Design and Implementation of Safe and Private Forward-Trading Platform for IoT-Based Transactive Microgrids

Aron Laszka  Abhishek Dubey  Scott Eisele  Michael Walker  Karla Kvatnick
University of Houston  Vanderbilt University  Vanderbilt University  Vanderbilt University  Siemens Corporate Technology

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. We address the problem of implementing transactive energy mechanisms in a distributed setting, providing both privacy and safety. Specifically, we design and implement an automated auction and matching system that ensures safety (i.e., satisfaction of line capacity constraints), preserves privacy, and promotes local trade and market efficiency for IoT-based transactive energy systems. This design problem is challenging because safety, market efficiency, and privacy are competing objectives. We implement our solution as a decentralized IoT-based trading platform, which is built on blockchain technology and smart contracts. To demonstrate the feasibility of our platform, we perform experiments with dozens of embedded devices and energy production and consumption profiles from a real dataset.

Index Terms—Transactive energy platform, Internet of Things, blockchain, privacy, security, safety, smart contract.

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, the battery technology costs per kWh have been dropping significantly [5], reaching grid parity [6]. These trends are enabling a different vision for the future of power-grid operations: a decentralized system in which local communities are arranged in microgrids [7]. In this vision, energy generation, transmission, distribution, and storage (i.e., electric vehicles or wall-mounted residential batteries) can be strategically used to balance load and demand spikes. A key feature of this vision is the support for local peer-to-peer energy trading within microgrids to reduce the load on the distribution system operators (DSO), leading to the development of Transactive Energy Systems (TES) [8], [9], [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.

This new vision of decentralized peer-to-peer energy market is synergistic with the recent advances and push towards Internet of Things, as shown by Volttron [11], OpenFMB [12], and the Resilient Information Architecture Platform for Smart Grid (RIAPS) [13], [14]. The latter is a platform developed by our extended research team, and it provides foundations for all algorithms, isolates the hardware details from the algorithms, and provides essential mechanisms for resource management, fault tolerance, and security.

With the assumption of the availability of these new IoT platforms in the microgrid, we can develop appropriate transaction management platforms (TMPs) that allow prosumers (consumers who may also produce electricity) within a microgrid to participate freely in a local peer-to-peer energy trading market. With the advent and anticipated ubiquity of residential energy storage in the form of EVs or wall-mounted battery units [15], an important challenge is to design TMPs that allow for the efficient trade of energy futures. However, the implementation of such TMPs remains difficult because of three integrated problems that must be addressed.

The first problem relates to ensuring the physical stability and safety of the grid apparatus, and is mainly concerned with dynamically balancing supply and demand without violating line capacity constraints. The second 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. The third problem 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 [16], and 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). Note that energy futures, whose delivery may happen several hours after the transaction is made, can play an important role in predicting and controlling microgrid load. In comparison, smart metering reveals only current (or past) usage.
In recent work [17], we have introduced Privacy preserving Energy Transactions (PETra), which is our distributed-ledger based solution that (1) enables trading energy futures in a secure and verifiable manner and (2) preserves prosumer privacy. Privacy in our previous work was implemented by using anonymized identifiers, a public distributed ledger, and a mixing service that prevents tracing the assets being traded back to the owner. However, the transactive mechanism implemented in [17] was opportunistic, where each consumer looked at the available asks from producers and chose the one that fit the needs of the consumer the best. In the present work, we consider an automated matching system that maximizes the amount of energy traded within the local market. Further, in the prior work, we did not consider system-wide safety constraints, only constraints on individual prosumers.

Building on our work in [17], we continue to use a blockchain to implement parts of our proposed TMP design. The use of blockchains (i.e. distributed ledgers) for implementations of TMPs is in line with the recent trends in the research community and industry focused on transactive energy markets [18], [19]. Although disintermediation of trust is widely regarded as the primary feature of blockchain-based transaction systems [20], 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 TESs since these are envisioned to involve the participation of self-interested parties, interacting with one another via a distributed computing platform that executes the transaction management.

**Contribution:** Our contributions in this paper are as follows:

- We co-design an automated matching mechanism and a decentralized IoT-based transaction management platform, whose goal is to support the energy trading workflow while ensuring privacy of the prosumers (i.e., their identity) and the safety of the system (i.e. satisfaction of line capacity constraints). This design problem is challenging because there is a direct conflict between safety, privacy, and market efficiency.

- We allow prosumers to consider the effect of energy storage in batteries by enabling them to specify multiple time intervals in which they could trade energy, which is necessary for taking full advantage of batteries.

- We describe the architecture and the protocol specification of our platform. Our solution combines the security and immutability of blockchain-based smart contracts with the efficiency of traditional computational platforms.

- Finally, we present an experimental evaluation of the proposed TMP and the resulting market performance, with and without the availability of prosumer-owned battery storage. We consider total energy trade throughput as the market performance metric.

**Outline:** We present our transactive microgrid model and a review of requirements in Section II. Then, we give an overview of the state of the art in Section III. We formalize the energy trading problem in Section IV. Our solution approach is described in Section V. The implementation architecture is summarized in Section VI. We present results and discussions in Section VII. Finally, we conclude the paper in Section VIII.

**II. System Model**

We consider a microgrid with a set of feeders arranged in a radial topology. A feeder 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, as shown in Figure 1. 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 [21]. 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 (it can be made either by each prosumer individually or by an automated matching mechanism). Once the matching is done, the energy transaction and the financial transaction are then handled at a later time.

A. System Requirements

We now describe requirements that must be considered for building a decentralized Transaction Management Platform (TMP) that supports the workflow across the microgrid described above.

1) Communication Fabric: The first requirement is the existence of an appropriate communication and messaging architecture. The TMP must collect participants’ offers and make them available to buyers and sellers, and the market algorithm must communicate clearing prices and buyer-seller matchings. In order to meet the operational and safety requirements described next, these messages must be reliably delivered under strict timing constraints, derived from the deadline by which a trade must clear. Moreover, the TMP must be capable of handling high volumes of micro-transactions anticipated in peer-to-peer trading scenarios. Finally, the communication fabric must support confidentiality, integrity, and non-repudiation of transactional data.

2) Operational Safety and Stability: 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. Therefore, it is important to ensure that local energy trade settlements involve power production and consumption which result in instantaneous power flows that never violate this safety constraint.

3) Market Security and Efficiency: The TMP shall include provisions for ensuring the protection of prosumer interests, as well as those of the DSO. Prosumer interests include being billed correctly based on energy prices set by the market and the measurements made by the smart meters. In the context of grid-connected microgrids, the system should match supply and demand as closely as possible, while respecting safety constraints. Therefore, the TMP should aim to maximize the amount of energy traded.

4) Privacy: 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, and the matching thereof, should remain anonymous to the other participants. A participant’s energy usage patterns and personal information, such as financial standing, shall not be inferable from the participant’s trading activity. For example, inference of energy usage patterns can be exploited by inferring the presence or absence of a person in their home.

III. Analysis of State of the Art

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 [22] and Ethereum [23], suffer from design limitations that prevent their direct application to validating energy trades. This is primarily due to the lack of additional constraints and checks required, beyond just the transactional integrity check provided by proof-of-work algorithms.

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 [24]. Additionally, there is still the problem of privacy—all transactions in these systems are public [25].

Most works discussing privacy look at it from the context of smart meters. McDaniel and McLaughlin discuss the privacy concerns of energy usage profiling, which smart grids could potentially enable [26]. Efthymiou and Kalogridis describe a method for securely anonymizing frequent electrical metering data sent by a smart meter [27] by using a third party escrow mechanism. Tan et al. study privacy in a smart metering system from an information theoretic perspective in the presence of energy harvesting and storage units [28]. 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, the transaction data provides more fine-grained information than the smart meter usage patterns [16].

Majumder et al. present an iterative double auction trading mechanism that preserves the participants’ privacy [29]. However, the privacy property pertains to the participants’ utility function models, not their identities.

Existing energy trading markets, such as the European Energy Exchange [30] and project NOBEL in Spain, employ the double-auction market mechanism [31], which can be implemented to preserve participant privacy. However, typical exchange implementations involve centralized database architectures prone to single points of failure.

3We are building a trading system in this paper and do not explicitly address the problem of billing. However, multiple billing approaches may be implemented on top of the blockchain, some of which could provide a very high level of privacy. This will be part of our future work.
Faqiry and Das present an auction mechanism for maximizing social welfare of buyers and sellers (if the supply is small) [32]. Their approach also provides some privacy meaning that a participant does not reveal their utility function. By constraining 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 pair as maximize the power sold, is maximized. In the later part of the paper, in order to make the trading fair, they consider an approach that discards the privacy maintained during the first phase. 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. ENERGY TRADING PROBLEM

In this section, we formulate the problem of matching energy future bids with asks (i.e., offers to buy energy to be delivered in the future with offers to sell energy) in the local energy trading market. Our formulation aims to promote market efficiency by maximizing the amount of energy futures traded within the microgrid, while satisfying microgrid safety requirements. We first introduce an initial problem formulation, in which we assume that all offers are available at the same time, and we clear the market at once. Then, we describe the generalized version of the problem, in which we consider a stream of incoming offers, which are cleared periodically.

A. Safety Requirements

While these trades are being cleared, we must consider safety requirements. At the distribution level, the amount of power that can be sent over a transmission line is typically limited by the thermal properties of the conductor, and it is physically enforced by protection equipment such as overcurrent relays. In traditional power systems, these capacity constraints are logically enforced by deploying some combination of load and generation curtailment schemes. Such schemes effectively impose upper bounds on the amount of power consumed by each load and the amount of power injected into the network by each source. For dispatchable generation, these upper bounds are typically calculated by solving some variant of an optimal power flow problem [33].

Since the setting of TES is fundamentally different from settings in which economic dispatch is appropriate (i.e. privately-owned DERs are not dispatchable by the DSO), we implement line capacity safety constraints by formulating them as constraints in the trading problem described below.

B. Problem Formalization

We begin by introducing notation for elements of the microgrid. For a list of symbols used in this paper, see Table I. We let \( F \) denote the set of feeders. For a feeder \( f \in F \), we let \( C_{ext}^f \) 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 \( C_{int}^f \) denote the maximum amount of power that is allowed to be consumed or produced within the feeder at any point in time.\(^4\) We assume that time is divided into intervals of fixed length \( \Delta \), and we refer to the \( t \)th interval simply as time interval \( t \).

The input of the energy trading problem is the set of buying and selling offers posted by the participants.\(^5\) For feeder \( f \in F \), we let \( S_f \) and \( B_f \) denote the set of selling and buying offers posted by participants located in that feeder, respectively.\(^6\) A selling offer \( s \in S_f \) is a tuple \((E_s, I_s, R_s)\), where

\(\text{Table I: List of Symbols}\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>set of feeders</td>
</tr>
<tr>
<td>( C_{ext}^f )</td>
<td>maximum net power consumption or net power production in feeder ( f \in F )</td>
</tr>
<tr>
<td>( C_{int}^f )</td>
<td>maximum total power consumption or total power production in feeder ( f \in F )</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>length of time intervals</td>
</tr>
<tr>
<td>( T_{clear} )</td>
<td>minimum number of time intervals between the finalization and delivery of a trade</td>
</tr>
<tr>
<td>( S_f )</td>
<td>set of selling offers from feeder ( f \in F )</td>
</tr>
<tr>
<td>( B_f )</td>
<td>set of buying offers from feeder ( f \in F )</td>
</tr>
<tr>
<td>( S, B )</td>
<td>set of all selling and buying offers, resp.</td>
</tr>
<tr>
<td>( S(t), B(t) )</td>
<td>set of selling and buying offers submitted by the end of time interval ( t ), resp.</td>
</tr>
<tr>
<td>( E_s, E_b )</td>
<td>amount of energy to be sold or bought by offers ( s \in S ) and ( b \in B ), resp.</td>
</tr>
<tr>
<td>( I_s, I_b )</td>
<td>time intervals in which energy could be provided or consumed by offers ( s \in S ) and ( b \in B ), resp.</td>
</tr>
<tr>
<td>( R_s, R_b )</td>
<td>reservation prices of offers ( s \in S ) and ( b \in B ), resp.</td>
</tr>
<tr>
<td>( M(s), M(b) )</td>
<td>set of offers that are matchable with offers ( s ) and ( b ), resp.</td>
</tr>
<tr>
<td>( I(s, b) )</td>
<td>( I_s \cap I_b )</td>
</tr>
<tr>
<td>( \hat{p}_{s,b,t} )</td>
<td>unit price for the energy provided by ( s ) to ( b ) in interval ( t )</td>
</tr>
<tr>
<td>( \hat{p}_{s,b,t} )</td>
<td>unit price for the energy provided by ( s ) to ( b ) in interval ( t )</td>
</tr>
<tr>
<td>( \text{Feasible}(S,B) )</td>
<td>set of feasible solutions given sets of selling and buying offers ( S ) and ( B )</td>
</tr>
<tr>
<td>( \hat{p}<em>{s,b,t}, \hat{p}</em>{s,b,t} )</td>
<td>finalized trade values</td>
</tr>
<tr>
<td>( L )</td>
<td>prediction window used by prosumers when posting selling and buying offers ((\text{min}(L) = 2))</td>
</tr>
<tr>
<td>( \Delta )</td>
<td>length of the time step used for simulating the real-interval of length ( \Delta )</td>
</tr>
<tr>
<td>( \Delta_s )</td>
<td>periodicity of solver that matches offers</td>
</tr>
</tbody>
</table>

\(^4\)In other words, limit \( C_{ext}^f \) is imposed on the net production and net consumption of all prosumers in feeder \( f \), while limit \( C_{int}^f \) is imposed on the total production and total consumption.

\(^5\)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.

\(^6\)If the DSO wants to participate in this energy trading market, it can be assigned to a “dummy” feeder in the problem formulation.
- \( E_s \) 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.

Similarly, a buying offer \( b \in B_f \) 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 \( S \) and \( B \) denote the set of all buying and selling offers (i.e., we let \( S = \bigcup_{f \in F} S_f \) and \( B = \bigcup_{f \in F} B_f \)).

We say that a pair of selling and buying offers \( s \in S \) and \( b \in B \) is matchable if
\[
R_s \leq R_b \quad (1) \\
I_s \cap I_b \neq \emptyset. \quad (2)
\]

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 \( s \in S \), we let the set of buying offers that are matchable with \( s \) be denoted by \( M(s) \). Similarly, we let the set of selling offers that are matchable with a buying offer \( b \) be denoted by \( M(b) \). Further, we let \( I(s, b) = I_s \cap I_b \).

A solution to the energy trading problem is a pair of vectors \((p, \pi)\), where
- \( p_{s,b,t} \) is the amount of power that should be provided by the seller \( s \in S \) and consumed by the buyer \( b \in M(s) \) in time interval \( t \in I(s,b) \),
- \( \pi_{s,b,t} \) is the unit price for the energy provided by seller \( s \in S \) to buyer \( b \in M(s) \) in time interval \( t \in I(s,b) \).

A pair of vectors \((p, \pi)\) 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:
  \[
  \forall s \in S : \sum_{b \in M(s)} \sum_{t \in I(s,b)} p_{s,b,t} \cdot \Delta \leq E_s \quad (3) \\
  \forall b \in B : \sum_{s \in M(b)} \sum_{t \in I(s,b)} p_{s,b,t} \cdot \Delta \leq E_b \quad (4)
  \]

- The amount of power flowing into or out of each feeder is below the safety limit in all time intervals:
  \[
  \forall f \in F, t : \left( \sum_{s \in S_f} \sum_{b \in B} p_{s,b,t} \right) - \left( \sum_{b \in B_f} \sum_{s \in S} p_{s,b,t} \right) \leq C_{f}^{\text{ext}} \quad (5) \\
  \forall f \in F, t : \left( \sum_{s \in S_f} \sum_{b \in B} p_{s,b,t} \right) - \left( \sum_{b \in B_f} \sum_{s \in S} p_{s,b,t} \right) \geq -C_{f}^{\text{ext}} \quad (6)
  \]

7We require the both the seller and buyer to produce a constant level of power during the time interval.

- The amount of energy consumed and produced within each feeder is below the safety limit in all time intervals:
  \[
  \forall f \in F, t : \sum_{b \in B_f} \sum_{s \in S} p_{s,b,t} \leq C_{f}^{\text{int}} \quad (7) \\
  \forall f \in F, t : \sum_{s \in S_f} \sum_{b \in B} p_{s,b,t} \leq C_{f}^{\text{int}} \quad (8)
  \]

- The unit prices are between the reservation prices of the seller and buyer:
  \[
  \forall s \in S, b \in M(s), t \in I(s,b) : R_s \leq \pi_{s,b,t} \leq R_b \quad (9)
  \]

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
\[
\max_{(p, \pi) \in \text{Feasible}(S,B)} \sum_{s \in S} \sum_{b \in M(s)} \sum_{t \in I(s,b)} p_{s,b,t}, \quad (10)
\]

where \( \text{Feasible}(S,B) \) is the set of feasible solutions given selling and buying offers \( S \) and \( B \) (i.e., set of solutions satisfying Equations (3) to (9) with \( S \) and \( B \)).

C. Advanced Problem Formulation

In our basic problem formulation, we assumed that all buying and selling offers \( B \) and \( S \) are available at once, and we cleared the market in one take. In practice, however, both the prosumers and the DSO may continuously submit new offers as their predictions, their physical state, and the market conditions change over time. 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. Here, we 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 \( t - T_{\text{clear}} \), where \( T_{\text{clear}} \) is a positive integer constant, which is set by the operator. Preventing “last-minute” changes can be crucial for safety and fairness since it allows both the DSO and the prosumers to prepare for delivering (or consuming) the right amount of power. In practice, the value of \( T_{\text{clear}} \) 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 \( B^{(t)} \) and \( S^{(t)} \) 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 \((p^*, \pi^*)\) to the extended energy trading problem:

\[
\max_{(p, \pi) \in \text{Feasible}(S^{(t)}, B^{(t)})} \sum_{s \in S^{(t)}} \sum_{b \in \mathcal{M}(s)} \sum_{\tau \in I(s, b)} p_{s, b, \tau} \tag{11}
\]

subject to

\[
\forall \tau < t + T_{\text{clear}}: \quad p_{s, b, \tau} = \hat{p}_{s, b, \tau} \tag{12}
\]

\[
\pi_{s, b, \tau} = \hat{\pi}_{s, b, \tau} \tag{13}
\]

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

\[
\hat{p}_{s, b, t + T_{\text{clear}}} := p^*_{s, b, t + T_{\text{clear}}} \tag{14}
\]

\[
\hat{\pi}_{s, b, t + T_{\text{clear}}} := \pi^*_{s, b, t + T_{\text{clear}}} \tag{15}
\]

By taking the above steps at the end of each time interval, trades are always cleared based on as much information as possible (i.e., considering as many offers as possible) without violating any safety or timing constraints. Next, we discuss how to implement the market using a blockchain-based decentralized platform.

V. SOLUTION APPROACH

In this section, we present a hybrid approach for solving the energy trading problem on a decentralized computing platform. Our hybrid approach combines the auditability and trustworthiness of blockchain-based smart contracts with the efficiency of more traditional computational platforms. We first show how to solve the problem by formulating it as a linear program. Then, we describe the computation and verification that need to be performed by the computational nodes and the smart contract in a decentralized microgrid.

A. 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:

\[
\max_{p, \pi} \sum_{s \in S} \sum_{b \in \mathcal{M}(s)} \sum_{t \in I(s, b)} p_{s, b, t} \tag{16}
\]

subject to Equations (3) to (9) and

\[
p \succeq 0 \text{ and } \pi \succeq 0. \tag{17}
\]

The extended energy trading problem introduced in Section IV-C can be reformulated as a linear program similarly, by considering \(S^{(t)}, B^{(t)}, \hat{p}, \hat{\pi}, \) and the additional constraints.

B. Hybrid Solver Implementation

Although solving linear programs is not computationally hard, it can be challenging with a large number of variables and constraints in resource-constrained computing environments. Since computation is relatively expensive on blockchain-based distributed platforms\(^9\), solving the energy trading problem using a blockchain-based smart contract would not be scalable in practice. In light of this, we propose a hybrid implementation approach, which 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 and then (2) use a smart contract to record the solution on the blockchain. To implement this hybrid approach securely and reliably, we must address the following issues.

- Computation that is performed off-blockchain does not satisfy the auditability and security requirements that smart contracts do. Thus, the results of any off-blockchain computation must be verified in some way by the smart contract before recording them on the blockchain.
- Due to network disruptions and other errors (including deliberate denial-of-service attacks), the off-blockchain solver might fail to provide the smart contract with a solution on time (i.e., before trades are supposed to be finalized). Thus, the smart contract must be able to proceed without an up-to-date solution.
- For the sake of reliability, the smart contract should accept solutions from multiple off-blockchain sources; however, these sources might provide different solutions. Thus, the smart contract must be able to choose from multiple solutions (some of which may come from a compromised computer).

1) Blockchain-based Smart Contract: We implement a smart contract that can (1) verify whether a solution \((p, \pi)\) is feasible and (2) compute the value of the objective function for a feasible solution. Compared to finding an optimal solution, these operations are computationally inexpensive, and they can easily be performed on a blockchain-based decentralized platform. Using these capabilities, the smart contract provides the following functionality:

- Solutions may be submitted to the contract at any time. The contract verifies the feasibility of each submitted solution, and if the solution is feasible, then it computes the value of the objective function. 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 \(t + T_{\text{clear}}\) based on the

Further, Solidity, the preferred high-level language for Ethereum, currently lacks built-in support for certain features that would facilitate the implementation of a linear programming solver, such as floating-point arithmetics.

\(^9\)Our current implementation uses Ethereum.
candidate solution.\footnote{If no solution has been submitted to the contract so far, which might be the case right after the trading system has been launched, \( p = 0 \) may be used as a candidate solution.}

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 \((p, \pi)\) that is feasible for sets \(S\) and \(B\) is also feasible for supersets \(S' \supseteq S\) and \(B' \supseteq B\). 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 final 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) Off-blockchain 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 again due to any feasible solution for sets \(S\) and \(B\) also being feasible for supersets \(S' \supseteq S\) and \(B' \supseteq B\).

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.\footnote{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.}

Thereby, we enable finding solutions in an efficient and very flexible manner, while reaping the benefits of smart contracts, such as auditability and trustworthiness.

VI. ENERGY TRADING SYSTEM

In this section, we outline a system that can provide the energy trading and market clearing functionality described in the preceding sections. We first describe the main components of the system, including a short background on the runtime platform. Then, we give an overview of the messages exchanged between these components in the trading workflow.

A. Background

In any technical system that relies on embedded computing, the most important ingredient is an “operating system” that provides the foundations for all algorithms, isolates the hardware details from the algorithms, and provides essential mechanisms for resource management, fault tolerance, and security. For example, in the work presented in this paper, this framework allows us to disseminate information between the actors. Our extended team has developed a platform called ‘Resilient Information Architecture Platform for Smart Grid’ (RIAPS) [34], where each actor is a composition of several libraries that provide (1) a component model that provides a concurrent model of computation for building distributed real-time applications, (2) a messaging framework for facilitating interactions among actors, (3) a resource-management framework for controlling the use of computational resources, (4) a fault-management framework for detecting and mitigating faults in all layers of the system, (5) a security framework to protect the confidentiality, integrity, and availability of system under cyber-attacks, (6) a fault tolerant time synchronization service, (7) a discovery framework for establishing the network of interacting actors of an application, and (7) a deployment and management framework for the administration and control of the distributed applications from a control room.

B. Trading System Architecture

Figure 2 shows the architecture of the energy trading system. The specific platforms and tools used in our implementation are shown in parentheses and as arrow labels. All components are written in Python and communicate with each
C. Providing Privacy

To protect their privacy, prosumers use anonymous addresses when interacting with the blockchain (e.g., posting offers). By generating new anonymous addresses at random periodically, they can prevent other participants from linking the anonymous addresses to their actual identities [17], thereby keeping their trading activities private. However, in contrast with our prior work [17], the addresses used in the workflow described in the next section are not completely anonymous. Since the blockchain-based smart contract has to check feeder-level safety constraints, each anonymous address must be linked to a feeder. Hence, an anonymous address can hide only the prosumer’s identity but not its feeder, which is a manifestation of the trade-off between safety and privacy.\footnote{Actually, participants can remain anonymous among a class of feeders with same number of participants and identical safety constraints.}

However, anonymous addresses pose a further threat to safety. Since participants can generate anonymous addresses at almost no cost, they could post selling and buying offers for large amounts of energy, without any intention of delivering and without facing any repercussions. A malicious or faulty participant could easily destabilize the grid with this form of reckless trading. Consequently, the amount of energy that may be traded by anonymous addresses belonging to a participant must be limited.

Thus, we use the concept of energy assets, initially introduced in [17] and used in the trading workflow described in the next section. An energy asset represents a permission to sell or buy a specific amount of energy in a specific set of time intervals. Each prosumer can ask the DSO to transfer assets to an anonymous address, who can check whether it would be safe to give permission to the participant. If it would be, then the DSO records this transfer on the blockchain. Later, when the participant posts an offer from the anonymous address, the smart contract can check whether the address has the assets required for the offer. Since only the DSO can link the anonymous address to the participant, assets enable enforcing participant-level constraints without violating privacy.

D. Trading Workflow

Figure 3 illustrates the trading workflow. For ease of presentation, the figure shows only one prosumer and only a single message of each type. In practice, a large number of prosumers may interact with the DSO and the smart contract, and each of them may exchange multiple messages.

The workflow includes the following messages:

- `withdrawAssets(anonAddress, energy, intervals, amount)`: message sent by a prosumer to the DSO, asking the DSO to transfer energy from the prosumer’s account at the DSO to an anonymous address. Before sending this message, the prosumer first generates a random anonymous address to protect her privacy. The message specifies the assets that the prosumer wishes to withdraw (i.e., amount of energy and time intervals) and the anonymous address to which the DSO should transfer them. Note that the prosumer may send this message long before actually engaging in trading, so the DSO does not have to be online continuously.
- `failedWithdrawal(anonAddress, msg)`: message sent by the DSO to the prosumer, notifying the prosumer that the requested assets cannot be withdrawn due to, e.g., energy safety requirements or insufficient funds.
- `addEnergy(anonAddress, energy, intervals)`: message sent by the DSO to the prosumer, notifying the prosumer that the requested assets cannot be withdrawn due to, e.g., energy safety requirements or insufficient funds.
- `addFinancialBalance(anonAddress, amount)`: message sent by the DSO to the prosumer, notifying the prosumer that the requested assets cannot be withdrawn due to, e.g., energy safety requirements or insufficient funds.

Fig. 3. Sequence diagram of the trading workflow. Solid lines represent messages, including smart-contract function calls (i.e., blockchain transactions), while dashed lines represent smart-contract events. For ease of presentation, we show only a single prosumer and one instance of each communication.
prosumer’s request, creating energy and financial assets on the blockchain and transferring them to an anonymous address. Before recording this transaction, the DSO must first verify that enabling the prosumer to trade these assets does not violate any safety constraints and that the anonymous address is linked to the correct feeder.

- AssetAdded(anonAddress, energy, intervals), FinancialAdded(anonAddress, amount): broadcast messages emitted by the smart contract (i.e., events logged on the blockchain), notifying the prosumer that the requested assets have been transferred to the anonymous address.
- postOffer(energy, intervals, price): smart-contract function called by a prosumer (from its anonymous address), publicly posting an energy bid or ask.
- OfferPosted(offerID, energy, intervals, price): message broadcast by the smart contract, notifying solvers that an offer was posted.
- submitSolution(powers, prices): smart-contract function called by a solver, submitting a new solution for the energy trading problem.
- SolutionFinalized(powers, prices): message broadcast by the smart contract, notifying both prosumers and solvers of energy trades that have been finalized.
- depositEnergy(energy, intervals), depositFinancial(amount): smart-contract functions called by a prosumer, depositing energy and financial assets to the prosumer’s account. Note that to protect privacy, the calls do not specify the prosumer, so the DSO has to keep track of which prosumer has used which anonymous address.
- EnergyDeposited(anonAddress, energy, intervals), FinancialDeposited(anonAddress, amount): messages broadcast by the smart contract, notifying the DSO that assets have been deposited from anonymous address, which triggers the transfer of these assets to the prosumer’s account at the DSO.

E. Implementation Considerations

1) Parameters: The system of prosumers, solvers, DSO, and smart contract operates mostly asynchronously. The only synchronous communication occurs between prosumers and the DSO. They all operate as independent processes running on remote nodes, with their own time bases. In particular, the solver can operate as a periodic process (with a period \( \Delta_s \)), waiting on information from the smart contract about all the offers that have been posted in the prior period. The prosumers can also operate as periodic processes, submitting their offers and bids to the smart contract. In practice, prosumers will be synchronized with real wall-clock time, making their bids and asks known for future intervals, depending upon the time at which they post their bids/asks and how far in the future they can predict their usage or operation. We make their prediction window \( L \) a parameter of the system. The value of this parameter is at least 2, because prosumers have to make a bid/ask for at least the next interval (we count the current interval in \( L \)). A larger value of this prediction window will increase the risk of uncertainty for the prosumer, since they are expected to be able to fulfill their bid or ask. Additionally, during our experiments, we can simulate \( \Delta \) as \( \tilde{\Delta} \) to speed up the process. These parameters are described in Table I.

2) Speed and Synchronization Considerations: A relevant problem for TMP is deciding how fast it can run and ensuring that trades for the next interval can clear before the \( T_{\text{clear}} \) parameter, which has a minimum bound of 1. In our system, regular network communication mechanisms are assumed to be fast. However, the communication with the smart contract is limited by the block mining rate, and we need multiple messages exchanged in each interval (see Figure 3), so the miners need to work fast enough to mine a few blocks in each time interval. In a closed environment, this can be achieved by reducing the difficulty of the cryptographic puzzle solved for proof-of-work consensus. In our system, as shown in the next section, we are able to clear transactions much faster than one time interval. For larger systems, the proof-of-work consensus may be replaced by, e.g., proof-of-stake, for scalability.

Another problem is the synchronization between the different agents. The runtime platform of RIAPS provides us with high-precision time synchronization [35]. However, even if we only have NTP as the synchronization mechanism, we can operate the system correctly. This is because intervals in practice are relatively (i.e., compared to typical communication delays) long (e.g., 15 minutes). Additionally, the smart contract can ensure that the system always proceeds to the next time interval (however, the accuracy of this is limited by the mining rate, see previous paragraph). Additionally, our system can tolerate or discard out-of-order and late messages due to the event chronology implemented in the blockchain platform. In practice, however, prosumers should try to post their offers early within a time interval, so that solvers will include them in the solution for the current time interval. On the other hand, solvers should wait some time before starting to work, so that they can collect all (or at least most) of the offers posted in the interval.

VII. Experimental Evaluation and Discussions

We consider a collection of load traces recorded by Siemens from a microgrid in Germany, containing 102 homes across 11 feeders (5 producers and 97 consumers). Figure 4 describes the feeder structure, the number of participants per feeder, and the feeder safety limits. We use \( \Delta = 15 \) minute intervals, resulting in a total of 96 intervals across the whole day. Figure 5 shows the total production and consumption across this microgrid. The horizontal axis shows the starting time for each of the 96 intervals. Since the dataset does not include prices, we assume reservation prices to be uniform in our experiments, and focus on studying the amount of energy traded and the performance of the system.

\[^{13}\text{The size of the prediction window is part of the prosumer strategy, which is not explored in this paper as our focus is on the implementation of the TMP.}\]

\[^{14}\text{Note that this is the amount of real time passed in the simulation before proceeding to the next interval. This allows us to speed things up for the experiment, since running the system slower would just be easier.}\]
Fig. 4. Feeder diagram. Brown nodes are feeder junctions, numbered 1 to 11 from top to bottom. Black nodes are the overcurrent relays, which ensure that the total power flowing in and out of the feeder is below 20 kW. The green nodes are the junction points for the producers (5), and the red nodes are junction points for the consumers (97). There are 102 prosumers in total.

As one of our primary contributions in this work is the ability to specify multiple time intervals for selling offers, we extended the trace that was collected by Siemens to allow each producer to have a battery with a total capacity of 90 kWh. With a battery, a producer can take the energy produced in a time interval and decide to make it available in future time intervals. Note that the resulting offers always span a contiguous set of time intervals, so they can be specified by their starting time and length. Figure 6 shows these intervals for one particular producer. The producers charge their batteries only when the total consumption is less than the total production, which happens just after 12:00 PM, see Figure 5.

A. Test Bed

The hardware test bed is a cluster of 31 BeagleBone Black (BBB) single-board computers (see Figure 7) acting as participants in the energy trading system. The BBBs are set up as light clients because they are resource constrained and therefore are not suitable for mining or acting as solvers. This means that they can safely access the blockchain but do not participate in the consensus process. In the dataset provided by Siemens, there are 95 consumers and 7 producers of power, see Figure 2. These participants are divided between the BBBs in the cluster. The PlaTIBART (Platform for Transactive IoT Blockchain Applications with Repeatable Testing platform) [36] platform provided us with the necessary devops support.

The block mining is provided by external hardware, locally or in a cloud server. We had a single miner instance that was responsible for maintaining the blockchain, and a single solver instance that used CPLEX [37] to solve the energy trading problem. This setup can be easily scaled to add more miners and solvers if there are enough computational resources available. The communication between the components was implemented using ZeroMQ, which is one of the communication protocols available in RIAPS.

B. Experiments

Table II describes the specifics of the four categories of tests that we ran. The tests vary the different implementation parameters (see Table I and Section VI-E). This allows us to
study how changing these parameters affects the total amount of energy traded.

Figure 8 shows the total energy traded for different tests. We varied the prediction window \( L \) for the participants from 2 to 13. That is, in each interval, the participants submitted offers starting from the next 1 to 12 intervals (current interval is always counted in the prediction window). The experiment simulated the whole day from the first interval starting at 0:00 (12:00 AM) to the 95\(^{th} \) interval ending at 23:59 (11:59 PM). As expected, increasing the prediction window without batteries has no effect on the total amount energy traded. This is because any production must be dispatched within one time interval, so the solver cannot optimize energy usage across multiple intervals even if future offers are available.

However, this is possible if there are batteries in the system. With batteries, the amount of energy traded increases as we allow the solver to match offers across multiple time intervals at once. Trading increases with the prediction window because of the increased analysis space available to the solver. Figure 5 shows the per interval trades for three of these cases (A, C, and D): without battery, with battery and \( L = 5 \), as well as with battery and \( L = 13 \).

Figure 9 shows the energy matched per interval in test case C for the first prosumer of the first feeder. Figure 10 shows a histogram of the time between posting an offer and recording a trade on the blockchain that includes the offer (also in test case C). The time length was always less than \( \Delta \), which was 120 seconds (see Table II).

### C. Discussion

We conclude our evaluation by providing a brief discussion of how our platform satisfies the requirements listed in Section II-A (see Table III for a summary). First, the communication architecture is provided by Ethereum and our middleware RIAPS. Our performance results in Figure 10 demonstrate that our platform can process and match trades much faster than what would typically be required in practice (Section VI-E). Second, we enforce feeder-level operational safety and stability constraints on trading using a blockchain-based smart contract (Section V-B). These are in addition to the prosumer-level constraints enforced by tracking energy assets (Section VI-C, [17]). Third, we ensure market efficiency and security by enabling the smart contract to validate and evaluate the trading solutions that it receives (Section V-B). Finally, we provide privacy for prosumers by allowing them to hide their identity using anonymous addresses (Section VI-C and [17]). However, prosumers are required to reveal which feeder they belong to in order to enable enforcing feeder-level safety constraints.

During the design and evaluation of our platform, one of the main challenges was the conflict between safety, privacy, and efficiency. For example, the enforcement of feeder-level safety
constraints required prosumers to reveal their feeder during trading, instead of staying completely anonymous. Further, feeder-level safety constraints can also prevent meeting energy demand with local supply, even if there is surplus production in the microgrid (Figure 5).

VIII. Conclusion

In this paper, we described the design and implementation of a transaction management platform (TMP) for IoT-based transactive microgrids. 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. Further, the ability to trade across multiple time intervals enables participants to take full advantage of batteries, thereby smoothening the load on the main grid. Finally, the use of blockchains provides decentralized trust and consensus capabilities, which protect from malicious actors.

REFERENCES


