TRANSAX: A Blockchain-based Decentralized Forward-Trading Energy Exchange for Transactive Microgrids

Aron Laszka
Computer Science
University of Houston
Houston, TX

Scott Eisele, Abhishek Dubey, Gabor Karsai
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN

Karla Kvaternik
Siemens Corporate Technology
Princeton, NJ

Published in the proceedings of the 24th IEEE International Conference on Parallel and Distributed Systems (IEEE ICPADS 2018).

***Abstract*—Power grids are undergoing major changes due to rapid growth in renewable energy and improvements in battery technology. Prompted by the increasing complexity of power systems, decentralized IoT solutions are emerging, which arrange local communities into transactive microgrids. The core functionality of these solutions is to provide mechanisms for matching producers with consumers while ensuring system safety. However, there are multiple challenges that these solutions still face: privacy, trust, and resilience. The privacy challenge arises because the time series of production and consumption data for each participant is sensitive and may be used to infer personal information. Trust is an issue because a producer or consumer can renege on the promised energy transfer. Providing resilience is challenging due to the possibility of failures in the infrastructure that is required to support these market based solutions. In this paper, we develop a rigorous solution for transactive microgrids that addresses all three challenges by providing an innovative combination of MILP solvers, smart contracts, and publish-subscribe middleware within a framework of a novel distributed application platform, called Resilient Information Architecture Platform for Smart Grid. Towards this purpose, we describe the key architectural concepts, including fault tolerance, and show the trade-off between market efficiency and resource requirements.**

***Index Terms*—smart grid, distributed ledger, decentralized application, transactive energy, system resilience, blockchain, smart contract, cyber-physical system**

I. Introduction

Power grids are undergoing major changes due to the rapid adoption of renewable energy resources, such as wind and solar power [1], [2]. For example, 4,143 megawatts of solar panels were installed in the third quarter of 2016 [3]. This capacity is predicted to grow from 4% of the total global energy production in 2015 to 29% in 2040 [4]. Simultaneously, battery technology costs per kWh have been dropping significantly [5], reaching grid parity [6]. These trends are enabling a decentralized vision for the future of power-grid operations in which local peer-to-peer energy trading within microgrids can be used to reduce the load on distribution system operators (DSO), leading to the development of Transactive Energy Systems (TES) [7]–[10]. Such mechanisms can improve system reliability and efficiency by integrating inverter-based renewable resources into the grid and by supplying power to the local loads when the main grid is interrupted.

To accomplish the goal of transactive energy, individual prosumers[1] need to engage in interactions, negotiate with each other, enter agreements, and make proactive run-time decisions—individually and collectively—while responding to changing demands and environmental conditions. In theory, these interactions could happen in a centralized manner by communicating relevant variables to a central controller, which would compute and broadcast the "optimal" control settings back to each individual prosumer. However, this system would not scale well as the number of coordinating parties increases. It would also adversely affect resilience because of increased risks for data corruption and loss during transmission. Further, the centralized controller would constitute a single point of failure. On the other hand, distributed optimization solutions might suffer from the same scalability challenges, and the "distribution" of the optimization problem often requires the over-simplification of objective functions, which would result in losing the guarantees of a globally "optimal" solution. In light of this, novel "decentralized" solutions are needed, in which individual prosumers operate with autonomous controllers that can trade on their behalf in a market, which is itself decentralized. However, creating such decentralized solutions is challenging due to a number of problems.

The first problem is ensuring the physical stability and safety of the grid apparatus, which requires dynamically balancing supply and demand without violating line capacity constraints. The second one is a distributed systems problem, which requires ensuring that this peer-to-peer market operates in a trustworthy manner even if some of the nodes are malicious, compromised, or faulty. The third problem

The information, data, or work presented herein was funded in part by Siemens, CT and Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000666.

[1] A prosumer is a home that can not only consume, but also produce surplus energy. Homes without production will be simply called consumers.

is related to privacy. While non-transactive smart metering systems require sharing prosumer information only with the DSO, transactive systems need to disseminate information among the participants to enable finding trade partners. The dissemination of trading information threatens the privacy of prosumers since it may expose their private information to anyone in the same microgrid. Further, data collected from energy transactions is expected to be more fine-grained than data collected by currently deployed smart meters [11], and it may be used to infer personal information about the market participants. For example, a participant's presence or absence at their residence might be inferable from their energy future offers (e.g., if a prosumer posts an energy selling offer, the residents are less likely to be at home). The fourth problem is resilience. Failures in distributed computing systems are a fact, and hence the transactive system must be able to tolerate failures by either mitigating faults or adapting the system to a different configuration.

Contributions: In this paper, we describe the design and implementation of TRANSAX, a transactive decentralized platform built over a distributed middleware, called Resilient Information Architecture Platform for Smart Grid (RIAPS) [12], [13]. RIAPS isolates the hardware details from the algorithms and provides essential mechanisms for resource management, fault tolerance, and security. An integrated distributed ledger and smart contracts provide us with the mechanisms to provide consensus and trust. This is in line with the recent trends in the research community and industry focused on transactive energy markets [14], [15]. Although disintermediation of trust is widely regarded as the primary feature of blockchain-based transaction systems [16], their use in TES is appealing also because they elegantly integrate the ability to immutably record the ownership and transfer of assets, with essential distributed computing services, such as Byzantine fault-tolerant consensus on the ledger state as well as event chronology. The ability to establish consensus on state and timing is important in the context of TES since these systems are envisioned to involve the participation of self-interested parties, interacting with one another via a distributed computing platform that executes transaction management. We provide privacy by using a mixing service, which prevents tracing assets being traded back to their owner, as described in our prior work [17][2]. However, unlike [17], we consider an automated matching system that maximizes the amount of energy traded within the local market, while satisfying safety constraints. Finally, we describe and evaluate an extension to the RIAPS framework that implements distributed fault detection and mitigation mechanisms. These mechanisms are critical for resilient operation of TRANSAX.

Outline: The outline of this paper is as follows. We explain the problem of transactive energy systems using an example in Section II. Then, we contextualize our contributions in TRANSAX using related research in Section III. We describe TRANSAX in Section IV, which is followed by an evaluation using a case study in Section V. Finally, we conclude with discussions in Section VI.

II. Transactive Energy Problem

Consider a microgrid with a set of feeders arranged in a radial topology.[3] A feeder[4] has a fixed set of nodes, each representing a residential load or a combination of load and distributed energy resources (DERs), such as rooftop solar and batteries. Each node is associated with a participant in the local peer-to-peer energy trading market. There is a distribution system operator (DSO), that also participates in the market and may thus use the market to incentivize timed energy production within the microgrid to aid in grid stabilization and promotion of related ancillary services [18]. In addition, the DSO supplies residual demand not met through the local market. The participants settle trades in advance, which allows them to schedule their transfer of power into the local distribution system. Alternatively, a mechanism can be responsible for matching the producers and consumers. There are typically three phases in these operations: discovery of compatible offers, matching of buying offers to selling offers (which may have been submitted either by each prosumer individually or by automated matching mechanisms). Once the matching is done, the energy transactions and financial transactions are then handled at a later time.

Example 1: Consider a community with two prosumers (P_1, P_2) and one consumer (C_1) on a single feeder. To make the problem of matching energy offers tractable, lets assume that the offers are made and matched for discrete time intervals. These intervals quantize the whole day, and their length can be a parameter of the problem setup. For the sake of example, lets assume that each day is divided into 15 minute intervals. Lets assume that P_1 has the ability to transfer 10 kW into the feeder during interval 48, which translates to 12:00pm–12:15pm. Assume similarly that P_2 can also provide 30 kW to the feeder in interval 48, but it has battery storage. Since P_2 has battery—unlike P_1, who must either transfer the energy or send the energy into the group—P_2 can delay the transfer until a future interval, e.g., interval 49. Now suppose that C_1 needs to consume 30 kW in interval 48 and 10 kW in interval 49. All the prosumers and consumer must provide these requirements to the market mechanism, which will then provide a matching solution. A possible solution would be to provide all 30 kW to C_1 from P_2 in interval 48. However, that will lead to the waste of energy provided by P_1. Thus, a better solution will be to consume 10 kW from P_1 in interval 48 and 20 kW from P_2 in interval 48. Then, transfer 10 kW from P_2 in interval 49, which is more efficient then the first matching as it allows more energy (summed across the intervals) to be transferred.

[2]Note that since the mixing implementation was discussed in [17], in this paper we focus on other key contributions, including resilience.

[3]The methods developed in this paper are extensible to more general tree topologies involving branching. We work with a radial topology to simplify our notation.

[4]A feeder element in electrical distribution is a power line transferring power from a distribution substation to distribution transformers or from distribution transformers to the end homes.

Note that the second solution requires the market to consider future intervals while solving the problem, which increases the size of the optimization problem. Further, it should be noted that if the information about C_1's offer is made public, then one can estimate that C_1 was doing heavy machine work during interval 48 and there was substantially lower activity during interval 49.

Based on this example, it is clear that there are five basic requirement that must be met by any solution.

- The first requirement is the existence of an appropriate communication and messaging architecture. The decentralized platform must collect participants' offers and make them available to buyers and sellers, and the market algorithm must communicate clearing prices and buyer-seller matchings. These messages must be reliably delivered under strict timing constraints, derived from the deadline by which a trade must clear.
- The trading activity shall not compromise the stability of the physical system operation. For example, capacity constraints along any feeder should be respected. Specifically, each feeder is rated for a maximal power capacity. For example, if the feeder capacity is only 10 kW, then C_1 should not consume 30 kW.
- There are a number of parameters of the system that should be made configurable. For example, the number of intervals to look ahead while solving the matching problem is one such parameter. Another parameter is the prediction window for each prosumer. Note that Example 1 required that the prosumers make their offers for future intervals available.
- Information such as the amount of energy produced, consumed, bought, or sold by any prosumer should be available only to the Distribution System Operator. All bids and asks as well as the matching thereof should remain anonymous to the other participants.
- The failure of a prosumer or market agent, including any solvers that are required to search for a matching solution, must not compromise the system. Further, there should be mechanisms to ensure that everyone agrees to and conforms to the decisions made by the market mechanism.

III. Related Work

Implementing a Transaction Management Platforms (TMP) requires a communication architecture, as well as trading mechanisms that provide the capability to match the bids and asks. Blockchain-based solutions have the potential to enable large-scale energy trading based on distributed consensus systems. However, popular blockchain solutions, such as Bitcoin [19] and Ethereum [20], suffer from design limitations that prevent their direct application to validating energy trades.

For example, Aitzhan and Svetinovic implemented a proof-of-concept platform for decentralized smart-grid energy trading using blockchains, but their system is based on proof-of-work consensus, and they do not consider grid control and stability, or scalability [21]. Additionally, there is still the problem of privacy—all transactions in these systems are public [22].

Most works discussing privacy look at it from the context of smart meters. McDaniel and McLaughlin discuss privacy concerns due to energy-usage profiling, which smart grids could potentially enable [23]. Efthymiou and Kalogridis describe a method for securely anonymizing frequent electrical metering data sent by a smart meter by using a third party escrow mechanism [24]. Tan et al. study privacy in a smart metering system from an information theoretic perspective in the presence of energy harvesting and storage units [25]. They show that energy harvesting provides increased privacy by diversifying the energy source, while a storage device can be used to increase both energy efficiency and privacy. However, transaction data from energy trading may provide more fine-grained information than smart meter based usage patterns [11].

Existing energy trading markets, such as the European Energy Exchange [26] and project NOBEL in Spain, employ the double-auction market mechanism [27], which can be implemented to preserve participant privacy. However, typical exchange implementations involve centralized database architectures which constitute single points of failure.

Majumder et al. present an iterative double auction trading mechanism that preserves the participants' privacy, in particular, it keeps their utility functions private [28]. Similarly, Faqiry and Das present an auction mechanism for maximizing social welfare of buyers and sellers (if the supply is small) [29]. Their approach also provides some privacy meaning that participants do not reveal their utility functions. By constricting the buyers' utility functions to be convex, the social welfare objective function is maximized when the micro-grid controller objective function, whose goal is to maximize the power sold, is maximized. In the later part of the paper, they consider an approach that discards the privacy maintained during the first phase in order to make trading fair. In their work, there is no mechanism to check whether the buyer can produce the power they claim they can supply, which could result in instability. The authors also mention in passing that their approach can be implemented as a distributed algorithm, but this was not carried out.

In contrast, the work presented in this paper is a distributed systems mechanism that considers the problem of a broader definition of privacy, safety, and protection from malicious actors as a combined problem.

IV. TRANSAX Platform

Next, we describe the solution implemented by the TRANSAX platform. Figure 1 describes the components of this decentralized and distributed platform. An agent runs on a computing node within the premises of each home. In the remainder of this paper, we refer to these agents as "prosumers" or "consumers" depending upon the context, but they are implemented as one type of entity that can both buy and sell energy. The solvers are nodes responsible for identifying the feasible and optimal trades. The miners are

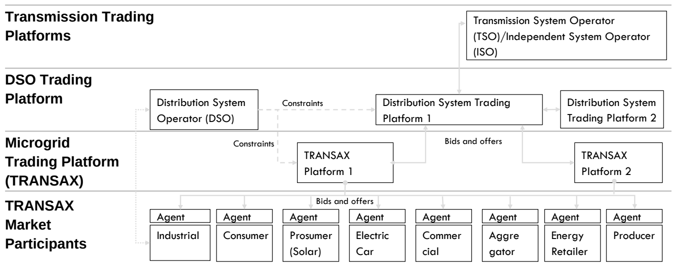

Fig. 1. Landscape of a Transactive Energy Systems with TRANSAX. TSO/ISO are responsible for bulk trading. Multiple instances of smaller scale trading platforms can co-exist at distribution and microgrid level.

responsible for reaching consensus on the market solutions using a distributed-ledger based "smart contract." Note that in this architecture, solvers and miners are not centralized. All agents communicate with each other providing offers and accepting the agreed upon market energy transfer schedule. The anonymization mixer implements privacy-preserving mechanisms ensuring that prosumers can remain anonymous to each other. The anonymization module has been explained in a previous paper and is not discussed further here [17].

First, we describe the market problem and then describe a smart contract solution and a protocol to setup the distributed system. Finally, we describe the distributed architecture of the implementation of the system.

A. Market Problem

Let $\mathcal{F}$ denote the set of feeders. For a feeder $f \in \mathcal{F}$, we let C_f^{ext} denote the maximum amount of power that is allowed to flow into or out of the feeder at any point in time. Similarly, we let C_f^{int} denote the maximum amount of power that is allowed to be consumed or produced within the feeder at any point in time.[5] We assume that time is divided into intervals of fixed length Δ, and we refer to the t-th interval simply as time interval t. For a list of symbols used in the paper, see Table I.

The input of the energy trading problem is the set of buying and selling offers posted by the participants.[6] For feeder $f \in \mathcal{F}$, we let $\mathcal{S}_f$ and $\mathcal{B}_f$ denote the set of selling and buying offers posted by participants located in that feeder, respectively.[7] A selling offer $s \in \mathcal{S}_f$ is a tuple (E_s, I_s, R_s), where

- E_s is the amount of energy to be sold,
- I_s is the set of time intervals in which the energy could be provided,
- R_s is the reservation price, i.e., lowest unit price for which the participant is willing to sell energy.

[5]In other words, limit C_f^{ext} is imposed on the net production and net consumption of all prosumers in feeder f, while limit C_f^{int} is imposed on the total production and total consumption.

[6]Participants may include both prosumers and the DSO. The DSO can shape load and trade energy futures by participating in the energy market the same way as prosumers do.

[7]If the DSO wants to participate in this energy trading market, it can be assigned to a "dummy" feeder in the problem formulation.

TABLE I
LIST OF SYMBOLS

Symbol	Description
	Microgrid
$\mathcal{F}$	set of feeders
C_f^{ext}	maximum net power consumption or net power production in feeder $f \in \mathcal{F}$
C_f^{int}	maximum total power consumption or total power production in feeder $f \in \mathcal{F}$
Δ	length of time intervals
T_{clear}	minimum number of time intervals between the finalization and delivery of a trade
	Offers
$\mathcal{S}_f$	set of selling offers from feeder $f \in \mathcal{F}$
$\mathcal{B}_f$	set of buying offers from feeder $f \in \mathcal{F}$
$\mathcal{S}$, $\mathcal{B}$	set of all selling and buying offers, resp.
$\mathcal{S}^{(t)}$, $\mathcal{B}^{(t)}$	set of selling and buying offers submitted by the end of time interval t, resp.
E_s, E_b	amount of energy to be sold or bought by offers $s \in \mathcal{S}$ and $b \in \mathcal{B}$, resp.
I_s, I_b	time intervals in which energy could be provided or consumed by offers $s \in \mathcal{S}$ and $b \in \mathcal{B}$, resp.
R_s, R_b	reservation prices of offers $s \in \mathcal{S}$ and $b \in \mathcal{B}$, resp.
$\mathcal{M}(s)$, $\mathcal{M}(b)$	set of offers that are matchable with offers s and b, resp.
$I(s,b)$	$I_s \cap I_b$
	Solution
$p_{s,b,t}$	amount of energy that should be provided by s to b in interval t
$\pi_{s,b,t}$	unit price for the energy provided by s to b in interval t
Feasible$(\mathcal{S}, \mathcal{B})$	set of feasible solutions given sets of selling and buying offers $\mathcal{S}$ and $\mathcal{B}$
$\hat{p}_{s,b,t}$, $\hat{\pi}_{s,b,t}$	finalized trade values
	Implementation Parameters
$T_{predict}$	prediction window used by prosumers when posting selling and buying offers
$T_{lookahead}$	number of time intervals considered in the future by the solver
$\hat{\Delta}$	length of the time step used for simulating the real-interval of length Δ

Similarly, a buying offer $b \in \mathcal{B}_f$ is a tuple (E_b, I_b, R_b), where the values pertain to consuming/buying energy instead of producing/selling, and R_b is the highest price that the participant is willing to pay. For convenience, we also let $\mathcal{S}$ and $\mathcal{B}$ denote the set of all buying and selling offers (i.e., we let $\mathcal{S} = \cup_{f \in \mathcal{F}} \mathcal{S}_f$ and $\mathcal{B} = \cup_{f \in \mathcal{F}} \mathcal{B}_f$).

We say that a pair of selling and buying offers $s \in \mathcal{S}$ and $b \in \mathcal{B}$ is *matchable* if

$$R_s \leq R_b \qquad \text{and} \qquad I_s \cap I_b \neq \emptyset. \tag{1}$$

In other words, a pair of offers is matchable if there exists a price that both participants would accept and a time interval in which the seller and buyer could provide and consume energy. For a given selling offer $s \in \mathcal{S}$, we let the set of buying offers that are matchable with s be denoted by $\mathcal{M}(s)$. Similarly, we let the set of selling offers that are matchable with a buying offer b be denoted by $\mathcal{M}(b)$. Further, we let $I(s,b) = I_s \cap I_b$.

A solution to the energy trading problem is a pair of vectors $(\boldsymbol{p}, \boldsymbol{\pi})$, where

- $p_{s,b,t}$ is a non-negative amount of power that should be provided by the seller $s \in \mathcal{S}$ and consumed by the buyer $b \in \mathcal{M}(s)$ in time interval $t \in I(s,b)$.[8]
- $\pi_{s,b,t}$ is the unit price for the energy provided by seller $s \in \mathcal{S}$ to buyer $b \in \mathcal{M}(s)$ in time interval $t \in I(s,b)$.

A pair of vectors $(\boldsymbol{p}, \boldsymbol{\pi})$ is a feasible solution to the energy trading problem if it satisfies the following constraints:

- The amount of energy sold or bought from each offer is at most the amount of energy offered:

$$\forall s \in \mathcal{S}: \sum_{b \in \mathcal{M}(s)} \sum_{t \in I(s,b)} p_{s,b,t} \cdot \Delta \leq E_s \tag{2}$$

$$\forall b \in \mathcal{B}: \sum_{s \in \mathcal{M}(b)} \sum_{t \in I(s,b)} p_{s,b,t} \cdot \Delta \leq E_b \tag{3}$$

- The amount of power flowing into or out of each feeder is below the safety limit in all time intervals:

$$\forall f \in \mathcal{F}, t: \left(\sum_{s \in \mathcal{S}_f} \sum_{b \in \mathcal{B}} p_{s,b,t} \right) - \left(\sum_{b \in \mathcal{B}_f} \sum_{s \in \mathcal{S}} p_{s,b,t} \right) \leq C_f^{ext} \tag{4}$$

$$\forall f \in \mathcal{F}, t: \left(\sum_{s \in \mathcal{S}_f} \sum_{b \in \mathcal{B}} p_{s,b,t} \right) - \left(\sum_{b \in \mathcal{B}_f} \sum_{s \in \mathcal{S}} p_{s,b,t} \right) \geq -C_f^{ext} \tag{5}$$

- The amount of energy consumed and produced within each feeder is below the safety limit in all time intervals:

$$\forall f \in \mathcal{F}, t: \sum_{b \in \mathcal{B}_f} \sum_{s \in \mathcal{S}} p_{s,b,t} \leq C_f^{int} \tag{6}$$

$$\forall f \in \mathcal{F}, t: \sum_{s \in \mathcal{S}_f} \sum_{b \in \mathcal{B}} p_{s,b,t} \leq C_f^{int} \tag{7}$$

- The unit prices are between the reservation prices of the seller and buyer:

$$\forall s \in \mathcal{S}, b \in \mathcal{M}(s), t \in I(s,b): \; R_s \leq \pi_{s,b,t} \leq R_b \tag{8}$$

The objective of the energy trading problem is to maximize the amount of energy traded. Formally, an optimal solution to the energy trading problem is

$$\max_{(\boldsymbol{p},\boldsymbol{\pi}) \in \mathit{Feasible}(\mathcal{S},\mathcal{B})} \sum_{s \in \mathcal{S}} \sum_{b \in \mathcal{M}(s)} \sum_{t \in I(s,b)} p_{s,b,t}\,, \tag{9}$$

where *Feasible*$(\mathcal{S}, \mathcal{B})$ is the set of feasible solutions given selling and buying offers $\mathcal{S}$ and $\mathcal{B}$ (i.e., set of solutions satisfying Equations (2) to (8) with $\mathcal{S}$ and $\mathcal{B}$).

[8]We require the both the seller and buyer to produce a constant level of power during the time interval.

1) Linear-Programming Solution:: We can solve the basic energy trading problem efficiently by formulating it as a linear program. First, create real-valued variables $p_{s,b,t}$ and $\pi_{s,b,t}$ for each $s \in S, b \in \mathcal{M}(s), t \in I(s,b)$. Then, the following reformulation of the matching problem is a linear program:

$$\max_{\boldsymbol{p},\boldsymbol{\pi}} \sum_{s \in \mathcal{S}} \sum_{b \in \mathcal{M}(s)} \sum_{t \in I(s,b)} p_{s,b,t} \tag{10}$$

subject to Equations (2) to (8) and

$$\boldsymbol{p} \geq \mathbf{0} \text{ and } \boldsymbol{\pi} \geq \mathbf{0}. \tag{11}$$

2) Trade Finalization: Equation (9) formulates the problem considering a single "snapshot" of all offers across all time intervals. However, in practice, prosumers may submit new offers at any time, resulting in continuously evolving sets of offers. Consequently, optimal solutions to Equation (9) may have to be found repeatedly as new offers are submitted, resulting in a series of evolving solutions. This presents a problem since prosumers need to know in advance what the "final" solution for a certain time interval is in order to be able to actually schedule energy production or consumption for that interval. Further, preventing "last-minute" changes can be crucial for safety and stability since it allows the DSO to prepare for satisfying energy demand that cannot be met locally.

As the set of submitted offers grows, the optimal solution to the energy trading problem may change, and the optimal value of each $p_{s,b,t}$ may vary. While each change can increase the amount of energy traded, the *trade values* $p_{s,b,t}$ and $\pi_{s,b,t}$ need to be *finalized* at some point in time. At the very latest, values for interval t need to be finalized by the end of interval $t-1$; otherwise, participants would have no chance of actually delivering the trade. We now extend the energy trading problem to consider a growing set of offers and a time constraint for finalizing trades. Our approach finalizes a minimum set of trades in each interval, which maximizes efficiency while providing safety.

We assume that all trades for time interval t (i.e., all values $p_{s,b,t}$ and $\pi_{s,b,t}$) must be finalized by the end of time interval $t - T_{clear}$, where T_{clear} is a positive integer constant, which is set by the operator. In practice, the value of T_{clear} must be chosen taking into account both physical constraints (e.g., how long it takes to turn on a generator) and communication delay (e.g., some participants might learn of a trade with delay due to network disruptions).

We let $\hat{p}_{s,b,t}$ and $\hat{\pi}_{s,b,t}$ denote the finalized trade values, and we let $\mathcal{B}^{(t)}$ and $\mathcal{S}^{(t)}$ denote the set of buying and selling offers that participants have submitted by the end of time interval t. Then, the system takes the following steps at the end of each time interval t:

- Find an optimal solution $(\boldsymbol{p}^*, \boldsymbol{\pi}^*)$ to the extended energy trading problem:

$$\max_{(\boldsymbol{p},\boldsymbol{\pi}) \in \mathit{Feasible}(\mathcal{S}^{(t)},\, \mathcal{B}^{(t)})} \sum_{s \in \mathcal{S}^{(t)}} \sum_{b \in \mathcal{M}(s)} \sum_{\tau \in I(s,b)} p_{s,b,\tau} \tag{12}$$

subject to

$$\forall \tau < t + T_{clear}: \; p_{s,b,\tau} = \hat{p}_{s,b,\tau} \quad (13)$$

$$\pi_{s,b,\tau} = \hat{\pi}_{s,b,\tau} \quad (14)$$

- Finalize trade values for time interval $t + T_{clear}$ based on the optimal solution $(\boldsymbol{p}^*, \boldsymbol{\pi}^*)$:

$$\hat{p}_{s,b,t+T_{clear}} := p^*_{s,b,t+T_{clear}} \quad (15)$$

$$\hat{\pi}_{s,b,t+T_{clear}} := \pi^*_{s,b,t+T_{clear}} \quad (16)$$

The problem in Equation (12) can also be reformulated as a linear program similarly, by considering $\mathcal{S}^{(t)}$, $\mathcal{B}^{(t)}$, $\hat{\boldsymbol{p}}$, $\hat{\boldsymbol{\pi}}$, and the additional constraints.

B. Market Solver

The role of the market solver is to periodically solve the linear program mentioned above as the offers stream in. To address the trustworthiness challenge, we implement a blockchain based solution as discussed previously. However, since computation is relatively expensive on blockchain-based distributed platforms, solving the energy trading problem using a blockchain-based smart contract would not be scalable in practice. In light of this, we adopt a *hybrid implementation approach*, which we introduced in earlier [30], to transactive energy systems. The hybrid approach combines the trustworthiness of blockchain-based smart contracts with the efficiency of more traditional computational platforms. The key idea of our hybrid approach is to (1) use a high-performance computer to solve the computationally expensive linear program *off-blockchain*[9] and then (2) use a smart contract to record the solution *on the blockchain*.

1) Blockchain-based Smart Contract: We implemented a smart contract[10] that verifies the feasibility of each solution $(\boldsymbol{p}, \boldsymbol{\pi})$ submitted by an off-blockchain solver. If the solution is feasible, then the contract computes the value of the objective function and compares it to the objective value of previously submitted solutions. The contract always keeps track of the best feasible solution submitted so far, which we call the *candidate solution*. At the end of each time interval t, the contract finalizes the trade values for interval $t + T_{clear}$ based on the candidate solution.[11]

This simple functionality achieves a high level of security and reliability. Firstly, it is clear that an adversary cannot force the contract to finalize trades based on an unsafe (i.e., infeasible) solution since such a solution would be rejected. Similarly, an adversary cannot force the contract to choose an inferior solution instead of a superior one. In sum, the only action available to the adversary is proposing a superior feasible solution, which would actually improve energy trading in the microgrid.

The contract is also reliable and can tolerate temporary disruptions in the solver or the communication network. Notice that any solution $(\boldsymbol{p}, \boldsymbol{\pi})$ that is feasible for sets $\mathcal{S}$ and $\mathcal{B}$ is also feasible for supersets $\mathcal{S}' \supseteq \mathcal{S}$ and $\mathcal{B}' \supseteq \mathcal{B}$. As the sets of offers can only grow over time, the contract can use a candidate solution submitted during time interval t to finalize trades in any subsequent time interval $\tau > t$. In fact, without receiving new solutions, the difference between the amount of finalized trades and the optimum will increase only gradually: since the earlier candidate solution can specify trades for any future time interval, the difference is only due to the offers that have been posted since the solution was found and submitted.

2) Solver: We complement the smart contract with an efficient linear programming solver, which can be run off-blockchain, on any capable computer (or multiple computers for reliability). The solver is run periodically to find a solution to the energy trading problem based on the latest set of offers posted. Once a solution is found by the solver, it is submitted to the smart contract in a blockchain transaction. Note that if new offers have been posted since the solver started working on the solution, the contract will still consider the solution to be feasible. This is again due to any feasible solution for sets $\mathcal{S}$ and $\mathcal{B}$ also being feasible for supersets $\mathcal{S}' \supseteq \mathcal{S}$ and $\mathcal{B}' \supseteq \mathcal{B}$.

From the perspective of the solver, being able to submit multiple solutions to the contract for the same problem has many advantages. For example, it allows the linear programming solver to be run as an anytime algorithm. Further, we can allow multiple—potentially untrusted—entities to try to solve the problem and submit solutions, since the smart contract will always choose the best feasible one. This is especially important in microgrids where a trusted third party is not guaranteed to always be present. In such settings, prosumers can be allowed to volunteer and provide solutions to the energy trading problem.[12] Thereby, we enable finding solutions in an efficient and very flexible manner, while reaping the benefits of smart contracts, such as auditability and trustworthiness.

3) Solver Lookahead Window: Since the energy trading problem (i.e., Equation (12)) can be formulated as a linear program, we can solve it efficiently, that is, in polynomial time. However, as the number of offers and the time intervals that they span increases, the number of variables $\{p_{s,b,t}\}$ may grow prohibitively high, which makes solving the trading problem very challenging in practice. A key observation that helps us tackle this challenge is that even though consumers and prosumers may post offers whose latest intervals are in the far future (i.e., for an offer s, the latest interval may be $\max I_s \gg t$, where t is the current interval), a solver only needs to consider a few intervals ahead of the finalization deadline. Indeed, we have observed that considering intervals in the far future has little effect on the optimal solution for the interval that is to be finalized next.

[9] We use CPLEX [31] as the MILP solver engine in TRANSAX.

[10] Source code is available upon request.

[11] If no solution has been submitted to the contract so far, which might be the case right after the trading system has been launched, $\boldsymbol{p} = \boldsymbol{0}$ may be used as a candidate solution.

[12] Of course, each prosumer will try to submit a solution that favors the prosumer. However, the submitted solution still needs to be superior with respect to the optimization objective, which roughly corresponds to social utility. Hence, each prosumer is incentivized to improve social utility by submitting a superior solutions that favors the prosumer. We leave the analysis of these incentives for future work.

Consequently, for practical solvers, we introduce a lookahead window $T_{lookahead} \geq T_{clear}$ that limits the intervals that need to be considered effectively: for any $\hat{t} > t + T_{lookahead}$, we set $p_{s,b,\hat{t}} = 0$, where t is the current interval. By "pruning" the set of fee variables, we can significantly improve the performance of the solver with negligible effect on solution quality. Figure 3 shows the memory usage of the solver (in time interval $t = 80$) and the energy traded, while varying the lookahead-window length $T_{lookahead}$.

Similar to memory, the lookahead parameter also impacts the CPU utilization of the solver. Thus, as a practical matter, we implemented a hierarchical controller to automatically adjust the lookahead window in TRANSAX solvers using resource limit callbacks, which we will describe in Section IV-D. The top-level controller sets the maximum lookahead value based on the available memory. The low-level controller sets the lookahead to a value between T_{clear} and the upper bound. The asynchronous architecture of TRANSAX enables multiple solvers to operate simultaneously and compete in providing a better matching solution, while obeying the limits imposed by available resources. This ensures that the solvers can be run on edge computing nodes in a community where other applications might also be co-hosted.

4) Other Parameters: In addition to the lookahead window $T_{lookahead}$, our implementation can also be configured with parameters that control the prosumers and the speed of the simulation. The solver operates as a periodic process, waiting on information from the smart contract about all the offers that have been posted. In our implementation, the prosumers also operate periodically, submitting their offers and bids to the smart contract in every interval. In a given interval, our prosumer implementation provides offers for up to $T_{predict}$ intervals in the future (including the current interval), where $T_{predict}$ is a parameter of the prosumer. We require that $T_{predict} > 1$ because we need at least one interval prediction for trading energy futures. Finally, during our experiments, we may speed up the simulation by letting the real-time length of the time interval be $\hat{\Delta} < \Delta$, but keeping the theoretical length of the interval at Δ. Note that $\hat{\Delta}$ is the amount of real time passed in the simulation before proceeding to the next interval. This allows us to speed up the experiments without compromising our results since running the system slower would be easier.

C. TRANSAX Protocol

As illustrated in Figure 2, when a participant (i.e., prosumer, consumer, or solver) receives the address of the smart contract, they submit a transaction to register themselves in the blockchain. After some time, the transaction is mined and triggers an event (e.g., `ProsumerRegistered`) notifying the participant that it can begin posting offers. When the participants contact the DSO, the response contains time information allowing the participants to determine the earliest interval for which the blockchain is accepting bids and solutions. Any bid or solutions that contains an end time after that interval is ignored. The interval returned by the DSO

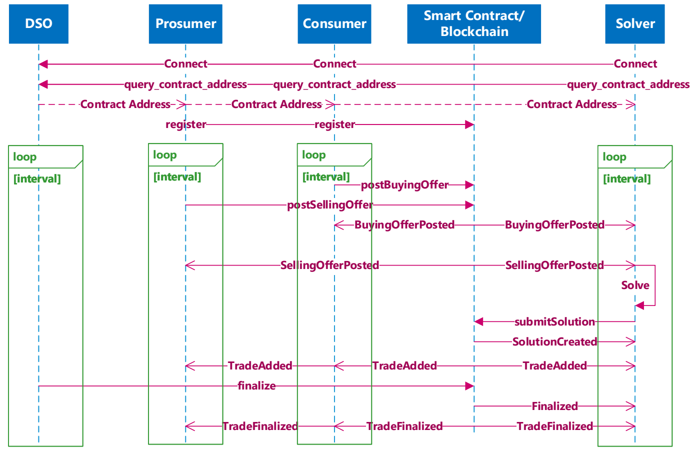

Fig. 2. Interaction diagram of TRANSAX components.

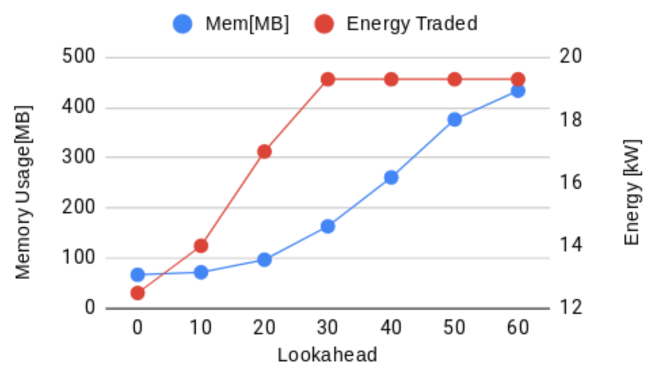

Fig. 3. Memory consumption and Energy traded during a single interval of the simulation (interval 80) for various values of $T_{lookahead}$.

is some number of intervals ahead of the current interval of the microgrid, since the power schedule must be determined before the time of actuation. The length of an interval in the case studies described here is 1 minute. For live deployments, the value can be configured by the system integrator. At the start of each interval, prosumers submit relevant bids to the blockchain. After trades have been added to the blockchain, the solver receives the `OfferPosted` event and will attempt to find a valid matching between bids and requests. The solver has a solving interval and attempts to find a better solution during each one. This continues until the DSO submits a `finalize` transaction to the blockchain, which triggers a `TradeFinalized` event. This event causes the solver to update its interval and begin working to find a solution for the next interval. The prosumers also receive this event informing them of the power they are expected to produce/consume during that interval when it arrives. Two concepts explained below are critical to this protocol

1) Offers: The middleware platform on which TRANSAX is build (described in next section) solves the problem of time synchronization, enabling all agents to correctly know the current interval and current time. Thus, the solvers and prosumers do not need to synchronize independently and can keep track of offers and intervals. The distributed ledger acts as a shared database and provides a notification log of events (e.g., submission of a new offer). In TRANSAX, these offers are sent as a structure of following form {`PROSUMER ID, START INTERVAL, END INTERVAL, ENERGY QUANTITY`}. Agents and solvers listen for these events and perform actions based on them.

2) Trade Finalization: Finalization of an interval means that the smart contract will not accept any more changes to the solution for that interval. Prosumers are notified of finalized trades using `TradeFinalized` events, which communicate matches as structures of the following form {`BUY OFFER ID, SELL OFFER ID, INTERVAL, POWER`}. This enables prosumers to act according to the solution of the energy trading problem since they know the identifiers of their offers and can filter on finalized trades. Note that even though we do not discuss penalizing prosumers who do not conform to the solution, it is straightforward to do this since the DSO can associate prosumers with their anonymous identifiers, and all offers and trades are permanently recorded on the ledger. Finally, the DSO can combine the recorded trades with actual power consumption and production values measured by electricity meters in order to bill prosumers (e.g., every month).

D. TRANSAX Implementation and Resilience

TRANSAX is implemented as an application in the Resilient Information Architecture Platform for Smart Grid (RIAPS) [13]. A RIAPS system is a collection of computing nodes, which are connected to power system sensors and actuators over a variety of interfaces, e.g. Modbus/UART, etc. Each RIAPS computing node executes a collection of platform services which run with the highest privileges on the system. These services are for discovering the other components in the distributed system (`riaps_disco`), for remotely deploying application (`riaps_deplo`) and for providing the computing nodes with a synchronized time [32], which is critical for the correct operation of TRANSAX. Control nodes are responsible for installing and removing distributed applications on these nodes. Each RIAPS application is a collection of *actors*, which are assembled from reusable *components*.

1) Fault Model: We extended RIAPS with additional capabilities to monitor and mitigate failures across three layers: physical and device level, platform services level, and application level. Typically, physical electrical-system architectures are designed with $N-1$ criterion, i.e., they have redundancy to tolerate the failure of any single physical device. Fault management at services level is implemented by the combination of a distributed hash table, which maintains information about all actors, and Zero MQ Zyre [33], an open-source framework for proximity-based peer-to-peer applications. Further, we extended RIAPS to provide resource management. These features are important to ensure that TRANSAX actors can work on remote nodes within the limits of available resources. These limits are enforced via the use of the cgroups interface, watchdogs, and custom zeroMQ pair connections in RIAPS. To support application-specific failure mitigation, RIAPS provides callbacks (see Table II), which can be implemented by the component developers, in this case the TRANSAX components. For example, we used the *handleCPULimit* to implement the controller for the lookahead window described in Section IV-B3.

TABLE II
CALLBACK HANDLERS IMPLEMENTED IN RIAPS FOR PROVIDING APPLICATION SPECIFIC MITIGATION ACTIONS. THESE HANDLERS RESPOND TO FAILURES AT THREE LEVELS: APPLICATION CODE, RIAPS SERVICES AND PHYSICAL DEVICES.

Source	Handler	Event
App	*handleCPULimit*	App CPU usage exceeds threshold
App	*handleMemLimit*	App Memory usage exceeds threshold
App	*handleSpcLimit*	App Disk usage exceeds threshold
App	*handleNetLimit*	App Network usage exceeds threshold
Phy	*handleNICStateChange*	RIAPS node network status has changed
App, service, Phy	*handlePeerStateChange*	Clean ingress/egress to network of peer
App, service	*handleDeadline*	App function duration exceeds threshold

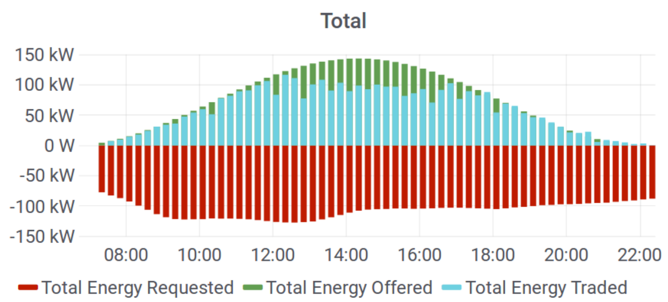

Fig. 4. Total energy production capacity (green), and energy demand (red) for each interval, as well as the total energy traded in each interval (blue) while subject to constraints. C_{ext} = 2MW, C_{int} = 2.5MW.

2) Mining Considerations: In the current implementation of TRANSAX, we use a private Ethereum network as the distributed ledger. To speed up the consensus protocol, we reduce the the difficulty of the cryptographic puzzle solved for proof-of-work consensus. For larger systems, the proof-of-work consensus may be replaced by, e.g., proof-of-stake, for scalability.

V. CASE STUDY

We consider a collection of load traces recorded from a microgrid in Germany, containing 102 homes (5 producers, 97 consumers) across 11 feeders. We show the nominal execution of the system as well as its resilience capabilities by illustrating execution under resource constraints and actor failures.

1) Nominal Evaluation: At this scale (102 prosumers), the current implementation was able to match offers during each simulation interval with $T_{lookahead} = 5$. The system-wide trading results can be seen in Figure 4. Each bar is a 15 minute interval. A green bar is the sum of all energy selling offers during that interval. A red bar is the sum of all energy buying offers during that interval. The blue bars are overlayed on the green bars, showing the total energy traded during that interval.

Early in the simulation, the buying offers exceed the selling offers. Then, as solar generation increases, the selling offers exceed the buying offers. The excess may be stored in batteries for use in future intervals, which increases the complexity of

TABLE III
SOLVING TIMES AS PROBLEM COMPLEXITY INCREASES

Real Time	Simulation Time	Variables	Constraints	Solving Time
10:52am	10:00am	2910	750	0.891s
11:00am	12:00am	6984	762	1.612s
11:04am	1:00pm	17751	778	4.35s

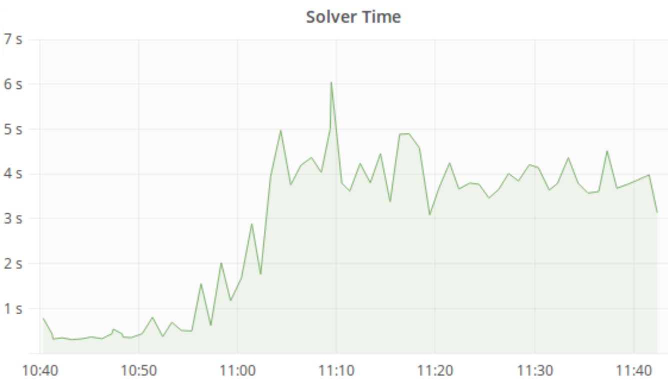

Fig. 5. Time taken by the solver to finish each time it runs. The solver runs every 5 seconds.

TABLE IV
MONETARY IMPACT OF TRADES SUBJECT TO CONSTRAINTS C_{ext} AND C_{int}, GIVEN \$0.12/KWH, AND TOTAL PRODUCTION AND DEMAND FOR ENERGY OF ~4.5MW AND ~8.3MW, RESPECTIVELY.

C_{ext}, C_{int}	Traded	Unused Energy	Unmet Demand
20kW, 25kW	2.288MW	50% (\$270.73)	73% (\$722.90)
2MW, 2.5MW	3.668MW	19% (\$105.18)	56% (\$557.35)

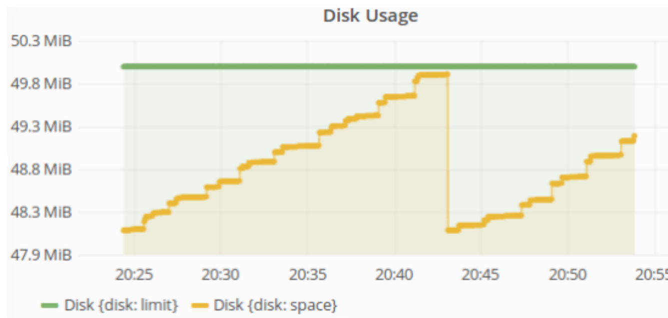

Fig. 6. The application disk usage grows from 48MB to the disk limit of 50MB, at which point the handler fires and allows the application to take corrective actions.

the MILP problem since offers can be matched across multiple intervals. Figure 5 shows evidence of this fact as we see an increase in solver time when selling offers exceed buying offers, around 11:00am. The increasing solver time is the result of increasing problem complexity, which is correlated with the number of variables and constraints in a problem. Some intervals and the corresponding numbers of variables and solve times are shown in Table III. Again, we see that as the selling offers exceed the buying offers, complexity increases, which results in increased solve time. These results provide insight into how the solver scales.

The scalability of TRANSAX is limited by the number of transactions that the distributed ledger supports, as well as the complexity of the MILP problem determined by the number of constraints and variables. Additionally, TRANSAX is able to scale by reducing the number offers in a particular interval by suggesting that solvers reduce their lookahead window.

The number of trades that are made in a day depends on the system parameters. In two experiments, we modulated the power flow constraints C_{ext} and C_{int}. The result of this can be seen in Table IV. In both cases, the total buying and selling offers remained constant, only the amount of power that was permitted to flow was changed. Changing the constraints increased the power traded by 1.4MW, thus reducing unused energy by 31%. In light of this, the efficiency of the platform is primarily dependent on the offers that prosumers make and the system constraints.

2) Resource Limit Evaluation: In this section, we show resource-limit monitoring and mitigation for disk usage and CPU usage. In Figure 6, we set the disk storage limit for the prosumer to 50 MB. When the limit is reached, $handleSpcLimit$ is triggered (see Table II), which forces the prosumer to rotate the logs.

To show the effect of the CPU resource constraint, we refer back to Section IV-B3. The actions of the top-level controller can be seen in Figure 7 as the yellow dots. When the solver consumes more than 30% [13] of the CPU, the top-level controller reduces the maximum value that the low-level controller may set. We see that over time, the maximum value decreases and the lookahead value (green dots) stays below the maximum. The low-level controller sets the value of the lookahead window, and its influence is shown by the green dots. The low-level controller is implemented as a proportional controller which monitors the solve time and has a solve-time set point of 0.5 seconds. This value was chosen for testing purposes only. The memory controller (not pictured) uses the same high-level control (when the threshold is crossed, it reduces the upper bound) and the same low level control.

Figure 3 demonstrates how TRANSAX can adapt to variation in the problem complexity. It shows the trade-off between resource consumption (memory) and trading efficiency during interval 80 (8:00pm) as the lookahead window varies. Trading efficiency stops increasing with a lookahead of 30, since interval 50 (12:30pm) is when selling offers become larger than buying offers, and thus the interval in which prediction becomes beneficial.

3) Failure Evaluation: Since TRANSAX is decentralized, there may be any number of solvers communicating asynchronously with the smart contract, being notified of trades and posting potential solutions. Thus, if any given solver fails, the system will continue unimpeded as long as other solvers are operational. The event $handlePeerStateChange$ has been implemented in the platform, and it is triggered when a node in the network fails or if it is disconnected for any reason. All the peers in the network may receive this message and take actions as appropriate. For example, in the transactive energy

[13] These instances can be seen in the *% CPU Utilization* plot of Figure 7.

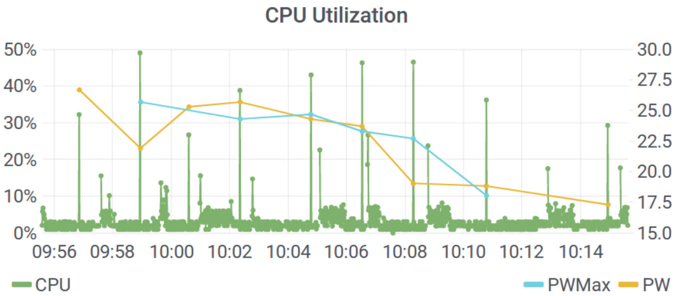

Fig. 7. $handleCPULimit$ reduces the maximum lookahead when the CPU limit is crossed.

case, if a node goes down, it may be unable to provide the energy it was supposed to, and so its trades should be removed. During the testing of the fault-tolerance features, node failure was detected on average in 0.14 seconds. The peers (other actors in the TRANSAX application) were notified that the node has recovered 1.88 seconds after the failure, and the node was able to fully reactivate in 6.52 seconds on average.

VI. Conclusions and Discussions

We described a decentralized platform for implementing energy exchange mechanisms in a microgrid setting. Our solution enables prosumers to trade energy without threatening their privacy or the safety of the system. Our hybrid solver approach, which combines a smart-contract based validator with an external optimizer, enables the platform to clear offers securely and efficiently.

In addition to the assurances provided by the distributed ledger, the resilience features provide necessary robustness for TRANSAX. For example, $handleDeadline$ can be used to adjust $T_{lookahead}$ in order to adapt to varying complexity when there are strict timing requirements, and $handlePeerStateChange$ can be used to monitor the health of neighboring prosumers, and if they become disconnected, their trades can be removed from the smart contract.

References

[1] US EIA, "Annual energy outlook," U.S. Energy Information Administration: Washington, D.C, Tech. Rep. DOE/EIA-0383, 2014.

[2] E. Santacana, G. Rackliffe, L. Tang, and X. Feng, "Getting smart," *IEEE Power and Energy Magazine*, vol. 8, no. 2, pp. 41–48, March 2010.

[3] "US solar market insight," 2017, http://www.seia.org/research-resources/us-solar-market-insight.

[4] T. Randall, "The way humans get electricity is about to change forever," Bloomberg, June 2015.

[5] A. Stock, P. Stock, and V. Sahajwalla, "Powerful potential: Battery storage for renewable energy and electric cars," *Climate Council of Australia Limited, Australia*, 2015.

[6] P. Bronski, J. Creyts, M. Crowdis, S. Doig, J. Glassmire *et al.*, "The economics of load defection: How grid-connected solar-plus-battery systems will compete with traditional electric service – why it matters, and possible paths forward," *Rocky Mountain Institute (RMI), Homer Energy*, 2015.

[7] K. Kok and S. Widergren, "A society of devices: Integrating intelligent distributed resources with transactive energy," *IEEE Power and Energy Magazine*, vol. 14, no. 3, pp. 34–45, 2016.

[8] F. A. Rahimi and A. Ipakchi, "Transactive energy techniques: closing the gap between wholesale and retail markets," *The Electricity Journal*, vol. 25, no. 8, pp. 29–35, 2012.

[9] W. Cox and T. Considine, "Structured energy: Microgrids and autonomous transactive operation," in *4th IEEE Conference on Innovative Smart Grid Technologies (ISGT)*, February 2013, pp. 1–6.

[10] R. B. Melton, "Gridwise transactive energy framework," Pacific Northwest National Laboratory, Richland, WA, Tech. Rep., 2013.

[11] A. Hussain, V. H. Bui, and H. M. Kim, "A resilient and privacy-preserving energy management strategy for networked microgrids," *IEEE Transactions on Smart Grid*, vol. 9, 2017.

[12] S. Eisele, I. Madari, A. Dubey, and G. Karsai, "RIAPS: Resilient information architecture platform for decentralized smart systems," in *20th IEEE International Symposium on Real-Time Computing (ISORC)*. IEEE, May 2017.

[13] S. Eisele, A. Dubey, G. Karsai, and S. Lukic, "Transactive energy demo with RIAPS platform," in *8th International Conference on Cyber Physical Systems (ICCPS)*, April 2017, pp. 91–91.

[14] L. Orsini, Y. Wei, and J. Lubin, "Use of blockchain based distributed consensus control," Apr. 13 2017, US Patent App. 15/292,783. [Online]. Available: https://www.google.com/patents/US20170103468

[15] "Power ledger whitepaper," July 2017. [Online]. Available: https://powerledger.io/whitepaper/

[16] "Special Report: Block-chain world," October 2017. [Online]. Available: https://spectrum.ieee.org/static/special-report-blockchain-world

[17] A. Laszka, A. Dubey, M. Walker, and D. Schmidt, "Providing privacy, safety and security in IoT-based transactive energy systems using distributed ledgers," in *7th International Conference on the Internet of Things (IoT)*, October 2017.

[18] O. Dag and B. Mirafzal, "On stability of islanded low-inertia microgrids," in *2016 Clemson University Power Systems Conference (PSC)*, March 2016, pp. 1–7.

[19] S. Nakamoto. (2008) Bitcoin: a peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[20] V. Buterin *et al.*, "Ethereum white paper," 2013.

[21] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, 2016.

[22] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *37th IEEE Symposium on Security and Privacy (S&P)*. IEEE, May 2016, pp. 839–858.

[23] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security & Privacy*, vol. 7, no. 3, 2009.

[24] C. Efthymiou and G. Kalogridis, "Smart grid privacy via anonymization of smart metering data," in *1st IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2010, pp. 238–243.

[25] O. Tan, D. Gunduz, and H. V. Poor, "Increasing smart meter privacy through energy harvesting and storage devices," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1331–1341, 2013.

[26] European Power Exchange, "EPEX SPOT operational rules," 2017. [Online]. Available: http://www.epexspot.com/en/extras/download-center/technical_documentation

[27] D. Ilic, P. G. D. Silva, S. Karnouskos, and M. Griesemer, "An energy market for trading electricity in smart grid neighbourhoods," in *6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, June 2012, pp. 1–6.

[28] B. P. Majumder, M. N. Faqiry, S. Das, and A. Pahwa, "An efficient iterative double auction for energy trading in microgrids," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*. IEEE, December, pp. 1–7.

[29] M. N. Faqiry and S. Das, "Transactive energy auction with hidden user information in microgrid." [Online]. Available: https://arxiv.org/abs/1608.03649

[30] S. Eisele, A. Laszka, A. Mavridou, and A. Dubey, "SolidWorx: A resilient and trustworthy transactive platform for smart and connected communities," in *2018 IEEE International Conference on Blockchain (Blockchain-2018)*, July 2018.

[31] IBM ILOG CPLEX, "V12. 1: User's manual for CPLEX," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[32] P. Volgyesi, A. Dubey, T. Krentz, I. Madari, M. Metelko, and G. Karsai, "Time synchronization services for low-cost fog computing applications," in *28th International Symposium on Rapid System Prototyping (RSP)*. New York, NY, USA: ACM, 2017, pp. 57–63.

[33] P. Hintjens, *ZeroMQ: Messaging for many applications*. O'Reilly Media, Inc., 2013.