Applied Energy 282 (2021) 116208

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

A secure distributed ledger for transactive energy: The Electron Volt Exchange (EVE) blockchain

Shammya Saha [a,*], Nikhil Ravi [a], Kári Hreinsson [a], Jaejong Baek [b], Anna Scaglione [a], Nathan G. Johnson [c]

[a] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA
[b] Laboratory of Security Engineering For Future Computing and with Center for Cyber-Security and Digital Forensics, Arizona State University, Tempe, AZ, USA
[c] The Polytechnic School, Arizona State University, Mesa, AZ, USA

ARTICLE INFO

Keywords:
Blockchain
Cyber–physical system
Cyber-security
Distribution system
Hyperledger fabric
Transactive energy

ABSTRACT

The adoption of blockchain for Transactive Energy has gained significant momentum as it allows mutually non-trusting agents to trade energy services in a trustless energy market. Research to date has assumed that the built-in Byzantine Fault Tolerance in recording transactions in a ledger is sufficient to ensure integrity. Such work must be extended to address security gaps including random bilateral transactions that do not guarantee reliable and efficient market operation, and market participants having incentives to cheat when reporting actual production/consumption figures. Work herein introduces the Electron Volt Exchange framework with the following characteristics: (1) a distributed protocol for pricing and scheduling prosumers' production/consumption while keeping constraints and bids private, and (2) a distributed algorithm to prevent theft that verifies prosumers' compliance to scheduled transactions using information from grid sensors (such as smart meters) and mitigates the impact of false data injection attacks. Flexibility and robustness of the approach are demonstrated through simulation and implementation using Hyperledger Fabric.

1. Introduction

The proliferation of Distributed Energy Resources (DERs), Electric Vehicles (EVs), grid-level energy storage, and networked grid-edge devices requires a trustworthy open energy trading platform for participants – i.e., a Transactive Energy (TE) framework. TE combines financial signals and dynamic control techniques to shift the timing and quantity of energy usage to achieve greater efficiency, increased use of renewable energy, reduced energy costs, and improved flexibility to manage shifts in net load locally. Such benefits have motivated the increasing body of research whose goal is to manage real-time demand and electricity supply in an open market where prosumers and utilities interact to establish a market-clearing price. Examples are the auction mechanisms proposed in [1], algorithms for co-simulation of transmission and distribution networks combined markets [2], multi-agent models capturing trading behaviors [3], and thermostatically controlled loads to participate in TE markets [4], to name a few examples.

Recently, many researchers have purported blockchain as the ideal enabling platform to implement TE. Blockchain can enhance cyber-security and traceability of Peer-to-Peer (P2P) transactions between mutually non-trusting parties in the TE marketplace [5]. There are several benefits to such an implementation: (1) once stored on the ledger, all transactions are transparent to all participants through an identical copy of the ledger, (2) new transactions are *hash-chained* when appended to the ledger, an operation that makes them immutable, mitigating cyber-attacks aimed at reducing the integrity and availability of the data, and (3) all functional aspects of TE enabled by blockchain, from bidding to pricing to billing, can be orchestrated running *Smart Contracts* [6].

1.1. Contribution

Blockchain ensures transparency and immutability of bidding and trading records in the ledger. However, records in the ledger have security gaps during the submission of bids and the verification of contractual obligations. In fact:

1. Threats exist internal to TE approaches because selfish players have an intrinsic incentive to cheat on reported consumption needs or production capacity during market clearing [7].
2. Ex-post, if the market stakeholders control smart meters, they can inject false data to hide discrepancies that would otherwise reveal cheating.

* Corresponding author.
E-mail address: shammya.saha@asu.edu (S. Saha).

https://doi.org/10.1016/j.apenergy.2020.116208
Received 21 July 2020; Received in revised form 4 November 2020; Accepted 7 November 2020
Available online 16 November 2020
0306-2619/© 2020 Elsevier Ltd. All rights reserved.

Nomenclature

$\mathcal{A}$	Calligraphic letters are sets
$\lvert\mathcal{A}\rvert$	Denotes the cardinality of set $\mathcal{A}$
T	Number of intervals in the decision horizon
$\mathcal{N}$	Set of aggregators
N	Number of aggregators
$\mathcal{B}$	Set of buses/nodes b in the power grid
$\mathcal{B}^{(n)}$	Subset of buses managed by aggregator $n \in \mathcal{N}$
$\mathcal{G}_e$	Electric grid graph
$\mathcal{G}_c$	Communication network graph
$\mathcal{E}_e$	Set of edges/lines connecting the set of buses $\mathcal{B}$ in $\mathcal{G}_e$
$\mathcal{E}_c$	Set of communication links between aggregators $\mathcal{N}$
$\mathcal{T}$	Set of time intervals in the decision horizon
GL	Global ledger
LL_n	Local ledger for aggregator $n \in \mathcal{N}$
$\boldsymbol{x}$	Boldfaced lower-case letters denote vectors and x_i denotes the ith element of a vector $\boldsymbol{x}$.
$\boldsymbol{X}$	Boldfaced upper-case letters denote matrices and X_{ij} denotes the ijth entry of a matrix $\boldsymbol{X}$
$\boldsymbol{X}^\intercal$	Transpose is denoted by $^\intercal$, so $\boldsymbol{X}^\intercal$ is the transpose of $\boldsymbol{X}$
$\boldsymbol{X}^\dagger$	Is the pseudo-inverse of $\boldsymbol{X}$, where $\boldsymbol{X}^\dagger = (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal$

To address these issues we propose the **Electron Volt Exchange (EVE)** blockchain architecture (Fig. 1). Novelty compared to other TE blockchain research lies in the following components:

1. TE blockchain designs commonly consider only bilateral transactions. Instead, the EVE approach utilizes a decentralized solution for the entire market economic dispatch problem whose formulation falls in the class of network utility maximization problems, first proposed for real time pricing in [8] (Section 3). The closest to our approach is found in [9], where the authors have incorporated controllable loads and generation to develop an iterative pricing algorithm using a smart contract that updates global variables of the distributed optimal power flow problem. The scheme still relies on a central update of variables to achieve convergence. Compared to [9], this work incorporates renewable generation, thermostatically controlled loads (TCLs), storage devices, deferrable appliances (DAs), and electric vehicles (EVs) and relies on a hierarchical, distributed architecture including aggregators [10] to "divide and conquer" the communication problem, avoiding congestion and yielding a scalable implementation for optimal price calculation.
2. The EVE architecture includes the first blockchain-based, distributed Robust State Verification (RSV) mechanism for TE transactions, where physical sensor measurements are cross-validated in a decentralized fashion to ensure prosumers abide by their market commitment (Section 4). Our algorithm, inspired by the work [11] on distributed state estimation in adversarial settings, is shown to be robust against False Data Injection Attacks (FDIAs) aimed at TE market theft.
3. The pricing and verification algorithms are tested via numerical simulations in Section 5. Implementation of the EVE blockchain framework onto a distributed ledger [12] is described in Section 6 using the open-source Hyperledger Fabric (HLF) framework. It includes a customized BFT-SMART [13] consensus protocol to provide security and improve performance with Byzantine Fault Tolerance (BFT) in an untrustworthy environment. This improves upon standard security features of HLF, such as Membership Service Provider (MSP), Fabric CA (Certificate Authority), Attribute-Based Access Control (ABAC) [14], and others that address common security concerns. Bench-marking for the proposed smart contracts using Hyperledger Caliper [15] is also described herein.

1.2. Related work

There is a substantial body of research and industrial efforts in TE; we focus our literature review on the relatively recent trend that includes blockchain as the backbone for managing P2P communications for market-related operations. These operations involve handling and securing prosumer bids and dispatch values, deciding a market-clearing algorithm, ensuring the balance between demand and supply to meet network constraints, ensuring cyber-security, and more. Readers are referred to [16] for a broad discussion on the applications of blockchain for smart grid Cyber–physical infrastructure, to [17] for potential benefits of blockchain for grid resiliency against cyber-attacks, and to [18] for details on how smart inverter, advanced metering infrastructure, and energy coordinator can support the digitization and decentralization of TE.

Existing literature has focused on co-simulation of the physical grid in tandem with blockchain TE implementation [19], quantifying energy losses caused by energy transactions in an energy blockchain [20], enforcing proportional fairness among DERs participating in voltage control through smart contracts to ensure voltage stability [21], developing a blockchain based energy trading platform for electric vehicles [22], integrating energy and carbon markets through a blockchain based trading framework [23], and more. The two-layer blockchain implementation in [24] consists of a first layer with smart meters forming a private blockchain and a second layer with aggregators forming a consortium blockchain to coordinate energy transactions within regions. These works assume security is implicit in transactions between aggregators and flexible demand assets. The authors in [25] use a continuous double auction mechanism, but the transaction mechanism exposed the prosumer's unique ID, posing a privacy concern. Similar concerns described in [26] indicate a public blockchain exposes the transactions and balances of each prosumer, and further, that the rate of transaction processing limits scalability. Authors in [27] present a double auction mechanism for localized energy exchanges between EVs, where local aggregators publicly audit and share transaction records without relying on a trusted third party. The use of local aggregators also appears in [28] to create an energy market that combines blockchain and IoT for two flexible community market players: an EV community and a DER community. In [9], the authors propose a blockchain implementation to clear the market using the Alternative Direction Method of Multipliers (ADMM) in a master–slave distributed architecture, where a central aggregator/master node updates global variables. The multi-layer smart contract implementation based on Ethereum in [29] addresses the mismatch in the settlement between *System Operators* and *Balance Responsible Parties*, yet it does not decentralize the Independent System Operator (ISO) nor include verification. Work done in [30] uses Hyperledger Composer to demonstrate a blockchain based TE market implementation while excluding a market clearing price calculation and assuming energy transactions are automatically verified. A prototype implementation of a TE market using Hyperledger Fabric for metering and billing purposes is proposed in [31], yet pricing and verification were not addressed. A related study that uses Hyperledger Composer in [32] determines the market clearing price by averaging bid prices offered by all buyers while sorting sellers by first-in, first-out basis. An approach similar to [32] using an Ethereum based blockchain architecture is found in [33]. However, such algorithms can be easily exploited by malicious prosumers or attackers to manipulate the clearing price and destabilize the TE

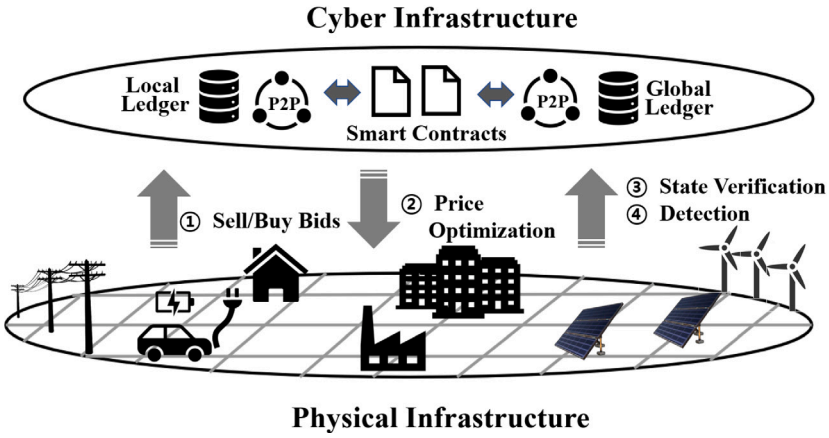

Fig. 1. Conceptual architecture of EVE illustrating cyber and physical layers.

market. Malicious prosumers can similarly influence the co-simulation framework presented in [34] where the ratio of total generation and consumption reported by the prosumers is used to determine the price. The private blockchain solution proposed in [35] requires a match between the energy producer and consumer regarding the amount of power to be generated and consumed, respectively, which is not practical for many prosumers and can violate physical constraints on the distribution network. The blockchain based energy trading model in [36] also lacks sufficient protection from physical constraint violations because the approach allows for an open trading platform across diverse types of power sellers without any optimized market pricing. In most studies, Smart Contracts orchestrate information exchanges among participants and during recording transactions, while still requiring a central entity to be in charge of calculating the market-clearing price in contrast to our fully decentralized solution. Also, prior methods focusing on security aspects and countermeasures against cyber-attacks to the market-clearing mechanism have ignored physical verification that must be tied to market records to work effectively as a continuous deterrent to theft. Relevant to this study are also prior works integrating smart metering with blockchain, specifically energy trading applications [37]. Even if they leverage the immutability of blockchain, these approaches leave data integrity and privacy concerns unresolved [38].

To address these gaps, our work incorporates insights from the considerable body of research that has been developed in cyber-security of electric power measurement systems to encompass stealth FDIAs in state estimation [39], non-stealth state estimation attacks such as data jamming [40], bias injection attack [41], and denial of service attacks [11]. To the best of our knowledge, the only work that discusses possible attack scenarios in blockchain-based energy trading is [42]; the scenarios mentioned by the authors include a malicious stakeholder attempting to modify market operations to produce an inaccurate clearing price, a malicious market operator attempting to modify operations of the market algorithm, and a malicious outsider trying to remotely tamper with communications among TE market participants. The RSV presented in this work addresses these security concerns to enhance the blockchain-based TE framework's cyber-security. The market modeling approach in [42] uses blockchain only to collect bid information from the prosumers and then utilizes a centralized architecture for determining the market clearing price. On the contrary, this work uses blockchain to manage prosumers, while the proposed decentralized price optimization algorithm uses the decentralized architecture of blockchain to run the iterative price determination algorithm. Moreover, using the inherent security features of Hyperledger Fabric and ABAC allows EVE to avoid the complex attribute based encryption for transaction security introduced in [43] while still achieving the same level of privacy.

2. EVE as a cyber–physical system architecture

Fig. 1 represents the interactions tied to the TE application layer for a generic blockchain based Cyber–physical infrastructure implementation. The specifics of our architecture are summarized in the following sections.

2.1. Physical infrastructure

The physical infrastructure includes:

- The electrical grid modeled as a connected graph $\mathcal{G}_e = (\mathcal{B}, \mathcal{E}_e)$. Lines and transformers are characterized by admittance parameters $y_{ij},\ \forall\ ij \in \mathcal{E}_e$.[1]
- Market Participants (prosumers and aggregators).
- Electrical loads, distributed generation, and storage assets that connect to buses on the electrical network. For clarity in our formulations, we model a single prosumer per bus b, making them equivalent.[2]
- Electrical sensors and control equipment.

2.2. Application layer

In EVE, scalability is achieved by dividing the application layer entities into agents with distinct tasks.

- The bottom layer includes *prosumers* who can buy and/or sell energy and control flexible loads, storage, and generation assets connected to the physical network. This layer can be broken into several physical regions.
- The top layer includes a set of local *aggregators* ($\mathcal{N}$) managing the prosumers connected to a subset of buses $\mathcal{B}^{(n)} \subset \mathcal{B}$ for all $n \in \mathcal{N}$.

The following remark is in order:

Remark 2.1. Individual prosumers have traditionally been unable to participate in energy markets, but aggregators may have access through the lumped capacity bids [44].[3] Aggregators can act as intermediaries between small consumers/producers and volatile markets, and thereby provide hedging solutions to reduce risk to individual market

[1] Voltage control and protective equipment are ignored because these do not have a direct impact on market operation and hence are not required for the description of the EVE architecture.

[2] In cases where multiple consumers connect to the same physical bus, we model those as separate buses connected through zero impedance edges, but omit this in diagrams for simplicity.

[3] A practical example can be found in [45].

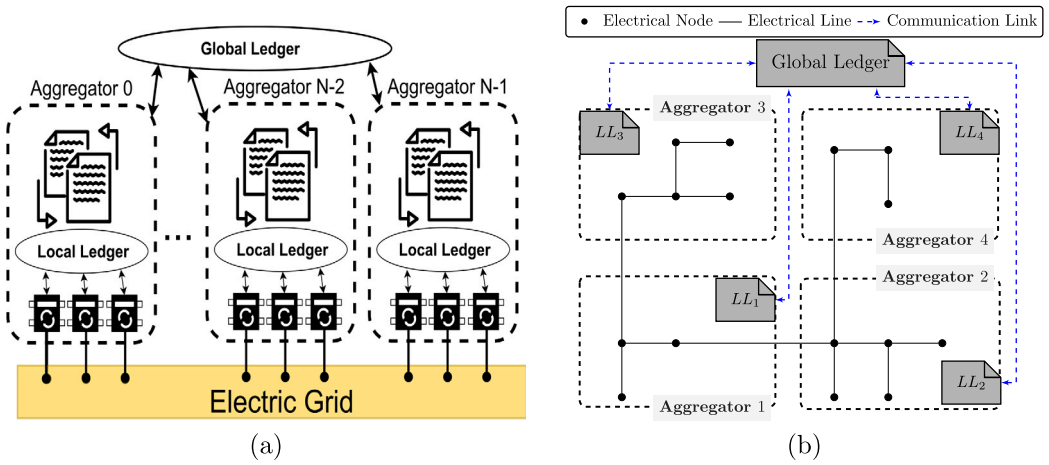

Fig. 2. (a) The hierarchical distributed ledger architecture. (b) Cyber infrastructure overlaid with the physical structure for a sample distribution network with 4 aggregators.

participants [46]. Aggregators can procure demands from consumers and sell to purchasers through trading frameworks proposed in prior works [47]. Moreover, resources needed for price optimization and verification processes can be more easily be obtained and justified for aggregators rather than each prosumer. Aggregators do not complicate management because blockchain is suited for a decentralized architecture. A solution without aggregators would otherwise increase communication latency and computation time for market-clearing as new prosumers are added, reducing scalability.

Policies in EVE can be divided into three main classes:

- **Pricing**: Policies deciding the optimum prosumers schedule and the price.
- **Verification**: Policies processing sensor information to verify that load/generation is correctly reported, contractual obligations have been met and billed accordingly.
- **Billing**: Policies for billing and ensuring compliance.

2.3. Cyber infrastructure

The cyber infrastructure includes security policies for settling transactions, communication/computation resources, and data archival based on blockchain. Building the cyber network requires all market participants to work together as a consortium using a set of policies agreed to during network initialization to determine the participants' permissions. For the shared database or ledger within the cyber architecture, EVE uses CouchDB [48] as it supports rich queries when data values are modeled as JSON. The cyber framework is generic to include or exclude Transport Layer Security (TLS); however, we recommend including TLS for additional security.

Application of the policies mentioned in Section 2.2 is handled through distributed ledgers. In EVE, a ledger consists of (a) a database that holds current values of a set of ledger states, and (b) a transaction log that records all changes that have resulted in the current system state. Our implementation of EVE consists of two types of distributed ledgers, a Local Ledger (LL) and a Global Ledger (GL). The smart contracts in this work handle interactions between the ledgers (GL and LL) and external applications to complete every transaction within EVE. Fig. 2a shows the hierarchical architecture of those ledgers whereas Fig. 2b overlays the distribution of physical nodes into aggregator zones. An aggregator uses the LL to collect bids submitted by prosumers, verify local state information, and update individual prosumer budgets after verification. Aggregators access the GL for distributed pricing and verification algorithms and for sharing global information with other aggregators. All information exchanges and history between participants are handled through smart contracts for reading, writing, and storing in distributed ledgers.

Fig. 3 depicts the implementation of policies under a single solving window $\mathcal{T}_i$ divided into three stages. Stage 1 refers to the Open Bidding Window in which prosumers submit bids for $\mathcal{T}_{i+1}$. Aggregators execute the verification algorithm based on measurements collected for $\mathcal{T}_{i-1}$. Hence stage 1 includes execution of the verification policy and then the billing policy. Stage 2 refers to the Closed Bidding Window in which the aggregators execute the distributed pricing algorithm for $\mathcal{T}_{i+1}$. Stage 3 refers to the Update Dispatch Window in which aggregators update the prosumers' schedules.

In Sections 3 and 4 the paper details the pricing algorithm and verification policies respectively, along with mechanisms that ensure the integrity of the decision-making process. The numerical performance via simulations in Section 5 and the implementation of the cyber architecture on HLF described in Section 6 conclude the paper.

3. EVE distributed pricing algorithm

Pricing and scheduling decisions are illustrated for a single solving window, as shown in Fig. 3. The period is split into T smaller discrete intervals of unit duration, all in the set $\mathcal{T}_i = \{iT, \ldots, (i+1)T-1\}$. Within each period, $\mathcal{T}_i$, bus b is connected to assets that either supply or demand power. Net real power generation of bus b at time t is denoted by $p_b(t)$. Positive and negative values of $p_b(t)$ indicate that bus b is supplying power to the grid or consuming from the grid, respectively. Neglecting losses, the total power schedule managed by aggregator $n \in \mathcal{N}$, $p^{(n)}(t)$, is the sum of power from each individual bus $p_b(t)$ associated with it:

$$p^{(n)}(t) = \sum_{b \in \mathcal{B}^{(n)}} p_b(t). \tag{1}$$

Each component $p_b(t)$ must have a certain cost (disutility) and must satisfy a set of constraints that depend on the generation and storage capacity available at the supply side, as explained in Section 3.1, that determines the optimal schedule. In Section 3.2, we describe a decentralized dual decomposition algorithm to solve a power balancing problem between different aggregators with a dual variable reflecting the price of energy. Related works [8,49] use a similar dual decomposition algorithm to solve a distributed problem between an aggregator and its customers, with the former reflecting its internal energy procurement cost function through iterative retail pricing and the latter trying to minimize deviation from a pre-determined aggregate power profile. Our work differs in that we are only modeling a decentralized energy market mechanism (hence no central provider) while leveraging the distributed ledger to ensure the liquidity of purchasers.

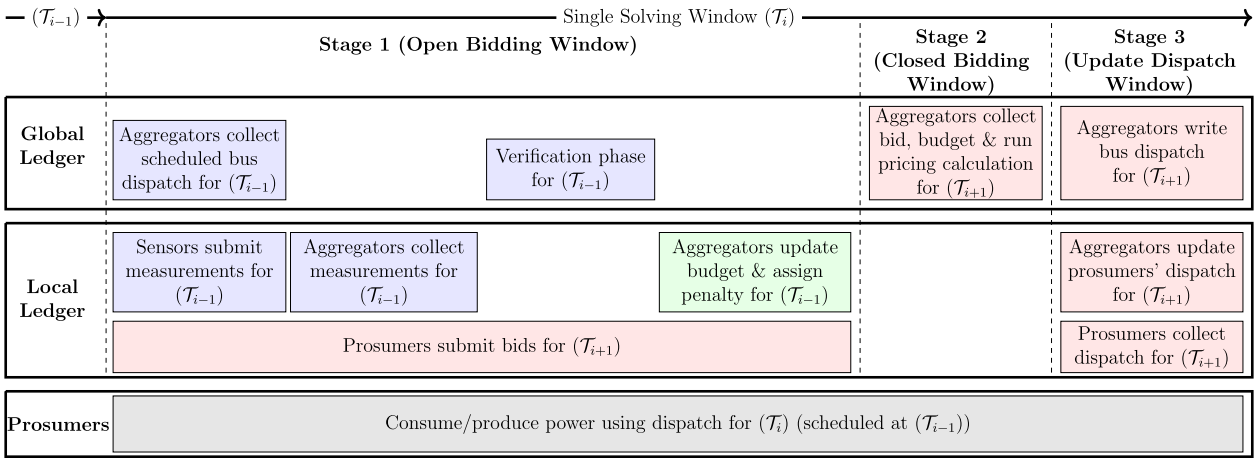

Fig. 3. Overview of EVE major tasks and timeline for a single solution window with red, blue, and green used to illustrate Pricing, Verification, and Billing policy execution steps, respectively.

3.1. Flexible resource model

The power injection trajectory $\boldsymbol{p}_b = [p_b(iT), \dots, p_b((i+1)T-1)]^\intercal \in \mathbb{R}^{T\times 1}$ is constrained depending on the type of energy service bus b provides. For example, in response to price signals, a participant may accept shifting to a less comfortable thermostat reference temperature, defer use of the dishwasher, dim lights, or other actions. This section describes in general terms[4] the inter-temporal constraints for demand and supply in a single period $\mathcal{T}_i$ (i is omitted for brevity) and discusses associated cost functions $C_b(\boldsymbol{p}_b)$.

For each aggregator n and bus b, the load profile can be split into a flexible component, that changes based on price, and an inflexible one that prosumers are willing to buy at any price. In the literature, $\boldsymbol{p}_b$ is typically modeled by a linear, affine function of a corresponding control signal $\boldsymbol{u}_b$, i.e., for all $b \in \mathcal{B}^{(n)}, n \in \mathcal{N}$:

$$\boldsymbol{p}_b = \boldsymbol{A}_b \boldsymbol{u}_b + \boldsymbol{\ell}_b, \quad p_b(t) \in \mathcal{S}_b^p, \quad \boldsymbol{u}_b \in \mathcal{U}_b \tag{2}$$

where $\boldsymbol{\ell}_b$ is the inflexible part of the load, $\boldsymbol{p}$ and $\boldsymbol{u}$ are column vectors, the set $\mathcal{U}_b$ is related to the flexibility that can be offered to adjust the shape of the profile, and $\mathcal{S}_b^p$ expresses operational constraints on how the asset can inject power.[5] Control signal constraints are mapped to $\boldsymbol{p}_b$ using $\boldsymbol{A}_b^\dagger$ as follows:

$$\boldsymbol{p}_b \in \mathcal{P}_b, \quad \mathcal{P}_b = \{\boldsymbol{p} | \boldsymbol{A}_b^\dagger(\boldsymbol{p} - \boldsymbol{\ell}_b) \in \mathcal{U}_b, \quad p(t) \in \mathcal{S}_b^p, \quad \boldsymbol{p}^\intercal \boldsymbol{\lambda} \le r_b\} \tag{3}$$

where the constraints $\mathcal{U}_b$ are linear, meaning that $\boldsymbol{p}_b$ lies within a polygon. In (3), $\boldsymbol{\lambda}$ denotes the price of energy over the horizon, r_b denotes the budget, and $\boldsymbol{p}_b^\intercal \boldsymbol{\lambda} \le r_b$ denotes the affordability constraint.

The cost to prosumer b, $C_b(\boldsymbol{p}_b)$, is the price the customer is willing to pay, or the price the supplier is willing to be paid to generate for a certain amount of power. $C_b(\boldsymbol{p}_b)$ is a convex function of $\boldsymbol{p}_b$. The cost function simultaneously reflects the utility and cost for prosumers that can switch between producing and consuming, respectively, with $p_b(t) \ge 0$ representing the supply utility function and $p_b(t) < 0$ representing the demand cost function.

This generic model can be applied to various resources including renewable generation, TCLs, storage devices, DAs, and EVs. Appendix A shows examples of such models and cost functions for supplemental study.

[4] Specific examples are given in Appendix A.

[5] Later integrality constraints will be relaxed in either $\mathcal{U}_b$ or $\mathcal{S}_b^p$, and be replaced with convex constraints to guarantee a convex market clearing problem required for the convergence of the pricing algorithm.

3.2. Distributed pricing and scheduling algorithm

Each aggregator can buy or sell power for the distributed resources connected to $\mathcal{B}^{(n)}$. Let $\mathcal{B}^{(n)} = \{n(1), \dots, n(|\mathcal{B}^{(n)}|)\}$, where $n(i)$ denotes the ith bus of aggregator n. The power profile of aggregator n is:

$$\boldsymbol{p}^{(n)} = \left(p^{(n)}(iT), \dots, p^{(n)}((i+1)T-1)\right)^\intercal = \sum_{b\in\mathcal{B}^{(n)}} \boldsymbol{p}_b \tag{4}$$

where $\boldsymbol{p}_b$ is the profile of one of the flexible resources at a bus $b \in \mathcal{B}^{(n)}$. We shall also define the following matrix:

$$\boldsymbol{P}^{(n)} = \left(\boldsymbol{p}_{n(1)}, \cdots, \boldsymbol{p}_{n(|\mathcal{B}^{(n)}|)}\right)^\intercal \tag{5}$$

whose rows are prosumers' profiles contributing to $\boldsymbol{p}^{(n)}$. The aggregated load profile is the sum of individual components, i.e., $\boldsymbol{p}^{(n)} = \boldsymbol{P}^{(n)\intercal}\boldsymbol{1}$, where $\boldsymbol{1} = \{1\}^{|\mathcal{B}^{(n)}|\times 1}$. The demand cost for aggregator n for a certain set of schedules $\boldsymbol{P}^{(n)}$ is described by:

$$\mathfrak{C}^{(n)}(\boldsymbol{P}^{(n)}) \triangleq \sum_{b\in\mathcal{B}^{(n)}} C_b(\boldsymbol{p}_b) \tag{6}$$

The feasible set of matrix profiles $\boldsymbol{P}^{(n)}$ that can be chosen by the aggregator must lie in the Cartesian product of the feasible sets for the tuple of profiles $\boldsymbol{p}_b \in \mathcal{P}_b$, $b \in \mathcal{B}^{(n)}$, meaning:

$$\boldsymbol{P}^{(n)} \in \mathcal{P}_{n(1)} \times \cdots \times \mathcal{P}_{n(|\mathcal{B}^{(n)}|)} \triangleq \mathcal{P}^{(n)} \tag{7}$$

Algorithm 1 Prosumer pricing interaction; A step-by-step implementation of (9) to (11) from the perspective of a prosumer $b \in \mathcal{B}^{(n)}$. LL_n refers to the local ledger to aggregator $n \in \mathcal{N}$, with $\Leftarrow$ and $\Rightarrow$ indicating writing to and reading from the ledger, respectively.

```
Wait until bidding interval starts:
LL_n ⇐ P_b (constraints) and C_b (cost function).
Wait until dispatch horizon starts.
if Dispatch instruction ready on ledger then
    LL_n ⇒ Read dispatch instruction p_b.
    Execute dispatch instruction p_b.
else
    Execute most recent p_b instruction.
end if
```

If transmission constraints are relaxed and the algorithm only balances instantaneous power, the optimum market clearing requires solving:

$$\min_{\boldsymbol{P}^{(1)},\dots,\boldsymbol{P}^{(N)}} \quad \sum_{n\in\mathcal{N}} \mathfrak{C}^{(n)}(\boldsymbol{P}^{(n)}) \quad \text{s.t.} \quad \sum_{n\in\mathcal{N}} \boldsymbol{P}^{(n)\intercal}\boldsymbol{1} = \boldsymbol{0}, \ \boldsymbol{P}^{(n)} \in \mathcal{P}^{(n)} \tag{8}$$

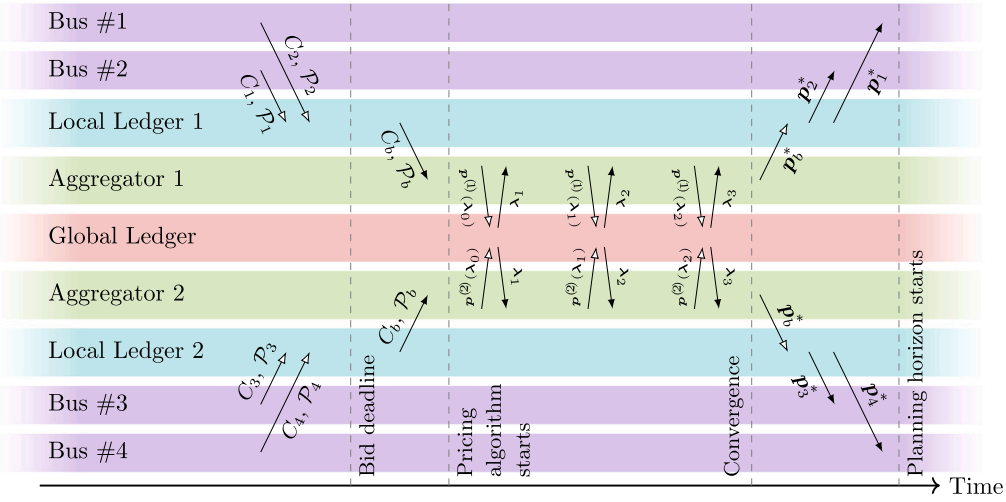

Fig. 4. A single solving window for distributed pricing ordered from (i) individual buses submit their costs and constraints, (ii) aggregators collect bus costs and constraints, (iii) aggregators iterate pricing algorithm, (iv) aggregators push dispatch solution to ledger, and (v) individual buses read dispatch solution.

The Lagrange multiplier of the balance constraint $\sum_{n\in\mathcal{N}} \boldsymbol{P}^{(n)\intercal}\mathbf{1} = 0$ in (8) $\boldsymbol{\lambda}$ expresses the shadow price of energy over the horizon in (3), $\boldsymbol{\lambda} = (\lambda(iT), \dots, \lambda((i+1)T-1))^\intercal$. Also, this is an instance of the network utility maximization problems which can be decomposed as detailed next.[6]

For each gradient descent, given the most recent price $\boldsymbol{\lambda}_k$ (the dual variable), each aggregator independently attempts to minimize its cost schedule through the following problem:

$$\min_{\boldsymbol{P}^{(n)}} \mathfrak{C}^{(n)}\boldsymbol{P}^{(n)} + [\boldsymbol{\lambda}_k]^\intercal \boldsymbol{P}^{(n)\intercal}\mathbf{1} \quad \text{s.t.} \quad \boldsymbol{P}^{(n)} \in \mathcal{P}^{(n)}, \quad \forall n \in \mathcal{N} \tag{9}$$

Let $\boldsymbol{P}^{(n)}_\star(\boldsymbol{\lambda}_k) \in \mathcal{P}^{(n)}$ be the solution of (9) in response to the kth iteration value of the vector $\boldsymbol{\lambda}_k$ and $\boldsymbol{p}^{(n)}_\star(\boldsymbol{\lambda}_k) = [\boldsymbol{\lambda}_k]^\intercal \boldsymbol{P}^{(n)\intercal}_\star\mathbf{1}$. Assuming a feasible solution exists, the algorithm updates the price as follows in GL:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha \sum_{n\in\mathcal{N}} \boldsymbol{p}^{(n)\intercal}_\star \tag{10}$$

Note that all the aggregators have to post their total injection based on the current price estimate, which will stop updating as soon as:

$$\boldsymbol{\lambda}_\star = \boldsymbol{\lambda}_{k^\star} : \quad \sum_{n\in\mathcal{N}} \boldsymbol{p}^{(n)}_\star = \mathbf{0} \tag{11}$$

These equations comprise the distributed and decentralized algorithms explained for prosumers and aggregators in Algorithms 1 and 2, respectively, and are visualized in Fig. 4.

A few interesting observations can be made:

1. Aggregators hide local bids and feasibility constraints from the GL, and instead they only show (expose) how they would react for the particular price scenarios iterated through $\boldsymbol{\lambda}_k$.
2. Because the cost and constraints of an aggregator are decomposable in terms of the prosumer profiles $\boldsymbol{p}_b$, the problem could be decomposed further to allow individual prosumers to keep $C_b(\boldsymbol{p}_b)$ and $\mathcal{P}_b$ private, interacting with the aggregator in a similar manner as shown in [51].

There are challenges with convergence of the distributed algorithm (9) not commonly found in comparable centralized formulations. First, the aggregate supply/demand curves must cross for a fixed price to emerge, a standard requirement in market theory. Second, a rogue aggregator may produce malicious values of $p^n_\star$ during the iterative phase of Algorithm 2, potentially preventing the algorithm from converging. There are numerous ways to detect such manipulations such as bounding the gradient step (10), however, the details of those are

Algorithm 2 Aggregator pricing algorithm; A step-by-step implementation of (9) to (11) from the perspective of aggregator $n \in \mathcal{N}$, with $\Leftarrow$ and $\Rightarrow$ indicating writing to and reading from the ledger, respectively.

Define algorithm time limit $\bar{\tau}$ and iteration limit $\bar{k}$.
Wait until prosumer bidding interval ends.
$LL_n \Rightarrow$ read all C_b, r_b, and $\mathcal{P}_b$ values available on ledger, building $\mathcal{B}^{(n)}$ based on submitted prosumers.
Initialize $\boldsymbol{\lambda}_\triangle$ and $\boldsymbol{p}^{(n)}_\triangle$ to most recent solutions $\boldsymbol{\lambda}_\star$ and $\boldsymbol{p}^{(n)}_\star$, or $\mathbf{0}$ if no prior solution exists.
Initialize $\boldsymbol{\lambda}_0 \leftarrow \boldsymbol{\lambda}_\triangle$, $k \leftarrow 0$ and $\hat{\alpha}$.
Build model (9) using $\boldsymbol{\lambda}_0$.
while $k < \bar{k}$ **do**
    Solve model (9) for $\boldsymbol{\lambda}_k$, enforcing the billing constraint for all prosumers.
    Retry writing solution $\boldsymbol{p}^{(n)}_\star$ to GL until ACK received.
    Start timer $\tau \leftarrow 0$.
    while Other aggregator solutions are not available from GL **do**
        if $\tau < \bar{\tau}$ **then**
            Wait ε seconds.
        else
            $LL_n \Leftarrow \boldsymbol{p}^{\triangle}_b$ for all prosumers $b \in \mathcal{B}^{(n)}$.
            $LL_n \Leftarrow r_b \leftarrow r_b + \boldsymbol{p}^{\triangle\intercal}_b\boldsymbol{\lambda}_\triangle$ for all prosumers $b \in \mathcal{B}^{(n)}$.
            Terminate algorithm, recycling last solutions.
        end if
    end while
    $GL \Rightarrow \boldsymbol{p}^{(m)\intercal}_\star\boldsymbol{\lambda}_k$ for all $m \in \mathcal{N} \setminus \{n\}$.
    Update $\boldsymbol{\lambda}_{k+1}$ following (10) using $\alpha = \hat{\alpha}/(k+1)$.
    if $|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k| < \epsilon$ **then**
        $\boldsymbol{\lambda}_\star \leftarrow \boldsymbol{\lambda}_k$.
        $LL_n \Leftarrow \boldsymbol{p}^{\star}_b$ for all prosumers $b \in \mathcal{B}^{(n)}$.
        $LL_n \Leftarrow r_b \leftarrow r_b + \boldsymbol{p}^{\star\intercal}_b\boldsymbol{\lambda}_\star$ for all prosumers $b \in \mathcal{B}^{(n)}$.
        Terminate.
    end if
    $k \leftarrow k+1$.
end while
Terminate.

[6] See e.g. [50] for its application to real time pricing.

beyond the scope of this paper and readers are referred to [52] for additional information. Third, due to the non-convexity introduced by the budget constraint in (3), there are no guarantees for convergence to the global optimum. However, as this constraint is only tight for a small number of prosumers, the problem often converges in practice.

4. EVE distributed robust state verification algorithm

While the blockchain TE framework ensures transparency and immutability of transactions placed on the chain, it is still insufficient to ensure that those transactions took place. Physical measurements are needed to confirm power injections at each bus in the electric grid. In a trustless system, aggregators have to cooperate to verify measurement accuracy, noting that some aggregators (or some of the prosumers they manage) may operate as adversaries by modifying market operations and/or measurements to commit energy theft. The essential tool described in Section 4.1 establishes accuracy by checking consistency with the grid's physical laws.

Before delving into algorithmic details, it is worth emphasizing that the notion of *distributed measurement verification* introduced here is new. The approach proposed for measurement verification consists of solving a regression problem closely related to state estimation in power systems fitting sensor data, but its goal is fundamentally different. Securely estimating the entire state vector for the grid is sufficient but not necessary to cross-validate the self-reported measurements. The state variables themselves are therefore not essential, and rather, we focus on the accuracy of reported *real power injections* within an aggregator region and *power flows between* different aggregators' regions. This approach ensures appropriate billing aggregators and prosumers and permits penalties to be levied for discrepancies in reporting. For measurements of the consumption/generation in time window $\mathcal{T}_i$, we run the verification mechanism in time window $\mathcal{T}_{i+1}$ as shown in Fig. 3. Considering an FDIA threat [53], our mechanism extends the idea in [11] and adapts it to serve as a decentralized cross-validation algorithm integrated within the EVE framework. The goal is to extrapolate the actual power injections $\boldsymbol{p}^{(n)}$ of each aggregator $n \in \mathcal{N}$ from sensor measurements. Since $\boldsymbol{p}^{(n)}$ are continuous variables, this is not a binary decision and amounts to solving a regression problem.

4.1. Physical constraints for the electric grid

Let $\boldsymbol{x}(t)$ be the vector of system variables at time t, consisting of bus injection variables and branch flow variables. Here, the variables in $\boldsymbol{x}(t)$ include real power injection, reactive power injection, and squared voltage magnitude expressed as $(p_b(t), q_b(t), v_b^2(t))^\intercal,\ \forall b \in \mathcal{B}$. Branch flow variables include the squared current magnitudes, real power flows, and reactive power flows expressed as $(c_l^2(t), P_l(t), Q_l(t))^\intercal,\ \forall l \in \mathcal{E}_e$. Let $\boldsymbol{x}_{\mathcal{A}}(t)$ include only the variables in $\boldsymbol{x}(t)$ that are directly-measured through a sensor. Measurements are noisy versions of physical parameters described by:

$$\boldsymbol{z}(t) = \boldsymbol{x}_{\mathcal{A}}(t) + \epsilon(t). \tag{12}$$

Within a margin of error due to the noise, these physical constraints are a set of non-linear homogeneous equations written in vector form as follows:

$$\boldsymbol{h}(\boldsymbol{x}(t)) = 0\ ;\quad \forall t \in \mathcal{T} \tag{13}$$

Appendix B specifies a possible $\boldsymbol{h}(\cdot)$ using the Distflow [54] equations.

4.2. Malicious agents behavior

We assume that the adversary (a malicious agent or a group of coordinating agents) is an insider who has legitimate physical and logical access to the network and ledgers through the certification mechanism. We also assume that the adversary is capable of manipulating sensors measurements, either by compromising sensors or compromising the communication between sensors and aggregators. The insider is motivated to disrupt the verification process and cheat the system for financial gain.

Utilizing sensor measurements reported at the LL level in (12) and physical constraints in (13), we formulate a decentralized optimization algorithm to complete the verification task using data from all sensors under any aggregator. The optimization algorithm (detailed later in Section 4.3.1) may be generalized into the following form:

$$\min_{\boldsymbol{x}^{(n)}} \sum_{n\in\mathcal{N}} f^{(n)}(\boldsymbol{x}^{(n)}) \tag{14a}$$

$$\text{s.t. Consensus Constraints} \tag{14b}$$

where n is a decentralized agent in set $\mathcal{N}$[7] and $f^{(n)} : \mathbb{R}^l \to \mathbb{R}$ is a cost function that agent n has to minimize while cooperatively minimizing the aggregate of cost functions from all the agents (14a), subject to some consensus constraints (14b). Common algorithms used to solve this problem using a certain communication graph $\mathcal{G}_c(\mathcal{N}, \mathcal{E}_c)$ (14) involve iterative consensus updates to $\boldsymbol{x}^{(n)} \in \mathbb{R}^l$ as follows:

$$\boldsymbol{x}_{k+1}^{(n)} = \sum_{m\in\mathcal{N}} a_{nm}\boldsymbol{r}^{(m)}(\boldsymbol{x}_k^{(m)}) \ \text{ at } \ k \geq 0 \tag{15}$$

where k is the iteration index, $a_{nm} \geq 0$ is a mixing weight $(n, m) \in \mathcal{E}$ such that $\sum_{m\in\mathcal{N}} a_{nm} = 1$, and $\boldsymbol{r}^{(m)} : \mathbb{R}^l \to \mathbb{R}^l$ is the information received by n from neighbor m. We see from (15) that neighbors on the communication graph exchange information with one another in each iteration.

The communications model is described as:

$$\boldsymbol{r}^{(m)}(\boldsymbol{x}_k^{(m)}) = \begin{cases} \boldsymbol{x}_k^{(m)}, & m \in \mathcal{R} \\ \star, & m \in \mathcal{M} \end{cases} \tag{16}$$

with $\mathcal{R} \subseteq \mathcal{N}$ and $\mathcal{M} \subseteq \mathcal{N}$ expressing the sets of regular and malicious agents, respectively. That is, regular agents report their true states, whereas malicious agents may inject false data and disrupt convergence of the algorithm to suit their goals.

An attack by an agent results in the following update for a neighbor n:

$$\widetilde{\boldsymbol{x}}_{k+1}^{(n)} = \boldsymbol{x}_{k+1}^{(n)} + \star \tag{17}$$

where $\widetilde{\boldsymbol{x}}_{k+1}^{(n)}$ is the false update and $\boldsymbol{x}_{k+1}^{(n)}$ is the true update (if all the neighbors communicated truthfully). For instance, a malicious agent will attempt theft by paying less (as a consumer) or receiving a larger compensation (as a producer). This occurs by under reporting energy usage or over reporting generation by altering the appropriate field of $\boldsymbol{z}^{(n)}(t)$ so that: $\widetilde{\boldsymbol{z}}^{(n)}(t) = \boldsymbol{z}^{(n)}(t) + \star$.

4.3. Robust state verification in the presence of FDIAs

Next, we provide details on the decentralized algorithm that allows aggregators to cross verify if measurements of power injections are to be trusted and, if not, which aggregators are likely responsible for the FDIA.

4.3.1. Modeling of the optimization problem

We pose the state verification problem as a decentralized optimization problem in which each aggregator $n \in \mathcal{N}$ has access only to their private cost function $f^{(n)} : \mathbb{R}^{l_n} \to \mathbb{R}$ and act on their own private vector of system variables, $\boldsymbol{x}^{(n)}(t)$. Here, $\boldsymbol{x}^{(n)}(t)$ includes copies of those variables in $\boldsymbol{x}(t)$ that pertain to buses and lines inside and at the periphery of aggregator region n (see Fig. 5). That is,

$$\boldsymbol{x}^{(n)}(t) = \boldsymbol{S}^{(n)}\boldsymbol{x}(t) \tag{18}$$

[7] We abuse the notation for the set of aggregators $\mathcal{N}$ to denote the set of decentralized agents in this subsection.

where $\boldsymbol{S}^{(n)}$ is a selection matrix that extracts the appropriate entries that make up $\boldsymbol{x}^{(n)}(t)$ from $\boldsymbol{x}(t)$. Since there are tie-lines between aggregator regions, some of the entries in $\boldsymbol{x}^{(n)}(t)$ will have identical counterparts in $\boldsymbol{x}^{(m)}(t)$ of a neighboring aggregator region m. For the extrapolated injections to be valid, the regions have to match values at these tie-lines. Failure to do so indicates an attack. The consensus constraint is, therefore, a consensus on the tie-lines variables, that can be written as:

$$\boldsymbol{S}_{nm}\boldsymbol{x}^{(n)}(t) = \boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t), \qquad \forall n \in \mathcal{N} \text{ and } m : nm \in \mathcal{E}_c \tag{19}$$

where $\boldsymbol{S}_{nm}$ is the selection matrix that extracts from $\boldsymbol{x}^{(n)}$ common variables between neighboring regions n and m.

Thus, the goal of the set of aggregators is to minimize their individual cost functions ($f^{(n)}, \forall n \in \mathcal{N}$) and the global cost function ($\sum_{n\in\mathcal{N}} f^{(n)}$) simultaneously subject to the consensus constraints (19) defined over the communication graph, $\mathcal{G}_c(\mathcal{N}, \mathcal{E}_c)$.

The optimization problem should find values of system variables, $\boldsymbol{x}^{(n)}(t)$, at each aggregator n that:

- Have the least residual error with respect to the measurements, $\boldsymbol{z}^{(n)}(t)$, to reduce measurement deviation from their corresponding system variables ($\boldsymbol{x}^{(n)}_{\mathcal{A}}(t)$) being minimized. Letting $\boldsymbol{S}^{(n)}_{\mathcal{A}}$ be the selection matrix that extracts available measurements from $\boldsymbol{x}^{(n)}(t)$, we can write

$$\boldsymbol{x}^{(n)}_{\mathcal{A}}(t) = \boldsymbol{S}^{(n)}_{\mathcal{A}}\boldsymbol{x}^{(n)}(t) \tag{20}$$

- Have the least residual error with respect to the scheduled injections, $\mathfrak{p}^{\star(n)}(t)$, where the ith element of the vector is given by $[\mathfrak{p}^{\star(n)}(t)]_i = p_{b_i}(t),\ \forall b_i \in \mathcal{B}^{(n)}$. This seeks to minimize deviation of the scheduled bus dispatch from their corresponding system variables ($\boldsymbol{S}^{(n)}_{\mathcal{P}}\boldsymbol{x}^{(n)}(t)$), where $\boldsymbol{S}^{(n)}_{\mathcal{P}}$ is the selection matrix that extracts active power injections variables from $\boldsymbol{x}^{(n)}(t)$, and
- Fit the physical model equations, $\boldsymbol{h}^{(n)}(\cdot)$,[8] with the least residual error.

Power injection components, $\boldsymbol{S}^{(n)}_{\mathcal{P}}\boldsymbol{x}^{(n)}$, from the optimization problem are used in EVE as the closest approximation for ground-truth billing. The severity of deviations from the schedule $\left(\mathfrak{p}^{\star(n)}(t) - \boldsymbol{S}^{(n)}_{\mathcal{P}}\boldsymbol{x}^{(n)}\right)$ that determines the penalties assigned to prosumers under a specific bus.

Finally, the state verification problem can be cast as the following optimization problem, written in a form analogous to (14a) that is amenable to ADMM decomposition [55]:

$$\min_{\{\boldsymbol{x}^{(n)}(t)\}^{t\in\mathcal{T}}_{n\in\mathcal{N}}} \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} f^{(n)}(\boldsymbol{x}^{(n)}(t)) \tag{21a}$$

$$\text{s.t.}\quad \boldsymbol{S}_{nm}\boldsymbol{x}^{(n)}(t) = \boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t)\ , \quad \forall nm \in \mathcal{E}_c \tag{21b}$$

$$\boldsymbol{x}^{(n)}(t) = \boldsymbol{S}^{(n)}\boldsymbol{x}(t)\ , \quad \forall n \in \mathcal{N} \tag{21c}$$

$$\boldsymbol{x}^{(n)}_{\mathcal{A}}(t) = \boldsymbol{S}^{(n)}_{\mathcal{A}}\boldsymbol{x}(t), \quad \forall n \in \mathcal{N} \tag{21d}$$

where:

$$f^{(n)}(\boldsymbol{x}^{(n)}(t)) := c_1 \left\| \boldsymbol{\Sigma}^{-1/2}_{z^{(n)}} \left(\boldsymbol{z}^{(n)}(t) - \boldsymbol{x}^{(n)}_{\mathcal{A}}(t) \right) \right\|^2_2 + c_2 \left\| \boldsymbol{h}^{(n)}(\boldsymbol{x}^{(n)}(t)) \right\|^2_2 + c_3 \left\| \mathfrak{p}^{\star(n)}(t) - \boldsymbol{S}^{(n)}_{\mathcal{P}}\boldsymbol{x}^{(n)}(t) \right\|^2_2 \tag{22}$$

Here, $\boldsymbol{\Sigma}_{z^{(n)}}$ is the diagonal matrix such that $[\boldsymbol{\Sigma}_{z^{(n)}}]_{jj} = \text{variance}([\boldsymbol{z}^{(n)}]_j)$. Additionally, (21b) enforces consensus among the common variables across neighboring regions for the constraints mentioned in (14b).

Remark 4.1. The terms c_1, c_2, c_3 denote the weight of the penalty levied on the solution if their respective objectives are violated. The third term penalizes a solution that is further from the schedule, and in noting this might hinder the verification step, practical applications can choose c_3 such that $c_3 \lll c_1, c_2$.

[8] Definitions of the functions $\boldsymbol{h}^{(n)}(\cdot)$ are given in Appendix B.

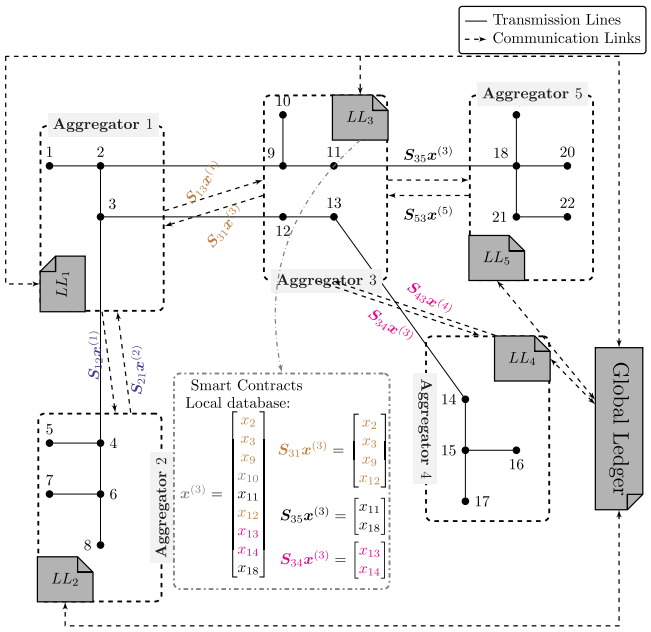

Fig. 5. State verification architecture for a test case with five aggregator regions.

4.3.2. Iterative update rule

In this section, we present equations that iteratively update state variables vectors to arrive at the solution to problem (21). First, we define two matrices that assist in formulating rules for iterative updates. Suppose that l_n is the number of variables pertaining to aggregator region n, then $\boldsymbol{D}^{(n)} \in \mathbb{R}^{l_n \times l_n}$ is a diagonal matrix with $[\boldsymbol{D}^{(n)}]_{jj}$ equal to the number of regions with which region n has the jth variable of $\boldsymbol{x}^{(n)}$ in common, $\forall j \in [l_n]$. Similarly $\overline{\boldsymbol{D}}^{(n)} \in \mathbb{R}^{l_n \times l_n}$ is also a diagonal matrix such that $[\overline{\boldsymbol{D}}^{(n)}]_{jj} = 1/[\boldsymbol{D}^{(n)}]_{jj}$ if $[\boldsymbol{D}^{(n)}]_{jj} \neq 0$, and $[\overline{\boldsymbol{D}}^{(n)}]_{jj} = 0$ if $[\boldsymbol{D}^{(n)}]_{jj} = 0$.

To arrive at the minimizer of the problem in (21) for an interval $t \in \mathcal{T}_i$, the updates in (23) are executed iteratively $\forall k \geq 0$ until a termination condition is satisfied[9]:

$$\boldsymbol{x}^{(n)}_{k+1} = \left(c_1 \boldsymbol{S}^{(n)\intercal}_{\mathcal{A}} \boldsymbol{\Sigma}^{-1}_{z^{(n)}} \boldsymbol{S}^{(n)}_{\mathcal{A}} + c_2 \boldsymbol{H}^{(n)\intercal}\boldsymbol{H}^{(n)} + c_3 \boldsymbol{S}^{(n)\intercal}_{\mathcal{P}} \boldsymbol{S}^{(n)}_{\mathcal{P}} + c_4 \boldsymbol{D}^{(n)} \right)^{-1} \times \left(c_1 \boldsymbol{S}^{(n)\intercal}_{\mathcal{A}} \boldsymbol{\Sigma}^{-1}_{z^{(n)}} \boldsymbol{z}^{(n)} + c_3 \boldsymbol{S}^{(n)\intercal}_{\mathcal{P}} \mathfrak{p}^{\star(n)} + c_4 \boldsymbol{D}^{(n)} \boldsymbol{v}^{(n)}_k \right) \tag{23a}$$

$$\boldsymbol{\psi}^{(n)}_{k+1} = \overline{\boldsymbol{D}}^{(n)} \sum_{m:nm\in\mathcal{E}_c} \boldsymbol{S}^{\intercal}_{nm}\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}_{k+1} \tag{23b}$$

$$\boldsymbol{v}^{(n)}_{k+1} = \boldsymbol{v}^{(n)}_k + \boldsymbol{\psi}^{(n)}_{k+1} - 0.5\left(\boldsymbol{\psi}^{(n)}_k + \boldsymbol{x}^{(n)}_k \right) \tag{23c}$$

where $\boldsymbol{H}^{(n)}$ is defined in Appendix B with variables initialized as:

$$\boldsymbol{x}^{(n)}_0 \in \mathbb{R}^{l_n} \qquad \ldots \text{initialize arbitrarily} \tag{24a}$$

$$\boldsymbol{\psi}^{(n)}_0 = \overline{\boldsymbol{D}}^{(n)} \sum_{m:nm\in\mathcal{E}_c} \boldsymbol{S}^{\intercal}_{nm}\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}_0 \tag{24b}$$

$$\boldsymbol{v}^{(n)}_0 = \frac{1}{2}\left(\boldsymbol{\psi}^{(n)}_0 + \boldsymbol{x}^{(n)}_0 \right). \tag{24c}$$

Algorithm 3 describes the iterative process in (23) with a pictorial representation of the algorithm given in Fig. 5. Buses in each aggregator region n have access to their local ledgers LL_n. The aggregators, in addition to their local ledger, also have access to GL. The aggregator collects measurements from their buses and stores them on LL_n. Results

[9] For the sake of readability, we write $\boldsymbol{x}^{(n)}$ instead of $\boldsymbol{x}^{(n)}(t)$ and use $\boldsymbol{x}^{(n)}_k$ to denote iterates of the algorithm where k is the iterate.

after an ADMM update are also saved in LL_n for each aggregator. Aggregators exchange state elements that correspond to tie-line variables with a neighbor as shown in Fig. 5.

Algorithm 3 Robust state verification; A step-by-step implementation from the perspective of aggregator $n\ \forall n \in \mathcal{N}$. Here $LL_n\ \forall n \in \mathcal{N}$ and GL represents the local and global ledgers respectively. The symbol $a \Leftarrow b$ corresponds to upload from b to a.

1: $LL_n \Leftarrow$ Collect local measurements $\boldsymbol{z}^{(n)}(t)$.
2: Initialize ADMM states according to (24), trust score $\boldsymbol{\pi} = \mathbf{0}$, and disagreements $d_{nm} = 0,\ \forall nm \in \mathcal{E}_c$.
3: **repeat**
4: $\quad \boldsymbol{\pi}^- \leftarrow \boldsymbol{\pi}$ and $[\boldsymbol{x}^{(n)}(t)]^- \leftarrow \boldsymbol{x}^{(n)}(t)$.
5: $\quad$ ADMM update of $\boldsymbol{x}^{(n)}(t)$ according to Eq. (23a).
6: $\quad$ Send common variable information to neighbors via the GL with ABAC control: $GL \Leftarrow \{\boldsymbol{S}_{ij}\boldsymbol{x}^{(n)}(t) \mid m : nm \in \mathcal{E}_c\}$.
7: $\quad$ Receive common variable information from neighbors: $\{\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t) \mid m : nm \in \mathcal{E}_c,\ m$ sent information$\} \bigcup$ $\{\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t-1) \mid m : nm \in \mathcal{E}_c, m$ did not sendinformation$\} \Leftarrow GL$.
8: $\quad$ Update intermediate states according to (23b) and (23c).
9: $\quad$ Run Algorithm 4 to update $\boldsymbol{\pi}$ and $d_{nm},\ \forall m : nm \in \mathcal{E}_c$.
10: **until** One of the termination conditions T1. or T2. is satisfied
11: Restart the algorithm with $\{\mathcal{G}_c \setminus (\arg\max_n \pi_n)\}$.

4.3.3. Threat model specific to RSV

The updates in (23b) to (23c) involve aggregating the shared state variable $\boldsymbol{x}^{(n)}$ from agent n with the same state variable from its neighbors, similar to the updates in (15). Yet Section 4.2 showed that approach to be vulnerable to FDI attacks via (17), thus indicating (23b) is also vulnerable to malicious injections by neighbors.

A malicious user in region m may modify values of its input measurements to affect the aggregator's measurement vector as follows:

$$\tilde{\boldsymbol{z}}^{(m)} = \boldsymbol{z}^{(m)} + \boldsymbol{a}^{(m)}$$

where the perturbation $\boldsymbol{a}^{(m)}$ has non-zero entries in the locations that correspond to false sensor measurements. Similarly, if the aggregator itself acts maliciously, it can inject false data (as seen in (16)) into the updates of $\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}$ that are passed to a neighbor aggregator n in (23b). Both changes lead to discrepancies in neighboring aggregators updates as follows:

$$\tilde{\boldsymbol{\psi}}_k^{(n)}(t) = \boldsymbol{\psi}_k^{(n)}(t) + \overline{\boldsymbol{D}}^{(n)}\boldsymbol{S}_{nm}^{\intercal}\boldsymbol{S}_{mn}\boldsymbol{a}_k^{(m)}(t) \tag{25}$$

$$\Rightarrow\quad \tilde{\boldsymbol{v}}_k^{(n)}(t) = \boldsymbol{v}_k^{(n)}(t) + \overline{\boldsymbol{D}}^{(n)}\boldsymbol{S}_{nm}^{\intercal}\boldsymbol{S}_{mn}\boldsymbol{a}_k^{(m)}(t) \tag{26}$$

$$\tilde{\boldsymbol{x}}_{k+1}^{(n)}(t) = \boldsymbol{x}_{k+1}^{(n)}(t) + c_4\boldsymbol{M}\boldsymbol{D}^{(n)}\overline{\boldsymbol{D}}^{(n)}\boldsymbol{S}_{nm}^{\intercal}\boldsymbol{S}_{mn}\boldsymbol{a}_k^{(m)}(t) \tag{27}$$

where $\boldsymbol{M} = \left(\boldsymbol{H}^{(n)\intercal}\boldsymbol{H}^{(n)} + \boldsymbol{S}_{\mathcal{A}}^{(n)\intercal}\boldsymbol{S}_{\mathcal{A}}^{(n)} + \boldsymbol{S}_{\mathcal{P}}^{(n)\intercal}\boldsymbol{S}_{\mathcal{P}}^{(n)} + c_4\boldsymbol{D}^{(n)}\right)^{-1}$ and (27) is analogous to the discrepancy in (17) where the inaccurate update is a modified version ($\tilde{\boldsymbol{x}}_{k+1}^{(n)}(t)$) of the true update ($\boldsymbol{x}_{k+1}^{(n)}(t)$).

The FDIAs will be successful at creating algorithm divergence, or convergence to a false optimum, if the ADMM updates in (23a) to (23c) are used to solve (21a). Algorithm divergence is a special case of a *Denial of Service* attack in which aggregators are unable to complete the verification process.

An additional area of concern is stealth attacks where the attacker injects a sparse vector, $\boldsymbol{a}^{(n)}$. Here, non-zero entries of the attack vector correspond to the sensors being attacked, such that the constraint in (28) is satisfied even with the perturbed state:

$$\boldsymbol{h}^{(n)}(\boldsymbol{x}^{(n)} + \boldsymbol{S}_{\mathcal{A}}^{(n)\intercal}\boldsymbol{a}^{(n)}) = 0 \tag{28}$$

where $\boldsymbol{x}^{(n)}$ corresponds to the true variables. Here, without any change in the loss function in the state verification problem (21a), the attacker is still able to alter the algorithm's output. These types of attacks are only possible when a malicious aggregator can gain complete knowledge about its neighbors' parameters. Such attacks are tough to detect, and even harder to mitigate, in the absence of a specially imposed structure on the actual measurement vectors. In practice, specially designed sparsity patterns for sensors can prevent such attacks.

4.3.4. Detection of the malicious agent

Methods proposed in [11] are employed to detect an attack as presented in Algorithm 4. Algorithm 4 is a detection subroutine with the robust state verification of Algorithm 3.

Algorithm 4 Detection loop; $F(d_{nm}, \boldsymbol{S}_{nm}\boldsymbol{x}^{(n)}(t), \boldsymbol{S}_{ji}\boldsymbol{x}^{(m)},\ \forall m : nm \in \mathcal{E}_c)$

1: Set α, $\epsilon = 10^{-16}$
2: Calculate $d_{nm}\ \forall m : nm \in \mathcal{E}_c$ according to (29).
3: Calculate $[\boldsymbol{B}]_{nm} \forall m : nm \in \mathcal{E}_c$ according to (30).
4: Submit $[\boldsymbol{B}]_{n:}$ to GL until ACK received.
5: $[\boldsymbol{B}]_{m:} \Leftarrow GL, \forall m \in \mathcal{N} \setminus \{n\}$
6: Compute $\boldsymbol{\pi}$, the left principle eigenvector of $\boldsymbol{B}$.
7: **return** $\boldsymbol{\pi}$ and $d_{nm},\ \forall m : nm \in \mathcal{E}_c$.

In Algorithm 4, each region n calculates a measure of disagreement, d_{nm}, in the shared variables with a neighboring region m as:

$$d_{nm} = (1-\alpha_k)d_{nm} + \frac{\alpha_k/4}{|\boldsymbol{S}_{nm}\boldsymbol{x}^{(n)}(t)||\mathcal{T}_i|}\sum_{t\in\mathcal{T}_i}\left\|\boldsymbol{S}_{nm}\boldsymbol{x}^{(n)}(t) - \boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t)\right\|_F^2 \tag{29}$$

and the matrix of normalized disagreement scores $\boldsymbol{B}$:

$$[\boldsymbol{B}]_{nm} = \frac{d_{nm}}{\sum_{m':nm'\in\mathcal{E}_c} d_{nm'} + \epsilon}. \tag{30}$$

The left principal eigenvector, $\boldsymbol{\pi}$, of $\boldsymbol{B}$ is then calculated. The value of $\|\boldsymbol{\pi}\|_2$ and the location of the highest element of $\boldsymbol{\pi}$ represent the presence of an attack and the index of the most likely attacker, respectively [11].

To mitigate the impact of FDIA, line 11 is added to Algorithm 3 to restart the algorithm after isolating the identified attacker. However, the structure of the communication graph can cause misidentification errors, resulting in divergence in the proposed algorithm. Convergence can be guaranteed provided the following condition is met when establishing the structure of the communication graph:

Theorem 1 (*Proposition 3 [11]*)**.** *Consider a system with $N > 2$ regions, if (i) there exists a 3-clique in the graph $\mathcal{G}_c$ and if (ii) for finite k the RSV does not converge, then the stationary distribution $\boldsymbol{\pi}_k$ exists and it is unique and can be computed.*

4.3.5. Termination conditions

Algorithm 3 terminates when either of the following two conditions is satisfied.

T1. The first condition is met when an aggregator converges, i.e.,

$$\left\|\boldsymbol{x}_k^{(n)}(t) - \boldsymbol{x}_{k-1}^{(n)}(t)\right\|_\infty \le \epsilon \tag{31}$$

T2. The second condition is met when all aggregators have agreed about the presence and identity of an FDI attacker, i.e.,

$$\|\boldsymbol{\pi}_k - \boldsymbol{\pi}_{k-1}\|_\infty \le \epsilon_\pi \quad \text{and} \quad \pi_m > \mu_m(\boldsymbol{\pi}) + \beta\sigma_m(\boldsymbol{\pi}) \tag{32}$$

for some $\beta > 0$, $m \in \mathcal{N}_n$, where each agent n calculates $\mu_m(\boldsymbol{\pi})$ and $\sigma_m(\boldsymbol{\pi})$ as the excluded average and excluded standard deviation, respectively:

$$\mu_m(\boldsymbol{\pi}) = \frac{1}{|\mathcal{N}_n| - 1}\sum_{m'\in\mathcal{N}_n\setminus\{m\}}\pi_{m'}$$
$$\sigma_m(\boldsymbol{\pi}) = \sqrt{\frac{1}{|\mathcal{N}_n| - 1}\sum_{m'\in\mathcal{N}_n\setminus\{m\}}|\pi_{m'} - \mu_{m'}(\boldsymbol{\pi})|^2} \tag{33}$$

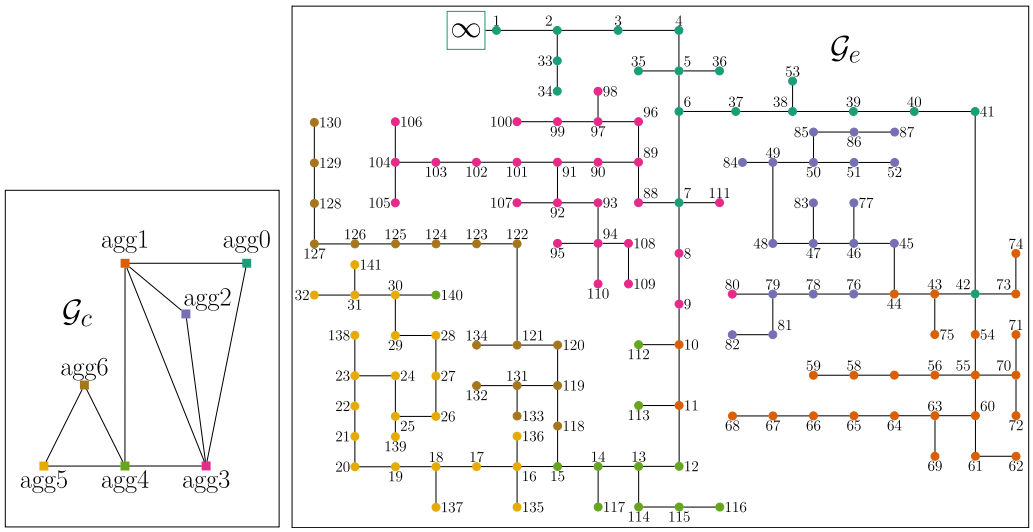

Fig. 6. $\mathcal{G}_e$ is the network graph corresponding to the 141 bus radial distribution network. The network in the box shows the communication graph $\mathcal{G}_c$ with the nodes representing aggregators.

4.3.6. Placement of measuring instruments

Partitioning the electric grid into N aggregator regions must be done according to a ruleset. Recovering a unique $\boldsymbol{\pi}_k$ during the convergence failure of the RSV algorithm requires the presence of at least one 3-clique in the communication graph $\mathcal{G}_c$ (Theorem 1). However, the radial structure of the distribution grid and the requirement for each aggregator region to contain a contiguous set of buses limits the number of possible 3-cliques on $\mathcal{G}_c$. A possible solution is to allow aggregators to access a small amount of sensor measurements from the neighboring aggregators to improve accountability at the expense of privacy.

5. Numerical simulation

This section demonstrates performance of the distributed pricing and robust state verification algorithms before describing implementation on HLF in Section 6.

5.1. Simulation setup

Fig. 6 shows the radial MATPOWER [56] 141 bus distribution network used as the demonstration case. The distribution network is separated into 7 aggregator zones ($N = 7$). The choice of N and distribution of buses within N is selected to increase common state variables between aggregators and to increase the number of 3-cliques in the communication graph ($\mathcal{G}_c$ in Fig. 6) to satisfy the required conditions introduced in Section 4.3.6. The utility is represented as the substation and belongs to the first aggregator. A total of 3900 prosumers are placed randomly across the distribution network. The total number of prosumers is arbitrary selected and sufficiently high to show algorithm scalability. The distributed pricing algorithm is run for six(6) ten-minute intervals to create a one-hour look-ahead window as a common duration of interest. The choice of interval length and the number of intervals can be selected to match local or regional guidelines on settlement time frames since the generalized formulations are independent of the length and number of intervals.

5.2. Distributed pricing

Each aggregator includes EVs, energy storage devices, DAs, thermostatically controlled loads (TCLs), and renewables, as summarized in Fig. 7, with each aggregator including a similar percentage of each prosumer type (generated randomly). Each prosumer has different properties and cost/utility functions uniformly sampled from an identical distribution. Renewables are configured as price-takers (they have no cost/utility), TCLs have a quadratic cost function demanding payments for deviations from their thermostat reference temperature, storage devices require a linear payment proportional to their usage, and DAs and EVs have a linear cost function. The slack bus has a quadratic cost function.

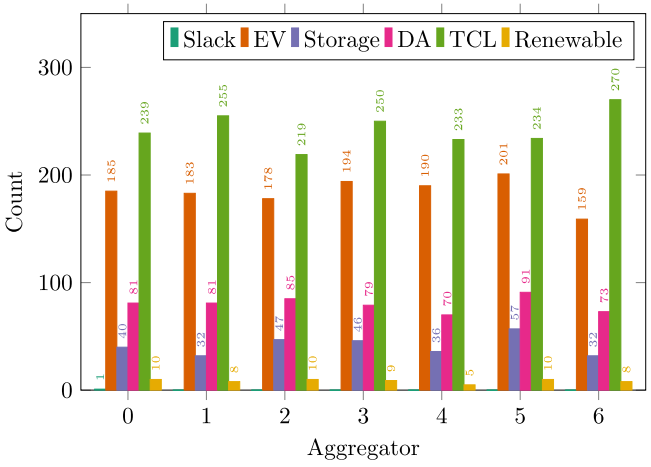

Fig. 7. Asset types within each aggregator region.

Cost/utility functions and prices are represented by an arbitrary monetary unit in which the price reflects the marginal cost of increasing load by a "single unit". Any changes in asset cost parameters are expected to influence the converged price of (9).

Power transfer between different resources is shown in Fig. 8a. The slack bus is a net generator (as the utility) since all other aggregators are composed mostly of consuming loads. Fig. 8b shows prices for each time interval in the one-hour look-ahead window and how those prices evolve as the distributed algorithm iterates to solve all dispatches and prices for all intervals at once. The price only fluctuates by approximately 2% across the iterations because of the significant load shifting behavior of flexible resources. A total of 28 out of 3900 prosumers are budget-constrained (need to curtail consumption due to insufficient funds); however, the algorithm converges without problems, even though convergence is not guaranteed.

To summarize, given a reasonable initial price for the algorithm, typically obtained from the previous solution, the algorithm solves

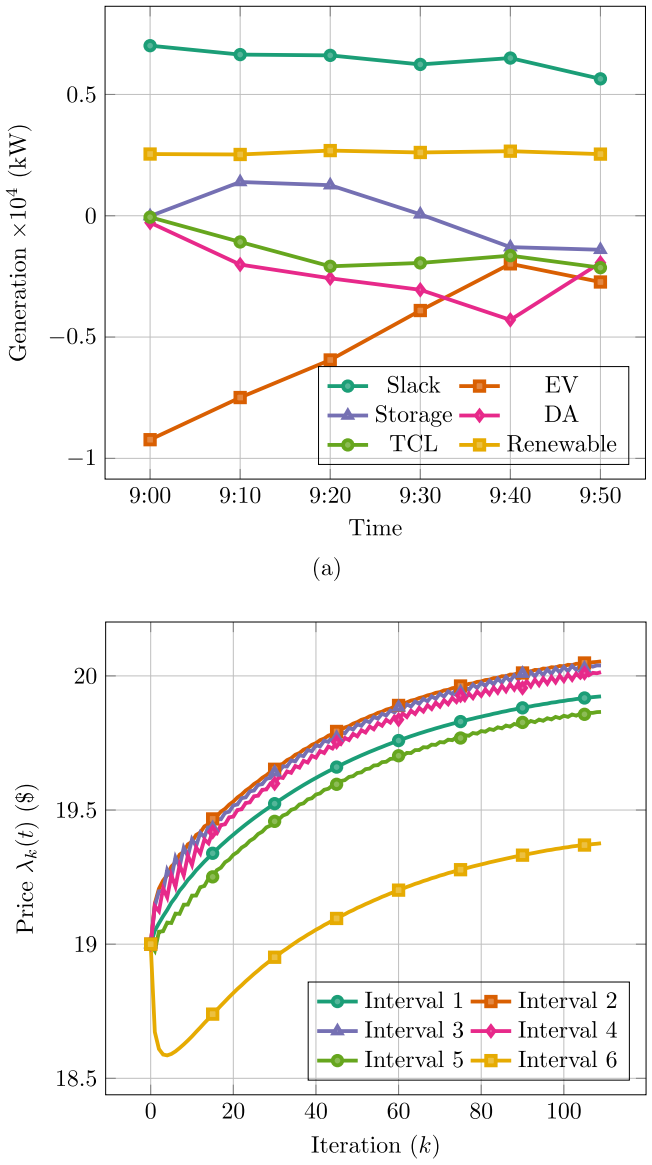

Fig. 8. (a) Power transfer between different resources (b) Progression of the shadow price $\boldsymbol{\lambda}_k$ as Algorithm 2 iterates, with each line denoting an element of $\boldsymbol{\lambda}$ which reflects price for a particular interval.

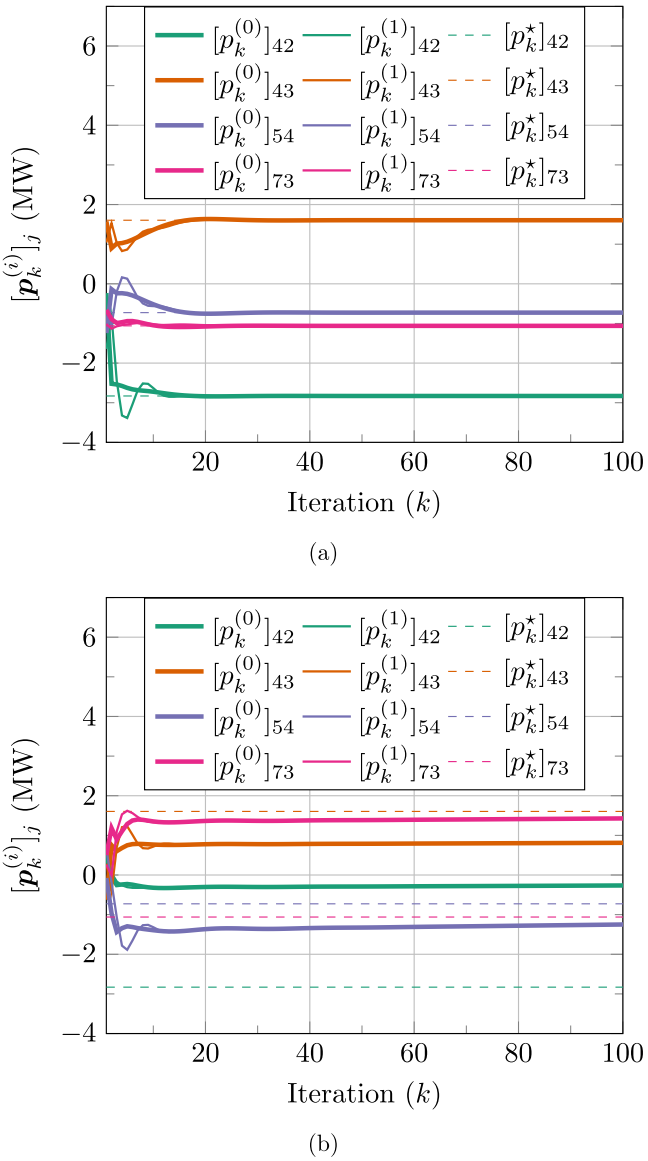

Fig. 9. Real power injections (in MW) by common nodes of aggregators 0 and 1 (a) Convergence to the optimal point under no attack (b) Convergence to a non-optimal point when aggregator 0 is an attacker.

in a distributed fashion the optimal schedule of all aggregators in a small number of iterations (< 100 in this illustrative example) while managing to honor the individual budget constraint. As mentioned in Section 3.2, convergence is not guaranteed due to the non-convexity introduced by the budget constraint in (3), which may lead to cyclic behavior in the gradient descent. However, for a low number of active constraints, the method often works as was the case in our experiments. This approach chooses an epsilon (Step 22 in Algorithm 2) that expresses the price difference between iterations to determine when prices have stabilized, and hence iterations conclude.

The approach developed here for distributed pricing can easily be adapted to the scenario when each prosumer can trade power directly rather than working through an aggregator. In that case, each prosumer will participate in the iterative pricing algorithm by interacting with the blockchain architecture through smart contracts as shown in Section 6.2. And while that approach is possible with the generalized mathematical framework introduced here, the authors advise caution as the approach will be computationally burdensome because the number of market participants engaging with the GL has an exponential effect on the number of transactions and hence slows the convergence process.

5.3. Distributed verification

Verification algorithm Algorithms 3 and 4 presented in Section 4 are demonstrated here with the following parameters: $\epsilon_\pi = 1e-3$, $\epsilon = 1e-3$, $\alpha_k = 1/k$, $c_3 = c_4 = 0.5$, $\beta_n = 2$, $\forall n \in \hat{\mathcal{N}}$. Measurements of available variables, $\boldsymbol{x}_{\mathcal{A}}$, were noisy versions of MATPOWER power flow output for the 141 bus radial distribution feeder case. The noise, w_n, was chosen to be Gaussian with zero mean and a variance of 1.

To illustrate the veracity of the verification algorithm, we set one of the aggregators, m, to be a malicious entity capable of injecting false data into its communications with neighboring aggregators. The FDI attack from aggregator m constitutes an injection from attack vector $\boldsymbol{a}^{(mn)}$ to the communications received by a neighboring aggregator n of the attacker in (23b). In each iteration of the algorithm, the attack

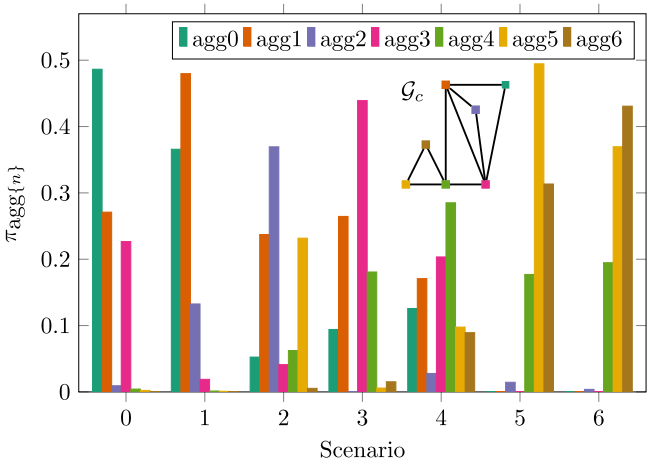

Fig. 10. The stationary distribution, $\boldsymbol{\pi}$, of $\boldsymbol{B}$ under FDI attack. $\boldsymbol{\pi}$ represents the trust score: higher the value of $\pi_{\text{agg}\{n\}}$, lower is the trust in aggregator n. In scenario m, agg$\{m\}$ is the attacker.

vector is chosen randomly subject to $\left\|\boldsymbol{a}^{(mn)}\right\|_2 = 0.5\sqrt{|\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t)|}$, where $|\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t)|$ is the length of the vector $\boldsymbol{S}_{mn}\boldsymbol{x}^{(m)}(t)$, i.e., the number of common variables shared by region m and its neighbor n.

Consider an example with aggregators 0 and 1 that has buses 42, 43, 54, and 73 as adjacent nodes with common variables between two aggregator regions as shown in the network topology of Fig. 6. Common variables include real power injection, reactive power injection, and voltage magnitude. Fig. 9a shows convergence of the four sets of common variables (one for each bus) shared by aggregators when there is no attack, whereas Fig. 9b shows their divergence when region 0 is an attacker. In Fig. 9a, both aggregators converge to the optimal point $p_b^\star$, $\forall b \in \{42, 43, 54, 73\}$. The attack in Fig. 9b creates a situation in which the parameters reach to the same value yet do not converge at the optimal point. This prevents the verification algorithm from completing. The subroutine described in Algorithm 4 aims to stop such attacks from occurring by tracking disagreements in the common variables between neighbors and identifying the most likely attacker.

The detection of FDI attacks from selfish entities on the TE market is accomplished using the stationary distribution $\boldsymbol{\pi}$ of the disagreement matrix $\boldsymbol{B}$, as discussed in Section 4.3.4. Fig. 10 shows results for $N = 7$ in which each aggregator is shown to be the attacker. Each set contains seven bars representing the element of vector $\boldsymbol{\pi}$ corresponding to each of the seven aggregators. As the height of the bar increases, that aggregator is seen as more untrustworthy by the other aggregators. For example, in scenario 0 where agg0 is the attacker, the corresponding bar plot indicates that the network of aggregators trust agg0 the least (i.e., π_{agg0} is highest, as calculated in line 6 of Algorithm 4). Similarly, in the other scenarios, we observe that the corresponding attacker amasses the lowest trust level.

It is worthwhile to discuss the distribution of distrust in the network. The distrust is spread among aggregators 0, 1, and 3 considering scenario 0. This can be explained by reflecting on node connectivity of the $\mathcal{G}_c$ graph in Fig. 10, in which agg1 and agg3 are neighbors of agg0. In scenario 4, where agg4 is the dishonest entity, we note that agg4 is the most untrustworthy, yet it is not as untrustworthy as the attackers in other scenarios. We also notice that distrust is more evenly spread across all aggregators in this scenario. In referring to $\mathcal{G}_c$, the reason for this outcome could occur because aggregator 4 has the highest betweenness centrality and is the only *cut vertex* in the network,[10] i.e., agg4 controls information flow between the two clusters (aggregators 0, 1, 2, 3; and aggregators 5, 6). For the communication graph $\mathcal{G}_c$, agg4 dictates the spread of disagreements amongst the aggregators from either cluster. This process to identify attackers will then permit restart of the algorithm to complete the verification process. As billing then occurs the guilty party is penalized, fined, or disconnected from participating in the transactive energy network.

[10] A vertex in an undirected connected graph is a cut vertex iff removing it (and edges through it) disconnects the graph or creates more components than the original graph.

6. Design and implementation on HLF

This section details implementation on HLF of the CPS described in Section 2, using the pricing algorithm from Section 3, and the verification algorithm in Section 4. The choice of a permissioned blockchain architecture, such as HLF, allows consensus protocols that are far less energy-intensive than the proof-of-work consensus protocols employed by permissionless blockchain architectures [57].

6.1. Network setup

The ordering service for any blockchain framework requires a consensus protocol to ensure unambiguous ordering of transactions and guaranteed integrity and consistency of the blockchain across distributed nodes. The developed framework is adaptive to the currently supported Solo, Apache KAFKA, and RAFT algorithms for withstanding crash faults as an ordering service.

Each aggregator is assigned a unique Certificate Authority (CA) [12] that is responsible for dynamically generating certifications (identities) for authenticating prosumers under each aggregator's purview. When joining the network, an individual prosumer shares information on their type of distributed energy resource and requests a new set of credentials with a unique prosumer identification (ID) from their associated aggregator. The credential issued by the aggregator includes a resource specific attribute in addition to the unique ID. Unlike prior works [32], here the ID is embedded in the prosumer's certificate to add additional security to the issued certificate for that prosumer. Any interaction between the blockchain network and an aggregator requires admin identities [12]. In contrast, any communication between EVE and a prosumer involves the use of prosumer's unique identity generated by the respective CA.

The use of "channels" here provides the required isolation between individual aggregators and prosumers [58]. Each transaction in EVE is channel-specific and no data can pass among channels, ensuring privacy and efficient handling of parallel transactions. The EVE blockchain uses $N + 1$ number of separate channels (one channel for each of the N individual aggregator for accessing the corresponding LL, plus one commonchannel among aggregators accessing the GL) to handle access to smart contracts and the ledger. A specific transaction with a specific ledger requires invoking the appropriate smart contract.

6.2. Implementation of EVE through smart contracts

Table 1 lists associations between channels, smart contracts, ledgers, and participant access control. Interactions between smart contracts is summarized in Figs. 11 and 12 for pricing and verification, respectively. Pricing and verification algorithms are written in Python as external applications. `Node.js` applications are developed to handle communications between external applications and smart contracts allowing read and write operations to appropriate ledgers. Note that each application interacts with different smart contracts to accomplish its required objective.

Account Contract (ACT) manages aggregator-level information such as total production/consumption and associated total cost at the bus-level. At the beginning of the bidding window, each aggregator is required to establish their respective bus information by creating timestamped entries in the GL through **ACT**.

Bid Contract (BC) contains a set of functions to manage individual prosumer bids, dispatch values, and budget information. This contract

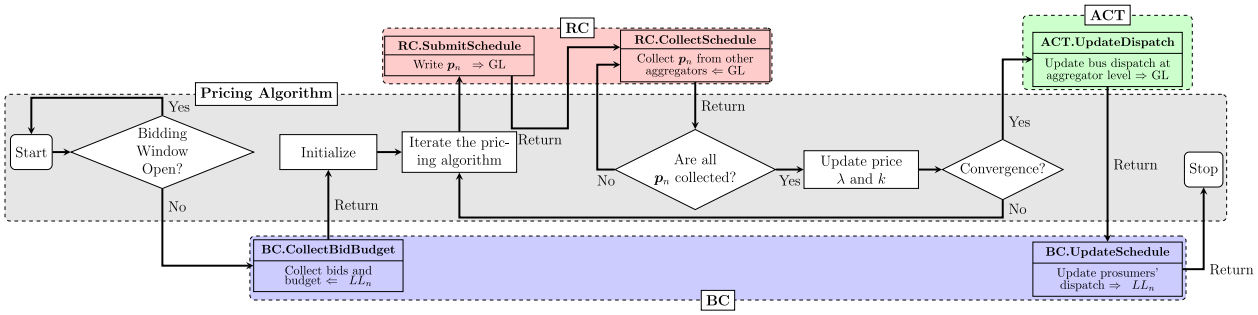

Fig. 11. Graphical depiction of pricing Algorithms 1 and 2 using smart contracts and ledgers for aggregator $n \in \mathcal{N}$.

Table 1
List of channels and associated smart contracts, ledgers, and participant access for $\mathcal{N} = \{0, \ldots, 6\}$.

Channel	Installed smart contract	Ledger access	Participant access
Commonchannel	**Account Contract, Record Contract**	GL	Aggregator n; $\forall n \in \mathcal{N}$
Agg{n}channel	**Bid Contract, Measurement Contract**	LL_n	Aggregator $n \in \mathcal{N}$ and all prosumers $b \in \mathcal{B}^{(n)}$; $n \in \mathcal{N}$

allows a prosumer to use its certificate to submit any number of bids within the bidding window. However, only the last submitted bid within the bidding window is accepted by each aggregator. When submitting a bid, the prosumer is required to provide its asset type (i.e., EV, renewable generation, DA, TCL, storage device, and inflexible load) and corresponding parameters. The smart contract extracts the prosumer's ID from the certificate and ties the submitted bid with the prosumer's unique identity to prevent malicious prosumers from impersonating other prosumers. After the bidding window closes, the aggregator queries submitted bids with their associated budgets from the LL through the aggregator's dedicated channel. The aggregator then executes the distributed pricing algorithm by exchanging information with other aggregators.

Record Contract (RC) is responsible for data exchange for the iterative pricing algorithm through GL. After achieving convergence, each aggregator updates its prosumers' dispatch and cost information in the LL, and bus dispatch values in the GL for future verification. This iteration can be computationally intensive because each interaction (reading and writing) with the GL is a transaction in HLF.

Measurement Contract (MC) handles local measurements from smart meters installed within individual aggregator zones for future verification purposes. Note that the developed algorithm is independent of the sensor location, allowing smart meters to be placed randomly in each aggregator zone for illustration purposes here. The EVE framework can accept measurements at each interval of $\mathcal{T}$ as separate transactions or all measurements for T intervals at the same time as a single transaction. The distributed verification step is always one time step behind the distributed pricing algorithm. For the simulated framework, data from smart meters are generated by solving a nonlinear AC power flow problem using the prosumers' dispatch values as input and then adding noise to it. The **RC** handles information exchange for the distributed verification process. Each transaction is assigned a `type` to ensure separation of entries inside **RC** for pricing and verification, thus allowing the use of a single smart contract to handle both iterative algorithms.

Algorithms 2 and 3 have different data sharing requirements. Each iteration of Algorithm 2 by one aggregator requires information from all aggregators, whereas each iteration of Algorithm 3 requires information from just neighboring aggregators. Private data sharing between neighboring aggregators can be achieved by creating neighbor-specific channels. Nevertheless, this process is burdensome because (1) more channels are required to handle private data sharing, and (2) changes are required in the existing blockchain network if the communication graph changes (due to changes in sensor deployments among aggregators or distribution network reconfiguration). Therefore, the developed framework leverages the ABAC feature of HLF to handle private communication using the existing commonchannel. When a new edge is created in the communication graph, the new relationship can be added to *RC* by upgrading the smart contract following the rules of the initial agreement.

Table 2
Benchmark results for Hyperledger Caliper to Test 200 iterations of information exchange for Algorithm 2.

N	Total transactions	Sent rate (tps)	Max latency (s)	Min latency (s)	Throughput (tps)
6	1200	6	1.8	0.31	6
7	1400	7	1.55	0.32	7
8	1600	8	1.77	0.36	8
9	1800	9	1.59	0.38	9
10	2000	10	1.49	0.39	10

Table 3
Security analysis of reviewed surveys.

Reference	EVE	[32,42]	[59]	[60]	[9,34,61]	[21]
Implementation framework	Hyperledger			Ethereum		
Threat model	✓	x	✓	✓	x	✓
Attack scenario	✓	x	✓	x	x	✓
Physical verification	✓	x	x	x	x	x

6.3. Results

Performance results of the proposed framework are generated using predefined use cases in Hyperledger Caliper. For test cases with varying numbers of aggregators (N) between 6–10, and assuming the pricing algorithm requires 200 iterations for convergence, Table 2 shows benchmark results in terms of the maximum latency, minimum latency, and throughput.[11] For each scenario shown in Table 2, the sending tps rate is equivalent to N because the number of aggregators is the maximum number of writes that can occur to the ledger.

6.4. Security analysis

Security and privacy of the proposed EVE blockchain introduced here are compared against other leading blockchain approaches. Table 3 summaries security considerations of the reviewed works that used blockchain for TE. Most do not fully consider security aspects such

[11] Latency = (time when response received - submit time) in second. Throughput = (total valid committed transactions/total time in seconds for all committed nodes in the network) in transactions per second (tps).

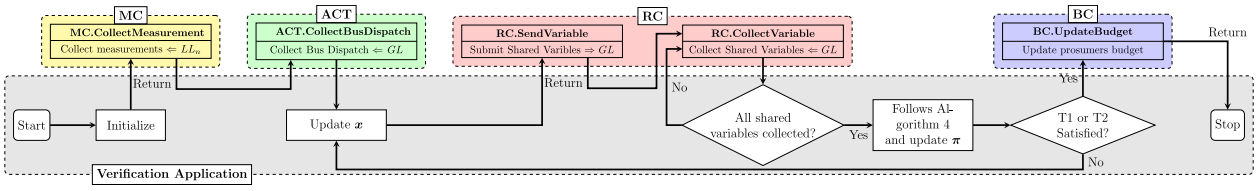

Fig. 12. Graphical depiction of verification Algorithm 3 using smart contracts and ledgers for aggregator $n \in \mathcal{N}$.

as threat model, attack scenario, and verification mechanism at the physical layer. This indicates that the security of these works depends on the built-in security mechanism of the blockchain framework and is not analyzed in-depth for additional threats or weaknesses. In this work, the proposed EVE blockchain framework has been designed and implemented with direct inclusion of cyber-security and specific threat models and attack scenarios. Table 4 provides a summary of potential threats and countermeasures referenced by the NIST Guide to Industrial Control Systems [62] and the HLF security mechanism as they relate to the threat model outlined in Section 4.2.

As the currently supported ordering mechanisms in HLF only provide crash fault tolerance and do not provide BFT [63], a customized BFT-SMART [13] state machine replication and a consensus library has been integrated into this work too. By reinforcing the design with this mechanism, our approach can achieve BFT resilience and durability to avoid a single point of failure. Also, using sensor measurements in the verification stage allows EVE to assure prosumers' compliance with the scheduled transaction.

7. Conclusion

A blockchain-enabled transactive energy platform entitled **Electron Volt Exchange** is presented in this paper. The integration of blockchain allowed a secured process for handling individual bids (prosumers) and collective bids (aggregators). Implementing aggregators for the distributed pricing algorithm allowed efficient use of the Hyperledger Fabric distributed architecture, as demonstrated here for a 141 bus radial network. A secure mechanism for pricing and later verifying economic transactions through a distributed consensus process is also presented. Future work will explore the implementation of additional Hyperledger Fabric features (e.g., idemix, smart contract packaging), other market mechanisms, and verification algorithms through distributed consensus for meshed networks.

CRediT authorship contribution statement

Shammya Saha: Conceptualization, Methodology, Software, Writing - original draft, Data curation. **Nikhil Ravi:** Methodology, Data curation, Writing - original draft, Visualization. **Kári Hreinsson:** Data curation, Writing - original draft, Visualization. **Jaejong Baek:** Software. **Anna Scaglione:** Conceptualization, Writing - review & editing, Software, Supervision, Funding acquisition. **Nathan G. Johnson:** Writing - review & editing, Supervision, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This study was funded in part by the United States Office of Naval Research (ONR) Defense University Research-to-Adoption (DURA) Initiative under award number N00014-18-1-2393.

Appendix A. Demand response resource models

A.1. Electric Vehicles (EVs)

An EV requires a certain amount of charge $\bar{u}$ over a period of length τ. Primary constraints are:

1. Rate of charge when grid-connected is $\dot{u}(t) = -p(t)$ for $0 \le u(t) \le \bar{u}$ and zero when the EV is full, with $p(t) < 0$ since charging is a load.
2. Charging is permitted only at a constant rate of $-\rho$, i.e., $p(t) \in \{0, -\rho\}$. Discharge to the grid is not permitted.
3. Charging has a deadline, i.e., $u(t_d) = u(t_a + \tau_c + \tau_s) = \bar{u}$.

Here t_a denotes the EV arrival time, t_d the EV departure time, τ_c the time needed for charging, and τ_s the leftover (slack) time. Considering a large number of loads, the constraint on charging rate can be relaxed as $-\rho \le p(t) \le 0$. In discrete time and vector form, assuming that the variable $u(t)$ is the energy normalized by the sampling period (i.e., the time that elapses between t and $t+1$), we can write:

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{\ell}, \; u(t_d) = \bar{u}, \;\; -\rho \le p(t) \le 0 \;\forall\; t \in \mathcal{T} \tag{A.1}$$

In (A.1), $\boldsymbol{A}$ computes the finite difference of the state of charge values in $\boldsymbol{u}$. Using $\boldsymbol{J}$ to denote an off-diagonal shift matrix, the following can be written:

$$\boldsymbol{A} = (\boldsymbol{J} - \mathbb{I}) \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}, \;\; \boldsymbol{\ell} = [u(t_a), 0, \dots, 0]^\intercal \in \mathbb{R}^{|\mathcal{T}|\times 1} \tag{A.2}$$

In this case $\boldsymbol{A}^\dagger = \boldsymbol{A}^{-1}$ is a triangular matrix of all negative ones. $\boldsymbol{A}^\dagger$ performs a cumulative sum of the entries of $-\boldsymbol{p}$, which is the inverse operation of taking the finite difference (similar to the integral is the inverse operation of the derivative). In general, we assume a prosumer is willing to pay more for having their EV charged earlier, and expects a discounted electricity price if not charging at full power. This conceptualization of energy price is a function of time, with price decreasing monotonically with time as $\boldsymbol{c}_1$ and $c_d < [\boldsymbol{c}_1]_t \;\forall t \in \mathcal{T}$:

$$C(\boldsymbol{p}) = \boldsymbol{c}_1^\intercal \boldsymbol{p} - c_d \min(\bar{u} - u_{|\mathcal{T}|}, \rho \max(0, \tau_c + \tau_s - |\mathcal{T}|)) \tag{A.3a}$$

$$= \boldsymbol{c}_1^\intercal \boldsymbol{p} - c_d \min(\bar{u} + \mathbf{1}^\intercal \boldsymbol{p}, \rho \max(0, \tau_c + \tau_s - |\mathcal{T}|)) \tag{A.3b}$$

A.2. Deferrable Appliances (DAs)

These loads include appliances such as washers, dryers, and water pumps that can be programmed to start their cycle at different times of day. The feasible set of power demand is based on a load profile $h(t)$, a minimum activation time $0 \le t_a \le |\mathcal{T}| - 1$, and a slack $\tau_s \ge 0$:

$$p(t) = -h(t - t_a - \tau), \quad 0 \le \tau \le \tau_s \tag{A.4}$$

where the price function depends on the slack τ_s. Let us assume without loss of generality that $t_a = 0$ as the arrival time can be embedded into the signal $h(t)$. Suppose also that the duration of $h(t)$ is d and that the slack and time are discrete. The signal $u(t)$ can indicate the time at which DA starts its cycle, which naturally means that $u(t) \in \{0, 1\}$ and the ℓ_1 norm $\|\boldsymbol{u}\|_1 = 1$. Considering a large enough population, this constraint can be relaxed to obtain an approximation of the feasible set as follows:

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{u}, \quad \|\boldsymbol{u}\|_1 = 1 \tag{A.5}$$

Table 4
Feasible threats and countermeasures in EVE.

Layer	Feasible threats	Countermeasures (HLF)	Countermeasures (EVE)
Application	Stealth FDIA, DoS attack, Smart Contract, Malware	MSP (Fabric CA)	MSP, FDIA Detection (Physical Verification)
Blockchain	Relay attacks, Privilege Elevation, Repudiation, Info disclosure, Byzantine Fault, Civil attack	Read/Write Set Validation, MSP Traceability with digital signature, Channel isolation	BFT-SMART with features from HLF
Network	DoS attack, Eclipse attack [64]	TLS	ABAC with features from HLF
Client	Identity Theft, Malware	MSP (Fabric CA), Hardware Security Module	ABAC with features from HLF

where $\boldsymbol{A} \in \mathbb{R}^{(|\mathcal{T}|+d)\times|\mathcal{T}|}$ equal to:

$$\boldsymbol{A}^{\intercal} = -\begin{bmatrix} h(0) & \dots & h(d) & 0 & \dots & 0 \\ 0 & h(0) & \dots & h(d) & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h(0) & \dots & h(d) \end{bmatrix} \tag{A.6}$$

The price a prosumer is willing to pay decreases as the delay increases. The price ranges from a maximum price the consumer is willing to pay to a minimum price that is the lowest possible energy cost. Suppose that we want the cost to grow linearly with time, and let c be the constant in the cost expression. Let $\tilde{c}_1 = \tilde{c}_1 \cdot (|\mathcal{T}|-1, |\mathcal{T}|-2, \dots, 1, 0)$. The cost can be obtained as follows:

$$C(\boldsymbol{p}) = \boldsymbol{c}_1^{\intercal}\boldsymbol{p} + c_0 = \tilde{c}_1 \sum_{t=0}^{|\mathcal{T}|-1} (|\mathcal{T}|-1-t)\cdot u(t) + \tilde{c}_0, \tag{A.7}$$

$$\Rightarrow \quad \boldsymbol{c}_1^{\intercal} = \tilde{\boldsymbol{c}}_1 \boldsymbol{A}^{\dagger}, \quad c_0 = \tilde{c}_0 \tag{A.8}$$

A.3. Thermostatically Controlled Loads (TCLs)

These loads include space heaters, air conditioners, and water heaters. Similar to prior works [65], we assume that the temperature dynamics of a heat pump based TCL can be modeled as a first-order differential equation:

$$C\dot{\theta}(t) = (\theta_o(t) - \theta(t))R^{-1} + p(t)\eta + \varepsilon(t)R^{-1}, \quad p(t) \in \{0, -\rho\} \tag{A.9}$$

with R being thermal resistance, C thermal capacitance, $\theta(t)$ the inside temperature, $\theta_o(t)$ the outdoor temperature, η the efficiency of the heat pump ($\eta > 0$ for cooling and $\eta < 0$ for heating), ρ continuous electrical power rating,[12] and $\varepsilon(t)$ denoting a random perturbation of temperature by external factors such as opening of windows/doors or operation of stoves. We denote the thermostats reference temperature as θ_r. Let:

$$\begin{aligned} u(t) &\triangleq \frac{\theta(t) - \theta_r(t)}{R\eta} \\ \tilde{\ell}(t) &\triangleq \frac{\theta_r(t) - \theta_o(t)}{R\eta} + \frac{\varepsilon(t)}{R\eta} + \frac{C}{\eta}\dot{\theta}_r(t), \quad \tau_h \triangleq CR \end{aligned} \tag{A.10}$$

To express the inter-temporal constraints as well as the comfort zone of the user, we can rearrange and relax the set of constraints in (A.9) as follows:

$$p(t) = \frac{C}{\eta}(\dot{\theta}(t) - \dot{\theta}_r(t)) + \frac{\theta(t) - \theta_r(t)}{R\eta} + \frac{\theta_r(t) - \theta_o(t)}{R\eta} + \frac{\varepsilon(t)}{R\eta} + \frac{C}{\eta}\dot{\theta}_r(t) \tag{A.11a}$$

$$= \tau_h\dot{u}(t) + u(t) + \tilde{\ell}(t), \ \underline{u} \le u(t) \le \overline{u}, \ -\rho \le p(t) \le 0 \tag{A.11b}$$

where we have relaxed the integer constraint $p(t) \in \{0, -\rho\}$.

It is notable that the $\tilde{\ell}(t)$ term in this affine relationship is random, since the outdoor temperature is random and so is the perturbation of the indoor temperature from sources other than the heat pump.

[12] Water heaters can be described using the same principles, with an additional energy loss component describing the hot water being replaced by cold water. However, in this paper, we will focus on heat pump based TCL because they are more dependent on external temperatures than water boilers.

In discrete time, we can average the behavior and approximate the relationship as follows:

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{\ell}, \quad \underline{u} \le u(t) \le \overline{u}, \quad -\rho \le p(t) \le 0 \tag{A.12}$$

Using (A.10) and (A.11a) the following can be written:

$$\begin{aligned} &\boldsymbol{A} = \tau_h(\mathbb{I} - \boldsymbol{J}) + \mathbb{I}, \ \boldsymbol{A}^{\dagger} = \boldsymbol{A}^{-1} = ((\tau_h + 1)\mathbb{I} - \tau_h\boldsymbol{J})^{-1} \\ &\boldsymbol{\ell} = \tilde{\boldsymbol{\ell}} - \tau_h[u(0), \boldsymbol{0}^{\intercal}]^{\intercal} \in \mathbb{R}^{|\mathcal{T}|\times 1} \end{aligned} \tag{A.13}$$

and $\tilde{\boldsymbol{\ell}} = (\tilde{\ell}(1), \dots, \tilde{\ell}(|\mathcal{T}|))$ with $\tilde{\ell}(t)$ being defined in (A.10). The price demand function should represent the prosumer's willingness to deviate from reference temperature. The prosumer is willing to pay less for larger temperature deviations than expected. We represent this cost as quadratic with the value of $\boldsymbol{u}$, i.e., proportional to $\|\boldsymbol{u}\|^2$. This means that the demand function is $\tilde{c}_0 - \tilde{c}_2\|\boldsymbol{u}\|^2$ and:

$$C(\boldsymbol{p}) = \boldsymbol{p}^{\intercal}\boldsymbol{C}_2\boldsymbol{p} + \boldsymbol{p}^{\intercal}\boldsymbol{c}_1 + c_0 = \tilde{c}_0 - \tilde{c}_2\left\|\boldsymbol{A}^{\dagger}(\boldsymbol{p} - \boldsymbol{\ell})\right\|^2 \tag{A.14}$$

where:

$$\rightarrow \quad \boldsymbol{C}_2 = -\tilde{c}_2\boldsymbol{A}^{-2}, \ \boldsymbol{c}_1 = 2\tilde{c}_2\boldsymbol{\ell}\boldsymbol{A}^{-2}, \quad c_0 = \tilde{c}_0 - \tilde{c}_2\boldsymbol{\ell}^{\intercal}\boldsymbol{A}^{-2}\boldsymbol{\ell} \tag{A.15}$$

A.4. Storage devices

Typically, battery rate of charge or discharge is constrained to be a constant value, meaning $p(t) \in \{-\rho, 0, \rho\}$. If we relax this non-convex constraint as done before, the load constraints are analogous to that of an EV, except that the unit can discharge:

$$\boldsymbol{p} = (\mathbb{I} - \boldsymbol{J})\boldsymbol{u} - [u(0), \boldsymbol{0}], \quad |p(t)| \le \rho, \quad 0 \le u(t) \le \overline{u} \tag{A.16}$$

where $\boldsymbol{J}$ is the shift matrix, shifting to the right each of the entries of $\boldsymbol{u}$. Here it is natural to assume that the price for discharging is higher than the price of charging, but it is also possible to express a cost that depends on the battery state. In addition, if storage is charged by the random injection of, for instance, solar PV, the forecast can be incorporated into the vector $\boldsymbol{\ell}$ in the model. We ignore other complexities here for simplicity. Considering $(a)_+ = \max(0, a)$, we can express the cost as:

$$C(\boldsymbol{p}) = (\boldsymbol{p})_+^{\intercal}(\boldsymbol{c}_1^+) + (-\boldsymbol{p})_+^{\intercal}(\boldsymbol{c}_1^-) \tag{A.17}$$

where $\boldsymbol{c}_1^+$ and $\boldsymbol{c}_1^-$ are non-negative cost vectors.

A.5. Renewables

The power injection from wind or solar PV has no marginal cost, therefore the only meaningful way for renewables to participate is posting a forecast of future production $\boldsymbol{p}$ with zero cost, i.e., $C(\boldsymbol{p}) = 0$.

A.6. Supply from the transmission grid

We assume that the transmission grid appears in the system as the slack bus and has a certain cost function for selling and a certain cost function for buying power below and above a schedule $\boldsymbol{p}_s$ that

was cleared in previous wholesale market stages. Therefore, at the substation bus we have a single supplier with supply function:

$$C(\boldsymbol{p}) = (\boldsymbol{p} - \boldsymbol{p}_s)_+^{\mathrm{I}}(c_1^+) + (-\boldsymbol{p} + \boldsymbol{p}_s)_+^{\mathrm{I}}(c_1^-) \tag{A.18}$$

For simplicity we assume that any deviation is feasible, so that the dispatch is always feasible, and the slack bus compensates for any shortfall or surplus of power subject to physical constraints in Appendix B.

Appendix B. Electric grid constraints

A radial electrical distribution system can be represented by a set of buses $\mathcal{B}$, edges $\mathcal{E}_e$, and a root node (commonly the substation node or slack bus) where each edge $\ell = (b, b') \in \mathcal{E}_e$ and $b, b' \in \mathcal{B}$. The *from* and *to* functions are defined as $f(\ell) = b$ and $t(\ell) = b'$ to return the source node and incident node for an edge, respectively. The inverse function $t^{-1}(b)$ gives back the edge pointing to bus b (for a radial graph the *to* bus b for each edge is unique) and $f^{-1}(b)$ returns all the edges originating at bus b. Dropping the time index for brevity, the power flow equations at each of the branches $\ell \in \mathcal{E}_e$ are given by [54]:

$$p_{t(\ell)} = P_\ell - \sum_{\ell' \in f^{-1}(t(\ell))} P_{\ell'} - \mathrm{Re}(z_\ell) c_\ell^2 \tag{B.1a}$$

$$q_{t(\ell)} = Q_\ell - \sum_{\ell' \in f^{-1}(t(\ell))} Q_{\ell'} - \mathrm{Im}(z_\ell) c_\ell^2 \tag{B.1b}$$

$$v_{f(\ell)}^2 = v_{t(\ell)}^2 + 2(\mathrm{Re}(z_\ell) P_\ell + \mathrm{Im}(z_\ell) Q_\ell) - |z_\ell|^2 c_\ell^2 \tag{B.1c}$$

$$v_{f(\ell)}^2 = \frac{P_\ell^2 + Q_\ell^2}{c_\ell^2}. \tag{B.1d}$$

All these equations are linear in the bus and branch quantities $(p_b(t), q_b(t), v_b^2(t))$ and $(P_\ell(t), Q_\ell(t), c_\ell^2(t))$ except (B.1d). However, including an auxiliary variable $x'_\ell = \frac{P_\ell^2 + Q_\ell^2}{c_\ell^2}$ can simplify the description of the physical constraints. By expressing $\boldsymbol{x}(t)$ as the auxiliary variables, then without loss of generality (B.1a) to (B.1d) can be written in the following linear form:

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{H}_{\mathcal{A}}\boldsymbol{x}_{\mathcal{A}} + \boldsymbol{H}_u\boldsymbol{x}_u = \boldsymbol{0}, \tag{B.2}$$

Though constraints in (13) are non-linear in general, here the constraints are relaxed by ignoring the non-linear relationships among the auxiliary variables and the remaining entries of the vector $\boldsymbol{x}$. The measurements are represented by $\boldsymbol{z} = \boldsymbol{x}_{\mathcal{A}} + \boldsymbol{\epsilon}$ and the physics of the system implies that $\boldsymbol{x}_{\mathcal{A}}$ satisfies (B.2).

From the vantage point of each aggregator region n, only a subset of the variables $\boldsymbol{x}_{\mathcal{A}}^{(n)}$ are measured, meaning that $\boldsymbol{z}^{(n)} = \boldsymbol{x}_{\mathcal{A}}^{(n)} + \boldsymbol{\epsilon}^{(n)}$. Also, not all constraints in (B.2) include the variables $\boldsymbol{x}^{(n)}$. Hence, the equations that involve buses/lines in region i can be isolated and written as:

$$\boldsymbol{h}^{(n)}(\boldsymbol{x}^{(n)}) = \boldsymbol{H}^{(n)}\boldsymbol{x}^{(n)} = \boldsymbol{0}. \tag{B.3}$$

These equations are used to verify that measurements and injections values are consistent with the laws of physics. The available measurements in each zone and the neighboring zones are used to interpolate the unknown variables as discussed in Section 4.3.1.

References

[1] Lin J, Pipattanasomporn M, Rahman S. Comparative analysis of auction mechanisms and bidding strategies for P2P solar transactive energy markets. Appl Energy 2019;255:113687. http://dx.doi.org/10.1016/j.apenergy.2019.113687.

[2] Nguyen HT, Battula S, Takkala RR, Wang Z, Tesfatsion L. An integrated transmission and distribution test system for evaluation of transactive energy designs. Appl Energy 2019;240:666–79. http://dx.doi.org/10.1016/j.apenergy.2019.01.178.

[3] Janko SA, Johnson NG. Scalable multi-agent microgrid negotiations for a transactive energy market. Appl Energy 2018;229:715–27. http://dx.doi.org/10.1016/j.apenergy.2018.08.026.

[4] Behboodi S, Chassin DP, Djilali N, Crawford C. Transactive control of fast-acting demand response based on thermostatic loads in real-time retail electricity markets. Appl Energy 2018;210:1310–20. http://dx.doi.org/10.1016/j.apenergy.2017.07.058.

[5] van Leeuwen G, AlSkaif T, Gibescu M, van Sark W. An integrated blockchain-based energy management platform with bilateral trading for microgrid communities. Appl Energy 2020;263:114613. http://dx.doi.org/10.1016/j.apenergy.2020.114613.

[6] Mohanta BK, Panda SS, Jena D. An overview of smart contract and use cases in blockchain technology. In: 2018 9th international conference on computing, communication and networking technologies (ICCCNT). 2018, p. 1–4. http://dx.doi.org/10.1109/ICCCNT.2018.8494045.

[7] Hussain SMS, Farooq SM, Ustun TS. Implementation of blockchain technology for energy trading with smart meters. In: 2019 innovations in power and advanced computing technologies (I-PACT), Vol. 1. 2019, p. 1–5. http://dx.doi.org/10.1109/i-PACT44901.2019.8960243.

[8] Li N, Chen L, Low SH. Optimal demand response based on utility maximization in power networks. In: 2011 IEEE power and energy society general meeting. IEEE; 2011, p. 1–8. http://dx.doi.org/10.1109/PES.2011.6039082.

[9] Münsing E, Mather J, Moura S. Blockchains for decentralized optimization of energy resources in microgrid networks. In: 2017 IEEE conference on control technology and applications (CCTA). IEEE; 2017, p. 2164–71. http://dx.doi.org/10.1109/CCTA.2017.8062773.

[10] Olivella-Rosell P, Lloret-Gallego P, Munné-Collado I, Villafafila-Robles R, Sumper A, Ottessen SO, et al. Local flexibility market design for aggregators providing multiple flexibility services at distribution network level. Energies 2018;11(4). http://dx.doi.org/10.3390/en11040822.

[11] Vuković O, Dán G. Security of fully distributed power system state estimation: Detection and mitigation of data integrity attacks. IEEE J Sel Areas Commun 2014;32(7):1500–8. http://dx.doi.org/10.1109/JSAC.2014.2332106.

[12] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth eurosys conference. EuroSys '18, New York, NY, USA: Association for Computing Machinery; 2018, http://dx.doi.org/10.1145/3190508.3190538.

[13] Sousa J, Bessani A, Vukolic M. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In: 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE; 2018, p. 51–8. http://dx.doi.org/10.1109/DSN.2018.00018.

[14] Yuan E, Tong J. Attributed based access control (abac) for web services. In: IEEE international conference on web services (ICWS'05). 2005, p. 569. http://dx.doi.org/10.1109/ICWS.2005.25.

[15] An Introduction to Hyperledger. Technical Report, The Linux Foundation; 2018.

[16] Musleh AS, Yao G, Muyeen SM. Blockchain applications in smart grid–review and frameworks. IEEE Access 2019;7:86746–57. http://dx.doi.org/10.1109/ACCESS.2019.2920682.

[17] Mylrea M, Gourisetti SNG. Blockchain for smart grid resilience: Exchanging distributed energy at speed, scale and security. In: 2017 resilience week (RWS). 2017, p. 18–23. http://dx.doi.org/10.1109/RWEEK.2017.8088642.

[18] Wu Y, Wu Y, Guerrero JM, Vasquez JC. Digitalization and decentralization driving transactive energy internet: Key technologies and infrastructures. Int J Electr Power Energy Syst 2021;126:106593. http://dx.doi.org/10.1016/j.ijepes.2020.106593.

[19] Hayes B, Thakur S, Breslin J. Co-simulation of electricity distribution networks and peer to peer energy trading platforms. Int J Electr Power Energy Syst 2020;115:105419. http://dx.doi.org/10.1016/j.ijepes.2019.105419.

[20] Sanseverino ER, Di Silvestre ML, Gallo P, Zizzo G, Ippolito M. The blockchain in microgrids for transacting energy and attributing losses. In: 2017 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). 2017, p. 925–30. http://dx.doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.142.

[21] Danzi P, Angjelichinoski M, Stefanović Č, Popovski P. Distributed proportional-fairness control in microgrids via blockchain smart contracts. In: 2017 IEEE international conference on smart grid communications (SmartGridComm). IEEE; 2017, p. 45–51. http://dx.doi.org/10.1109/SmartGridComm.2017.8340713.

[22] Silva FC, A. Ahmed M, Martínez JM, Kim Y-C. Design and implementation of a blockchain-based energy trading platform for electric vehicles in smart campus parking lots. Energies 2019;12(24). http://dx.doi.org/10.3390/en12244814.

[23] Hua W, Jiang J, Sun H, Wu J. A blockchain based peer-to-peer trading framework integrating energy and carbon markets. Appl Energy 2020;279:115539. http://dx.doi.org/10.1016/j.apenergy.2020.115539.

[24] Lu X, Guan Z, Zhou X, Du X, Wu L, Guizani M. A secure and efficient renewable energy trading scheme based on blockchain in smart grid. In: 2019 IEEE 21st international conference on high performance computing and communications; IEEE 17th international conference on smart city; IEEE 5th international conference on data science and systems (HPCC/SmartCity/DSS). 2019, p. 1839–44. http://dx.doi.org/10.1109/HPCC/SmartCity/DSS.2019.00253.

[25] Zhao S, Wang B, Li Y, Li Y. Integrated energy transaction mechanisms based on blockchain technology. Energies 2018;11(9). http://dx.doi.org/10.3390/en11092412.

[26] Agung AAG, Handayani R. Blockchain for smart grid. J King Saud Univ - Comput Inf Sci 2020. http://dx.doi.org/10.1016/j.jksuci.2020.01.002.

[27] Kang J, Yu R, Huang X, Maharjan S, Zhang Y, Hossain E. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. IEEE Trans Ind Inf 2017;13(6):3154–64. http://dx.doi.org/10.1109/TII.2017.2709784.

[28] Ferreira JC, Martins AL. Building a community of users for open market energy. Energies 2018;11(9). http://dx.doi.org/10.3390/en11092330.

[29] Danzi P, Hambridge S, Stefanović Č, Popovski P. Blockchain-based and multi-layered electricity imbalance settlement architecture. In: 2018 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm). 2018, p. 1–7. http://dx.doi.org/10.1109/SmartGridComm.2018.8587577.

[30] Jogunola O, Hammoudeh M, Adebisi B, Anoh K. Demonstrating blockchain-enabled peer-to-peer energy trading and sharing. In: 2019 IEEE Canadian conference of electrical and computer engineering (CCECE). 2019, p. 1–4. http://dx.doi.org/10.1109/CCECE.2019.8861525.

[31] Gür AÖ, Öksüzer c, Karaarslan E. Blockchain based metering and billing system proposal with privacy protection for the electric network. In: 2019 7th international istanbul smart grids and cities congress and fair (ICSG). 2019, p. 204–8. http://dx.doi.org/10.1109/SGCF.2019.8782375.

[32] Pipattanasomporn M, Kuzlu M, Rahman S. A blockchain-based platform for exchange of solar energy: Laboratory-scale implementation. In: 2018 international conference and utility exhibition on green energy for sustainable development (ICUE). 2018, p. 1–9. http://dx.doi.org/10.23919/ICUE-GESD.2018.8635679.

[33] Christidis K, Sikeridis D, Wang Y, Devetsikiotis M. A framework for designing and evaluating realistic blockchain-based local energy markets. Appl Energy 2021;281:115963. http://dx.doi.org/10.1016/j.apenergy.2020.115963.

[34] Coignard J, Munsing E, MacDonald J, Mather J. Co-simulation framework for blockchain based market designs and grid simulations. In: 2018 IEEE power energy society general meeting (PESGM). 2018, p. 1–5. http://dx.doi.org/10.1109/PESGM.2018.8586124.

[35] Che Z, Wang Y, Zhao J, Qiang Y, Ma Y, Liu J. A distributed energy trading authentication mechanism based on a consortium blockchain. Energies 2019;12(15). http://dx.doi.org/10.3390/en12152878.

[36] Zhang Y, Shi Q. An intelligent transaction model for energy blockchain based on diversity of subjects. Alexandria Eng J 2020. http://dx.doi.org/10.1016/j.aej.2020.10.005.

[37] Pop C, Cioara T, Antal M, Anghel I, Salomie I, Bertoncini M. Blockchain based decentralized management of demand response programs in smart energy grids. Sensors 2018;18(1):162. http://dx.doi.org/10.3390/s18010162.

[38] Andoni M, Robu V, Flynn D, Abram S, Geach D, Jenkins D, et al. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. Renew Sustain Energy Rev 2019;100:143–74. http://dx.doi.org/10.1016/j.rser.2018.10.014.

[39] Xie L, Mo Y, Sinopoli B. False data injection attacks in electricity markets. In: 2010 first IEEE international conference on smart grid communications. IEEE; 2010, p. 226–31. http://dx.doi.org/10.1109/SMARTGRID.2010.5622048.

[40] Deka D, Baldick R, Vishwanath S. Optimal data attacks on power grids: Leveraging detection & measurement jamming. In: 2015 IEEE international conference on smart grid communications (SmartGridComm). IEEE; 2015, p. 392–7. http://dx.doi.org/10.1109/SmartGridComm.2015.7436332.

[41] Luo X, Wang X, Zhang M, Guan X. Distributed detection and isolation of bias injection attack in smart energy grid via interval observer. Appl Energy 2019;256:113703. http://dx.doi.org/10.1016/j.apenergy.2019.113703.

[42] Wang S, Taha AF, Wang J, Kvaternik K, Hahn A. Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. IEEE Trans Syst Man Cybern: Syst 2019;49(8):1612–23. http://dx.doi.org/10.1109/TSMC.2019.2916565.

[43] Guan Z, Lu X, Yang W, Wu L, Wang N, Zhang Z. Achieving efficient and privacy-preserving energy trading based on blockchain and abe in smart grid. J Parallel Distrib Comput 2021;147:34–45. http://dx.doi.org/10.1016/j.jpdc.2020.08.012.

[44] Mahmoudi N, Saha TK, Eghbal M. A new trading framework for demand response aggregators. In: 2014 IEEE PES general meeting | conference exposition. 2014, p. 1–5. http://dx.doi.org/10.1109/PESGM.2014.6938882.

[45] California's Community Choice Program. 2020, URL: https://cal-cca.org/about/.

[46] Koch S. Chapter 2 - assessment of revenue potentials of ancillary service provision by flexible unit portfolios. In: Du P, Lu N, editors. Energy storage for smart grids. Boston: Academic Press; 2015, p. 35–66. http://dx.doi.org/10.1016/B978-0-12-410491-4.00002-6.

[47] Mahmoudi N, Saha TK, Eghbal M. A new trading framework for demand response aggregators. In: 2014 IEEE PES general meeting | conference exposition. 2014, p. 1–5. http://dx.doi.org/10.1109/PESGM.2014.6938882.

[48] Gupta S, Rani R. A comparative study of elasticsearch and couchdb document oriented databases. In: 2016 international conference on inventive computation technologies (ICICT), Vol. 1. 2016, p. 1–4. http://dx.doi.org/10.1109/INVENTIVE.2016.7823252.

[49] Chang T-H, Alizadeh M, Scaglione A. Coordinated home energy management for real-time power balancing. In: 2012 IEEE power and energy society general meeting. IEEE; 2012, p. 1–8. http://dx.doi.org/10.1109/PESGM.2012.6345639.

[50] Li L, Scaglione A, Manton JH. Distributed principal subspace estimation in wireless sensor networks. IEEE J Sel Top Sign Proces 2011;5(4):725–38.

[51] Karakoç N, Scaglione A, Nedić A. Multi-layer decomposition of optimal resource sharing problems. In: Proc. in IEEE Conf. on decision and control (CDC). Miami Beach, FL; 2018, p. 1–6. http://dx.doi.org/10.1109/CDC.2018.8619777.

[52] Ravi N, Scaglione A. Detection and isolation of adversaries in decentralized optimization for non-strongly convex objectives. IFAC-PapersOnLine 2019;52(20):381–6. http://dx.doi.org/10.1016/j.ifacol.2019.12.185, 8th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2019.

[53] Liu Y, Ning P, Reiter MK. False data injection attacks against state estimation in electric power grids. ACM Trans Inf Syst Secur 2011;14(1):13. http://dx.doi.org/10.1145/1653662.1653666.

[54] Baran ME, Wu FF. Network reconfiguration in distribution systems for loss reduction and load balancing. IEEE Trans Power Deliv 1989;4(2):1401–7. http://dx.doi.org/10.1109/61.25627.

[55] Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 2011;3(1):1–122. http://dx.doi.org/10.1561/9781601984616.

[56] Zimmerman R, Murillo-Sánchez C, Thomas R. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Trans Power Syst 2011;26(1):12–9. http://dx.doi.org/10.1109/TPWRS.2010.2051168.

[57] Wang S, Taha AF, Wang J. Blockchain-assisted crowdsourced energy systems. In: 2018 IEEE power & energy society general meeting (PESGM). IEEE; 2018, p. 1–5. http://dx.doi.org/10.1109/PESGM.2018.8585864.

[58] Aw N. Private data collections: A high-level overview. 2018, URL: https://www.hyperledger.org/blog/2018/10/23/private-data-collections-a-high-level-overview.

[59] Gai K, Wu Y, Zhu L, Qiu M, Shen M. Privacy-preserving energy trading using consortium blockchain in smart grid. IEEE Trans Ind Inf 2019. http://dx.doi.org/10.1109/TII.2019.2893433.

[60] Laszka A, Eisele S, Dubey A, Karsai G, Kvaternik K. Transax: A blockchain-based decentralized forward-trading energy exchanged for transactive microgrids. In: 2018 IEEE 24th international conference on parallel and distributed systems (ICPADS). 2018, p. 918–27. http://dx.doi.org/10.1109/PADSW.2018.8645001.

[61] Sabounchi M, Wei J. Towards resilient networked microgrids: Blockchain-enabled peer-to-peer electricity trading mechanism. In: 2017 IEEE conference on energy internet and energy system integration (EI2). 2017, p. 1–5. http://dx.doi.org/10.1109/EI2.2017.8245449.

[62] Stouffer K, Falco J, Scarfone K. Guide to industrial control systems (ICS) security. NIST Spec Publ 2015;800(82).

[63] Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus. Technical Report, Hyperledger Architecture Working Group; 2017.

[64] Heilman E, Kendler A, Zohar A, Goldberg S. Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX security symposium (USENIX security 15). 2015, p. 129–44.

[65] Callaway DS. Tapping the energy storage potential in electric loads to deliver load following and regulation, with application to wind energy. Energy Convers Manage 2009;50(5):1389–400. http://dx.doi.org/10.1016/j.enconman.2008.12.012.