluster (https://blog.bigchaindb.com/bigchaindb-2-0-is-byzantine-fault-tolerant-5ffdac96bc44)

Each BigchainDB node can be seen as a *state-machine.* While Tendermint is responsible for the connection of the nodes and the consensus (the "agreement" on the current state), BigchainDB manages the assets creation and ownership transfer, ensuring that only the owners of an asset can transfer (or "spend" it) and implementing the rules to *create* valid transaction and *verify* the validity of a transaction.

From the performance point of view, Tendermint provides a secure consensus protocol, being capable to achieve thousands of transactions per second[28] and a latency for block finalisation of 1 to 3 seconds[29], faster than other “traditional” blockchains (Bitcoin or Ethereum, for example) which successfully solve the consensus problem in public settings without a central authority.

The BigchainDB Transaction model

In summary, the most used transactions on a BigchainDB network are Create transactions and Transfer transactions. A Create transaction can be used to register divisible or indivisible of assets, along with arbitrary metadata. An asset can have zero, one, or several owners. The owners can specify conditions which must be satisfied by anyone wishing transfer the asset to new owners. BigchainDB verifies that the conditions have been satisfied as part of checking the validity of Transfer transactions, prevents double-spending of an asset, and grants the immutability of validated transactions[30].

More specifically, according to BigchainDB transaction specification v2.0, a transaction is composed by the following fields (see Table 2).

Table 2: BigchainDB transaction fields

Id	SHA3-256 hash of the transaction
Version	Indicates the transaction validation rules to be used when validating the transaction
Inputs	A list of transaction inputs
Outputs	A list of transaction outputs
Operation	Indicates the kind of transaction. The allowed values are: • "CREATE" • "TRANSFER" • "VALIDATOR_ELECTION" • "CHAIN_MIGRATION_ELECTION" • "VOTE"
Asset	The primary component of the transaction. In BigchainDB, transactions are used to register, issue, create or transfer assets
Metadata	User-provided transaction metadata

We will focus on the most basic kind of transaction allowed by BigchainDB: *Create* and *Transfer* transactions.

Create Transactions

A Create transaction can be used to register a single asset in BigchainDB. Each entity to be managed by BigchainDB it is called “asset”, even if it is not representing an actual asset.

A Create transaction can have one or more input and each output has an associated amount of “shares” of that output; this way, divisible assets are supported. Associated with each output, there is also an associated *condition* that must be met in order to spend the output, and a list of all the public keys associated with the conditions, representing the list of *owners,* a Create transaction must be signed by all the owners.

Transfer Transactions

A Transfer transaction can spend one or more outputs on other Create or Transfer transactions, associated with the same asset. Each transaction input must satisfy the condition on the output it is trying to spend. Just like a Create transaction, a Transfer transaction can have one or more outputs.

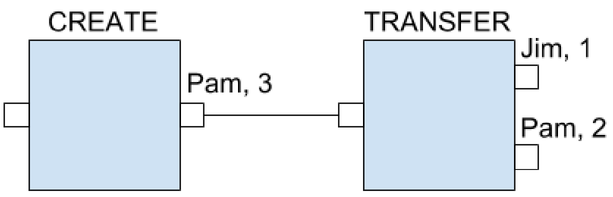


Figure 14: Transaction example

In the example representation in Figure 15, we see a Create transaction with one output: Pam controlling all the three shares of the created asset.

The Subsequent Transfer transaction, spends the first transactions output as an input and produces two outputs: Pam controlling 2 shares and Jim controlling the remaining one.

Since the transfer transaction is spending the output of the create transaction, the input on the transfer transaction must contain a valid signature from Pam to be validated.

Moreover, we refer to the output of the create transaction as a *spent transaction output*, and to the outputs of the transfer transaction as *unspent transaction outputs* (UTXOs).

Transaction Validity

BigchainDB validates transactions at a structural level and a graph level. A transaction is structurally valid if it satisfies some constraints on its internal structure, in isolation. For example, in a create transaction the asset must contain a *data* key, representing the asset payload, while in a transfer transaction the asset must contain an id key. Another check is related to the transaction inputs: while a Create transaction should have exactly one input, a Transfer transaction should have at least one input.

Each of the conditions in an output, must be one the following:

- An ED25519 signature condition, or
- A threshold condition.

Finally, transactions must always be signed and hashed, with the hash computed starting from an associative array key-value pairs for each transaction field.

A transaction is graph valid if it satisfies some constraints on the directed graph of transactions, where the graph nodes are transactions and the directed edges connect transaction inputs and outputs. One of the constraints is that the transaction doesn't *double spend*: the transaction cannot have two or more inputs spending the same output, and cannot spend an output that is spent by another transaction. The block order determines which one of two transactions spending the same output is the double spend. The second constraint is that the transaction cannot be a duplicate of another transaction. Other graph validity constraints include, for example, that for a Transfer transaction all the inputs fulfil the conditions on the spent outputs, or the sum of the amounts on the input must equal the total amount on the outputs.

4.1.1. Prototype Implementation

The proposed solution is based on the development of a new component, called *"smart meter aggregator"*, interacting with the FIWARE[31] -based IoT solution proposed for the field data layer of the architecture.

FIWARE is an open-ßsource IoT platform that offers a wide range of services, from the processing and storage of low-level data to the analysis and visualisation of high-level data. FIWARE has often been used as the IoT platform of choice for various European projects and will be the fundamental IoT platform in the eDREAM project.

FIWARE is an open middleware platform for the IoT, supported by the European Commission Union that offers the possibility of working with real-time data thanks to the Business API Ecosystem and interoperable protocols for the creation of new Internet services and applications.

Furthermore, open-source reference implementations of its components are freely available.

The FIWARE platform uses the FIWARE NGSI v2 as a reference data context model, an enhanced version of Open Mobile Alliance NGSI, which allows the adoption of FIWARE Harmonised Data Models for the creation of specific data models for the representation of the data of interest of the specific domain.

Figure 15 depicts the architecture of the FIWARE-enabled local node that will be used to retrieve data from the field. The data will be used also from the blockchain platform through a blockchain-enabled communication system to be stored in BigchainDB.

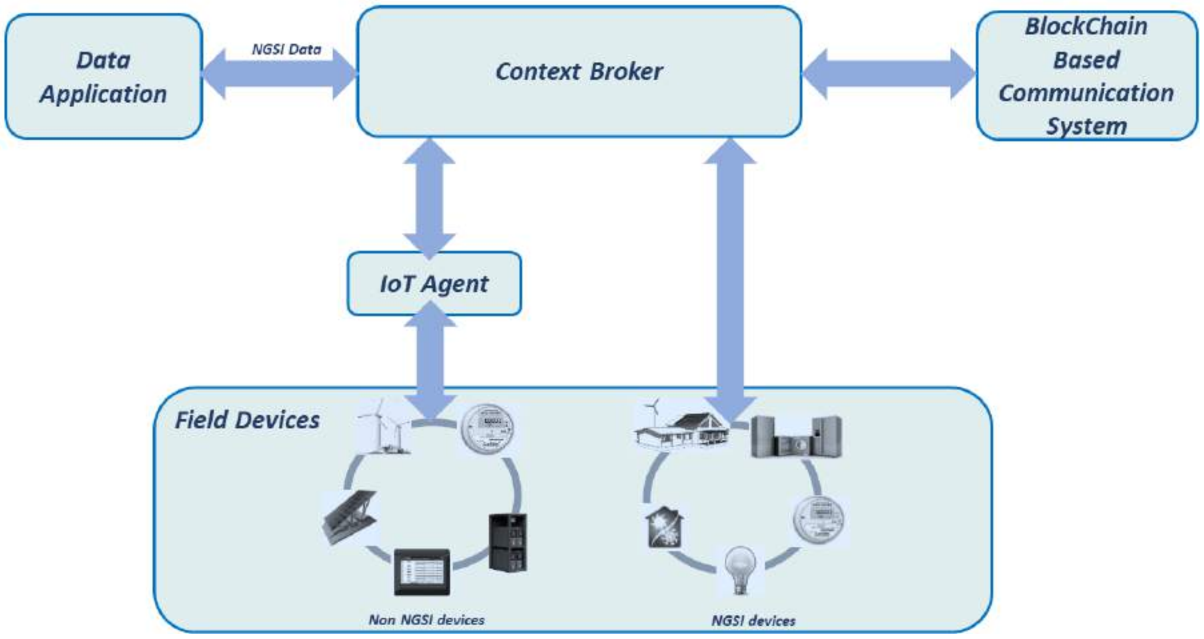


Figure 15: Local node architecture for data retrieving and processing

The data are modelled as NGSI entities based on JSON format and managed by FIWARE Context Broker. The Context Broker allows the management of context information on the base of the FIWARE NGSI open specifications, including queries (even based on geographic position) and subscriptions for asynchronous notifications on changes. Context Broker keeps the current value of all the different entities managed by the platform updated with the information provided by the different IoT agents, isolated from the specific IoT layer.

In this way, the data can be used by the components of the eDREAM platform not based on a decentralised platform, connecting to the Context Broker and querying/subscribing the entities using the NGSI v2 API, regardless of the source of such entities, local or remote. While, the blockchain-based communication mechanism will receive the information necessary to carry out DR flexibility trading and management, generating access and usage logs, and retrieving the required data.

In this context, the **smart meter aggregator** is responsible for the communication with the NGSI broker through which it will receive real time data from smart meters, will aggregate data on 15 minutes base and will store the resulting aggregated data using BigchainDB transactions.

```
{
  "subscriptionId": "5c8fbdc3f382b7ba2d5c660d",
  "originator": "localhost",
  "contextResponses": [
    {
      "contextElement": {
        "type": "SMX",
        "isPattern": "false",
        "id": "urn:ngsi-ld:SMX:BBB6099",
        "attributes": [
          {
            "name": "P",
            "type": "W",
            "value": "-84.891",
            "metadatas": [
              {
                "name": "TimeInstant",
                "type": "ISO8601",
                "value": "2019-03-18T16:13:45.275Z"
              }
            ]
          }
        ]
      },
      "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
      }
    }
  ]
}
```

Figure 16: Example of Smart Meter payload received from the subscribed Broker

Figure 17 shows the proposed architecture, highlighting the new aggregator component.

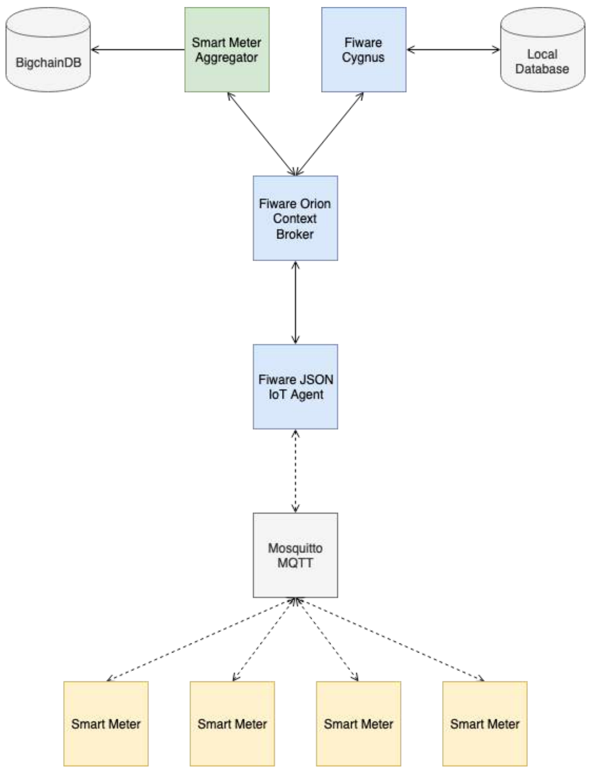


Figure 17: BigchainDB integration through smart meter aggregator

The smart-meter aggregator acts as a connector in charge of persisting NGSI data in configured BigChainDB storage, providing the means to manage the context history created as a data stream that can be injected into BigChainDB.

Therefore, the smart-meter aggregator is a Data Wrapper, specifically designed to cope with the NGSI data managed by the Context Broker to retrieve the data, translate them in the appropriate format and send them to BigchainDB through a specific communication channel.

The component will generate the key pairs representing the digital identities to track and the related smart meters as assets. The assets will be registered using Create transactions, so will be stored in BigchainDB with no possibility to delete them (immutability) once the transaction ends up in a validated block.

```
meter = {
    ...:     'data': {
    ...:         'meter': {
    ...:             'serial_number': 'eDream0001',
    ...:             'manufacturer': 'Smeter',
    ...:         },
    ...:     },
    ...: }
```

Figure 18: Smart Meter Digital Asset Definition

In order to track the measures provided by the smart meter, new transactions containing the updated values will be appended to the blockchain. Since this kind of updates does not imply a change of ownership, the transfer transaction will still have the previous owner as beneficiary, but a new metadata field, representing the updated context related to the smart meter.

In the Figure 19 is presented as an example of a transaction in which the meter values reported in the example of the NGIS payload in Figure 16 are contained as metadata.

In particular, recvTime represent the datetime of receiving, attrName the name of the attribute (P, Power), attrType is referring to the measurement unit (W, Watt) and attrValue represent the updated value.

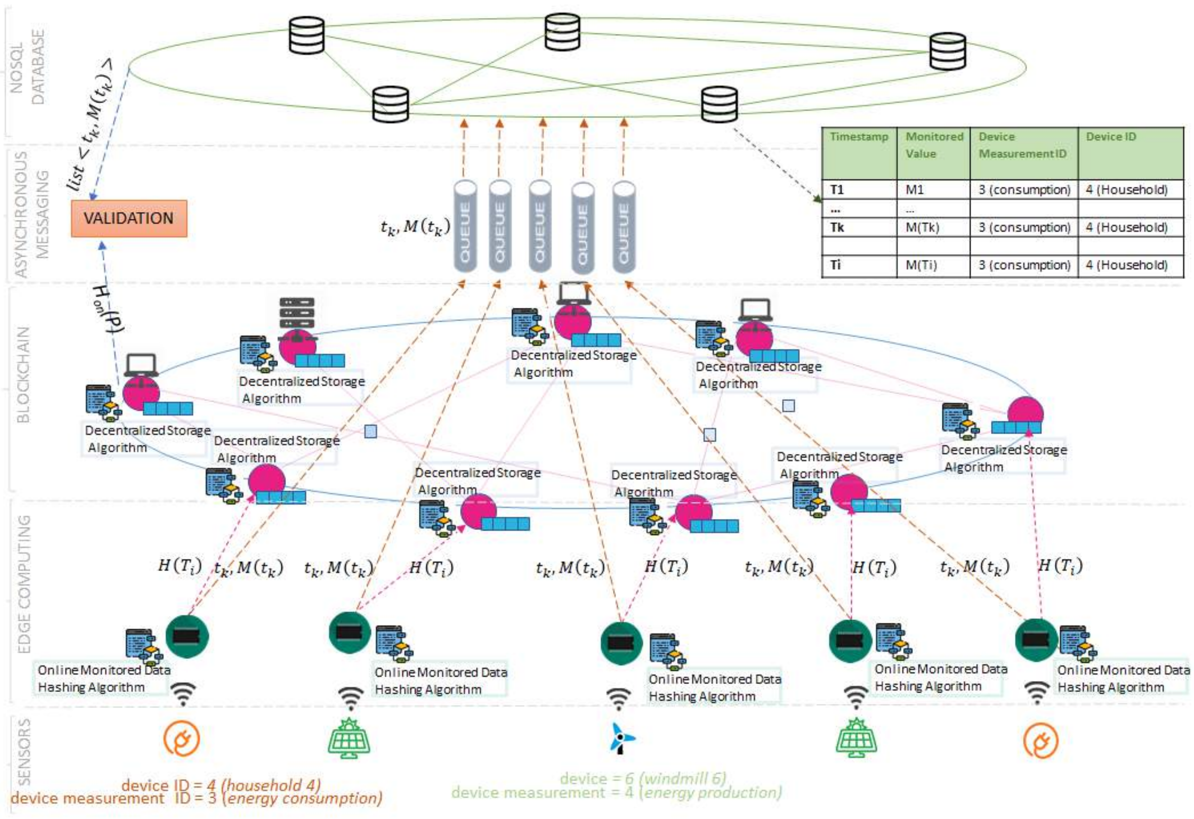
```
[
    {
        "metadata": {
    "recvTime" : ISODate("2019-03-18T16:13:45.275Z"),
    "attrName" : "P",
    "attrType" : "W",
    "attrValue" : "-84.891"
}
        "id":
"51ce82a14ca274d43e4992bbce41f6fdeb575f846e48e170a3bbb3b0cf8e2404"
    },
]
```

Figure 19: Smart Meter Transaction Metadata

4.2. eDREAM second-tier Energy Data Storage

We propose a hybrid solution in which real-time energy data that are collected from IoT metering devices in a time interval are hashed-linked back by the edge device in the order in which they had been sampled and stored off chain in a distributed database. An energy transaction is created and signed for the entire interval by the edge device publishing on the blockchain the average energy value registered and the associated hash fingerprint generated.

On blockchain, once the transaction is sealed, the average of the monitored values will be registered for further validation and business logic assessment, while the digital fingerprint will be hashed-back with the digital fingerprints of previous time intervals. Whenever historical data is requested, the digital fingerprint of the off-chain stored data will be computed and checked against the on-chain registered fingerprint. In case the hashes coincide, it is concluded that the off-chain real-time registered data have not been tampered with. Otherwise, further inquiries can be made in order to detect the exact interval where the data has been modified, thus leading to a tamper-evident system, where no changes can go unnoticed.



NOSQL DATABASE
ASYNCHRONOUS MESSAGING
BLOCKCHAIN
EDGE COMPUTING
SENSORS
$list < t_k, M(t_k) >$
VALIDATION
$H_{on}(P)$
$t_k, M(t_k)$
QUEUE
Decentralized Storage Algorithm
Online Monitored Data Hashing Algorithm
$H(T_i)$
device ID = 4 (household 4)
device measurement ID = 3 (energy consumption)
device = 6 (windmill 6)
device measurement = 4 (energy production)

Timestamp	Monitored Value	Device Measurement ID	Device ID
T1	M1	3 (consumption)	4 (Household)
...	...		
Tk	M(Tk)	3 (consumption)	4 (Household)
Ti	M(Ti)	3 (consumption)	4 (Household)

Figure 20: Proposed second-tier solution for energy data storage

Figure 20 presents the layered architecture of our proposed solution. There are several off-chain layers:

- the sensor device layer where data are fetched from a sensor associated with a device identifier and a measurement type;
- the edge-computing level that gathers data from the associated sensors, runs the hashing algorithms and implements the connections to store the raw values;
- an off-chain storage layer that provides scalable storage capabilities for the real time data received from the sensors;
- the asynchronous messaging layer implemented by a queue system that ensures data sequentiality and functionality in case of fluctuations in the monitoring data sampling rate.

Consider an energy meter device that provides monitored data over a time interval P. We split interval P in N smaller disjoint intervals T_i, where each interval T_i is delimited by a start time and an end time $T_i = (T_S^i, T_E^i]$:

$$P = \cup_{i=1}^{N} T_i \; and \; \cap_{i=1}^{N} T_i = \emptyset$$

In each time interval T_i there can be L-ordered discrete timestamps in which sensors are sampling new data:

$$T_i = \{t_k | t_k < t_{k+1}, \forall k = 0..L-1\}$$

We denote the monitored value sent by a sensor at a timestamp t_k as $M(t_k)$.

From the sensor level, the monitored data value $M(t_k)$ is sent to the edge device at timestamp t_k. The edge device will forward this information directly to the asynchronous messaging system that stores the new sample data in the off-chain storage system in a sequential manner. At edge device level we had defined an online hashing algorithm that for each interval T_i will compute the digital fingerprint of all the monitored data received from the sensor at each discrete timestamp from that interval. At the end of intervalT_i, the edge device will sign and register the digital fingerprint of the monitored data on blockchain, thus ensuring an immutable log of this value. The blockchain will implement a decentralised storage algorithm that is responsible for storing and compute a hash of period P, composed of all the digital fingerprints received for each interval T_i.

4.2.1 Digital Fingerprinting Energy Data

We had defined an online algorithm (see Figure 21) that aims at computing the hash of the raw monitored energy values as they are sampled for each discrete timestamp over an interval T_i. Thus the inputs of the algorithm are the monitored value at each timestamp $t_k$and a seed hash ($HASH_{SEED}$) that is uniquely generated for each prosumer, providing the genesis hash value for the linked back data structure. The outputs expected to be returned by the algorithm are the hash representing the digital fingerprint of all the monitored energy data values over the period T_i($H(T^i_E)$) and the average energy value over the entire interval.

The first step of the algorithm at line 1 is to initialise the hash of the period T_i with the value of the seed. For each new energy monitored datum $M(t_k)$ at timestamp t_k in the interval T_i (line 2), the hash $H(t_k)$ will be computed (line 3) by hashing the current value, with the timestamp and the previous hash registered at t_{k-1} (for time t_1, the previous hash is $H(T_S^i)$ – the seed hash). At the same time the sum of energy it is calculated (line 4) so that at the end of the period T_i (line 6) the average will be computed and returned (line 7) together with the fingerprint $H(T_E^i)$, which is the hashed-linked back values computed for each time t_k over the entire period T_i.

```
INPUT: $T_i\ M(t_k), HASH_{SEED}$
OUTPUT: $Energy_{Transaction} < H(T_E^i), AVG_{Energy}(T_i) >$
Begin   1. $H(T_S^i) = HASH_{SEED}$
        2. $foreach(M(t_k), T_S^i < t_k \leq T_E^i)\ do$
            3. $H(t_k) = HASH(M(t_k), t_k, H(t_{k-1}))$
            4. $SUM_{Energy}(t_k) = SUM_{Energy}(t_{k-1}) + M(t_k)\ , count++$
        5. $endforeach$
        6. $AVG_{Energy}(T_i) = \frac{SUM_{Energy}(t_k)}{count}$
        7. $return\ Energy_{Transaction} < H(T_E^i), AVG_{Energy}(T_i) >$
End
```

Figure 21: Hashing sensor sampled energy data on the edge

The final hash obtained at the end of the interval 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. Since at each point in time the only value required is the previous hash (as depicted in Figure 22), this structure can be easily used by devices with very low hardware specifications, as opposed to a Merkle-tree structure, where previous data must be kept in memory in order to compute pair-wise hashes of data.

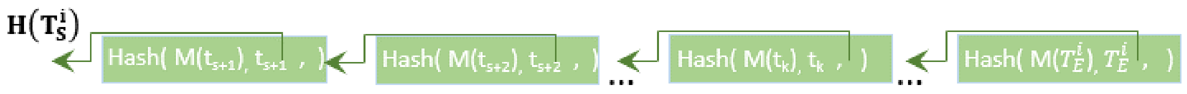

$H(T_S^i)$
Hash($M(t_{s+1}), t_{s+1}$,)
Hash($M(t_{s+2}), t_{s+2}$,)
...
Hash($M(t_k), t_k$,)
...
Hash($M(T_E^i), T_E^i$,)

Figure 22: Hashed linked back monitored values

During time interval T_i each monitored energy value is sent to a distributed database which can easily handle large amounts of data (NoSQL Database). However, at the end of the time interval T_i, only the digital fingerprint $H_{t_k}$is registered and saved on chain together with the average of the monitored energy values AVG_{Energy}. From this point, the immutability of the fingerprint is ensured by the blockchain, furthermore the immutability of the actual data $M(t_k), t_k$is ensured in the sense that any tampering will not go unnoticed, thus rendering the off-chain data tamper-evident and the on-chain data tamper-proof.

For fingerprinting and storing the energy transactions, a second hashed-linked back structure is defined and used. From the transactional throughput point of view, depending on the chain specifications and on the number of devices installed, the granularity of interval T_i should be set such that the chain is able to efficiently process the entire number of transactions received from the network. From the storage point of view, since during the entire period P, there will be N digital fingerprints specific for each interval T_i, the storage space would increase proportionally with N. Thus, we define a second hashed-linked back structure to keep by hash-linking back all the digital fingerprints $H(T_i)$ of all intervals T_i. The input of the algorithm depicted in Figure 23 consists of the sampling number of intervals, N, that the period P is split in, generated energy transaction in the current interval, and the seed hash $HASH_{SEED}$, required for initialising the hash of the period P (line 1). For each new interval $i$registered before the end of the period (line 2 and 3), the hash H^{T_i} is computed as the hash of the current interval energy transaction hashed together with the hash of the previous interval $H^{T_{i-1}}$ (for interval T_1 the previous hash is H^{T_0} – the seed hash). The output of the algorithm is the hash value generated for the entire interval P.

```
INPUT: N, $Energy_{Transaction} < H(T_i), AVG_{Energy}(T_i) >, HASH_{SEED}$
OUTPUT: $H_P$
Begin
        1. $H^{T_0} = HASH_{SEED}$
        2. $foreach(0 < i \leq N)do$
        3. $H^{T_i} = H\big(Energy_{Transaction} < H(T_i), AVG_{Energy}(T_i) >, H^{T_{i-1}}\big)$
        4. $endforeach$
        5. $return H_P$
End
```

Figure 23: On-chain hashed linked back hashing algorithm

The advantage of our proposed approach is that by the end of period P, one fixed-length hash will depict the digital fingerprint of energy data acquired during the entire period regardless of the number of registered intervals N. Each hash corresponding to a time interval is mined in a block as depicted in Figure 24 and validated by the network nodes rendering it immutable once the enough blocks are mined on top of it.

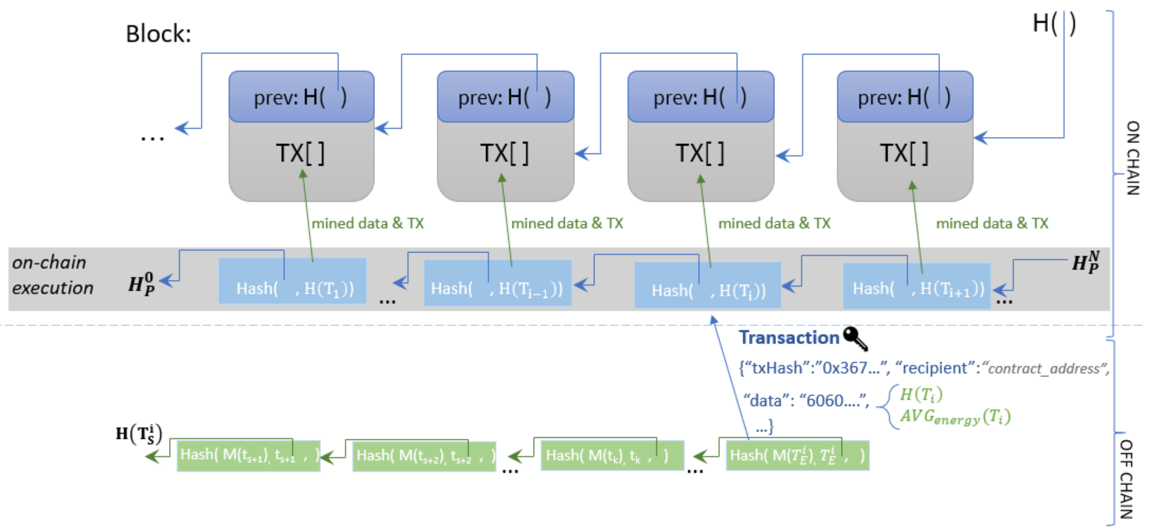
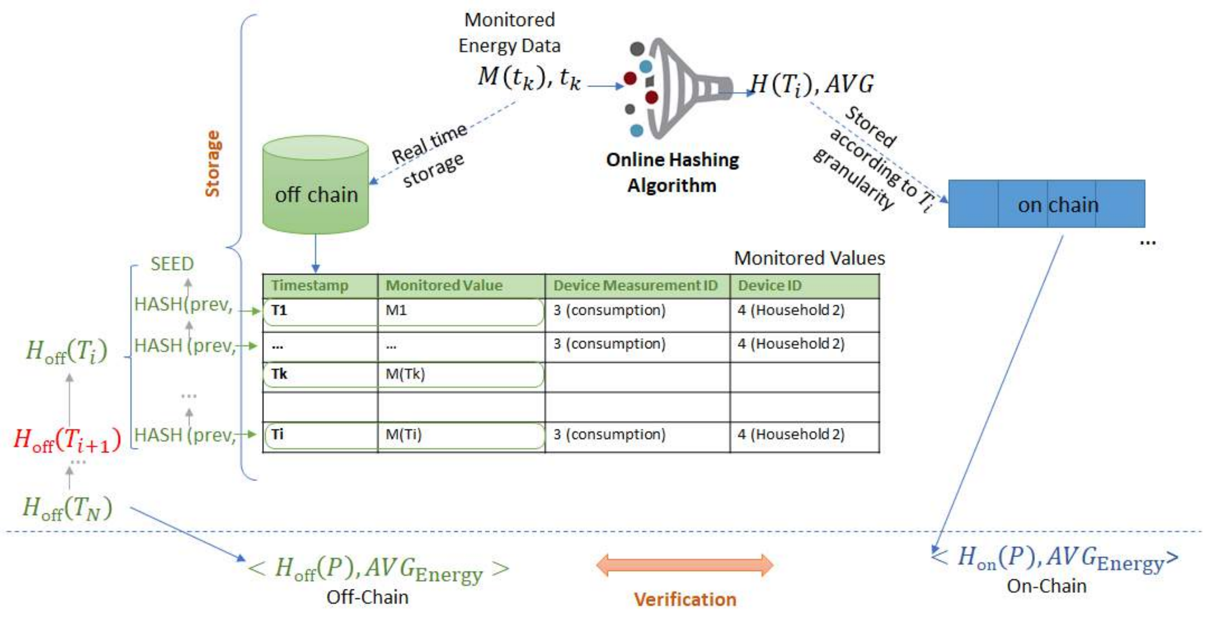


Figure 24: Hashed linked back on-chain energy transactions

When historical monitored energy data values need to be retrieved for a period P, all the stored values are validated against the data fingerprints registered on chain. All the values that are fetched from the database are passed through the algorithms presented in an offline manner, resulting in the digital fingerprint of the data, H_P^{off} over the time period P. Since the chain already keeps for each period P a tamper-proof fingerprint, H_P^{on}, considering that the data stored off chain has not been tampered with, H_P^{off} will equal H_P^{on} as depicted in Figure 25.

Any change that is made to monitored energy values or the associated timestamps will be detected by the algorithm 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 T_i that has been tampered with. Thus, a trade-off must be considered for choosing the granularity of T_i for which the digital fingerprint is registered on chain. A high sampling rate is beneficial to narrow as much as possible the subset of monitored data that contains the unreliable tampered value, but at the same time could affect the ledger's scalability since the blockchain can handle a limited throughput.



Timestamp	Monitored Value	Device Measurement ID	Device ID
T1	M1	3 (consumption)	4 (Household 2)
...	...	3 (consumption)	4 (Household 2)
Tk	M(Tk)		
Ti	M(Ti)	3 (consumption)	4 (Household 2)

Figure 25: Off-chain Validation Mechanism

4.2.2 Prototype Implementation and Evaluation

We have implemented the above presented scalable energy storage solution using Ethereum and Solidity [32] for the on-chain components and Cassandra[33] and RabbitMQ[34] for the off-chain ones. For evaluation we have considered a setup in which P is 24 hours. We have set the T_i interval granularity to an hour and we have considered that the energy data is sent by the smart meters every minute:

$$T_i = \{t_k | t_k < t_{k+1}, \forall k = 1..59\}$$

Finally, the energy transactions are published on chain at the end of each hour.

The on-chain implementation is enforced by a smart contract that is responsible to process all the digital fingerprints received from one sensor.

Table 3 presents the obtained results in terms of throughput and response time of the eDREAM second-tier solution, the response time corresponds to the time elapsed from the moment the edge device publishes the energy value, to the point the queue acknowledges the request.

Table 3: evaluation of the On-chain Off-chain System throughput

	eDREAM 2nd Tier Energy Data Storage
Response Time	0.002 s
Throughput	50 000 tx/sec

The eDREAM second-tier energy data storage, allows a throughput of up to 50000 transactions per second. The results show that the hybrid system that combines on-chain hashes of data and off-chain real-time recording of monitored data, provides a great improvement in terms of scalability while at the same time ensuring the tamper proofing of data.

Figure 26 shows the relation between the average monitoring sampling rate, t_k, and the number of energy transactions that can be process by the queuing system. As it can be seen for up to 2000 prosumers, a set average monitoring sampling rate of 0.2 seconds is feasible and does not lead to congestion of the queue messaging system.

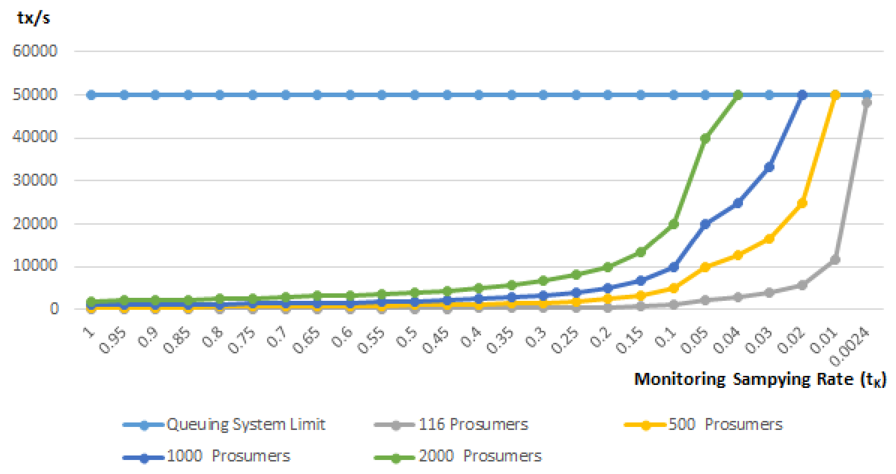


Figure 26: Off-chain monitoring sampling rate and the number of prosumers

Figure 27 shows the relation between the minimum energy transaction time interval considered T_i and the number of prosumers in a microgrid. According to the results in case of a microgrid of approximately 2000 prosumers, the minimum interval T_i for efficiently register energy transactions is of approximately 9 minutes.

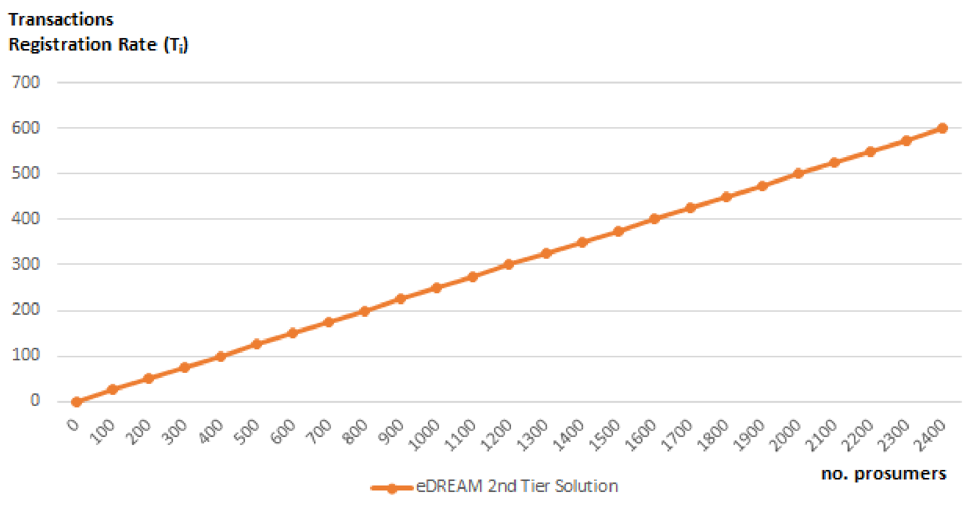


Figure 27: On-chain energy transactions registration rate and the number of prosumers

5. Conclusion

In this document, we described two different solutions exploring the usage of the blockchain in the energy sector to store the measures provided by smart meters.

The first one is a novel hybrid solution built on top of distributed databases and Ethereum smart contracts, while the second one is based on the integration of BigchainDB with the existing IoT field data aggregation layer. Both the solutions aim to overcome the scalability and performance limits inherent "by design" to blockchains, maintaining at the same time the advantages that this technology provides. While the solution based on BigchainDB can be useful if there is a need for a plug-and-play solution in a storage-only context, the first one better fits use cases in which there is the need to include more advanced capabilities, supporting the execution of smart contracts natively and in the same blockchain.

References

1 https://media.voog.com/0000/0042/0957/files/SOFIE_D2.1-State_of_the_Art_Report.pdf
2 Bitcoin: A Peer-to-Peer Electronic Cash System
3 https://www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
4 http://gavwood.com/paper.pdf
5 https://www.pwc.com/gx/en/industries/assets/pwc-blockchain-opportunity-for-energy-producers-and-consumers.pdf
6 http://www.ourenergypolicy.org/wp-content/uploads/2018/07/Discussion_Paper_Livingston_et_al_Blockchain_OR_0.pdf
7 https://www.blockchaintechnology-news.com/2018/04/10/chile-to-put-energy-data-on-ethereums-blockchain/
8 https://www.windpowerengineering.com/business-news-projects/microgrid-investment-in-blockchain-future-of-energy-systems-says-globaldata/
9 Bosco, Croce, Raveduto: Blockchain technology for financial services facilitation in RES investments
10 M. Pustišek, A. Kos and U. Sedlar, "Blockchain-based Autonomous Selection of Electric Vehicle Charging Station," 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, 2016, pp. 217-222.
11 https://www.sciencedirect.com/science/article/pii/S1364032118307184
12 MIGUEL CASTRO, BARBARA LISKOV: Practical Byzantine Fault Tolerance and Proactive Recovery
13 https://interledger.org/rfcs/0027-interledger-protocol-4/draft-6.html
14 https://eips.ethereum.org/EIPS/eip-1283
15 https://www.ethgasstation.info/
16 https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
17 https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
18 Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie and Massimo Bertoncini, Blockchain-based Decentralized Management of Demand Response Programs in Smart Energy Grids, Sensors 2018, 18(1), 162.
19 Andreas M. Antonopoulos, Gavin Wood, “Mastering Ethereum” , https://ethereumbook.info/
20 Andreas M. Antonopoulos , “Mastering Bitcoin: Unlocking Digital Cryptocurrencies” , ISBN 1449374042. 2015
21 Tudor Cioara, Ionut Anghel, Claudia Pop, Massimo Bertoncini, Vincenzo Croce, Dimosthenis Ioannidis, Konstantinos Votis, Dimitrios Tzovaras, Luigi D'Oriano, Enabling New Tehnologies for Demand Response Descentralized Validation using Blockchain, IEEE 18th International Conference on Environment and Electrical Engineering and 2nd Industrial and Commercial Power Systems Europe 2018.
22 McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., ... & Granzotto, A. (2016). BigchainDB: a scalable blockchain database. white paper, BigChainDB.
23 Buchman, Ethan. Tendermint: Byzantine fault tolerance in the age of blockchains. Diss. 2016.
24 MongoDB , Available online at https://www.mongodb.com/
25 BigchainDB smart contracts , Available online at http://docs.bigchaindb.com/en/latest/smart-contracts.html
26 Oraclize, Available online at https://docs.oraclize.it/
27 https://coinreport.net/wp-content/uploads/2016/02/BigchainDB-Primer-20160210.pdf
28 https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf
29 https://blog.cosmos.network/consensus-compare-casper-vs-tendermint-6df154ad56ae
30 https://docs.bigchaindb.com/en/latest/assets.html#how-bigchaindb-is-good-for-asset-registrations-transfers
31 https://www.fiware.org/
32 Solidity, Available online at https://solidity.readthedocs.io/en/latest/
33 Cassandra, Available online at http://cassandra.apache.org/
34 Rabbitmq, Available online at https://www.rabbitmq.com/