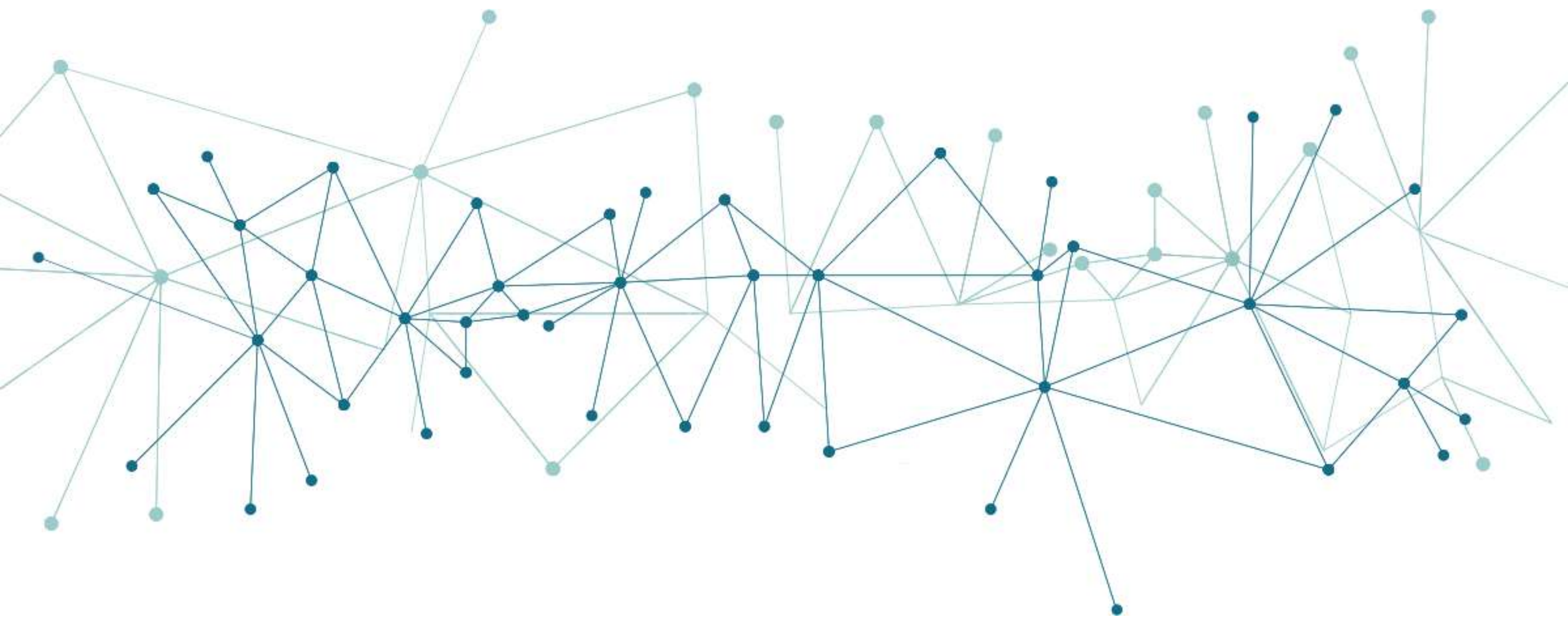
The eDREAM project is co-founded by the EU's Horizon 2020 innovation programme under grant agreement No 774478

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

DELIVERABLE: D5.3 Consensus based techniques for DR validation and financial settlement

Authors: Vincenzo Croce, Giuseppe Raveduto, Matteo Verber, Tudor Cioara, Claudia Pop, Ioan Salomie



Imprint

Consensus based techniques for DR validation and financial settlement, April 2020

Contractual Date of Delivery to the EC:	30.04.2020
Actual Date of Delivery to the EC:	30.04.2020
Author(s):	Vincenzo Croce (ENG), Giuseppe Raveduto (ENG), Matteo Verber (ENG), Tudor Cioara (TUC), Claudia Pop (TUC), Ioan Salomie (TUC)
Participant(s):	ENG, TUC, E@W, ASM
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:	T5.3 – Closed Loop DR Verification for near real time DR Financial Settlement
Confidentiality:	public
Version:	0.99

Legal Disclaimer

Copyright

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

API	Application Programming Interface
BFT	Byzantine Fault Tolerance
BA	Byzantine Agreement
DEP	Distributed Energy Prosumer
DR	Demand Response
DSO	Distributed System Operator
eDREAM	enabling new Demand Response Advanced, Market oriented and secure technologies, solutions and business models
GBR	Gradient Boosted Tree Regression
GHOST	Greedy Heaviest Observed Subtree
IoT	Internet of Things
JSON	JavaScript Object Notation
LSTM	Long-Short Term Memory-based Neural Networks
MAPE	Mean absolute percentage error
MLP	Multi-Layer Perceptron
P2P	Peer to Peer
PoA	Proof-of-Authority
PoS	Proof of Stake
REST	Representational State Transfer
RPC	Remote procedure call
S&PN	Salt-and-pepper noise
SVR	Support Vector Regression
URI	Uniform Resource Identifier
WP	Work Package

Executive Summary

In this deliverable we present the work done in the direction of defining blockchain-based techniques for Demand Response (DR) programs validation and their energy and financial settlement. We have defined a consensus-based validation of prosumers peers energy transactions that leverages on a quorum based approach using stakes, we have implemented a Proof of Authority and smart contracts solution for DR programs statement both financial and energy imbalances and we have defined and analysed dynamic model for incentivisation / penalization of prosumers in decentralized DR programs.

In regards with the consensus-based validation, we have started by conducting a state-of-the-art review of existing consensus-based validation approaches pinpointing their advantages and disadvantages to the case of decentralized energy-efficient DR programs management in smart energy grids and we have defined a quorum and stake based consensus solution for prosumers energy transactions validation over the eDREAM 2nd tier distributed ledger for energy data introduced in D5.1. The defined consensus is leveraging on an adapted version of the two-phase commit and validation stakes and on an overlay peer-to-peer network of nodes allowing them to agree on the validity of each energy transaction generated by each peer (based on its monitored energy data) before is submission on the chain. This allows us to minimize the risk of misbehaviour due to malicious data provided by energy meters or altered by attackers on the communication network. For flexibility actual delivery and financial settlement in DR programs we have developed a mechanism that is based on smart contracts and PoA as the algorithm for blocks validation. Smart contracts implemented and used for determining the actual flexibility delivery after the DR flexibility order had been agreed upon and issued to the enrolled prosumers. They allow the measuring in the prosumers or aggregators rate of success or failure to deliver the promised flexibility in near-real-time, registering potential imbalances in terms of deviations from flexibility order signals and then conducting the financial settlement using dynamic incentivization / penalization.

For the financial settlement of decentralized DR programs, we defined a model to calculate a coefficient to be applied to an incentive value to obtain the actual incentive or penalty payment to be applied to prosumers in a specified timeframe, based on its flexibility delivery. The model takes as input the flexibility order in terms of expected profile request the associated tolerance range, the normal behaviour of the prosumer in terms of its energy baseline and the actual profile followed by the prosumer (i.e. actual flexibility delivery). The model allows to pay an incentive to the prosumer every time is energy profile follows the flexibility order signal inside the tolerance range and to claim for compensation when its profile is out from this range.

The evaluation results obtained on a test case scenario implemented using data provided by Kiwi Pilot site are showing the effectiveness of our proposed solution for validating energy transactions, detecting the imbalances in terms of deviations between submitted energy plans and associated flexibility order request of the aggregator and the actual delivery energy of prosumers from its portfolio. The dynamic model for decentralized programs' financial settlement is effective in dynamically calculating the prosumers' penalties / incentivises. In our evaluation each time the imbalances passed the threshold tolerance range; the prosumer was penalized for the deviation in the amount of delivered flexibility versus the expected one otherwise it was rewarded.

1 Introduction

1.1 Purpose

This report provides an overview of the work carried out in Task 5.3, analysing the applicability of consensus-based techniques for DR validation and financial settlement.

We have worked on the development of blockchain based techniques for validating the energy transactions of individual prosumer peers and for settlement of imbalances and flexibility deviations in DR programs and for financial statement of DR using innovative penalization/incentivisation schemes.

The use of such blockchain enabled techniques is improving the reliability of the DR programs implementation bringing the energy imbalances and financial statement in flexibility delivery closer to the real time.

1.2 Relation to other activities

WP5 uses some of the output of WP2 in terms of requirements and use-cases as well as the outputs of WP3 in terms of energy demand/generation forecasting and prosumers' baseline assessment. In particular T5.3 builds upon the 2nd tier distributed ledger for storing energy transactions (output of T5.1) and on the smart contracts-based platform of decentralized DR management through blockchain-driven flexibility services and peer to peer trading (output of T5.2).

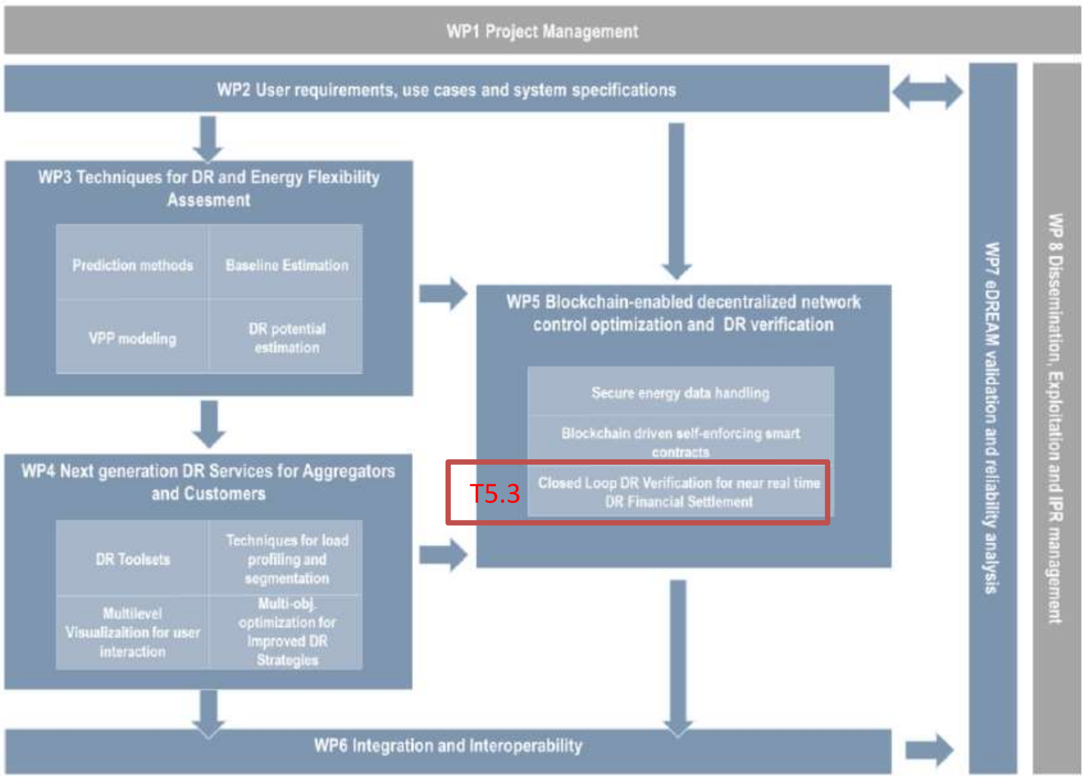

WP1 Project Management
WP2 User requirements, use cases and system specifications
WP3 Techniques for DR and Energy Flexibility Assesment
Prediction methods
Baseline Estimation
VPP modeling
DR potential estimation
WP5 Blockchain-enabled decentralized network control optimization and DR verification
Secure energy data handling
Blockchain driven self-enforcing smart contracts
T5.3
Closed Loop DR Verification for near real time DR Financial Settlement
WP4 Next generation DR Services for Aggregators and Customers
DR Toolsets
Techniques for load profiling and segmentation
Multilevel Visualization for user interaction
Multi-obj. optimization for improved DR Strategies
WP6 Integration and Interoperability
WP7 eDREAM validation and reliability analysis
WP 8 Dissemination, Exploitation and IPR management

Figure 1. eDREAM PERT chart showing WP5 in relation to other work packages

1.3 Structure of the document

The remainder of the document is organized as follows:

- Section 2 presents the quorum and stake based consensus technique defined for validating the prosumers peers energy transactions and the PoA and smart contracts implemented for settlement of DR programs;
- Section 3 describes and analyses a dynamic model for calculating prosumers' incentives and eventual penalties for not complying to the flexibility order in decentralized DR programs;;
- Section 4 concludes the report.

2 Blockchain based DR Validation and Settlement

In this chapter we will report the blockchain based techniques for validating the energy transactions of individual prosumer peers and for settlement of imbalances and flexibility deviations in DR programs.

2.1 Consensus based Validation of Peers Energy Transactions

We start by conducting a state-of-the-art review of existing consensus-based validation approaches pinpointing their advantages and disadvantages to the case of decentralized efficient management of smart energy grids and we define a consensus-based energy transactions validation solution over the eDREAM 2nd tier distributed ledger for energy data introduced in D5.1.

2.1.1 Technology Review

In general, distributed systems consist of a set of processing nodes interconnected by a communication network that exchange messages [3]. With the adoption of IoT monitoring and control devices, deployment of decentralized RES and overlay data exchange infrastructure the smart energy grid had become lately o good example of such a large-scale distributed system. In this case the representative nodes of the systems will be the energy prosumers that may produce energy, consume energy or both.

However, with the growing deployment of small-scale prosumers such as combined heat and power plants, distributed energy generation units, electric cars, and batteries, the architecture of energy grid systems needs to be decentralized to overcome the increasing complexity and new challenges of energy management operations. Thus, the energy systems are transitioning towards more diverse cooperative and decentralized models where energy management may effectively take place by coordinating in a decentralized fashion the small-scale prosumers to offer valuable energy services. A fundamental problem in making these decentralized models successful is the to ensure that the prosumer nodes of such an energy network can agree on the result of a distributed computation and network state.

With the advent of blockchain technology lot of attention is being put on the consensus algorithms for solving this problem. There are two important factors that should be consider when building such algorithms that that influence their performance and reliability: i) the message delivering bound δ and ii) the potential number of faulty or corrupt nodes in the system . If the message delivery bound δ does not exist, the distributed system is classified as asynchronous [2]. If the bound δ exists but is not known, the system is classified as partially synchronous, while when the bound δ is known, the system is synchronous [7].

Other influential factor is that it is impossible for a decentralized system to be at the same time consistent, available and partition-resilient, thus the consensus algorithms should make a trade-off on availability and consistency of the results since the tolerance to partitioning is mandatory. Thus, in the blockchain based systems, the distributed state consistency is achieved over time, by mining blocks only on longest chains and removing forks. This makes the blockchain based systems eventually consistent [46]. Furthermore, in the case of distributed system with asynchronous communication and at least one faulty process ($\delta \to \infty \ and \ \geq 1$), no consensus algorithm can be developed with the property of guaranteed termination. Of course, this is not reasonable for decentralized smart energy grid systems so synchronous communication is implied for this case [8]. In [7] it is shown that consensus can be achieved in networks with partially synchronous communication given by an existing but unknown bound for message latency, even if up to $\leq$ of the nodes are faulty. Furthermore, is the bound for message latency is known, the system can reach consensus even if more that 50% of the nodes are faulty. In the case of synchronous distributed systems, exist several solutions for the distributed consensus problem (i.e. also known as the Byzantine Generals problem [9]) such as

Draper's FTMP [10] Honeywell's MMFCS [11], SIFT **Errore. L'origine riferimento non è stata trovata.**, and other more advanced ones derived from the Practical Byzantine Fault Tolerance (PBFT) algorithm [13].

Figure 2 depicts a taxonomy of the Consensus Algorithms and shows two main classes of algorithms: Non-Byzantine fault tolerant algorithms and Byzantine Fault tolerant algorithms. The difference is given by the ability of algorithms to reach agreement, integrity and termination in case of existing faulty or attacker nodes in the distributed system, thus Non-Byzantine Fault Tolerant Protocols rely on the assumption that all the nodes are correct ($= 0$) while the Byzantine Fault Tolerant Protocols can handle situations when the number of faulty nodes is as high as half of the total number of nodes ($\sim 50\% *$)

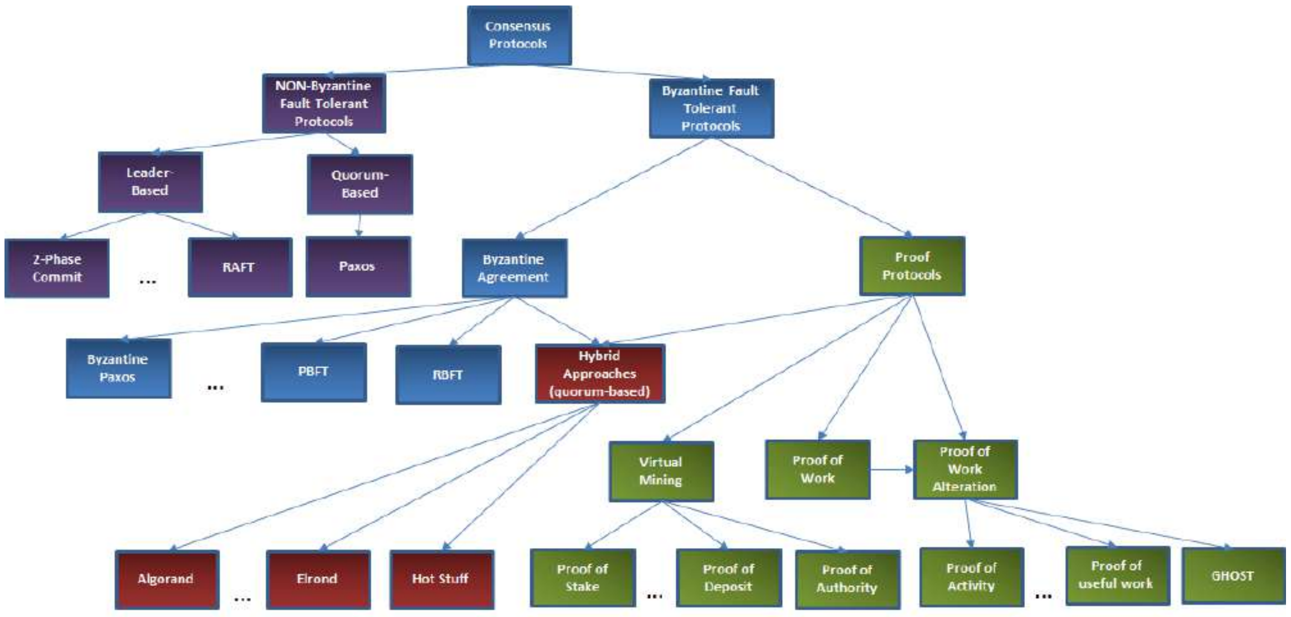


Figure 2. Taxonomy of consensus algorithms for distributed systems

The Non-Byzantine Fault Tolerant Protocols are leader-based, such as 2-Phase Commit [3] and RAFT [23], where a leader election algorithms is used to select a leader that will centralize the votes and commit the transaction. Furthermore, Non-Byzantine Fault Tolerant Protocols are quorum based, where a subset of the processes is selected to validate the transaction using a voting scheme. A well-known algorithm of this class is the Paxos [24] that solves consensus in a network of processes that may fail but are correct (there exist no faulty processes that may lie).

The rest of our review will be focused on consensus algorithms that can handle faulty or incorrect nodes into the network since this is highly desirable feature for energy grid management. The Byzantine fault tolerant protocols aim to assure that the peers are be able to agree on a system valid state even in case some of them feature faulty or malicious behaviours [1]. The idea is to find a model and protocol for a network of message passing processes, some of them being faulty, such that a general agreed state can be extracted from the distributed system. The Byzantine Fault Tolerant Protocols are classified as traditional Byzantine Agreement (BA) protocols and Proof-of-X protocols [22]. The Byzantine Agreement protocols use a quorum-based mechanism where a subset of the nodes must agree on a transaction validity. Examples of such algorithms are the Byzantine Paxos algorithm [25], the Practical byzantine fault tolerance algorithm [13] and variants that address the robustness such as Ardvark [14] and RBFT [15] or that address the performance problems of PBFT, such as Q/U [16], HQ [17], Zyzzyva [18] and ABsTRACTs [19]. An interesting byzantine fault tolerant distributed commit protocol is proposed in [3], where the authors enhance the classical 2-Phase Commit protocol by replicating the coordinator to successfully terminate when the coordinator failed and by building a quorum of coordinators to validate transactions and identify malicious participants.

The Proof Protocols are used by most of the public DLT systems [27], [28] for supporting the consensus mechanisms in order to ensure the consistency of the ledger state across the network nodes. The Proof Protocols have defined two categories of nodes: Provers and Verifiers, where the Prover who may have unlimited resources needed to convince Verifier nodes with limited resources, about the truthfulness of a statement. As opposed to traditional Byzantine Agreement (BA) protocols, that use a quorum of participants to validate a transaction by voting, the Proof-of-Work (PoW) algorithm validates a transaction (or a set of transactions) by solving a computationally intensive problem by a Prover that require a lot of physical resources and make infeasible for an attacker to cast an erroneous vote. The time needed by the prover to solve the computationally intensive problem gives the mining rate and directly influence the throughput (number of transactions) and the network scalability.

From the initial implementation of PoW in Bitcoin [20], where one block was generated every 10 minutes, PoW variations have been proposed aimed at improving the mining rate in order to obtain a higher throughput of transactions per second. The Greedy Heaviest Observed Subtree (GHOST) protocol [12] proposed by Ethereum increases the mining rate from 1 block per 10 minutes to 1 block per ~15 seconds. In order to avoid the potential problems that may arise due to delayed propagation of blocks, GHOST uses references to orphan blocks or Uncles (valid blocks that were not accepted in the main chain due to network delays) in order to increase the weight of the longest chain. In this sense, each new block can contain references to previous Uncles and for each of the referenced uncles, the miner will receive a small incentive, consequently, the miner of the uncle will also be rewarded when a new block makes a reference to it. This mechanism discourages the faulty miners to mine on forked chains and from perusing long-range attacks. Other variations of PoW have been considered in order to impose some restrictions on the hardware devices used for mining by encouraging the implementation of ASIC (Application Specific Integrated Circuits) resistant algorithms for hashing. This came as a result of the Bitcoin's early years when hardware companies started to profit from the popularity of blockchain solutions by developing ASICs in order to increase the hash rate of the computing nodes. However, one such circuit may cost around 3000 dollars [29], which makes it unprofitable for a simple user to invest in such hardware and gives more power and control to large companies and to the manufacturer. In order to avoid this problem, the next generation of DLT solutions researched and applied new hash functions that are ASIC resistant. ASIC resistant algorithms try to shift their strategy from CPU intensive algorithms to memory intensive algorithms, called Memory hard puzzles because the performance of processors has increased over time at an exponential rate, as opposed to the memory which has known a more linear increase. The purpose of these algorithms is to design a method that requires large amounts of data to be stored, that cannot be efficiently parallelized. Scrypt [30] is one of the first ASIC resistant algorithms and is currently widely used by many applications. However, Litecoin, which is one of the top platforms that use this algorithm set the memory size at 128 KB [31] thus making it possible to be stored at the CPU cache level. This restriction was applied since the Scrypt algorithm requires the same resources for solution verification as for the solution discovery and higher requirements would stress too much the regular non-mining nodes. Dagger Hashimoto [32], [33] on the other hand, is an algorithm that provides an easy verification solution, thus allowing the Prover's requirements in memory size to increase up to 1 GB RAM. Equihash is also a widely used hashing algorithm but its main disadvantage, as its authors state [34], is the fact that it is parallelizable, which is not a quality desired in ASIC resistant algorithm. Finally, the Cuckoo hash cycles [35], (i.e. used in GRIN [36] and Aeternity [37]), is also considered a reasonable solution when talking about ASIC resistance.

Even though the PoW algorithms have good results in reaching a distributed consensus on the system state their main problem lies in the high energy consumption needed for solving the computationally intensive problem. This aspect makes unfeasible their usage for smart energy grid systems in general and for energy efficiency applications in particular. For example, the validation of transactions in Bitcoin demand an amount of electricity similar with the one of approximatively 32 US households [22], while the entire Bitcoin blockchain consumes as much electrical energy as Switzerland.

The *first category of consensus solutions addressing the energy consumption issue* are based on adaptations of PoW aiming to give a purpose for all the energy and computational resources of the network. Since the network uses large computational resources whose only purpose is to prove and validate the next block of the blockchain, the concept of Proof of Useful Work is launched as an alternative to trying to use the computational power for a publicly beneficial domain. Such implementations aim to use the computational power across the network to solve some of the world's problems (e.g. research simulations for Proof-of-Research). For example, CureCoin [38] is implementing such an algorithm called SigmaX that aim to perform protein unfolding in order to find a cure for different diseases.

The *second category are the Virtual Mining Protocols* that offer an alternative to the PoW by keeping a high cost for the Prover but changing the type of system resource consumed. If the cost of the Prover in PoW is directly related to the energy consumed, which is lost if the Prover does not offer honest work to be validated and rewarded by the network, in the case of the virtual mining Protocols the Prover cost is a deposit of coins that are offered as insurance for its honest work. If up until now the node was chosen based on its result to the computationally intensive problem, now the node will be elected in a pseudo-random way, and the chance of winning will be proportional to the number of coins / stakes of the owner of the system. As result in the Virtual Mining Protocols, the clients have the mining potential proportional to the percentage of the stake they hold. Three such virtual mining approaches have been identified that aim at incentivizing the honest work of the miner by promising as a reward a sum of coins greater than the initial insurance: Proof of Stake considers the age of the coin in the algorithm, thus requiring for some coins not to be spent for a period of time; Proof of Burn requires for a relevant amount of coins to be destroyed and a proof of the destroying transaction to be provided; Proof of Deposit requires for some coins to be put away for some time in a vault. Proof of Activity [39] is a hybrid algorithm build upon PoW and Proof of Stake (PoS) found in [40]. The algorithm starts as a simple Proof of Work algorithm until one correct hash is found; the block is then transmitted in the network, but it is not yet added to the blockchain. To be validated the block needs to be signed by N peers in the blockchain network. The hash is used to generate N numbers that correspond to N coins generated since the genesis of the blockchain. Each of these coins has one current stakeholder who will be required to sign the current block. In case that some of the stakeholders are not online and cannot sign, then new hashes are generated, and other stakeholders are asked to sign the block. This approach makes attacks upon the network more difficult since it makes use of the advantages brought by both systems.

From existing Virtual Mining Protocols, the Proof-of-Stake has a good potential of becoming the most used consensus protocol in DLTs because it addresses fundamental problems of the PoW protocol such as computational waste and high-power demand [47]. Anyway, in case of PoS algorithm, since the nodes propose a new block by guaranteeing with their own stake it gives rise to the "nothing-atstake" vulnerability. This means that when a fork appears in the context of a network partitioning, a 12-attacker node can propose a block on either chain, hoping that at least one block will be accepted. The node guarantees each proposed block with its own stake, but due to network partitioning it is difficult for other nodes to observe and penalize this misbehaviour. This situation can lead to other forks or to the fact that the attacker node receives rewards for proposing new blocks. In PoW algorithms, the "nothing-at-stake" vulnerability is avoided due to the fact that when proposing a new block, the node has to solve a computational puzzle that consumes electrical energy, and by proposing two blocks on two chains from a fork means that the node has to solve twice the problem, thus doubling its costs. The Casper version of PoS [41] is considered a suitable alternative for the permissioned systems, by considering only a fixed set of users as validators of blocks but its implementation is not yet available. Another flavour of Proof-of-Stake commonly used for permissioned systems is the Delegated Proof of Stake in which N peer nodes witnesses are periodically selected by stakeholders of the system to propose the next block, and then be rewarded for its contribution. Proof of Authority [44], on the other hand, suggests that only trusted parties are entitled to provide commits to the system, which can be required where high-security properties need to be implemented, like in the case of private Enterprise solutions. In Quorum, RAFT [42] algorithm is used, where a predetermined leader is creating a block that is sent to each node in the cluster. In terms of finality, proof-protocols are known not to be final, however they offer probabilistic finality, since once many blocks are sealed over, the probability of a block's state to change is very low.

The PoS solutions can be grouped in two main categories [41]: i) chain-based PoS that mimics PoW by assigning pseudo randomly the right to generate new blocks to various nodes and ii) Byzantine Fault Tolerant PoS that is based on BFT research. They address the "nothing-at-stake" vulnerability in different ways. The chain based PoS are penalizing nodes when sending multiple blocks on competing chains (e.g. Slasher [50], [51] or Casper). The BFT PoS mechanisms allow validators to vote on blocks by casting several messages, with two rules: finality condition (to determine when a hash is finalized) and slashing conditions (to determine when a validator misbehaved and must be excluded). A block is considered finalized once enough votes have been cast and all nodes from the DTL agree on adding it to the canonical history. This involves sending many messages in the network to make aware other nodes that a new block was proposed and running a version of Byzantine Agreement on the new block. Propagating many messages in the network impacts system scalability, thus methods to reduce the number of messages is needed. Two techniques are found in literature addressing this: i) quorum based voting – when a node is selected randomly as the prover and a subset of nodes are selected to be verifiers that run a Byzantine Agreement protocol (Algorand [47]); and ii) sharding-based approaches – where the blockchain is split into shards for inter-shard transactions and only transactions that involve nodes from two different shards need message propagation between shards (Casper, Elrond [48]). Algorand is based on a new and fast Byzantine Agreement Protocol used to generate a new block through a binary Byzantine Agreement (BA*) protocol that enhances the traditional BA protocol to work in rounds in a synchronous environment with at least 2/3 players being honest. Furthermore, a cryptographic sortition based on Random Verifiable Functions is used to select a subset of the users to be members of the BA* algorithm. A cryptographic function is used to select a new leader based on a previous block. The leader will be in charge to propose the new block. A set of verifiers is used to check the validity of the new proposed block. The choice of the leader is not predictable, thus making impossible for an attacker to alter the new block. Furthermore, leaders learn of their role without informing others only after proposing the new block, thus avoiding attacks. After a new block is proposed, the leader has no importance for the algorithm. However, the verifiers must agree on the new block, and they run the BA* algorithm in rounds, at each step players being replaced, thus avoiding cases when many verifiers are corrupt. Elrond is based on a sharding approach, splitting the blockchain and account state in several shards where parallel validation can occur using a consensus algorithm based on a secure PoS. The consensus algorithm follows a similar approach as Algorand with a prover and a set of validators chosen randomly within a shard and running a Byzantine Agreement algorithm to validate the proposed block. Finally, Hot Stuff [49] proposes a consensus algorithm using a leader-based Byzantine fault-tolerance protocol for partially synchronous distributed system models where a chosen leader drives the consensus decision at the rate of the maximum delay allowed by the network. Table 1 shows a comparison between main consensus algorithms considering the relevant features discussed above.

Table 1: Comparison among consensus solutions in blockchain

Features	Proof of Work	BFT based Proof of Stake			
	Bitcoin	**Casper**	**Algorand**	**Elrond**	**Hot Stuff**
Partition Resilient	Yes, eventually consistent	Partition resilient and available with a trade-off in consistency, leading to attacks such as Nothing-at-Stake			
Network assumptions	*Permission-less*				*Permissioned*
	Synchronous network	Synchronous network	Synchronous network	Synchronous within shards; Asynchronous cross-shard.	Partial synchrony model
Max number of faulty nodes	49%	33%	33%	33%	33%
Scalability (Network Size)	>10.000 full nodes	-	50-500K nodes	16 shards, total number of nodes N/A	128 nodes
Transaction throughput	2 TPS (1 block every 10 minutes)	15 TPS	250 TPS (1 block in less than 10 minutes)	>10.000 TPS	50 TPS
Transaction Finality	More than 1 hour (6 blocks)	Yes	1 block in less than 10 minutes	1 block in less than 10 minutes	Yes - Proven
Smart Contracts	No	Yes	Yes, but not Turing complete	Yes - EVM compliant engine	No
Sharding	No	Yes	No	Yes	No
Prover	First Miner that solves the computational problem	Node from Dynamic Validator Set	Node chosen randomly using Verifiable Random Functions and last block hash	Block proposer from an eligible set committing stake	Leader chosen from the network, center of the star communication network.
Validator	Any other full node can check the transactions from the newly proposed block	Dynamic Validator Set selected according to stake. Nodes can join and leave the set dynamically.	Validator set chosen randomly using Verifiable Random Functions and last block hash.	Other nodes from eligible set	Nodes from the validator set
How nodes agree on new block	Each full node can validate the block	Message passing protocol BA	BA run by validators	Modified PBFT	Modified Practical; BFT run in three phases.
Generation of new tokens	Yes, the miner is rewarded	Yes	No need for incentives for validators	Yes (ERD)	-
Existence of Forks	Yes, miners will propose blocks only on longest chains	Yes	No – Only Adversaries can create forks (small probability)	Yes – Only within shards; Forks of maximum 2 blocks long.	-

Protocol Finality	Yes, mining process takes a finite time	Ongoing research	Yes - BA	Modified PBFT	Yes - Proven

Analysing the above state of the art consensus approaches none of them are suitable to be re-used in our blockchain based energy and DR management platform. The most likely candidates are the PoS approaches because are addressing the energy consumption issue of the consensus process but each them are either in their initial development phases and miss some relevant features desirable for the eDREAM platform. Ethereum Casper is not deployed yet, it does not have operational shards and PoS ready for platform implementation and use, Algorand does not offer programming language to write Turing-complete Smart Contracts needed for eDREAM energy and DR platform implementation, similar Elrond do not have at this point a blockchain version with a Ethereum Virtual Machine capable of running Smart Contracts and finally Hot Stuff does not offer proper documentation and assistance for platform developing.

2.1.2 eDREAM Solution

Considering the 2nd tier eDREAM distributed leader for energy transactions (implemented in D5.1) and the smart contracts based platform for energy and DR management (implemented in D5.2) we have proposed a consensus-based algorithm based on adapted version of the two-phase commit and validation stakes to validate the each prosumer peer energy transactions prior to the submission on the chain. We propose the use of a Proof-of-Authority (PoA) consensus algorithm at blockchain tier (implementation based on Ethereum) for blocks validation and to implement an overlay peer-to-peer network of nodes and a consensus algorithm allowing them to agree on the validity of each energy transaction generated by each peer (based on its monitored energy data) prior to is submission on the chain.

Figure 3 presents the main tiers of the prosed consensus solution for energy transactions validation:

- **Prosumer Tier** – contains the prosumers peer nodes and associated smart metering devices. It implements the tamper evident energy hashing solution of the eDREAM 2 tier distributed ledger as described in Deliverable D5.1. Each prosumer smart meter sends energy monitored data M at timestamp to an edge node with which it is associated at the Overlay P2P Network Tier.
- **Overlay P2P Network Tier -** acts as an intermediary between the edge tier and the blockchain tier, aiming to filter out malicious energy monitored values and associated transactions by implementing a consensus algorithm that considers the stake of the nodes when casting votes. This allows to minimize the risk of misbehaviour due to malicious data provided by energy meters or altered by attackers on the communication network.
- **Blockchain Tier – implements** the business logic of eDREAM energy and DR management platform using Smart Contracts in Ethereum, while at this tier the consensus and blocks validation is achieved through PoA.

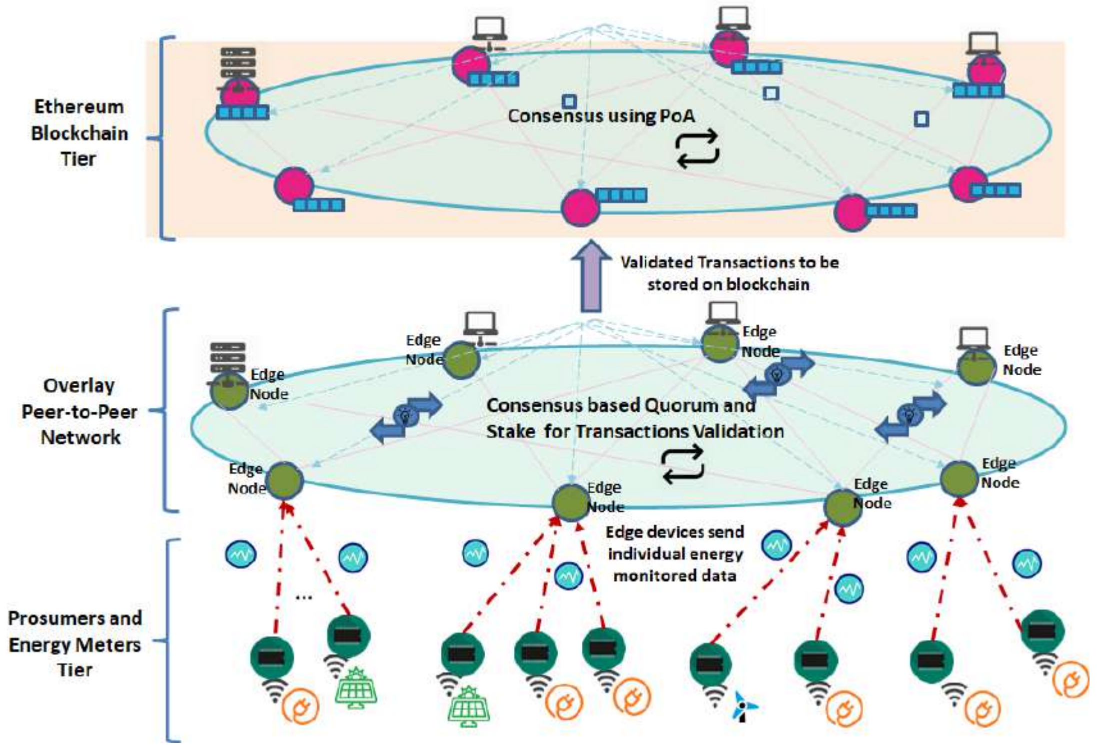

Ethereum Blockchain Tier
Consensus using PoA
Validated Transactions to be stored on blockchain
Overlay Peer-to-Peer Network
Edge Node
Consensus based Quorum and Stake for Transactions Validation
Edge devices send individual energy monitored data
Prosumers and Energy Meters Tier

Figure 3. Consensus based energy transaction validation in eDREAM blockchain based platform

We consider that each smart energy meter associated with a prosumer node collects monitored data in the format , ∈, meaning that the energy value) was registered at timestamp . The data is sent to the edge node being previously signed using a private key s generating a digital signature:

$$Si = encrypt(, s)$$

The public key p is publicly available for all the other edge nodes from this tier. The edge node generates energy transactions are being generated $Energ < H, AV >$ (i.e. average of monitored energy values over the interval).

The Overlay Peer to Peer Network Tier consists of M edge nodes, denoted , $k \in \{1..M\}$, each such device being able to run validations of monitored data based on its knowledge of the sensors. Furthermore, each such node has a stake), that is proportional to the hardware capabilities and validation accuracy of the node.

Each such edge node has a local buffer of $toBeProcessed$ where it stores messages with monitored energy data registered by associated prosumers peers. When an edge node prepares to commit an energy transaction of a prosumer to the blockchain tier, it extracts from the local buffer values associated to that prosumer metering device and computes an average for it over a time interval . This value AV will be then validated and consent by other peer nodes before committing it to the blockchain.

$$AV = *, \in$$

This information is broadcasted by the edge node to a mesh of edge nodes from the Overlay P2P Network and eventually saved in their local buffers. Every edge node has the possibility to decrypt the message and see the actual values as well as the prosumer ID.

$$= decrypt(Si, p)$$

The problem that appears is to detect if the value transmitted by the prosumer's smart meter is the real measured value or it was altered on purpose. We have defined and used a set of validation rules to be run by each validator edge node to identify the malicious energy data values from the network. Depending on their capability and stake, the nodes will run various anomaly detection algorithms to validate the proposed monitored values. If the value is validated with success, they will receive a reward directly proportional to their stake. Otherwise, the node will receive a penalty. This will lead to the situation where the validator edge node will have a stake proportional to the number of success validations, that can be proved empirically to depend on the complexity of the validation process run locally.

Considering their hardware capabilities, the edge nodes may use rules-based system requiring low hardware resources or more complex algorithms based on machine learning for anomaly detection. The simplest approach for validating a monitored value is to compare it to the minimum and maximum values reported for the given prosumer, i.e. the prosumer flexibility bands. The machine learning techniques can be used to classify the monitoring patterns of each smart meter. This requires that every edge node holds a database with monitored data from the smart meter, and continuously updates it with valid data agreed through consensus by other edge nodes. Using this database, it uses supervised learning techniques to identify consumption patterns and uses them to classify anomalous consumption patterns.

Regardless of the validation technique used locally by the edge nodes, the result of the validation can be expressed as:

$validat(=$

$$=$$

The proposed consensus algorithm is leveraging on concepts specific to the 2-Phase-Commit algorithm and PoS consensus allowing the individual peer nodes to vote on the validity of an energy transaction guaranteeing this with their own stake. We consider that due to the fact edge nodes use their stake in validation process at least the high-stake ones not malicious (i.e. they may lose their stake otherwise) thus there is no need for a BFT consensus. This relaxation allows the implementation of consensus validation derived from proof-of-stake to validate transactions (monitored values from sensors) before committing them to the blockchain.

The goal is to forbid tampered or not valid monitored energy values to reach the blockchain as energy transactions (i.e. only values labelled as true by the $validat$ method, are propagated). However, because sometimes the edge nodes do not have full knowledge of the sensors to validate individually each value, we propose a consensus algorithm where a set of edge nodes vote for valid values and reach quorum-based agreement on which values to drop and which to propagate further.

The proposed consensus algorithm ensures the following properties:

- termination (or liveness) – meaning that after a finite amount of time all edge nodes reach a consensus regarding the validity of received monitored energy value. This is achieved by running a quorum-based consensus algorithm in two phases;
- integrity (or validity) meaning that all nodes that propose a value AV , as being valid should all agree on that value (i.e. all the nodes vote Yes, and none votes No);
- agreement (or consistency) meaning that all edge nodes should agree and only one of them will send the value $Energ < AV >$ to the blockchain layer, while all other nodes will mark the value as sent.

The main steps of the consensus algorithm are depicted in Figure 4. There are two roles:

- **Initiator** role, of the edge node that decides to validate the monitored energy values and energy transaction received from an smart energy meter associated with a linked prosumer peer in order to gain an incentive $incentiv$, and raise its stake, and

- **Participant** role, that will aid the Initiator to validate the energy transaction and will gain in exchange a part of the validation incentive.

The algorithm will be run in three phases, the first one will create the committee of P participants used for voting through a randomized technique, the second phase will be a validation phase, where participants validate locally the monitored values and cast a vote backup up by a stake they will to lose in case their vote is not majoritarian, and finally a commit phase, where the validated monitored value is committed to the blockchain and the financial settlement is finished.

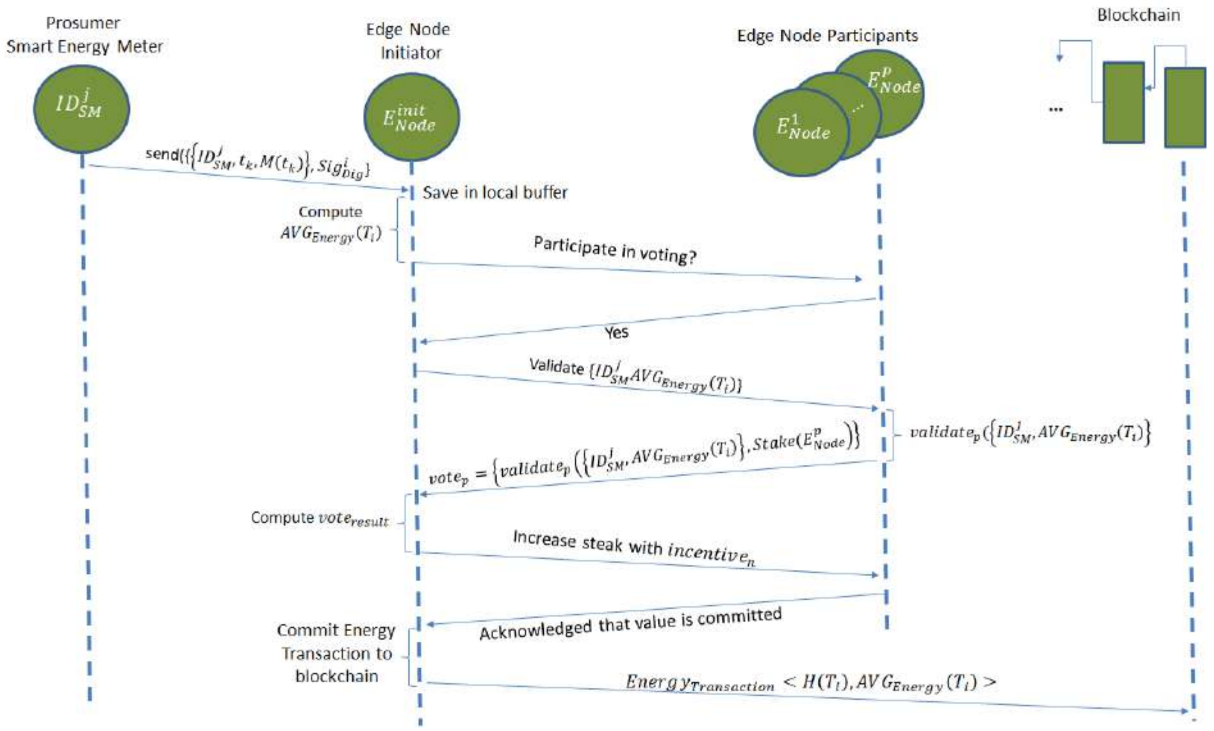

Prosumer Smart Energy Meter
Edge Node Initiator
Edge Node Participants
Blockchain
ID_{SM}^{j}
E_{Node}^{init}
E_{Node}^{P}
E_{Node}^{1}
...
send({$\{ID_{SM}^{j}, t_k, M(t_k)\}, Sig_{Dig}^{j}$}
Save in local buffer
Compute $AVG_{Energy}(T_i)$
Participate in voting?
Yes
Validate $\{ID_{SM}^{j} AVG_{Energy}(T_i)\}$
$validate_p(\{ID_{SM}^{j}, AVG_{Energy}(T_i)\}$
$vote_p = \{validate_p(\{ID_{SM}^{j}, AVG_{Energy}(T_i)\}, Stake(E_{Node}^{P})\}$
Compute $vote_{result}$
Increase steak with $incentive_n$
Acknowledged that value is committed
Commit Energy Transaction to blockchain
$Energy_{Transaction} < H(T_i), AVG_{Energy}(T_i) >$

Figure 4. Sequence of steps performed by the consensus algorithm for energy transaction validation

The algorithm run by the initiator is presented in Figure 5. The algorithm has as input the monitored values M over interval on whcih the transaction $Energ < AV >$ is generated and as output a vote true/false validating the energy transaction and preparing it to be committed to the blockchain tier.

The edge node receives a set of monitored energy value digitally signed by a prosumer and saves them in the local buffer of $toBeProcessed$ messages. When it chooses to validate the monitored values over the time interval the node computes the average of those values AV for smart energy meter with I. Then the node runs the first phase of the algorithm, when it should select a committee formed by a set of peer edge nodes which will also validate the value as well. Node runs a randomized selection through which it selects a committee of P nodes from the total N nodes: $Committee = \{, p\epsilon, P \leq N\}$ (line 1). Then, it sends a message to every node of the committee, collecting the answers of those that can participate (see lines 2-3).

```
Input:  monitored energy value registered by prosumer smart meter $I$
Output: $true$ if value is valid and transaction ready to be committed to blockchain
        $false$ if value is not valid and transaction discarded
begin
    1.  Select random committee members $Committee = \{, p\epsilon, P \leq N\}$
```

```
2.    Send message of "participate?" to each member of the $Committee$
3.    Collect answers from members that will participate in voting
4.    Send $validat($ to each member of $Committee$
5.    Collect $P$ votes from $Committee$ members: $vot =$
6.    Compute $vot =$
7.    If $vot > 0$ then
8.        the value $AV$ is considered valid and will be committed to the blockchain.
9.        edge nodes that voted with 1 receive an $incentiv$
10.       edge nodes that voted with -1 (the monitored value is valid) lose their stake )
11.       return $true$;
12.   end if
13.   If $vot \leq 0$ then
14.       the value $AV$ is considered invalid and the transaction will be discarded
15.       nodes that voted with -1 receive an $incentiv$
16.       nodes that voted with 1 (the monitored value is valid) lose their stake )
17.       return $false$;
18.   end if
end
```

Figure 5. Consensus algorithm for energy transactions validation

The Voting Phase of the algorithm involves the initiator edge node sending the monitored value AV to every edge node from the committee and wait for their votes (see line 4). Each edge node from the $Committee$ set will validate locally the value and send back to the initiator their votes weighted with a stake they will to lose in case their vote is not majoritarian:

$$vot =.$$

The initiator will collect the P votes (see line 5) and pass to the Committing Phase. The initiator collects the votes from the $Committee$ and selects the majority of the votes weighted with the stake of the (see line 6).

$$vot =$$

Finally, the initiator evaluates the result of the vote and computes the incentive of each node that validated correct the monitored value (see lines 7-18), according to the equation:

$$incentiv = incentiv *$$

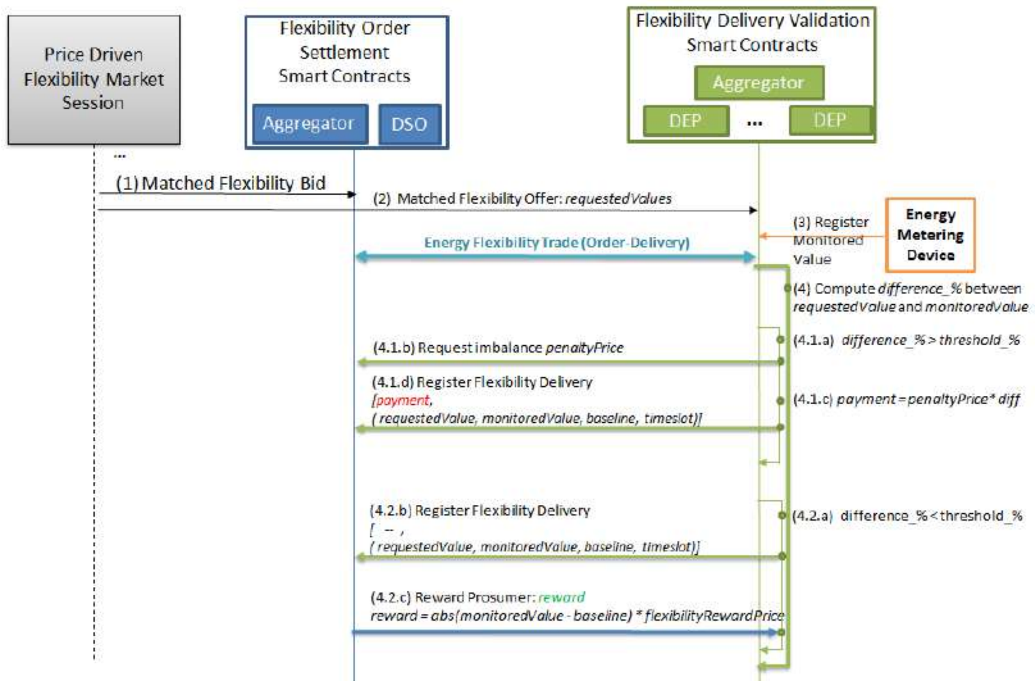
If $vot > 0$ then the value AV is considered valid and the transaction will be committed to the blockchain layer. Nodes that voted with 1 receive an incentive $incentiv$ direct proportional with the stake proposed. Furthermore, all nodes that voted with -1 (the monitored value is valid) lose their stake).

If $vot \leq 0$ then the value AV is considered invalid and will be discarded. Nodes that voted with -1 receive an incentive $incentiv$ direct proportional with the stake proposed. Furthermore, all nodes that voted with 1 (the monitored value is valid) lose their stake).

2.2 PoA and smart contracts for settlement of DR programs

For flexibility actual delivery and financial settlement in DR programs we have developed at the Blockchain Tier a mechanism that is based on smart contracts and PoA as algorithm for blocks validation.

Figure 6 presents the interaction between smart contracts defined and used for determining the actual flexibility delivery after the DR flexibility order signalled had been agreed upon and issued to the enrolled prosumers. They allow the measuring in the prosumers or aggregators rate of success or failure to deliver the promised flexibility in near-real time, registering potential imbalances in terms of deviations from flexibility order signals and then conducting the financial settlement and validation of the program.



Price Driven Flexibility Market Session
Flexibility Order Settlement Smart Contracts
Aggregator
DSO
Flexibility Delivery Validation Smart Contracts
Aggregator
DEP
...
DEP
(1) Matched Flexibility Bid
(2) Matched Flexibility Offer: requestedValues
Energy Flexibility Trade (Order-Delivery)
(3) Register Monitored Value
Energy Metering Device
(4) Compute difference_% between requestedValue and monitoredValue
(4.1.b) Request imbalance penaltyPrice
(4.1.a) difference_% > threshold_%
(4.1.d) Register Flexibility Delivery [payment, (requestedValue, monitoredValue, baseline, timeslot)]
(4.1.c) payment = penaltyPrice * diff
(4.2.b) Register Flexibility Delivery [-- , (requestedValue, monitoredValue, baseline, timeslot)]
(4.2.a) difference_% < threshold_%
(4.2.c) Reward Prosumer: reward
reward = abs(monitoredValue - baseline) * flexibilityRewardPrice

Figure 6. Flexibility delivery validation and financial settlement

The sequence of operations involved, follows the initial flexibility request / flexibility order matching which may be done either by direct agreement of the aggregator with their enrolled prosumers (see D5.2) or via a price driven flexibility marketplace (to be delivered in D5.5). After the flexibility market session clearing phase has finished, the aggregators or the DSO are notified about their matched Flexibility Bids (Step 1), and the prosumers or other aggregators are notified about their matched Flexibility Offers (Step 2). Once notified about the matched Flexibility Offer, the DEP (Distributed Energy Prosumer) smart contract registers the flexibility order signal to be followed in near real time during the actual delivery of flexibility (considering the matched flexibility requested values).

The following sequence of steps will repeat unit the end of the flexibility request period (i.e. end of the DR program):

- Step 3: The smart meter of a prosumer and the corresponding edge node registers the monitored energy value and associated energy transaction on the blockchain, and the DEP smart contract is triggered.
- Step 4: The DEP smart contract is responsible to evaluate the difference between the actual monitored energy value and the flexibility order value agreed upon. Based on the difference, there are two possible alternatives:

- Step 4.1: The prosumer was not able to deliver the promised flexibly energy thus is failing to follow the flexibility order curve. If the deviation percentage (*difference_%)* is higher than the threshold imposed by the flexibility buyer (4.1.a), then the prosumer will have to pay a penalty. The payment is computed (4.1.c) based on the registered deviation multiplied by the penalty price per unit (4.1.b), imposed by the flexibility buyer. The payment is delivered to the buyer (4.1.d) together with all the information required to verify that the payment registered was correct.
- Step 4.2: The prosumer delivered the promised flexibility energy by successfully following the flexibility order curve. The monitored value is accordance to the flexibility requested by the buyer (4.2.a), then the prosumer must prove this by offering all the information regarding his activity at the current timeslot: the requested value, the baseline value and the monitored value. The Buyer evaluates the delivered flexibility and a reward is issued and transferred to the prosumer.

The steps from 3 to 4 will repeat unit the end of the flexibility request period (i.e. end of the DR program).

In the code snippet depicted Figure 7 the computation and validation of actual flexibility delivery against the flexibility request. Firstly (see line 2), the prosumer registering the monitored energy transaction value is authenticated to ensure that the transaction is indeed signed by the prosumer owning the DEP smart contract. Once the issuer is authenticated, the absolute deviation with respect to the requested profile is computed (see line 5) and the deviation percentage (see line 6).

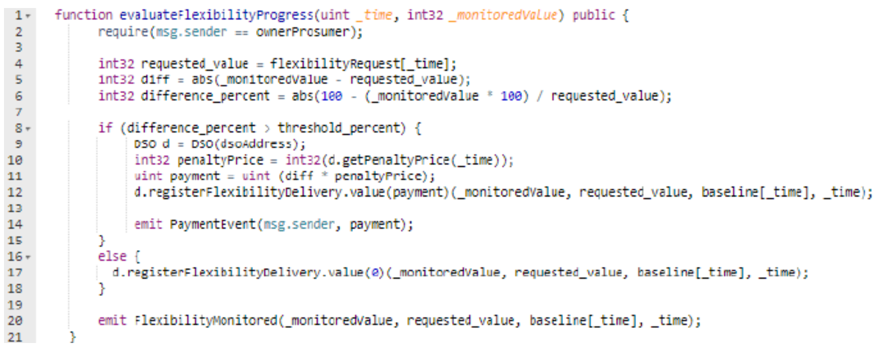

```
1   function evaluateFlexibilityProgress(uint _time, int32 _monitoredValue) public {
2       require(msg.sender == ownerProsumer);
3   
4       int32 requested_value = flexibilityRequest[_time];
5       int32 diff = abs(_monitoredValue - requested_value);
6       int32 difference_percent = abs(100 - (_monitoredValue * 100) / requested_value);
7   
8       if (difference_percent > threshold_percent) {
9           DSO d = DSO(dsoAddress);
10          int32 penaltyPrice = int32(d.getPenaltyPrice(_time));
11          uint payment = uint (diff * penaltyPrice);
12          d.registerFlexibilityDelivery.value(payment)(_monitoredValue, requested_value, baseline[_time], _time);
13  
14          emit PaymentEvent(msg.sender, payment);
15      }
16      else {
17        d.registerFlexibilityDelivery.value(0)(_monitoredValue, requested_value, baseline[_time], _time);
18      }
19  
20      emit FlexibilityMonitored(_monitoredValue, requested_value, baseline[_time], _time);
21  }
```

Figure 7. Validation of actual flexibility delivery

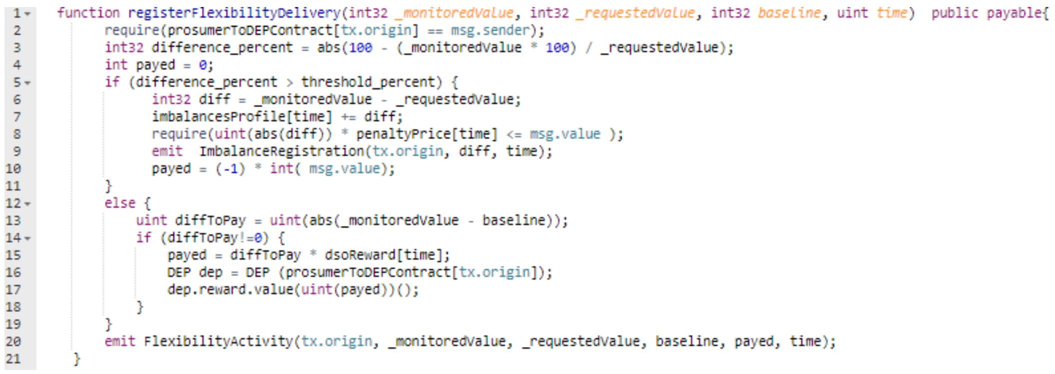
If the deviation percentage is higher than the allowed threshold, imposed by the flexibility order the prosumer will need to pay for the imbalance created. Firstly, the DEP smart contract will get the penalty price (price per unit Wh) then the required payment is computed as the product of the penalty price and the imbalance created (see line 11). The flexibility requester is notified (see line 12) about the monitored value and the reference values (requested value and baseline value) in order to allow its own evaluation about the DEP actual flexibility delivery. Being a payable function, together with the data, the DEP smart contract will also deliver the payment required for the generated imbalance. However, if the prosumer successfully manages to deliver the flexibility requested (see line 16), associated incentives will be distributed to its wallets during the financial settlement.

For flexibility order financial settlement, the flexibility requester associated smart contract stores the incentives reward for successful delivery of flexibility and the penalty price for failing to meet flexibility order. Both prices are registered as price per unit (see Table 2).

Table 2. State variables used for DR financial settlement

State Variable	Description	Unit
Incentives	The incentive offered as a reward for making available the flexibility.	Gwei / Wh
Penalty	The penalties imposed for noncompliance.	

In Figure 8 is depicted the code snippet of the smart contract responsible to evaluate and financial settle the prosumer for its flexibility delivered. Firstly, the contract authenticates the prosumer and the associated contract (DEP smart contract) (see line 2) and validates that the current prosumer is registered as a matched prosumer responsible to deliver flexibility.



```
1   function registerFlexibilityDelivery(int32 _monitoredValue, int32 _requestedValue, int32 baseline, uint time)  public payable{
2       require(prosumerToDEPContract[tx.origin] == msg.sender);
3       int32 difference_percent = abs(100 - (_monitoredValue * 100) / _requestedValue);
4       int payed = 0;
5       if (difference_percent > threshold_percent) {
6           int32 diff = _monitoredValue - _requestedValue;
7           imbalancesProfile[time] += diff;
8           require(uint(abs(diff)) * penaltyPrice[time] <= msg.value );
9           emit  ImbalanceRegistration(tx.origin, diff, time);
10          payed = (-1) * int( msg.value);
11      }
12      else {
13          uint diffToPay = uint(abs(_monitoredValue - baseline));
14          if (diffToPay!=0) {
15              payed = diffToPay * dsoReward[time];
16              DEP dep = DEP (prosumerToDEPContract[tx.origin]);
17              dep.reward.value(uint(payed))();
18          }
19      }
20      emit FlexibilityActivity(tx.origin, _monitoredValue, _requestedValue, baseline, payed, time);
21  }
```

Figure 8. Flexibility delivery financial settlement

The smart contract will check the deviation percentage (see line 3) in order to ensure the correct payment of the prosumer. In case the deviation is higher than a predefined threshold (see line 5), the contract will verify that the penalty payed by the DEP contract (see line 8) is well calculated and validated. Otherwise, if the prosumer managed to deliver the requested flexibility (see line 12) then the contract will firstly compute the amount of flexibility the prosumer have delivered with respect to the baseline (see line 13) and the prosumer will be rewarded proportional considering to the predefined incentive value (see lines 15-17).

To validate and enforce the financial settlement, the PoA algorithm is used by the nodes to ensure consensus between all participants (see Figure 9). Once new monitored energy data values are registered, the Edge Node will sign an energy transaction, specifying the average value as a payload and specifying the DEP contract address as a recipient of the transaction. Once the transaction reaches the Validator node, the DEP smart contract (i.e. recipient) will be executed (in an RPC-similar approach) considering as input the state registered in the chain in the latest block (i.e. Block K). Once the states are updated, based on the obtained contract state and balance states a new block (Block K+1) is proposed by the validator by recomputing the storage root and the and the state root. The newly created block is then broadcasted to the entire network of peers. Upon receiving the new block, each node is responsible to validate the block and append it to their local chain.

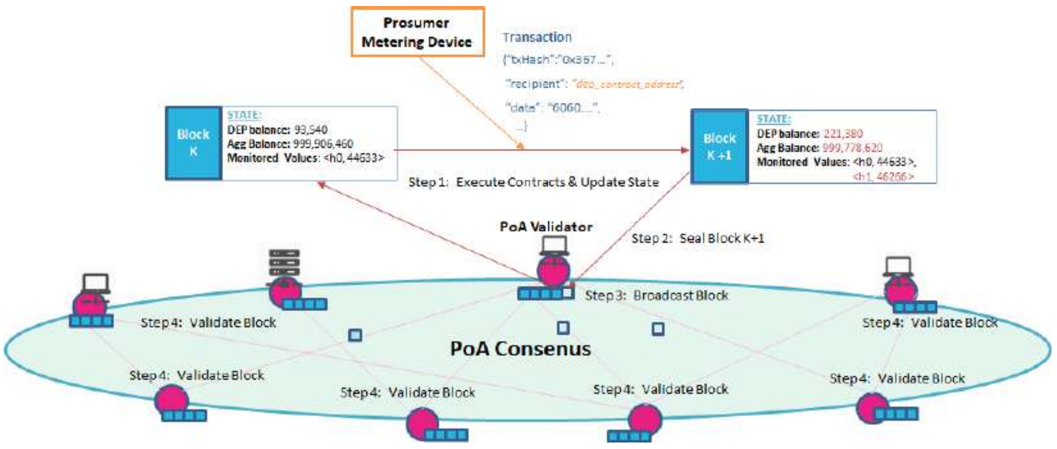


Figure 9. State update and agreement using PoA consensus

PoA is a reputation-based consensus algorithm, that comes as an alternative to Proof-of-Stake, such that the validators are no longer staking their coins but their own reputation. The PoA Validator nodes are required to reveal their real identities. In our approach we have leveraged on the AURA PoA implementation used by Ethereum (Parity) [56]. The algorithm considers N validator nodes, that are trustworthy entities. At each step of the algorithm one validator is selected to issue the next block. The selection process is based on the step S which a new block needs to be proposed such that:

$$= S \ mod \ N$$

In this case the rule of the "Longest Chain" imposed by the Proof-of-Work algorithms is adapted, such that an honest validator node must propose at each step the new block on top of the chain with the highest score (i.e. score function is defined and used to evaluate the score of the chain).

2.3 Obtained results

For evaluation purposes we have considered as scenario the energy and financial settlement of imbalances in DR programs (see Figure 10).

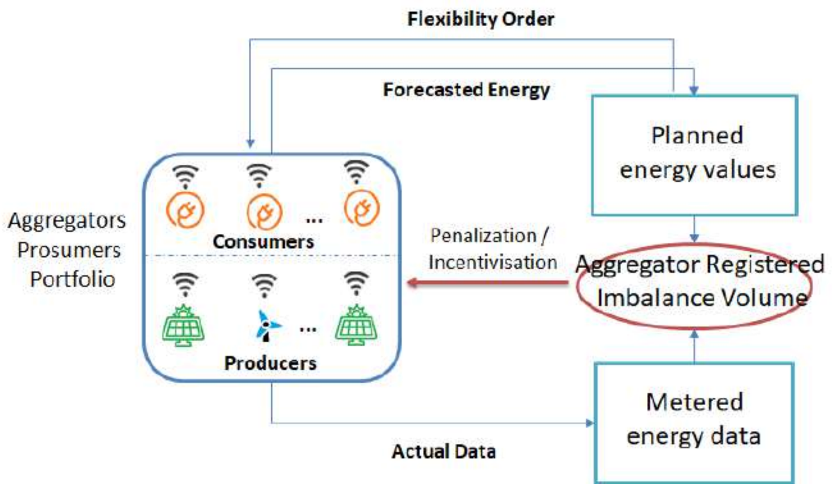


Figure 10. Aggregator energy and financial settlement of imbalances in flexibility delivery

Nowadays this process features significant delays (i.e. a few months) due to latency in energy volumes actualization, reconciliation and confirmation. The blockchain is a potential technology for reducing the delays to a minimum. The use of smart contracts as presented in section 2 has the potential of tracking in a near real time fashion (i.e. half an hour in our case) the identity of the prosumer generating the imbalance by not delivering the agreed flexibility amount and as result the billing process is brought closer to the real time.

We have considered an aggregator that is managing a portfolio of 12 prosumers directly or as results of being matched by the price driven flexibility marketplace (i.e. 6 producers and 6 consumers). In response to the DR matched request it submits energy flexibility offers by aggregating the flexibility of the prosumers from its portfolio. In case it fails to deliver the aggreged flexibility it is financially accountable for deviations from DR programs is case of success delivery is receives a financial reward. The incentives are divided to the participants prosumers based on how much they have adjusted their energy demand profiles to deliver the expected amount flexibility. At the same time the penalties are supported by the prosumers that are not delivering the expected flexibility causing imbalances.

The energy profiles of the consumers are provided by Kiwi project pilot containing 12-month minute by minute average power consumption, while the generation profiles have been synthetically generated with the same sampling frequency feature, considering different types of energy production (i.e. wind, solar, etc.).

For evaluation purposes we have considered a setup in which the DR program time interval for flexibility delivery is set for 12 hours, while for interval we had considered a granularity to a half an hour, the energy data being sampled by the energy meters at every minute:

$$= \{| <, \forall k = 1..30\}$$

Thus, energy transactions associated with the consumption and generation of modelled prosumers are published on chain at the end of each half an hour.

We have evaluated the effectiveness of our solution to detect the imbalances in terms of deviations between submitted energy plans and associated flexibility order by the aggregator and the actual delivery energy of prosumers from its portfolio. We have implemented and deployed smart contracts allowing to assess at each half an hour the deviations encountered at the level of the aggregator and to track the identity of the responsible prosumer. Figure 11 presents the energy consumption plan constructed by the aggregator, the values being split on each consumer from its portfolio.

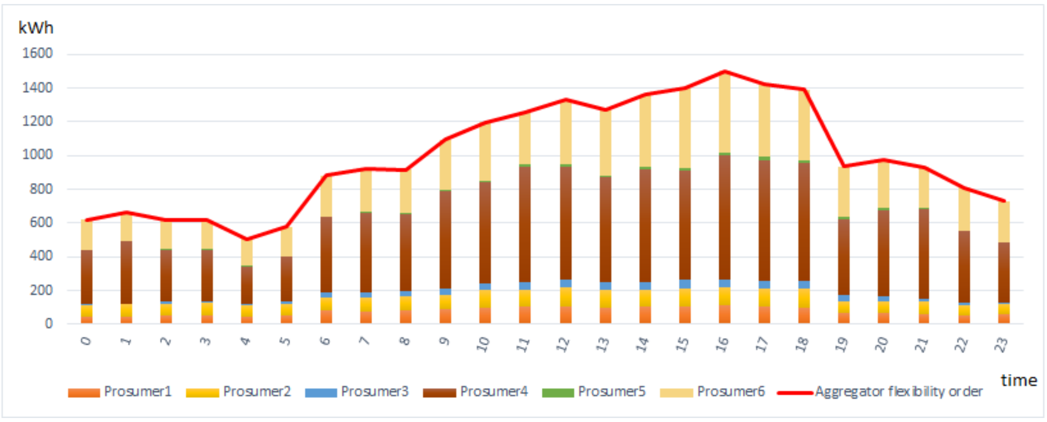


Figure 11. Aggregator scheduled energy plan in relation with enrolled prosumers energy

In our evaluation case we have considered that at certain time frames the actual energy consumption of some prosumers do not match the plan (i.e. they not able to comply to the flexibility order signals provided by the aggregator) generating deviations in terms of shifted and delivered flexibility amount for which the aggregator will be financially accountable (see Figure 12).

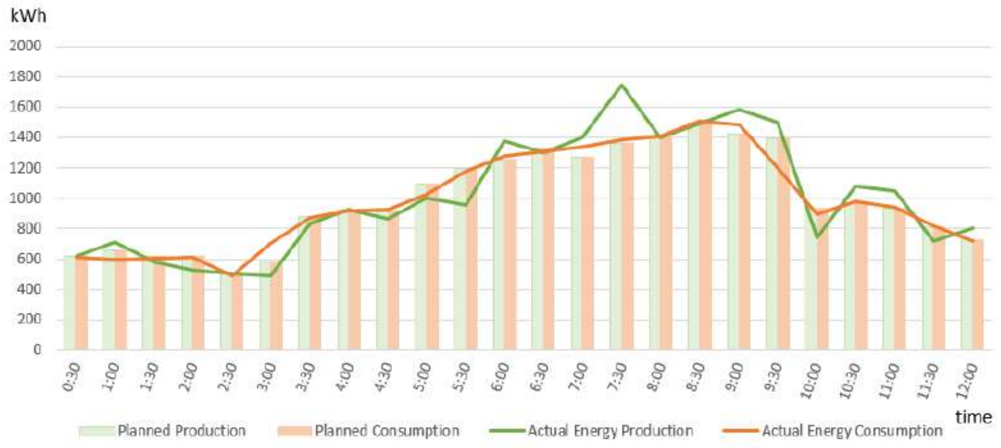


Figure 12. Aggregators computing the total planed order vs actual energy monitored values at every half an hour

Each prosumer's flexibility delivery activity is validated at each half an hour against the plan by means of the smart contracts deployed on chain. The corresponding smart contracts are triggered by new energy transactions registered in the distributed ledger every half an hour and evaluate the difference between the ordered energy consumption and the actual monitored one (as shown by monitored energy transactions registered in the distributed ledger).

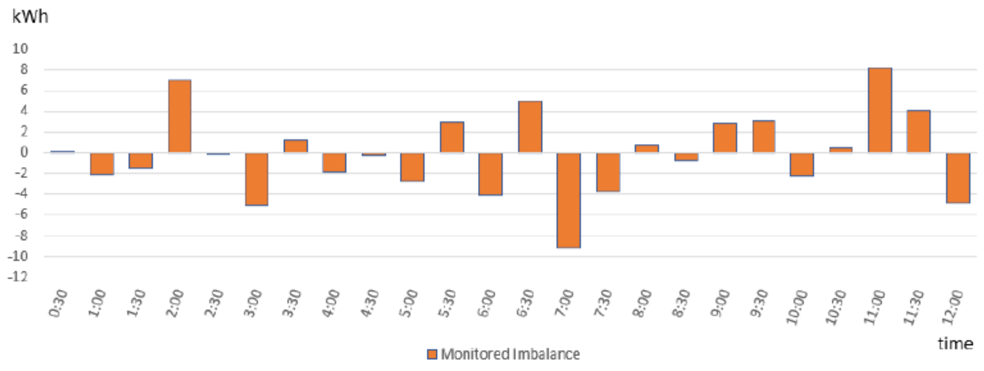


Figure 13. Validating the deviations of Prosumer1 from energy plan based on the blockchain registered energy transactions

Figure 13 shows the deviations calculated for Prosumer1 where negative values represent the downward regulation (i.e. prosumer's consumption is higher than anticipated) and positive values represent the upward regulation (i.e. prosumer's consumption is lower than anticipated) that must be activated to restore the balance. Each time the imbalances pass the threshold minimum allowed, the aggregator is penalized for the deviation in amount of delivered flexibility versus the expected one (for example hours 2, 7 and 11).

The deviations generated by the prosumers are further reported to the aggregator which may take advantage of the blockchain smart contracts to achieve an internal mitigation of flexibility deviation. For financial settlement we considered the price for the electricity of 0.2 euro per kWh and the price of Ether of 330 Euro, resulting in a reference price of 606 Gwei per Wh. Considering this, Figure 14 presents the financial settlement of Prosumer1 as a consequence of its energy monitored deviations in near real time from the aggregators expected values. We refer to this value as a "basic incentive" and, in section 3, we will see in detail how it is possible to improve or worsen it considering also how much the prosumer has moved from his expected behaviour trying to meet the demand.

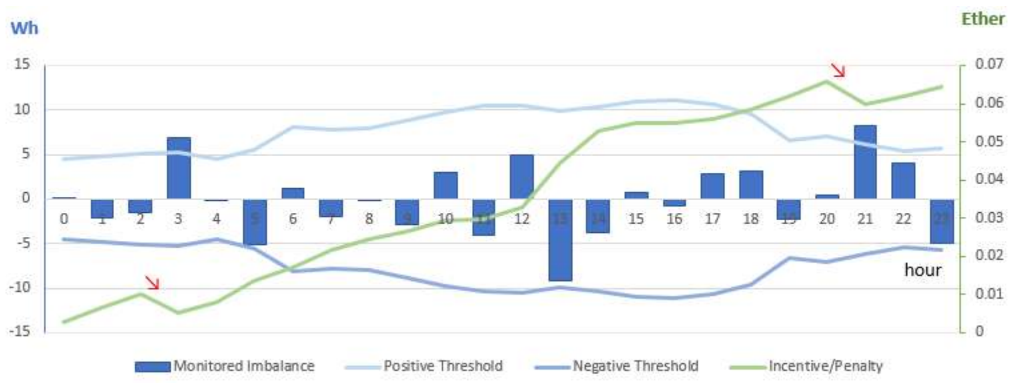


Figure 14. Prosumer1 financial settlement as result of flexibility delivery validation

The imbalances generated by the prosumers are further reported to the aggregator which may take advantage of the blockchain smart contracts to achieve an internal balancing between generation and production in real-time or it may pay penalties if it doesn't manage to delivered the overall aggregated flexibility requested (see Figure 15).

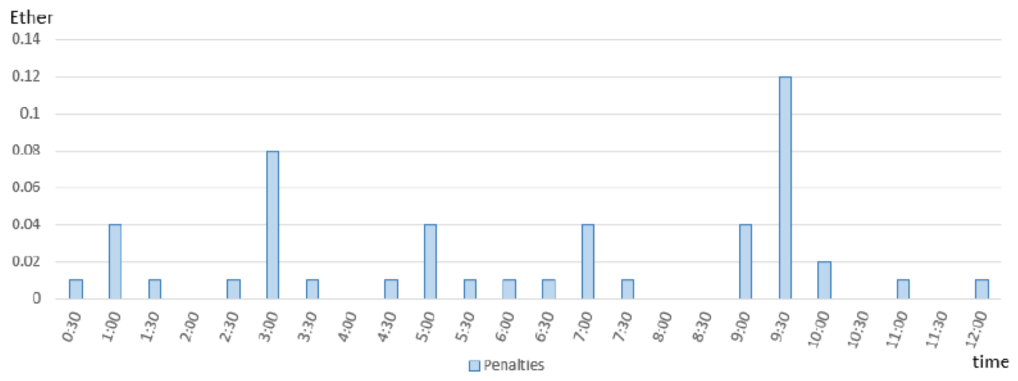


Figure 15. Aggregator financial settlement as result of energy transactions validation and deviations registration

As it can be seen in the above scenario a fundamental piece of information for the successful implementation of DR programs is the forecasting of energy demand and flexibility. In the eDREAM blockchain based implementation this process is being done using monitored energy data for which energy transactions are being generated and registered to the blockchain. Since up to the blockchain registration the energy and transaction information can be tampered is may influence the effectiveness of the predictions as well the financial settlement of the DR programs due to inaccurate reporting of flexibility shifting.

As already presented in the previous section to avoid this in eDREAM we have proposed a quorum and stake-based consensus and the Overlay Peer to Peer Network tier which allows the edge nodes to validate an energy transaction prior to its registration on chain.

Thus, we have evaluated the impact of this on the accuracy of the energy forecasting individual learners delivered in D3.1 aiming to determine the negative impact and prediction error increase due to data tampering as well as the enhancement brought by our consensus algorithm that can identify the timeslots with tampered sensor data and invalidate this data before reaching the blockchain and databases, thus removing it from the prediction history.

To simulate the impact of a prosumer associated Smart Meter sending malicious data onto the energy forecasting service in terms of prediction accuracy we have considered a random data tampering pattern. We have considered a Salt-and-pepper noise (S&PN) overlapped on the monitored data from the sensors. The input features for the prediction algorithm are computed considering the additive noise model $Z()$:

$$M = M + Z()$$

The noise model is defined considering a probability density function , a threshold and a maximum allowed noise value $Nois$:

$$Z =$$

We further present the evaluation results considering two scenarios:

- Scenario A - data modified using the noise model fed by the monitored devices is not validated at the Overlay Peer to Peer Network tier and is persisted in the Cassandra DB and the Blockchain Tier (see D5.1 energy ledger implementation description) and
- Scenario B - data modified using the noise model is passed through the P2P Overlay Network of Edge Nodes that used our proposed Quorum and Stake based Consensus to validate the energy transactions and monitored energy value and do not commit the anomalous one to the persistent storage. In the second scenario, invalidated values will be replaced in using interpolation.
- Reference Scenario - data not modified using the noise model.

Errore. L'autoriferimento non è valido per un segnalibro. shows the impact of allowing tampered energy data (i.e. by varying the probability density function of a Smart Meter altering its monitored values) in the absence of consensus-based validation.

Table 3. Prediction accuracy in MAPE values for the defined scenarios

	SVR			MLP			GBR			LSTM		
Noise Probability	Reference scenario	Scenario A	Scenario B	Reference scenario	Scenario A	Scenario B	Reference scenario	Scenario A	Scenario B	Reference scenario	Scenario A	Scenario B
0.1	4.72	4.88	4.73	5.68	6.54	5.94	4.75	4.91	4.8	13.17	14.54	13.16
0.2		4.97	4.77		5.57	5.93		4.91	4.94		13.17	13.17
0.3		6.19	4.57		6.62	5.76		5.33	4.89		13.51	13.17
0.4		5.17	4.76		6.6	5.98		5.08	4.79		13.17	13.16

On Reference Scenario the Support Vector Regression (SVR) forecasting obtained a MAPE error of 4.72%. On Scenario A, when data is altered using a Salt-and-pepper noise that varies from 0.1 to 0.4 probability, the MAPE error of the SVR algorithm increases from 4.88% up to 6.19%. However, in Scenario B, by passing the data through the proposed consensus and validation solution and replacing the missing data using interpolation, the MAPE errors of the SVR algorithm are kept close to the reference values, with a variation less than 1.25%.

Similar results are obtained for other prediction algorithms evaluated: Multi-Layer Perceptron (MLP), Long-Short Term Memory-based Neural Networks (LSTM) and Gradient Boosted Tree Regression (GBR). For each of them, in Scenario A when tampered and noisy data is allowed in the persistent data storage, the errors increase with the noise, displaying 5% up to 20% increase in the MAPE error when the noise varies from 0.1 up to 0.4. However, by using the validation of the energy transactions data through consensus in the overlay P2P network, the MAPE error of the forecasting process varies less than 6% compared to the MAPE of the reference scenario, even when the noise has a probability of 0.4.

3 Financial Settlement of DR

Smart meters and innovative communication systems facilitate the coordination of the participants in a smart grid [57]. The improved ability of prosumers to communicate with one another and make decisions about how and when to produce and consume electrical power, allowing customers to shift towards a 24/7-based demand response where the customer sees incentives for controlling load all the time customers are still largely influenced by economic incentives and are reluctant to relinquish total control of their assets to utility companies [58].

Demand Response (DR), by promoting the interaction of the customers, improve the reliability of the power system and, in the long term, lowering peak demand, DR reduces overall plant and capital cost investments and postpones the need for network upgrades.

One advantage of a smart grid application is time-based pricing. Customers who traditionally pay a fixed rate for consumed energy (kWh) and requested peak load can set their threshold and adjust their usage to take advantage of incentives, also in the form of a reduced price.

Another advantage, mainly for large customers with generation, is being able to closely monitor, shift, and balance load in a way that allows the customer to save peak load and not only save on kWh and kW/month but be able to trade what they have saved in an energy market. Again, this involves sophisticated energy management systems, incentives, and a viable trading market.

3.1 Incentives for DR programs

In this scenario, as we described in Section 2, it is very important to correctly determine the incentives to be awarded to prosumers involved in DR programs. Furthermore, since we are defining a decentralised system, it is equally important to determine the penalties to be applied in the event of non-compliance, in order to discourage opportunistic behaviour by participants.

3.1.1 Objectives

We defined a model to calculate a coefficient to be applied to a '*basic*' incentive value to obtain the actual incentive or penalty payment to be applied to a prosumer in a specified timeframe, based on its behaviour. The model takes as input the expected profile ($request$), the associated tolerance range ($range\_inf$ and $range\_sup$), the normal behaviour of the prosumer (i.e. without the flexibility request, $forecast$), and the actual profile followed by the prosumer ($real$).

More in detail, the incentives (or penalties) are calculated in a specified time interval $[,]$. The whole time-interval is discretized in n discrete times t and, for each t, represents the expected power consumption at time t, represents the real power consumption measured at time t, and f_t represents the power consumption forecast for time t. The model is completed by two constants, R and M. R represents the width of the tolerance range, while M is the multiplier applied to calculate the penalties when the prosumer's real consumption is out of the request's tolerance range.

The rationale for the algorithm is to pay an incentive to the prosumer every time d_t is contained inside the tolerance range and to claim for a compensation when is out from this range. The more the actual measured power consumption is close to the requested profile, the greater the incentive will be. Vice-versa, greater deviations from the request will cause greater penalties. Thus, the maximum theoretical incentive is given when the actual profile is following exactly the requested profile, namely =, whereas the incentive will be zero when is on the boundary of the tolerance range. In addition, the incentive, or the penalty, is calculated considering whether is an improvement or a worsening respect the forecasted profile.

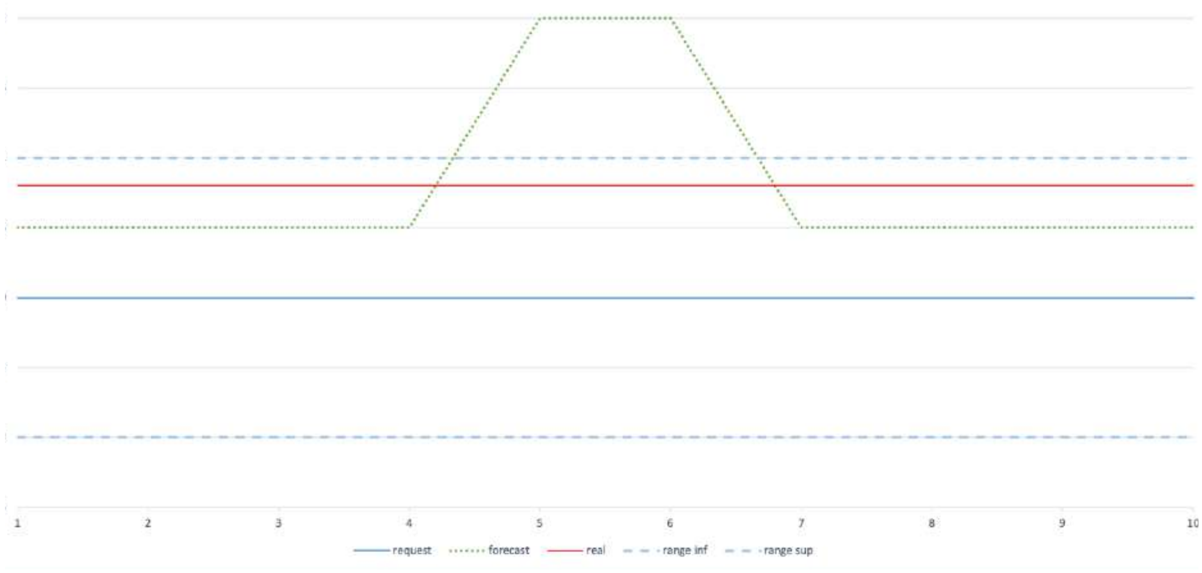


Figure 16. Example 1

The example in Figure 16 shows in blue the request , in green the forecast , and the actual measured value in red. The two dotted lines define the tolerance range associated with the request. In this basic example, the entire is included within the tolerance range but the only improvement of respect of are shown in and in , where the actual profile is closer to the requested profile than the forecasted value. This will result in greater incentives at time and , even if is constant in time.

3.1.2 Dynamic Model

To fulfil the objectives presented in section 3.1.1, we defined the following function:

$$I(t, R, M) =$$

To understand how the model performs, the incentive function $I(t, R, M)$ is applied to the example case shown in Figure 17, and the resulting incentives and penalties are shown in Figure 18.

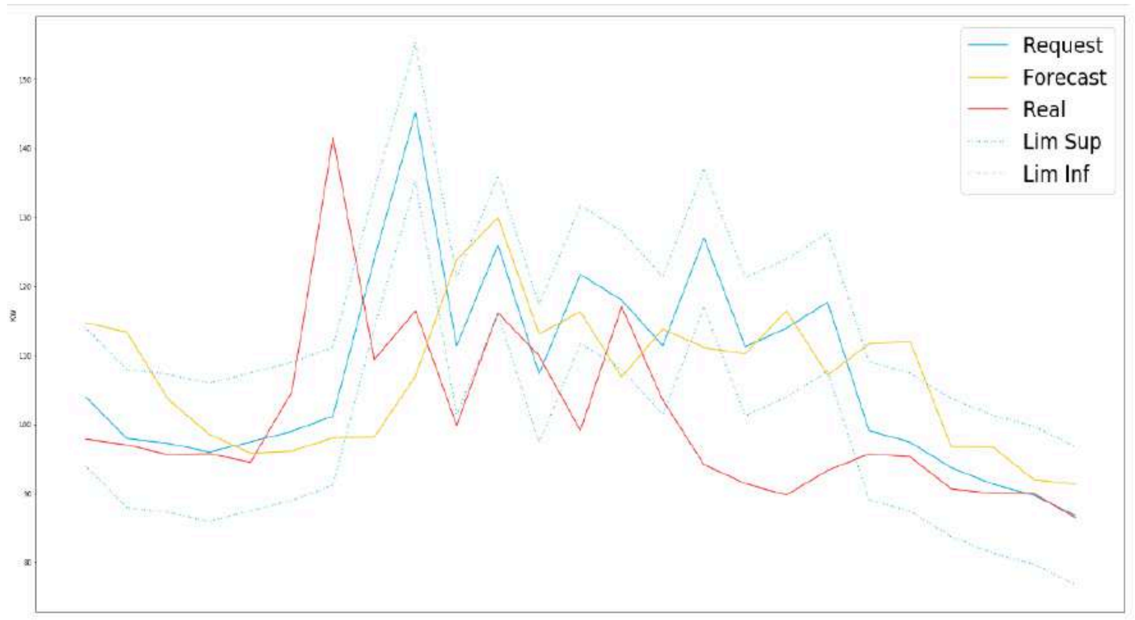


Figure 17. Application of the incentive function $I(t, R, M)$

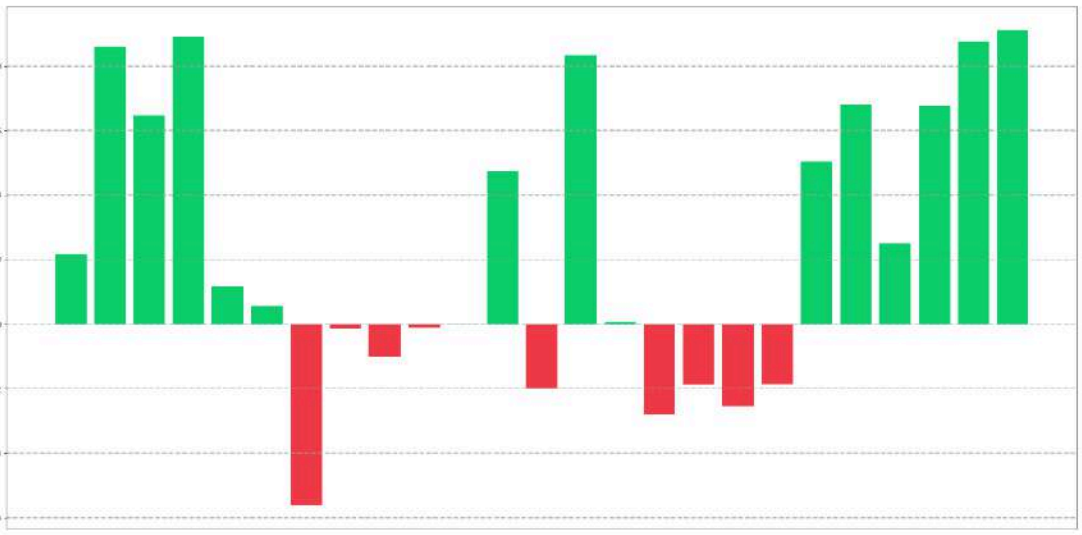


Figure 18. Incentives and penalties calculated for the different timestamps

The resulting incentives and penalties for the first 20 timestamps in the example, evaluated configuring $R = M = 5$, are also shown in Table 4. We awarded an incentive for every timestamp in which the prosumer's profile was contained inside the tolerance range $[-5, +5]$, or

$$| - | < R = 5$$

On the other hand, the prosumer is penalized for every timestamp in which his profile resulted outside of the tolerance range, or

$$> R = 5$$

In the example case, the total cumulative incentive for the prosumers resulted 2.81 times the value of the basic incentive.

T	R_T	F_T	D_T	I
T_1	104016	114769	97867	0,2174
T_2	97951	113394	97051	0,8585
T_3	97272	103651	95630	0,6461
T_4	95998	98579	95763	0,8915
T_5	97477	95840	94528	0,1164
T_6	98997	96149	104721	0,0573
T_7	101229	98077	141569	-0,5612
T_8	123993	98140	109402	-0,0156
T_9	145223	107006	116466	-0,0993
T_{10}	111413	123840	99901	-0,0103

T_{11}	126011	129978	116190	0,0015
T_{12}	107357	113120	109946	0,4729
T_{13}	121745	116336	99100	-0,1992
T_{14}	118029	106905	117098	0,8341
T_{15}	111417	113827	103529	0,0080
T_{16}	127042	111139	94189	-0,2817
T_{17}	111261	110254	91431	-0,1869
T_{18}	113971	116489	89799	-0,2554
T_{19}	117705	107176	93353	-0,1863
T_{20}	99122	111671	95721	0,5032

Table 4. Resulting incentives and penalties

We can see from Table 4 how the incentive is maximum where d_t is close to the request r_t and also represents a significative improvement respect to the forecast (e.g. in $t =$, $t =$). Conversely, we can see how the maximum penalty is applied when is out of tolerance range and represents also a worsening respect of the forecast (e.g. $t =$).

3.2 Experimental Results

The service was developed as a Python module and exposes a dedicated REST API to determine the incentives for a series of time intervals, following the model specified in section 3.1.

The API is described as follows:

$$http(s)://[host]:[port]/incentive/[request\ id]/[forecast\ id]/[actual\ profile\ id]/[M]/[R]$$

where M and R are the constants to apply and request id, forecast id, and actual profile id are the database identifiers representing the requested profile, the forecasted load, and the actual measured prosumer's profile respectively.

The implementation was tested using data from the project's Italian pilot, analysing data gathered every ten minutes from the smart meters located in the ASM district in Terni, Italy. In particular, our simulations were based upon data gathered from the smart meter measuring the consumption of the ASM headquarters building.

A typical scenario is represented in Figure 19. The chart shows the requested profile () in blue, the tolerance range ($\pm R$) in dotted blue, the forecasted consumption () in yellow, and the actual power consumption () in red. In this case we configured $M = 0.2$ and $R = 10kW$.

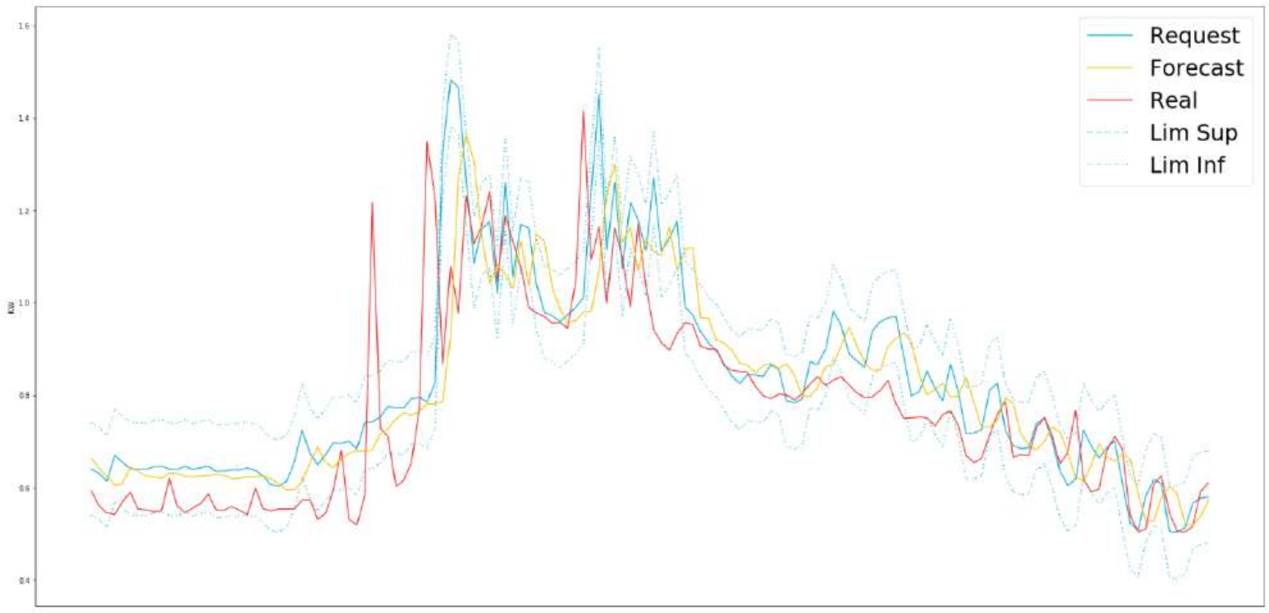


Figure 19. Scenario 1 (07/11/2018)

Figure 20 shows the incentives calculated for each ten minutes window, based on the behaviour represented above. The incentives are represented in green, the penalties in red.

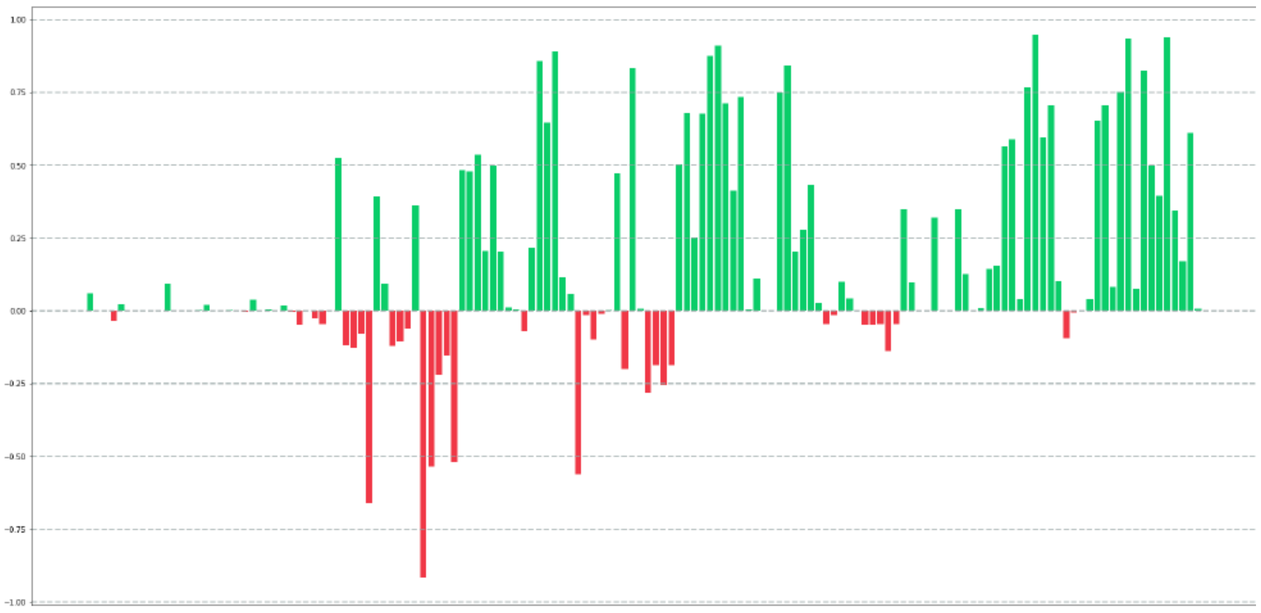


Figure 20. Incentives for scenario 1

In this case, the prosumer was awarded a total incentive of about 23.5 times the basic incentive.

In the second example, represented in Figure 21, we can see how narrowing the tolerance range affects the final results, even without modifying the multiplier M. For the second scenario, we configured $R = 2kW$ and $M = 0.2$.

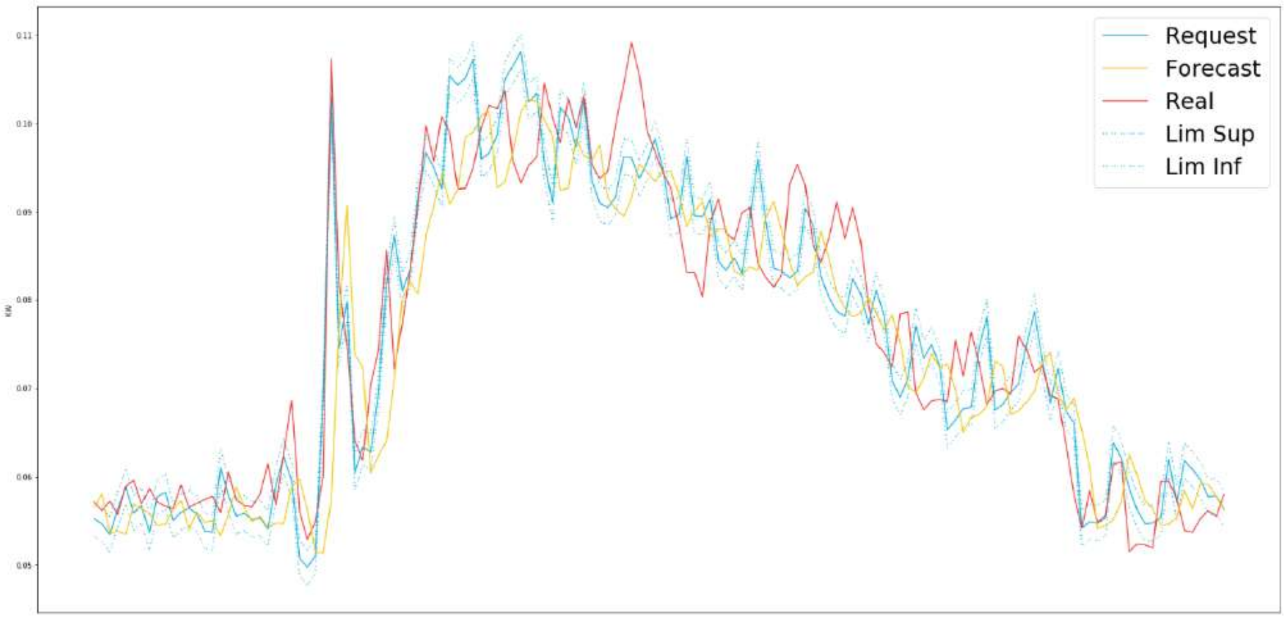


Figure 21. Scenario 2 (18/04/2019)

In this scenario, as we can see from Figure 22, the sum of the different incentives applied for each time window results -9.55 times the basic incentive, meaning that the prosumer was penalized for his behaviour.

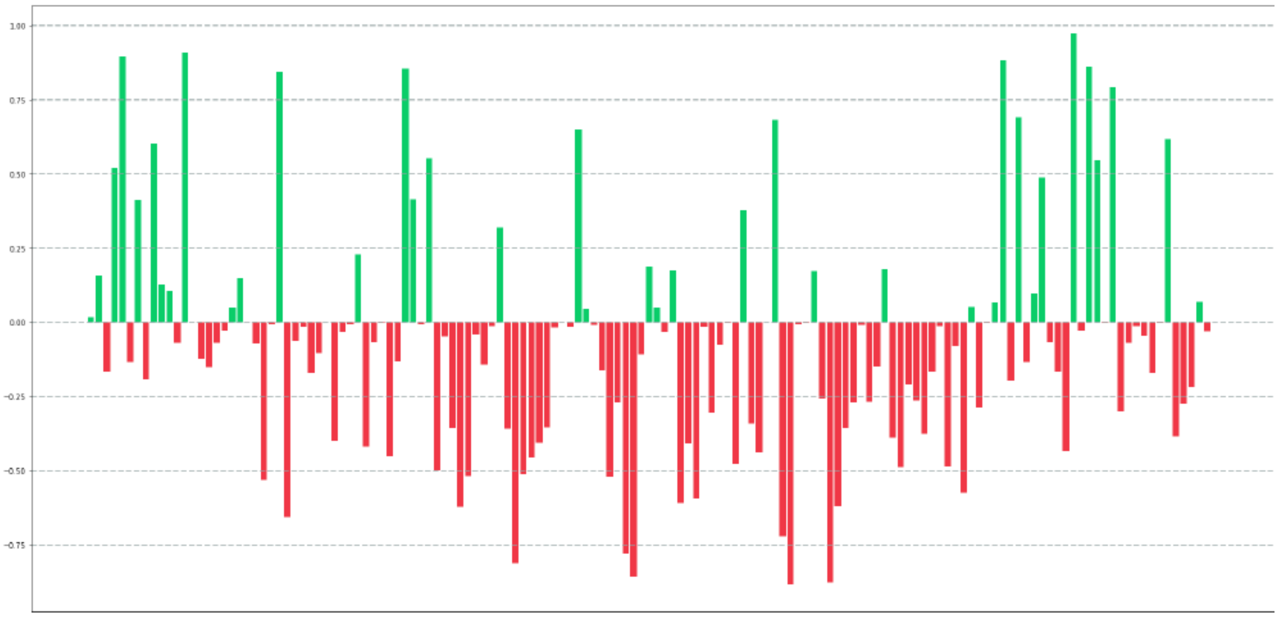


Figure 22. Incentives for scenario 2

In the final example, represented in Figure 23, we can see how the multiplier M affects the final results, using $R = 5kW$ and $M = 1$.

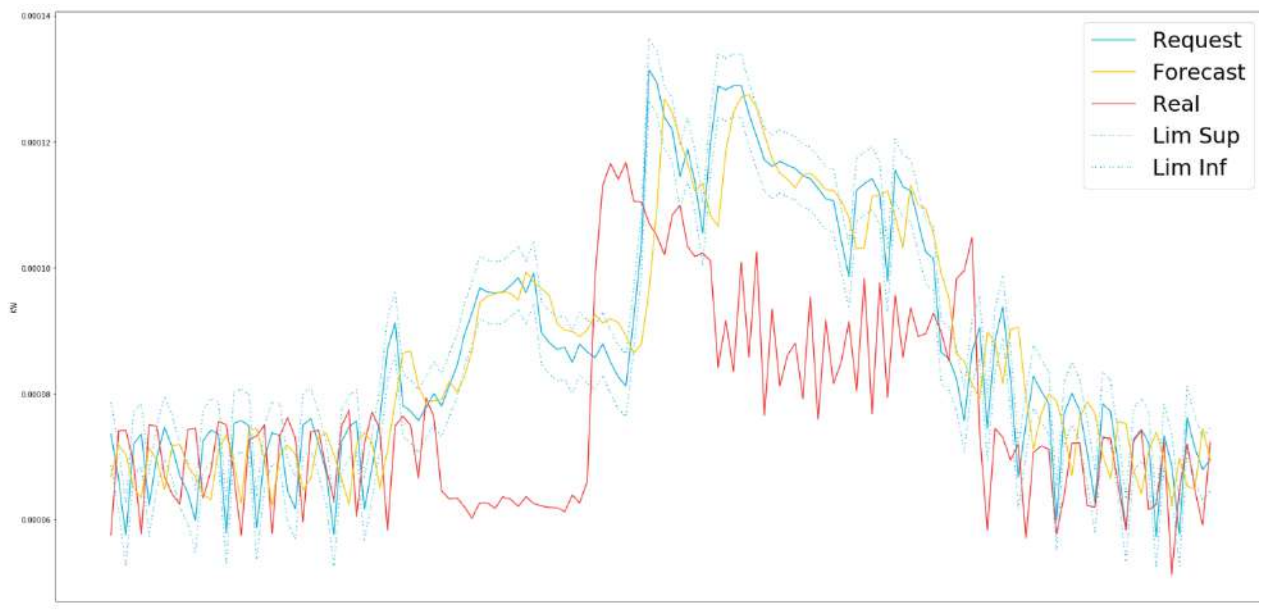


Figure 23. Scenario 3 (11/01/2020)

We can see from Figure 24 how, in this case, the penalties applied are visibly higher, in absolute value, compared with the previous scenarios.

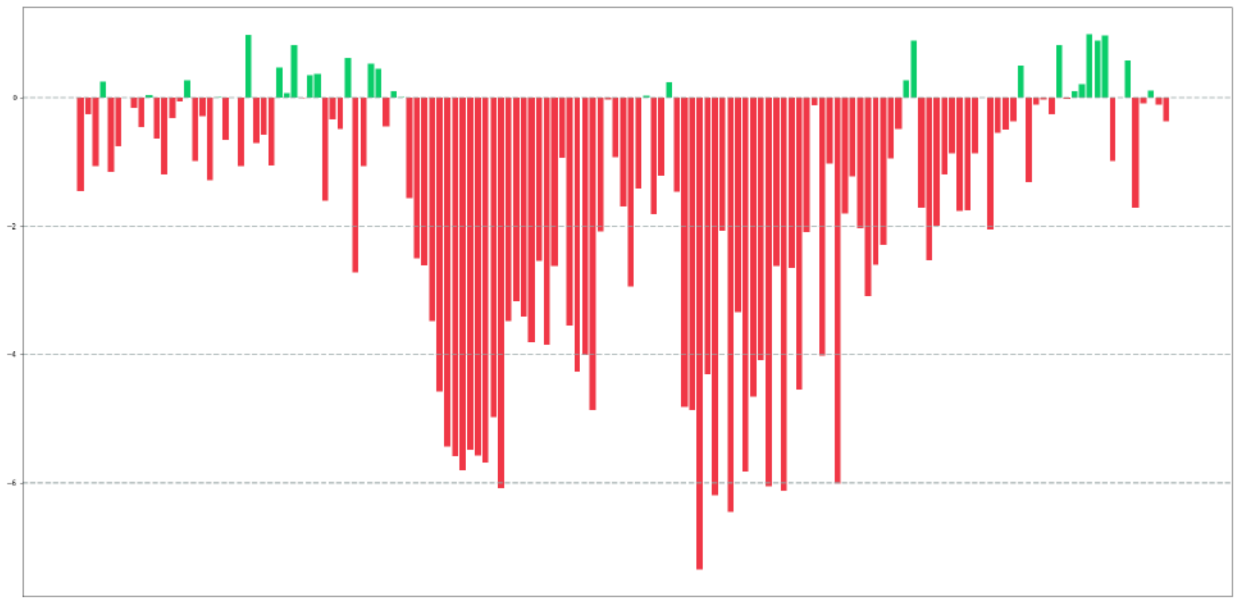


Figure 24. Incentives for scenario 3

The final result is, in fact, a penalization for about 247 times the basic incentive value.

4 Conclusions

In this deliverable, we have presented the definition of consensus-based techniques energy transactions and DR programs validation and settlement as well as a model for dynamic financial incentivisation of prosumers in decentralized DR programs. This work completes the definition of the blockchain-based platform for the distributed control and management of energy micro-grid, providing a mechanism for validating the energy transaction and a model for calculating the incentives, or penalties, for each prosumer involved.

The experimental results show that is possible to determine, to a very good degree, the presence of tampered energy data and associated transactions by means of consensus avoiding their submission on the blockchain. Also using smart contracts, the DR programs energy imbalances settlement is brought as close as possible to the real time making for the aggregator to identify in a reliable way the compliant prosumers or any possible flexibility deviations generated by the non-compliant ones.

Aggregators may capitalize on this information mitigating the effects deviations via the blockchain smart contracts and dynamic incentivization, imposing sanctions to the untrustworthy prosumers, and rewarding the fair prosumers with incentives. To determine the amount of the incentives or penalties to assign, we implemented a dynamic model considering also, in addition to the distance from the requested profile, the behaviour of the prosumer as the effort made by the prosumer to fulfil the request, compared to the normal prosumer's behaviour.

References

[1] SIFT: design and analysis of a fault-tolerant computer for aircraft control". Microelectronics Reliability. 19 (3): 190. 1979. doi:10.1016/0026-2714(79)90211-7. ISSN 0026-2714

[2] Pease, M., Shostak, R., and Lamport, L. (1980). Reaching agreement in the presence of faults. Journal of the ACM, 27(2):228–234.

[3] Coulouris, George; Jean Dollimore; Tim Kindberg (2001), Distributed Systems: Concepts and Design (3rd Edition), Addison-Wesley, p. 452, ISBN 978-0201-61918-8

[4] Available Online: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ (last accessed 3rd of March 2020)

[5] Available Online: https://github.com/ethereum/wiki/wiki/Sharding-FAQ (last accessed 3rd of March 2020)

[6] Fishcer, Lynch, Paterson, Impossibility of Distributed Consensus with One Faulty Process, 1985

[7] Dwork, Lynch, Stockmeyer, Consensus in the Presence of partial Synchrony, Journal of the Association for Computing Machinery, Vol. 35, No. 2, April 1988, pp. 288-323.

[8] Zhong Fan, Parag Kulkarni, Sedat Gormus, Costas Efthymiou, Georgios Kalogridis, Mahesh Sooriyabandara, Ziming Zhu, Sangarapillai Lambotharan, and Woon Hau Chin, Smart Grid Communications: Overview of Research Challenges, Solutions, and Standardization Activities, https://arxiv.org/pdf/1112.3516.pdf

[9] Leslie Lamport, Shostak, Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, Pages 382-401

[10] Hopkins, Albert L.; Lala, Jaynarayan H.; Smith, T. Basil (1987). "The Evolution of Fault Tolerant Computing at the Charles Stark Draper Laboratory, 1955–85". The Evolution of Fault-Tolerant Computing. Dependable Computing and Fault-Tolerant Systems. 1. pp. 121–140. doi:10.1007/978-3-7091-8871-2_6. ISBN 978-3-7091-8873-6. ISSN 0932-5581

[11] Driscoll, Kevin; Papadopoulos, Gregory; Nelson, Scott; Hartmann, Gary; Ramohalli, Gautham (1984), Multi-Microprocessor Flight Control System (Technical Report), Wright-Patterson Air Force Base, OH 45433, USA: AFWAL/FIGL U.S. Air Force Systems Command, AFWAL-TR-84-3076

[12] Sompolinsky Y., Zohar A. (2015) Secure High-Rate Transaction Processing in Bitcoin. In: Böhme R., Okamoto T. (eds) Financial Cryptography and Data Security. FC 2015. Lecture Notes in Computer Science, vol 8975. Springer, Berlin, Heidelberg

[13] Castro, M.; Liskov, B. (2002). "Practical Byzantine Fault Tolerance and Proactive Recovery". ACM Transactions on Computer Systems. Association for Computing Machinery. 20 (4): 398–461. CiteSeerX 10.1.1.127.6130

[14] Clement, A.; Wong, E.; Alvisi, L.; Dahlin, M.; Marchetti, M. (April 22–24, 2009). Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults (PDF). Symposium on Networked Systems Design and Implementation. USENIX. Archived

[15] Aublin, P.-L.; Ben Mokhtar, S.; Quéma, V. (July 8–11, 2013). RBFT: Redundant Byzantine Fault Tolerance. 33rd IEEE International Conference on Distributed Computing Systems. International Conference on Distributed Computing Systems. Archived from the original on August 5, 2013.

[16] Abd-El-Malek, M.; Ganger, G.; Goodson, G.; Reiter, M.; Wylie, J. (2005). "Fault-scalable Byzantine Fault-Tolerant Services". ACM Sigops Operating Systems Review. Association for Computing Machinery. 39 (5): 59. doi:10.1145/1095809.1095817

[17] Cowling, James; Myers, Daniel; Liskov, Barbara; Rodrigues, Rodrigo; Shrira, Liuba (2006). HQ Replication: A Hybrid Quorum Protocol for Byzantine Fault Tolerance. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation. pp. 177–190. ISBN 1-931971-47-1.

[18] Kotla, Ramakrishna; Alvisi, Lorenzo; Dahlin, Mike; Clement, Allen; Wong, Edmund (December 2009). "Zyzzyva: Speculative Byzantine Fault Tolerance". ACM Transactions on Computer Systems. Association for Computing Machinery. 27 (4): 1–39. doi:10.1145/1658357.1658358

[19] Guerraoui, Rachid; Kneževic, Nikola; Vukolic, Marko; Quéma, Vivien (2010). The Next 700 BFT Protocols. Proceedings of the 5th European conference on Computer systems. EuroSys. Archived from the original on 2011-10-02. Retrieved 2011-10-04.

[20] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008 (Available Online: https://bitcoin.org/bitcoin.pdf)

[21] Wahab, Memood, Survey of Consensus Protocols, 2018

[22] A Survey of Consensus Algorithms in Crypto : Available Online: https://medium.com/@sunflora98/a-survey-of-consensus-algorithms-in-crypto-e2e954dc9218 (last accessed 3rd of March 2020)

[23] RAFT algorithm: https://raft.github.io/ (last accessed 3rd of March 2020)

[24] Lamport, The Part-Time Parliament, ACM Transactions on Computer Systems 16, 2 (May 1998), 133-169.

[25] Lamport, Byzantizing Paxos by Refinement, 2010

[26] W. Zhao, "A Byzantine Fault Tolerant Distributed Commit Protocol," Third IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC 2007), Columbia, MD, 2007, pp. 37-46. doi: 10.1109/DASC.2007.10

[27] Jakobsson Markus.,Juels Ari ," Proofs of Work and Bread Pudding Protocols", Communications and Multimedia Security,pp 258-272, 1999

[28] Shafi Goldwasser, Silvio Micali, Charles Rackoff, "The knowledge complexity of interactive proof-systems" , Proceedings of 17th Symposium on the Theory of Computation, Providence, Rhode Island. 1985

[29] Bitcoin Mining Hardware ASICs, 2019, Available online at https://www.buybitcoinworldwide.com/mining/hardware/, Last accessed on July 2019

[30] Colin Percival, "Stronger key derivation via sequential memory-hard functions", Self-published, Available online at https://www.tarsnap.com/scrypt/scrypt.pdf, 2009

[31] Pedro Franco, "Understanding Bitcoin: Cryptography, Engineering and Economics", ISBN: 978-1-119-01916-9, 2014

[32] Vitalik Buterin. Dagger: A memory-hard to compute, memory-easy to verify scrypt alternative. Tech Report, hashcash.org website, 2013

[33] Dryja, Thaddeus. "Hashimoto: I/O bound proof of work"(PDF). Semantic Scholar. Archived on 2017, Available online at

https://web.archive.org/web/20170810043640/https://pdfs.semanticscholar.org/3b23/7cc60c1b9650e260318d33bec471b8202d5e.pdf, Last accessed on July 2019

[34] Alex Biryukov, Dmitry Khovratovich," Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem", Cryptology ePrint Archive: Report 2015/94. Available online at https://eprint.iacr.org/2015/946

[35] John Tromp, "Cuckoo Cycle: a memory bound graph-theoretic proof-of-work", International Conference on Financial Cryptography and Data Security, pp 49-62, 2015

[36] GRIN, Available online at https://github.com/ignopeverell/grin, Last accessed on July 2019

[37] AEternity , Available online at http://www.aeternity.com/, Last accessed on July 2019

[38] CureCoin, Available online at https://www.curecoin.net/, Last accessed on July 2019

[39] Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. (2014). Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. IACR Cryptology ePrint Archive, 2014, 452.

[40] Jepson, Christina. "DTB001: Decred Technical Brief." Available at https://cryptorating.eu/whitepapers/Decred/decred.pdf, Last accessed on July 2019

[41] Buterin, V., & Griffith, V. (2017). Casper the friendly finality gadget. arXiv preprint arXiv:1710.09437.

[42] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14). 2014.

[43] Larimer, D. (2014). Delegated proof-of-stake (dpos). Bitshare whitepaper. Available online at https://bitshares.org/technology/delegated-proof-of-stake-consensus/ Last accessed on July 2019

[44] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2018). Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain

[45] Intel: Sawtooth Lake (2017). https://intelledger.github.io/

[46] Available Online: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787125445/1/ch01lvl1sec11/cap-theorem-and-blockchain (last accessed 05 march 2020)

[47] J. Chen, S. Micali, Algorand: A secure and efficient distributed ledger, Theoret. Comput. Sci. 2019

[48] Elrond A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake, Available at: https://elrond.com/ Last accessed on March 2020

[49] M. Yin, HotStu: BFT Consensus in the Lens of Blockchain, 2019, Available at: https://blogs.vmware.com/research/2018/11/06/hotstuff-bft-consensus-in-the-lens-ofblockchain/ Last accessed on March 2020

[50] V. Buterin Slasher: A Punitive Proof-of-Stake Algorithm, Available at: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/, 2014 accessed on March 2020

[51]I. Bentov, Cryptocurrencies without Proof of Work, Available at: https://fc16.ifca.ai/bitcoin/papers/BGM16.pdf 2017 Last accessed on March 2020

[52] Available Online: https://medium.com/@CryptoAssassin4/algorands-resiliency-against-adversarial-attacks-f4cf135a4309 (last accessed 10th of March 2020)

[53] Y. Gilad , Algorand: Scaling Byzantine Agreements for Cryptocurrencies, MIT CSAIL, 2018

[54] Available Online: https://github.com/ethereum/wiki/wiki/Sharding-FAQ#but-doesnt-the-cap-theorem-mean-that-fully-secure-distributed-systems-are-impossible-and-so-sharding-is-futile (Last accessed 11 march 2020)

[55] Vlad Zamfir, Introducing the \Minimal CBC Casper" Family of Consensus Protocols, 2018

[56] Aura - Authority Round - Wiki, https://wiki.parity.io/Aura

[57] Pierluigi Siano Demand response and smart grids—A survey

[58] Smart Grid projects in Europe: Lessons learned and current developments.

Available Online: https://ses.jrc.ec.europa.eu/sites/ses.jrc.ec.europa.eu/files/documents/ld-na-25815-en-n_final_online_version_april_15_smart_grid_projects_in_europe_-_lessons_learned_and_current_developments_-2012_update.pdf (Last accessed 19 march 2020)