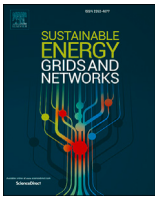
Sustainable Energy, Grids and Networks 43 (2025) 101762

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

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



Blockchain-based hierarchical smart contracts to prevent user profiling in decentralized energy trading systems

Joan Ferré-Queralt, Jordi Castellà-Roca, Alexandre Viejo*

Universitat Rovira i Virgili, Departament d'Enginyeria Informàtica i Matemàtiques, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Països Catalans 26, Tarragona, E-43007, Catalonia, Spain

ARTICLE INFO

Keywords:
Blockchain
Decentralized energy trading
Hierarchical smart contracts
Privacy protection
User profiling prevention
Smart grid technology

ABSTRACT

Smart grid technology has transformed electricity generation, distribution, and consumption by incorporating advanced communication systems and distributed energy resources, including solar panels and energy storage solutions. This integration enables prosumers to actively participate in energy markets, benefiting from real-time monitoring, dynamic pricing, and load balancing. However, the detailed data collected during these processes raise significant privacy concerns, as it may expose sensitive information about users' lifestyles. This work presents an innovative energy trading system operating within decentralized energy distribution networks. The system leverages blockchain-based hierarchical smart contracts to enhance privacy protection for users. It automates energy trades, ensures accurate transaction verification, and obscures user identities and energy consumption patterns through its hierarchical structure, preventing unauthorized profiling or data breaches. Additionally, mechanisms to detect and penalize dishonest behavior are incorporated, ensuring the integrity and fairness of the energy market. The feasibility of the proposed system is experimentally evaluated in an IoT environment through a small-scale implementation using actual IoT devices, yielding positive results in terms of scalability and privacy features. Lastly, a comparative analysis is presented to demonstrate the advantages of the proposed system over existing state-of-the-art solutions.

1. Introduction

The advent of smart grid technology represents a transformation in the way electricity is generated, distributed, and traded [1]. Smart grids leverage advanced information and communication technologies to enable a more interactive, efficient, and resilient energy system. This paradigm shift is driven by the increasing integration of renewable energy sources, the growing demand for more reliable electricity supply, and the necessity to improve energy efficiency and sustainability [2]. By incorporating sophisticated digital solutions, smart grids not only enhance the reliability and robustness of the power supply but also enable new models of energy trading, empowering prosumers to actively participate in decentralized energy markets.

At the core of this transformation is the integration of distributed energy resources (DERs), such as solar panels, wind turbines, and energy storage systems [3]. These resources are often located close to the point of consumption, allowing prosumers—entities that both produce and consume electricity—to trade surplus energy with other market participants [4]. Unlike traditional electricity grids, which are characterized by the one-way flow of electricity from centralized power plants to consumers, smart grids facilitate the bidirectional flow of electricity and information, enabling real-time energy trading, dynamic pricing, and load balancing [5]. This interactivity is essential for optimizing energy usage, integrating renewable energy sources, and ensuring the stability of decentralized energy trading systems.

As smart grids evolve and incorporate more distributed elements, energy trading becomes a fundamental mechanism for balancing supply and demand across decentralized networks. However, new challenges arise regarding security, privacy, and scalability. While advanced metering infrastructure (AMI) enables detailed energy monitoring and supports real-time optimization of trading activities, it also raises critical privacy concerns. Specifically, *user profiling*—the analysis of prosumers' energy consumption patterns—can reveal sensitive information about their daily routines and lifestyle. Protecting this data from unauthorized access is crucial not only for preserving user privacy but also for building trust in decentralized energy markets [6].

* Corresponding author.
Email addresses: joan.ferreq@urv.cat (J. Ferré-Queralt), jordi.castella@urv.cat (J. Castellà-Roca), alexandre.viejo@urv.cat (A. Viejo).

https://doi.org/10.1016/j.segan.2025.101762
Received 20 March 2025; Received in revised form 14 May 2025; Accepted 30 May 2025
Available online 2 June 2025

Moreover, ensuring that privacy-preserving energy trading solutions are scalable and efficient, particularly in decentralized and IoT environments, is equally important. As smart grids shift towards more decentralized control and peer-to-peer energy trading, traditional centralized security models face limitations. Distributed systems, as highlighted in [7], offer promising alternatives with their inherent redundancy and tamper-proof nature. However, achieving a balance between safeguarding privacy and delivering a scalable, lightweight energy trading solution remains a significant challenge that this work seeks to address.

1.1. Related work

The works presented in [8,9] were among the first to leverage distributed systems for facilitating peer-to-peer energy transactions. However, despite their decentralized ambition, these solutions ultimately relied on a central entity to match consumers and producers, introducing a central point of failure and replicating the security concerns associated with centralized systems.

A significant step towards a truly decentralized electricity trading system was presented in [10], where the authors proposed an energy currency based on blockchain technology, drawing on the work of Satoshi Nakamoto in [11]. This approach paved the way for the seamless integration of blockchain and smart contracts into the smart grid, enabling secure and transparent peer-to-peer energy trading without the need for a central authority. As a result, several works, such as [12–16], implemented blockchain-based smart grid architectures, showcasing the versatility and potential of blockchain technology in revolutionizing the energy sector.

Despite these advances, the aforementioned blockchain-based solutions primarily employed the Proof-of-Work (PoW) consensus mechanism. While PoW offers strong security guarantees, it is notorious for its high energy consumption due to the substantial computing power required to validate transactions [17]. This high energy demand contradicts the principles of a sustainable smart grid, making PoW-based approaches unsuitable for large-scale adoption.

In response to this challenge, other works, such as [10,18–20], explored alternative, more energy-efficient consensus mechanisms. However, these proposals often overlooked the importance of protecting user privacy, which, as stated in [21], can lead to security vulnerabilities, undermining the trust and adoption of any proposed system. For example, works such as [22] demonstrated that by analyzing a user's transactions, even if they are anonymous, an attacker can detect patterns of electricity consumption or production, infer their identity, and carry out successful profiling attacks.

Initially, proposals such as [23–25] aimed to tackle these privacy issues by employing permissioned blockchains in the first two cases and a private blockchain in the latter, rather than public blockchains. The primary goal of this approach was to restrict network participation to authorized entities, thereby mitigating the risk of external attackers identifying users through transaction analysis. However, this solution was only partially effective, as it does not prevent internal users within the permissioned network from conducting profiling attacks.

The authors in [26] introduce a framework for microgrid energy transactions using a multiple authorities attribute-based verification mechanism within blockchain. The proposed system focuses on enhancing privacy through attribute-based signatures and distributed verification. However, it primarily aims to protect transaction authenticity and data integrity rather than specifically mitigating profiling attacks based on behavioral data patterns. As a result, it may not be fully resilient against such attacks.

More recent works, including [27], employ encryption of data sent to blockchain-based smart contracts, which helps protect against user profiling but introduces new challenges. For example, while encrypting consumption and production data enhances privacy, these systems often do not verify whether the reported values are accurate. As a result, dishonest users may submit false energy production or consumption records, misleading the system and other participants and potentially causing imbalances or unfair outcomes. This issue also arises in proposals such as [28], which use zero-knowledge proofs to validate hidden electricity values. Although these mechanisms ensure data confidentiality, they prevent any trusted party from confirming the authenticity of the submitted data, leaving the system vulnerable to dishonest transactions and unable to detect the malicious parties.

Similarly, other approaches, such as those presented in [29–31], utilize homomorphic encryption to encrypt consumption and production transactions. These encrypted transactions are aggregated and then uploaded to the blockchain, revealing only overall totals. However, this aggregation hides individual usage details, making it impossible for the system to perform accurate, user-level billing.

Finally, addressing the issue of false energy production and consumption, [32] introduces a reputation system to penalize misbehaving users. However, in this scheme, no entity maintains the relationship between smart grid accounts and the real identities of users. As a result, it is unable to effectively apply coercive measures against users who engage in dishonest behavior.

1.2. Contribution and plan of this paper

This work presents a novel energy trading system within decentralized energy distribution networks, allowing consumers and producers to participate while preserving their privacy against user profiling attacks.

Building on [33], the new proposal incorporates hierarchical smart contracts for secure transaction validation while obscuring sensitive user details and conducts a thorough feasibility evaluation. The system also employs smart contracts for accurate, automated record-keeping and includes mechanisms to identify and penalize misbehaving users, thus maintaining trust and efficiency without compromising privacy.

The paper is structured as follows. Section 2 provides background on the blockchain technology used in this proposal. Section 3 presents an overview of the system, including key aspects, general architecture, and design requirements. Section 4 details the hierarchical structure of smart contracts. Section 5 describes the system's key protocols. Section 6 evaluates system functionality through a distributed implementation and performance analysis. Section 7 examines security and privacy aspects. Section 8 presents a comparative analysis with state-of-the-art proposals and a SWOT evaluation of the system. Finally, Section 9 summarizes the findings and outlines future research directions.

2. Background

This section provides the foundational concepts of blockchain systems relevant to the proposed energy trading platform. It first introduces the structure and operation of blockchain networks and consensus mechanisms, followed by a description of smart contracts and their execution model. Finally, it explains the concept of gas, which represents the computational cost of executing smart-contract functions and plays a central role in performance and security evaluation throughout the manuscript.

2.1. Blockchain technology

A blockchain is a replicated, append-only ledger jointly maintained by a peer-to-peer network of nodes [11]. Each block includes a set of state transitions (transactions) and a header containing a cryptographic hash of the previous block, thereby forming an immutable hash chain.

Because every node verifies the entire history, tampering with past data would require recalculating the hash pointers of all subsequent blocks and convincing a majority of honest nodes. This makes large-scale forgery computationally or economically infeasible.

A consensus protocol ensures that the network converges on a single canonical chain, even in the presence of asynchrony and potentially Byzantine participants. To achieve this, blockchains rely on validators, which are nodes responsible for proposing and confirming new blocks. These validators follow a specific consensus mechanism to agree on the next state of the ledger.

The two most widely adopted consensus schemes are Proof-of-Work (PoW) and Proof-of-Stake (PoS). PoW assigns voting power based on computational effort, while PoS assigns it based on the amount of native tokens a participant is willing to stake.

In public blockchains such as Bitcoin and Ethereum, any node can participate as a validator, supporting open and decentralized governance. In contrast, *permissioned* blockchains, such as Hyperledger Fabric, restrict validation rights to a fixed set of approved entities. This approach sacrifices openness in favor of higher throughput and simpler governance.

2.2. Smart contracts

Smart contracts are *deterministic* programs that reside at on-chain addresses and execute within the blockchain's virtual machine. Once deployed, their bytecode and storage become part of the ledger's replicated state.

External users or other contracts invoke functions by broadcasting a transaction that specifies: (i) the contract address; (ii) the function selector and encoded parameters; and (iii) a fee budget known as *gas*.

To ensure consistent results across the network, every full node re-executes the transaction and must reach the same resulting state. As a result, smart contract logic must avoid nondeterministic operations, such as relying on wall-clock time or external sources of randomness [34].

2.2.1. Gas and transaction fees

Gas is the metering unit that prices low-level opcodes and persistent storage access in the Ethereum Virtual Machine (EVM). It serves three complementary purposes:

- *Resource accounting*: Computationally intensive operations (e.g., SHA-256 hashing or writing to storage) consume more gas than simple arithmetic operations, ensuring that fees reflect actual compute and I/O usage.
- *Denial-of-service (DoS) safety*: Transaction senders must specify the maximum amount of gas they are willing to consume (`gasLimit`), which prevents infinite loops or excessive computation from exhausting network resources.
- *Economic incentive*: Gas fees reward validators for processing transactions and maintaining network security.

3. System overview

This section discusses the background and requirements of the proposed system and provides a general overview of its architecture and operation. All nomenclature symbols used in this section are listed in Table 1.

3.1. Key concepts

The following definitions establish key concepts fundamental to understanding the new proposal.

Definition 1 (Electricity transaction). An electricity transaction τ is defined as:

$$\tau = \{u_k, d, t, E_{d,t}^{u_k}\}$$

where u_k is a user identifier, d is the transaction date, t is the corresponding time slot, and $E_{d,t}^{u_k}$ is the amount of electricity consumed or generated by user u_k. Each day is divided into T time slots, denoted as $T = \{t_0, t_1, \dots, t_{|T|-1}\}$. The energy value $E_{d,t}^{u_k}$ is positive when the user consumes electricity and negative when the user generates it.

Definition 2 (Protected transaction). An electricity transaction τ (see Definition 1) is univocally linked to a user identity u_k. Publicly disclosing users' transactions τ on the blockchain through a smart contract would reveal this link, enabling re-identification and subsequent user profiling.

Table 1
Nomenclature of key symbols.

Symbol	Description
τ	Electricity transaction
u_k	User identifier
d	Transaction date
t	Time slot index
T	Set of all time slots, $T = \{t_0, \dots, t_{\|T\|-1}\}$
$E_{d,t}^{u_k}$	Electricity amount consumed/generated by u_k at (d,t)
τ'	Protected transaction
$\alpha_i^{u_k}$	Blockchain address i of user u_k
A^{u_k}	Set of addresses of user u_k: $\{\alpha_0^{u_k}, \dots\}$
$E_{d,t}^{\alpha_i^{u_k}}$	Electricity amount for address $\alpha_i^{u_k}$
$EC_{d,t}^{u_k}$	Total consumption: $\sum_{\alpha \in A^{u_k}} E_{d,t}^{\alpha}$
D	Set of days from registration to present
$EP_t^{u_k}$	Avg. consumption of u_k at slot t
EP^{u_k}	Electricity pattern vector
$\delta(u_a, u_b)$	Distance function
U	Set of all users $\{u_0, \dots, u_{\|U\|-1}\}$
γ	Electricity profile (cluster)
Γ	Set of all profiles
μ_t^{γ}	Mean pattern of profile γ at slot t
μ^{γ}	Pattern vector
SM_i	Smart meter of prosumer i
P_i	Producer i
G_x	Group of prosumers x
MG	Main grid layer
MGp	Main grid producers
SG	Smart grid layer
Co_x	Coordinator for group G_x
O_i	Operator i
$\Gamma(U)$	Objective function
$\gamma(u_i)$	Classification function
mSC	Main smart contract
iSC_n	Intermediate smart contracts
gSC_m	Group smart contracts

To mitigate this risk, our proposal divides each τ into multiple protected transactions τ'. Instead of using u_k as the user identifier, one of the user's blockchain addresses α^{u_k} is used. Since each user possesses a set A^{u_k} of different blockchain addresses, i.e., $A^{u_k} = \{\alpha_0^{u_k}, \dots, \alpha_{n-1}^{u_k}\}$, a different α^{u_k} is assigned to each smart contract in which they participate.

$$\tau' = \{\alpha_i^{u_k}, d, t, E_{d,t}^{\alpha_i^{u_k}}\}$$

Definition 3 (Electricity consumption). Since each electricity transaction is divided into multiple protected transactions (see Definition 2), where the user's unique identifier u_k is replaced by different blockchain addresses, the electricity consumption $EC_{d,t}^{u_k}$ for a user u_k during a given time slot t on date d must be computed as the sum of the electricity consumption across all blockchain addresses $\alpha_n^{u_k}$ associated with u_k in the corresponding smart contracts:

$$EC_{d,t}^{u_k} = \sum_{\alpha \in A^{u_k}} E_{d,t}^{\alpha}$$

Definition 4 (Electricity pattern). A user's electricity pattern EP^{u_k} is the daily average electricity consumption/production for each time slot $t \in T$:

$$EP_t^{u_k} = \frac{1}{|D|} \sum_{d \in D} EC_{d,t}^{u_k}$$

$$EP^{u_k} = \{EP_0^{u_k}, EP_1^{u_k}, \dots, EP_{|T|-1}^{u_k}\}$$

where D denotes the set of all days from the day the user joined the smart grid until the present day.

Electricity patterns enable comparisons between users based on the similarity of their consumption profiles, forming the foundation for user clustering and profiling within the network.

Definition 5 (Distance). The distance δ between users u_a and u_b is a metric that quantifies their similarity across all time slots. It is defined as:

$$\delta(u_a, u_b) = \sum_{t \in T} |EP_t^{u_a} - EP_t^{u_b}|$$

Analyzing the consumption patterns of a large user group reveals distinct patterns based on user habits, preferences, and behaviors. This analysis enables users to be clustered into categories called electricity profiles.

Definition 6 (Electricity profile). Let the set of all users be denoted as $U = \{u_0, u_1, \ldots, u_{|U|-1}\}$, where each $u_i \in U$ has an associated electricity pattern EP^{u_i} that represents the user's daily average electricity consumption/production pattern. Therefore, an electricity profile γ is defined as a group of users from the set U who exhibit similar electricity consumption patterns.

The complete set of electricity profiles is denoted as Γ, encompassing all identified groups of users. The set U can then be partitioned into $|\Gamma|$ profiles, where $|\Gamma| \leq |U|$. Each electricity profile γ is associated with a representative mean electricity pattern, denoted as μ^γ, which characterizes the average electricity consumption/production across all users in the profile for each time slot:

$$\mu_t^\gamma = \frac{1}{|\gamma|} \sum_{u \in \gamma} EP_t^u$$
$$\mu^\gamma = \{\mu_0^\gamma, \mu_1^\gamma, \ldots, \mu_{|T|-1}^\gamma\}$$

Definition 7 (Objective function). The process of creating the set Γ involves minimizing the variance of electricity consumption across each time slot for all users u_i within each profile γ, thereby ensuring that users in each profile exhibit closely similar patterns:

$$\Gamma(U) = \arg\min_{\Gamma} \sum_{\gamma \in \Gamma} \sum_{u \in \gamma} \delta(u, \mu^\gamma)^2$$

Definition 8 (Classification function). A new or unknown user u_i can be classified into an existing profile γ_j using the classification function:

$$\gamma(u_i) = \arg\min_{\gamma_j} \delta(u_i, \mu^{\gamma_j})$$

3.2. General architecture

The proposed architecture, depicted in Fig. 1, consists of three interconnected layers. The *prosumer layer* facilitates localized electricity trading among users. The *smart grid layer* manages transaction oversight and coordinates energy needs with the *main grid layer*, which comprises primary electricity providers. The following key actors are involved within these layers:

- *Smart meters* (SM_i): Devices that track the energy usage of each prosumer.
- *Prosumers*: Individuals capable of both producing and consuming energy. Depending on their net energy balance, they can function as either *Consumers* (C_i) or *Producers* (P_i). They are organized in groups G_x.
- *Main grid* (MG): This layer distributes energy, forecasts consumption patterns, and provides additional electricity to meet demand. It includes *Main grid producers* (MGp), which are responsible for supplying large-scale electricity production.
- *Smart grid* (SG): Manages transactions and payment processing.
- *Coordinators* (Co_x): Entities acting as distribution system operators (DSOs), that is, the owners of the physical feeders supplying each prosumer group G_x. Since each feeder has a single owner, the DSO inherently has full visibility over the energy entering and leaving its network. In the proposed model, the DSO is treated as a *trusted third party (TTP)* responsible for verifying transactions.

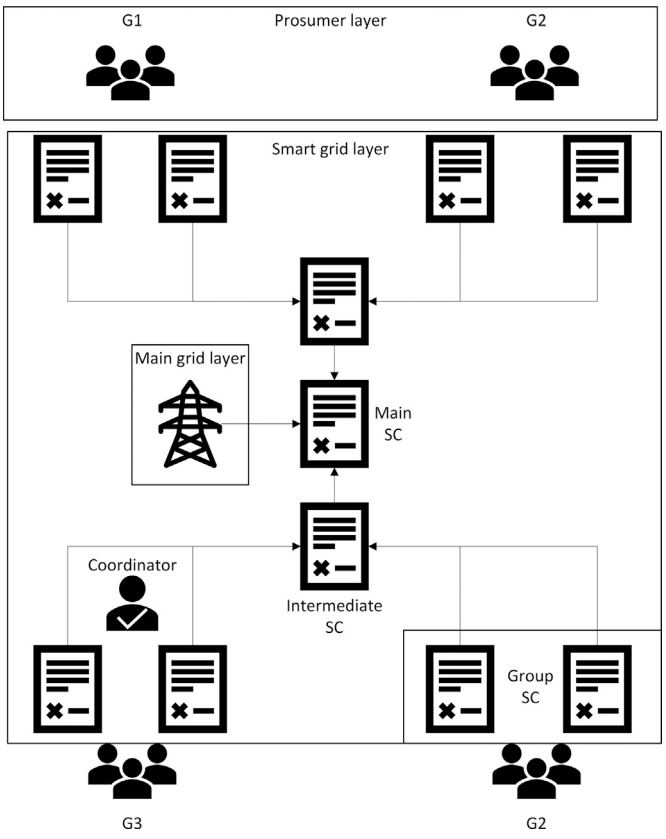


Fig. 1. Architecture scheme.

- *Operators* (O_i): Service entities responsible for registering and authorizing prosumers, maintaining network connectivity, and managing access permissions to participate in the SG. Operators are trusted agents acting on behalf of the DSO and are required to follow a strict registration process to ensure that prosumers receive only the necessary credentials and accounts.
- *Hierarchical structure of smart contracts*: Responsible for managing decentralized energy trading and communication among prosumer groups and the MG. It consists of three levels: the main smart contract, intermediate smart contracts, and group smart contracts.

3.3. Functional requirements

The envisaged energy trading system should meet a set of functional requirements that ensure its viability and efficiency within decentralized energy distribution networks. The proposed system will be evaluated according to the following key requirements:

- *Integration with existing IoT devices (F1)*: The system must seamlessly integrate with existing IoT devices used within smart grid infrastructures to ensure compatibility and operability without significant modifications.
- *Scalability (F2)*: The system must demonstrate the ability to scale effectively, accommodating an increasing number of blockchain transactions and smart contracts without a significant drop in performance. Scalability will be evaluated based on two key metrics: (i) blockchain gas consumption and (ii) response time.

3.4. Security and privacy requirements

To ensure a robust and trustworthy energy trading system within decentralized energy distribution networks, the proposed solution must adhere to the following security and privacy requirements:

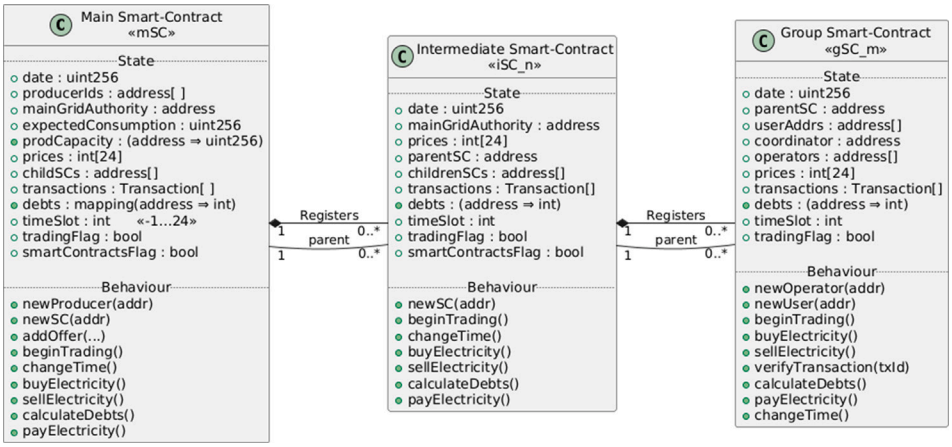


Fig. 2. Smart contract classes diagram.

- *Anonymity of honest users (S1)*: The system must preserve the anonymity of honest users participating in the energy market. By protecting user identities, the system safeguards user privacy against unauthorized profiling or tracking, thereby fostering trust and encouraging participation.
- *Penalization of misbehaving users (S2)*: While maintaining user anonymity, the system must also possess the capability to identify and penalize users who engage in dishonest or malicious activities. This capability should rely on accurate and integrity-resistant recording of electricity transactions. By ensuring comprehensive transaction logging and guaranteeing that data cannot be tampered with, the system can enable fair enforcement measures without compromising the privacy of honest participants.

3.5. System overview

The system functions through four key protocols: *Initialization*, *Price Calculation*, *Energy Trading*, and *Debt Liquidation*.

- *Initialization*: This protocol establishes the foundation for the energy trading system by organizing key actors, their roles, and the deployment of a smart contract-based framework. The main grid (MG) registers main grid producers (MGp) to manage energy production and distribution. Coordinators (Co_x) set up local mechanisms for tracking and trading energy within prosumer groups (G_x) using smart contracts. Operators (O_i) register prosumers, install smart meters (SM_i), and provide secure credentials for system participation.
- *Price Calculation*: This protocol determines energy prices by adopting standard practices from existing electricity markets, facilitated through a smart contract-based framework. The protocol is aligned with practices adopted in several countries, including those in the European Union, where the price is determined by the most expensive accepted bid required to satisfy demand [35]. Main grid producers (MGp) participate in a bidding process to set electricity prices for various time slots, considering production costs, market demand, and other factors. Using smart contracts, producers submit their bids and production capacities, which are verified to calculate final prices.
- *Energy Trading*: This protocol coordinates the trading day by managing interactions across all system levels, from the main grid (MG) to individual users (U_k) and their smart meters (SM_k). The process begins with the MG authority initiating the trading day and cascading the start signal through the hierarchy of smart contracts to the group level. Users then determine their electricity usage or production for each time slot, divide it into multiple protected transactions τ', and submit these to randomly selected group smart contracts (gSC_n) within G_x, which store and manage them. Group coordinators (Co_x) verify transactions, while the trading process progresses by updating time slots and propagating energy surplus or deficit data up the hierarchy. At each level, energy transactions are aggregated and reconciled until the main grid consolidates overall consumption and production, interacting with main grid producers (MGp) to adjust supply. At the end of the trading day, debts are calculated and propagated throughout the hierarchy, ensuring a comprehensive settlement of balances across all participants.
- *Debt Liquidation*: This protocol facilitates payment settlement based on each user's electricity consumption or production. After the trading day concludes and the smart contract hierarchy has computed all outstanding balances, users initiate the process by settling their local debts within their assigned contracts. Each smart contract performs the necessary cryptocurrency transactions to reconcile its balance and, if required, interacts with its parent contract to address additional cryptocurrency transactions. This process propagates upward through the hierarchy until the main contract resolves the final balance with the main grid producers (MGp), ensuring precise and efficient debt settlement across the system.

4. Hierarchical structure of smart contracts

This section details the three-level hierarchy of smart contracts employed in the proposed system to manage decentralized energy trading and communication among prosumer groups and the main grid. The structure of this hierarchy is illustrated in Fig. 2.

4.1. Main smart contract

The main smart contract (mSC) serves as the cornerstone of the proposed hierarchical structure, facilitating seamless communication and coordination between the main grid and the smart grid. Its primary purpose is to centralize and manage critical information related to main grid producers (MGp), including their energy production capacities and pricing. By verifying and storing this data, the mSC ensures accurate price calculations for different times of the day and supports efficient transaction management across the network. Additionally, the mSC acts as an intermediary, enabling communication and balancing energy surpluses or deficits between intermediate smart contracts (iSC_n) and the main grid, thereby maintaining the integrity and stability of the decentralized energy trading system.

The structure of the mSC includes the following attributes:

- *Date*: Specifies the trading day.
- *Producer identities*: A list of registered producers' blockchain addresses.

- *Main Grid Authority*: The public key identifying the main grid authority.
- *Expected electricity consumption*: The estimated energy consumption for the trading day.
- *Production capacity*: The energy production capacity reported by the MGp.
- *Prices*: Electricity prices for each time slot.
- *Intermediate smart contracts*: Addresses of child iSC_n instances.
- *Transactions*: A list of electricity transactions, including the amount and price.
- *Debts*: Records of debt for each child iSC.
- *Time Slot*: The current time slot (ranging from −1 to 24).
- *Trading Flag*: Indicates whether trading is active.
- *Smart Contracts Flag*: Indicates whether new iSC can be registered.

Methods for interaction with the mSC include:

- *newProducer()*: Adds a new MGp.
- *newSC()*: Registers a new iSC_n when the *Smart Contracts flag* is set to "True"
- *addOffer()*: Allows MGp to submit offers for energy trading.
- *beginTrading()*: Activates trading by setting the *Trading Flag* to "True".
- *changeTime()*: Advances the *Time Slot* to the next interval.
- *buyElectricity()/sellElectricity()*: Facilitates transactions between iSC_n and MG during trading.
- *calculateDebts()*: Computes and records debts after trading, deactivating trading by setting the *Trading Flag* to "False".
- *payElectricity()*: Enables iSC_n instances to settle debts with producers (P_i).

When a new mSC is deployed, the MG must provide key parameters, including the *Date* of the trading day and the *Expected energy consumption*, which is determined based on anticipated energy demand. Additionally, the MG initializes certain attributes in the mSC with default values: the *Trading Flag* is initialized to "False", indicating that trading is not yet active, and the *Time Slot* is initialized to -1.

4.2. Intermediate smart contracts

Following the deployment of mSC, intermediate smart contracts (iSC_n) are registered with it for electricity pricing and authorization management, ensuring that only valid protected transactions (τ') are recorded.

The structure of each iSC_n includes the following attributes:

- *Date*: Specifies the trading day.
- *Main Grid Authority*: Public key for the iSC_n connector.
- *Prices*: Electricity prices per time slot.
- *Parent SC*: Address of the parent node.
- *Children SC*: Addresses of children iSC_n or group smart contracts (gSC_m).
- *Transactions*: List of electricity transactions, including amount and price.
- *Debts*: Debt records for each child and parent.
- *Time Slot*: The current time slot (ranging from −1 to 24).
- *Trading Flag*: Indicates whether trading is active.
- *Smart Contracts Flag*: Indicates whether new SCs can register with iSC_n.

Methods for interaction with iSC_n include:

- *newSC()*: Registers a new iSC_n or gSC_m with MG when the *Smart Contracts Flag* is "True".
- *beginTrading()*: Starts the trading day.
- *changeTime()*: Advances the *Time Slot* to the next interval and calculates the energy balance for the previous slot.
- *buyElectricity()*, *sellElectricity()*: Handle energy transactions during active trading.
- *calculateDebts()*: Computes debts after trading activities conclude.
- *payElectricity()*: Settles debts and distributes funds accordingly.

When a new iSC_n is deployed, it requires the *Date* and *Parent SC Address*. By default, the *Trading Flag* is set to "False" and the *Time Slot* is initialized to -1.

4.3. Group smart contracts

After deploying the iSC_n, the next step is to deploy the group smart contracts (gSC_m). Each gSC_m is linked to a specific group G_x and is responsible for tracking the energy usage and pricing for a subset of users within that group. These smart contracts prioritize localized electricity trading within the group. To further enhance user privacy, each group G_x can have multiple gSC_m contracts, distributing user electricity usage data across different records and thereby reducing traceability.

The structure of each gSC_m includes the following attributes:

- *Date*: Specifies the trading day.
- *Parent SC address*: The address of the iSC_n with which the gSC_m is registered.
- *User addresses*: The addresses of registered users authorized to trade electricity.
- *Coordinator*: The public key of the group's coordinator (Co_x). Each coordinator uses a unique public key for every gSC_m they manage.
- *Operators*: Public keys of the O_i entities authorized to register new users.
- *Prices*: Electricity prices for each time slot.
- *Transactions*: A record of protected electricity transactions τ', each including quantities, prices, and a *Verified* status. The *Verified* status is initially set to "False" when a user uploads the transaction and is updated during validation.
- *Debts*: Records of outstanding debts for each user.
- *Time Slot*: The current time slot (ranging from −1 to 24).
- *Trading Flag*: Specifies whether trading is currently active.

Methods for interaction with each gSC_m include:

- *newOperator()*: Used by Co_x to add a new operator O_i.
- *newUser()*: Allows an O_i to register a new user authorized to trade electricity via a group signature scheme. This scheme ensures that external observers cannot link a registration to a specific operator, preserving anonymity.
- *beginTrading()*: Invoked by iSC_n to initiate the trading day. This method sets the *Trading Flag* to "True" and updates the electricity prices for the group.
- *buyElectricity()*, *sellElectricity()*: Enables users to record their electricity consumption or production during trading, storing transaction details such as amounts and prices in *Transactions*.
- *verifyTransaction()*: Used by Co_x to validate user-submitted transactions, ensuring their integrity by setting the *Verified* status of each transaction to "True". The validation process follows existing methods employed by electricity companies to detect anomalies in smart meter readings. These methods are beyond the scope of this work.
- *calculateDebts()*: Invoked by iSC_n to calculate outstanding debts after trading activities have concluded.
- *payElectricity()*: Allows users to settle their debts and distribute funds accordingly among other users or iSC_n.
- *changeTime()*: Utilized by Co_x to advance the *Time Slot* to the next interval and communicate the group's energy surplus or deficit to iSC_n.

When deploying a new gSC_m, Co_x must provide the *Date* and the *Parent Smart Contract Address*. By default, the *Trading Flag* is initialized to "False" and the *Time Slot* to −1.

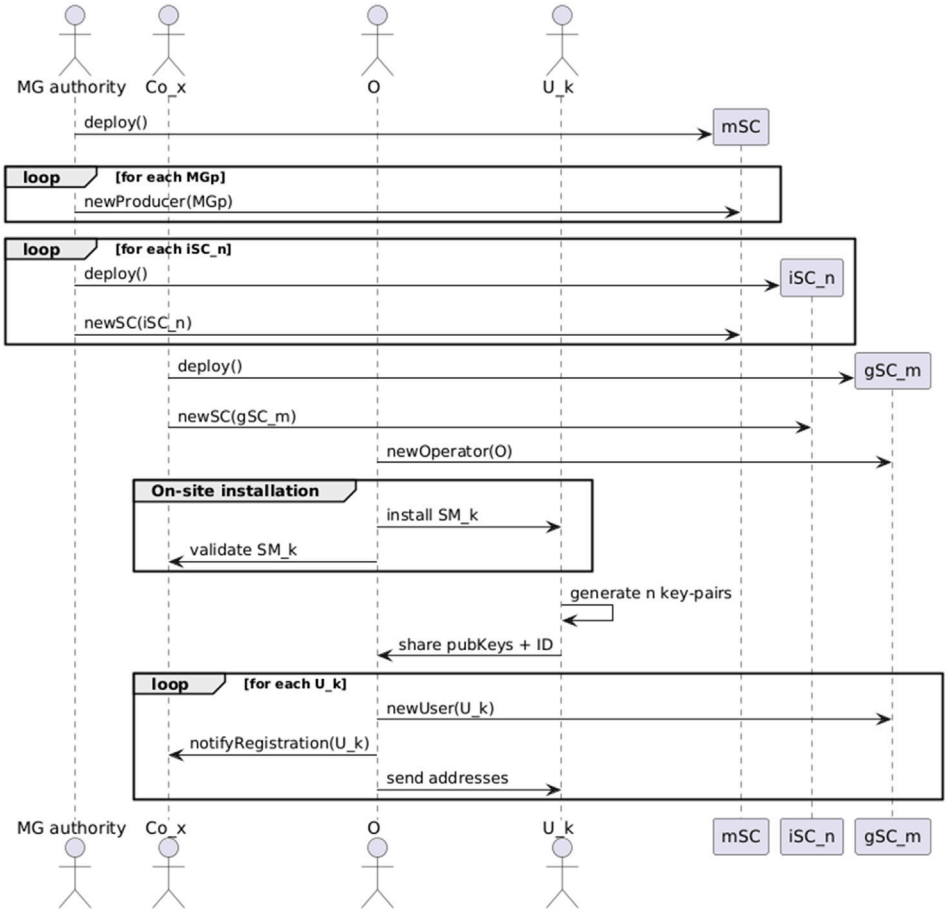


Fig. 3. Initialization of hierarchical smart contract sequence diagram.

5. Protocols

This section outlines the formal protocols that govern the proposed system: *Initialization*, *Price Calculation*, *Energy Trading*, and *Debt Liquidation*.

5.1. Initialization

This protocol establishes the smart contract infrastructure necessary for managing electricity transactions. The process is illustrated in the sequence diagram in Fig. 3, and its main steps are as follows:

1. The MG authority executes the following actions:
   (a) Deploys the main smart contract mSC.
   (b) Registers each MGp with mSC by invoking *newProducer()*.
   (c) Deploys all intermediate smart contracts iSC_n.
   (d) Uses *newSC()* to link each child SC to its parent SC by communicating the respective addresses.
2. Each Co_x for group G_x performs the following actions:
   (a) Deploys the group smart contracts gSC_m associated with G_x.
   (b) Registers each gSC_m with the corresponding iSC_n using *newSC()*.
   (c) Registers each O with the respective gSC_m using *newOperator()*.
3. Each O performs the following steps for each U_k:
   (a) Installs a sealed smart meter SM_k at U_k's domicile.
   (b) Connects SM_k to U_k's electricity generation and consumption line.
   (c) Establishes a connection between SM_k and Co_x for validation.
4. Each U_k performs the following actions:
   (a) Receives a blockchain account and the corresponding private key.
   (b) Generates n key pairs/addresses using a Key Derivation Function.
   (c) Shares the generated public keys and personal identification with O.
5. Each O performs the following steps for each U_k:
   (a) Registers U_k into d group smart contracts gSC_m using *newUser()*.
   (b) Notifies Co_x about U_k's registration.
   (c) Provides U_k with the addresses of all relevant smart contracts in the hierarchy, including gSC_m, iSC_n, and mSC.

5.2. Price calculation

This protocol establishes energy prices following standard practices in existing electricity markets, as detailed in [35]. The key innovation is the use of mSC for interactions between the different MGp and the system, replacing conventional centralized entities. This approach enhances transparency and auditability while remaining compatible with traditional market mechanisms.

The steps of the price calculation mechanism are as follows:

1. Each MGp determines electricity prices for various time slots through a bidding process. These prices are calculated based on production costs, market demand, and other economic factors.
2. Each MGp submits its bids and production capacities directly to mSC via the *addOffer()* method.
3. The mSC verifies the bids and calculates the final prices for each time slot.

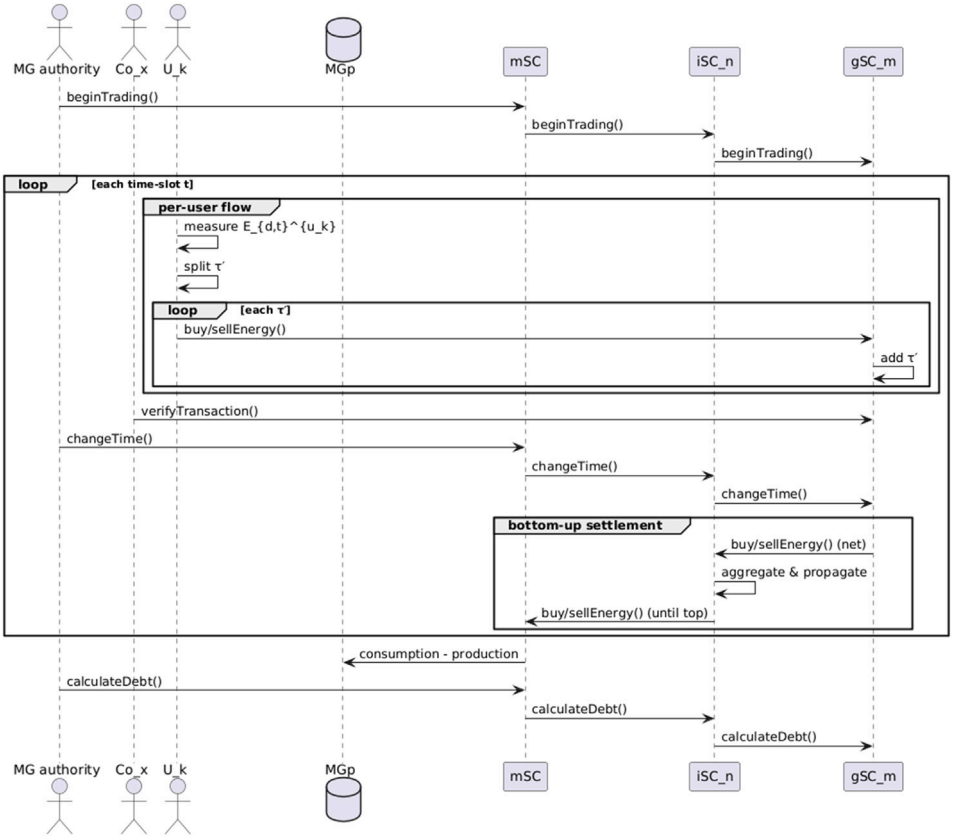


Fig. 4. Energy trading sequence diagram.

5.3. Energy trading

This protocol, initiated by the MG authority, coordinates the trading day by enabling communication between mSC, iSC_n, gSC_m, and U_k/SM_k. The corresponding steps, which include trading initiation, transaction management, and verification, are illustrated in the sequence diagram in Fig. 4 and are described below:

1. The MG authority invokes the *beginTrading()* method in mSC to initiate the trading day.
2. The mSC triggers the *beginTrading()* methods of all its iSC_n children.
3. Each iSC_n recursively calls the *beginTrading()* methods of its subordinate smart contracts until reaching the group smart contracts gSC_m at the lowest level of the hierarchy.
4. Each U_k/SM_k, for every time slot, carries out the following actions:
   (a) Determines its positive (consumption) or negative (production) electricity usage, $E_{d,t}^{u_k}$.
   (b) Splits $E_{d,t}^{u_k}$ into m protected transactions τ' randomly.
   (c) For each transaction, selects a gSC_m linked to its group G_x at random and invokes the appropriate *buyEnergy()* or *sellEnergy()* method.
   (d) The selected gSC_m adds the unverified transaction to its *Transactions* list, including the current price.
5. Each Co_x, for every time slot, invokes the *verifyTransaction()* method in each gSC_m associated with G_x to verify the stored transactions
6. The MG authority invokes the *changeTime()* method in mSC to progress the trading process.
7. The mSC triggers the *changeTime()* methods of all its iSC_n child contracts.
8. Each iSC_n recursively calls the *changeTime()* methods of its subordinate smart contracts, propagating the action until the group smart contracts gSC_m at the lowest hierarchy level are reached.
9. Each gSC_m evaluates its validated transactions, determines the resulting positive (consumption) or negative (production) electricity usage, and invokes the corresponding *buyEnergy()* or *sellEnergy()* method in its parent iSC_n.
10. Each iSC_n aggregates the positive or negative electricity usage from its child contracts and invokes the appropriate *buyEnergy()* or *sellEnergy()* method in its own parent iSC_n. This process continues recursively until the mSC at the top of the hierarchy is reached.
11. The mSC interacts with the MGp to reconcile the overall positive (consumption) or negative (production) electricity usage across the network.
12. At the end of the trading day, the MG authority calls the *calculateDebt()* method in the mSC to finalize the trading process and compute outstanding balances. This call propagates through the hierarchy of smart contracts, with each smart contract at higher levels invoking the *calculateDebt()* methods of its child contracts. The process continues recursively until the *calculateDebt()* methods of the group smart contracts gSC_m at the bottom of the hierarchy are executed, ensuring that all levels compute and record outstanding balances accurately.

5.4. Debt liquidation

This protocol finalizes payments based on the electricity consumption or production of each U_k/SM_k. It is executed after the trading day ends and the entire smart contract hierarchy has completed the *calculateDebt()* methods. This ensures that all smart contracts and U_k/SM_k have precise information about the cryptocurrency amounts they need to pay or receive.

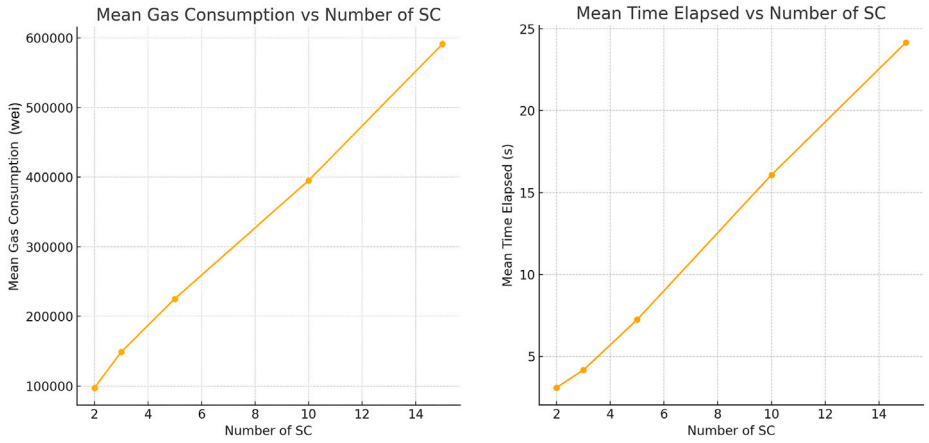


Fig. 5. Gas consumption and time elapsed per time slot operation.

The steps of the debt liquidation mechanism are as follows:

1. Each U_k/SM_k invokes the *payElectricity()* method in their assigned gSC_m.
2. Upon invocation, each gSC_m conducts the cryptocurrency transactions required to settle its local balance, calculated in the prior step. If additional electricity was sourced from a parent iSC_n, the gSC_m invokes the *payElectricity()* method in the parent contract.
3. Each iSC_n, triggered by a child gSC_m, executes its own cryptocurrency transactions to settle its balance.
4. If required, iSC_n recursively calls the *payElectricity()* methods of its parent contracts, propagating upward until the mSC settles the final debt with the MGp.

6. Functional evaluation

This section evaluates the proposed system's functional requirements, as outlined in Section 3.3, focusing on feasibility (F1) and scalability (F2).

Feasibility is assessed through a small-scale implementation using actual IoT devices, replicating a real-world environment. This practical evaluation highlights the system's compatibility with existing IoT infrastructures and its ability to function without significant modifications.

Scalability is examined through theoretical modeling and experimental testing, focusing on two key metrics: blockchain gas consumption and response time. Blockchain gas consumption is a particularly relevant indicator in blockchain-based systems, as it reflects the computational cost of transactions and directly affects fees, scalability, and user experience. Together, these metrics provide a comprehensive understanding of the system's capacity to efficiently manage increasing volumes of transactions and smart contracts.

The following sections detail the test scenario used to evaluate these functional requirements and present the insights gained from the study.

6.1. Test scenario and methodology

The following considerations were taken into account when designing the evaluation testbed:

- *Distributed Ledger*: The open-source distributed ledger IOTA[1] was selected for the test environment due to its low computational requirements and suitability for IoT applications.
- *Dataset*: The dataset used for the evaluation is sourced from [36] and includes a variety of electricity user profiles over an entire year. Specifically, it contains records of electricity consumption and production in 15-min intervals for 51 users, encompassing a mix of households and a public building. Notably, some households are equipped with solar panels, adding variability to the consumption and production patterns.
- *Implementation*: The test environment follows a three-layer architecture comprising user nodes, distribution nodes, and a master node. The user and distribution nodes were deployed on seven Raspberry Pi 3 devices with 1.2 GHz quad-core ARM Cortex-A53 CPUs, 1GB RAM, and Raspbian OS, powered by AC supply. The master node was hosted on a desktop computer with greater computational capacity. Components communicated via a Local Area Network (LAN) with predefined IP addresses and ports, using WiFi for message exchange.
- *Methodology:* To evaluate the network's ability to handle parallel transactions from smart meters, tests were conducted under controlled conditions using the laboratory's Local Area Network (LAN), which features standard Ethernet connections with a nominal bandwidth of 1 Gbps. Tests were repeated multiple times at different hours and on different days to capture typical variations in background traffic and concurrent network activity. The reported results correspond to the average values obtained across these runs, providing a robust assessment of the network's capacity to support parallel smart meter transactions.
- *Number of Smart Contracts per Group*: This parameter is critical for enhancing system privacy. Increasing the number of smart contracts per group improves privacy but may also affect performance, particularly in terms of the computational cost of executing smart contract operations (i.e., blockchain gas consumption) and system response time. To evaluate this trade-off, tests were conducted with configurations ranging from 2 to 15 smart contracts per group.

6.2. Results

Fig. 5 shows that both gas consumption and response time increase linearly with the number of smart contracts per smart meter. However, response time remains within acceptable limits: Even with 15 smart contracts per smart meter, it stays under 25 s, which is well below the 15-min time slot duration. This ensures that IoT devices can process transactions without delays.

Gas consumption is a key factor for large-scale deployments, as it impacts cost and system sustainability. The IOTA EVM ledger mitigates this concern by operating on a fee-less model, keeping transaction costs low. The most common transaction type (*sell/buy* electricity) costs approximately $0.00019 at current IOTA prices. For a user participating in 20 smart contracts with 15-min time slots, the total daily cost would be $0.36, which remains a manageable expense.

[1] IOTA, https://www.iota.org/

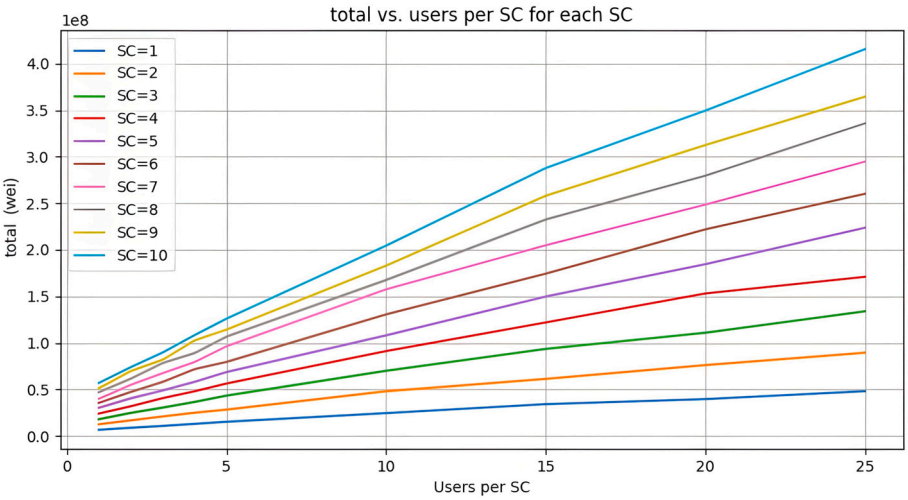


Fig. 6. Total gas consumption of all three operational stages.

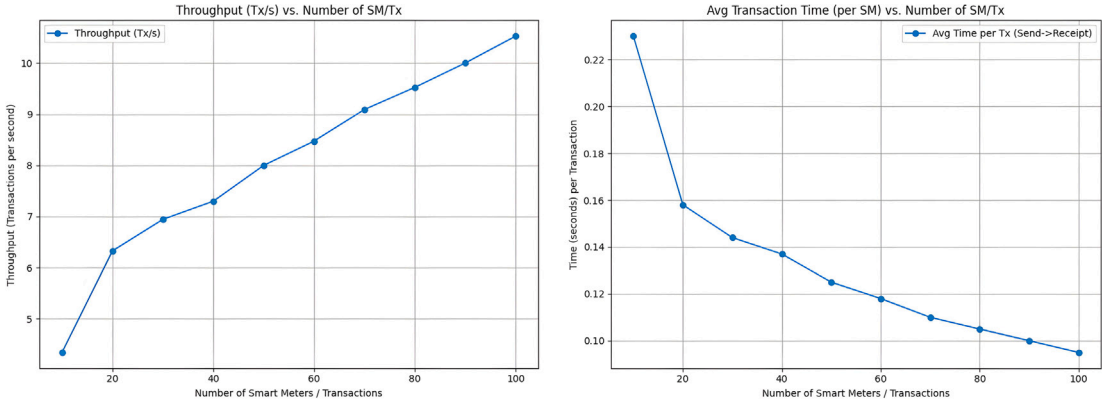


Fig. 7. Throughput and time per transaction.

To assess the overall cost of executing smart contracts, we measured the total gas consumption across all three operational stages. As shown in Fig. 6, gas consumption increases linearly with the number of users per group, with a more significant rise at higher smart contract counts. Since each contract requires repeated operations and deployment, overall gas usage can be controlled by adjusting the number of smart contracts per group while keeping the number of users fixed.

Finally, beyond gas consumption, we evaluated the network's ability to handle parallel transactions from smart meters. Fig. 7 shows that transaction latency remains stable at approximately 0.1 s, even as the number of simultaneous *sell/buy* transactions increases. Throughput scales accordingly, reaching around 10 transactions per second with 100 parallel transactions, confirming the system's scalability. Prior research [37] supports these findings, highlighting the high block confirmation rate of the IOTA EVM (testnet), which is critical for smart grid applications.

For large-scale deployments, additional capacity may be needed. The IOTA ledger addresses this by supporting multiple layer-2 chains,[2] allowing for parallel execution. Instead of a single set of smart contracts for all grid users, layer-2 chains can distribute the load across smaller user groups (e.g., local communities or cities), improving efficiency and scalability.

7. Security and privacy evaluation

This section evaluates the security and privacy properties of the proposed system, with a focus on the two core requirements defined in Section 3.4: preventing unauthorized user profiling (S1) and punishing misbehaving users (S2). To enable a broader analysis, the section begins by defining the system's attacker model, which outlines the capabilities and threat potential of different actors. This model serves as a foundation for identifying relevant attack vectors and for assessing how the system satisfies the S1 and S2 requirements.

7.1. Attacker model

The attacker model includes both *external adversaries*, who lack valid credentials, and *internal adversaries*, who are legitimate users of the system. In line with standard cryptographic assumptions, adversaries are considered to be polynomially bounded and are assumed incapable of breaking hash collision resistance, forging digital signatures, or inverting pseudonymisation mechanisms.

Coordinators and *operators* are treated as fully trusted entities. Coordinators function as distribution system operators (DSOs) and inherently possess complete visibility over the energy flows within their infrastructure, eliminating incentives to subvert the protocol for additional information. Operators are responsible for registering prosumers and assigning accounts for participation in smart contracts. They are assumed to follow the registration process correctly and to refrain from creating fraudulent or duplicate identities.

[2] https://wiki.iota.org/isc/introduction/

Smart contract logic is assumed to be immutable and publicly auditable once deployed, guaranteeing transparency and resistance to tampering.

Based on this model, potential attacks can be classified into two broad categories:

- *Active attacks*: These involve modifying, injecting, or disrupting valid messages and protocol steps. They primarily threaten integrity and availability, and are typically mounted by internal adversaries:
  - *Message tampering*: Attempting to alter a prosumer's off-chain payload (energy amount, price, or time slot) before commitment. Given a secure blockchain infrastructure, this is prevented by the integrity guarantees of the consensus layer.
  - *Replay attacks*: Reusing a previously executed transaction to perform multiple unauthorized trades. This is mitigated by smart contract logic, which disallows the execution of already processed transactions.
  - *Sybil attacks*: Attempting to obtain multiple fake identities to manipulate the protocol. As operators are trusted and enforce strict registration, adversaries cannot obtain unauthorized accounts.
  - *Denial-of-Service (DoS)*: Flooding the network with false transactions to degrade system performance. While possible in principle, the use of a permissioned or scalable distributed ledger makes large-scale DoS attacks impractical.
  - *Fake data injection*: Submitting forged consumption or production records in an attempt to gain unfair economic advantage or distort the system's operation. As this constitutes a form of user misbehavior, it is addressed in detail in Section 7.3, where mechanisms for detection and punishment are described.
- *Passive attacks*: These involve observing network traffic or inspecting the public ledger without interfering with protocol execution. The primary goal is to compromise confidentiality and unlinkability:
  - *Ledger analysis*: Monitoring on-chain transaction data to correlate blockchain accounts and reconstruct user consumption or production patterns. This threat, central to profiling risks, is examined in depth in Section 7.2.

This attacker model provides the foundation for evaluating the system's ability to meet its two primary security requirements. The profiling threats posed by passive adversaries directly relate to requirement S1, while the active attacks associated with dishonest behavior are addressed under requirement S2. The following subsections build upon this model to demonstrate how the proposed system mitigates these threats and upholds the intended privacy and integrity guarantees.

7.2. Preventing unauthorized user profiling

As stated in the definition of electricity consumption (see Definition 3 in Section 3.1), the actual consumption value of user u_k can only be determined by summing the electricity consumption across all their blockchain addresses. These addresses are used in different group smart contracts SC_{G_n} associated with the group G_x to which u_k belongs.

An attacker outside the group G_x (i.e., an *external* attacker) cannot link two group smart contracts, gSC_i and gSC_j, from G_x without the collaboration of the coordinator Co_x. This is because an external attacker can only obtain the following information from a gSC_n:

- *SC address*: Each gSC_n has a unique address, independent of any relationship with other contracts in the group.
- Co_x *address*: Coordinators use unique addresses for every gSC_m they manage. Therefore, this data cannot be used to link gSC_i and gSC_j.
- *User addresses*: Each gSC_n contains the addresses of its registered users, which are unique within that specific smart contract. These addresses do not have a direct relationship with those used in other gSC_n contracts.
- *Users' protected transactions* τ': The electricity consumption value $EC_{d,t}^{\alpha^{U_k}}$ in each τ' appears random and does not have a direct correlation with consumption values in other gSC_n contracts or within the same gSC_n.
- *Number of users*: Different gSC_n contracts within the same group may have varying numbers of registered users.

Since external attackers cannot link two group smart contracts from the same group, they will not be able to link u_k's addresses either, making profiling impractical.

For an attacker who belongs to the group G_x (i.e., an *internal* attacker), knowledge of z different gSC_n within G_x can be exploited to infer user identities. The attacker may attempt to link shared addresses across gSC_n by selecting at most one address per contract while avoiding duplicate selections for a given u_k. In this scenario, the total number of possible combinations is $|gSC|^{z-1}$.

This number can be reduced by focusing on specific time slots t during analysis. For example, excluding periods of low solar generation can significantly narrow the range of possible combinations. However, increasing the number of group smart contracts and users enhances privacy, as both factors raise the difficulty of user re-identification

The theoretical findings discussed above are validated through a simulated attack on privacy-protected electricity consumption data. This analysis focuses on a subset of 50 users observed over a full year, using data from [36]. To identify distinct consumption profiles within the dataset, the k-means clustering algorithm is applied to users' daily electricity patterns. This method partitions the data into k clusters by minimizing within-cluster variance, assigning each user a profile label γ, as described in Section 3.1.

The daily electricity pattern EP^{u_k} of each user is privacy-protected through a randomization method. Specifically, each user's EP^{u_k} is distributed across multiple segments by multiplying it with a randomized matrix containing values between -1 and 1. Each column of this matrix is normalized to sum exactly to one, ensuring that recombining the segments perfectly reconstructs the original consumption pattern. This approach prevents direct identification, as no single segment fully reveals the user's daily consumption, thereby significantly reducing the risk of re-identification. Fig. 8 shows the real and anonymized electricity consumption and production values of a prosumer with photovoltaic (PV) generation from the dataset, distributed across five smart contract transactions.

The simulated adversarial attack models an attacker attempting to re-link these protected, randomized data segments to their original user profiles. The attacker employs a pre-trained k-means clustering model, initially trained on actual user electricity consumption, to classify users into specific consumption profiles. During the attack, the model classifies each randomized segment of EP^{u_k}. The success of the attack is evaluated based on whether the model can correctly assign anonymized segments to their original profiles, thereby assessing the effectiveness of the privacy protection measures.

To evaluate the robustness of the privacy-preserving protocol, two key parameters are systematically varied:

- *Number of Profiles* ($|\Gamma|$)*:* The number of clusters is adjusted from 2 to 50 to analyze its effect on the adversary's success rate in inferring a user's profile.
- *Number of Smart Contracts:* Daily electricity patterns are split into multiple segments, each linked to a different smart contract. The number of smart contracts is varied from 1 to 50 to determine the minimum obfuscation level needed to counter the adversary.

Fig. 9 presents the simulation results, showing the adversary's accuracy in identifying user profiles under different configurations. The x-axis represents the number of clusters, $|\Gamma|$, used in the k-means algorithm, while the y-axis indicates the proportion of correct adversary predictions. Each solid line corresponds to a different number of smart

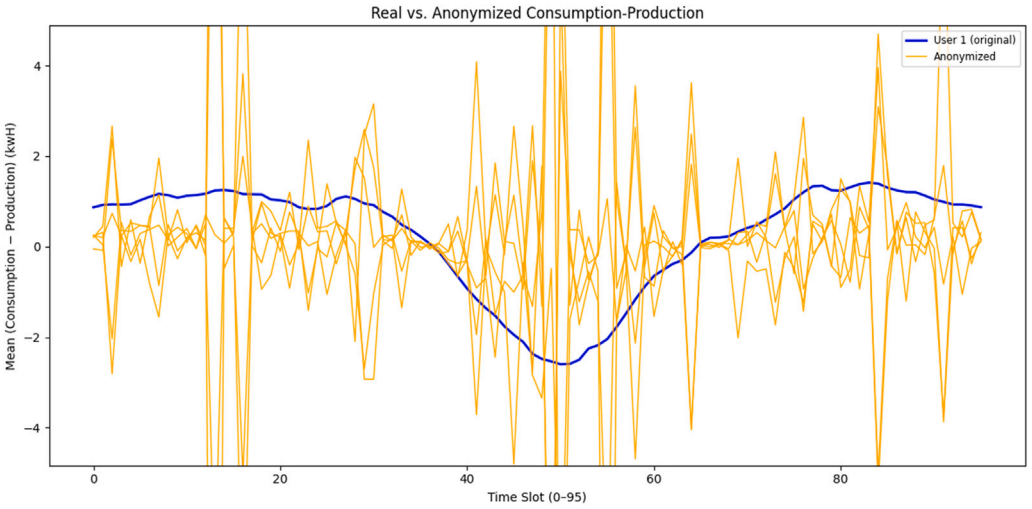


Fig. 8. Real and anonymized electricity consumption and production values for a prosumer.

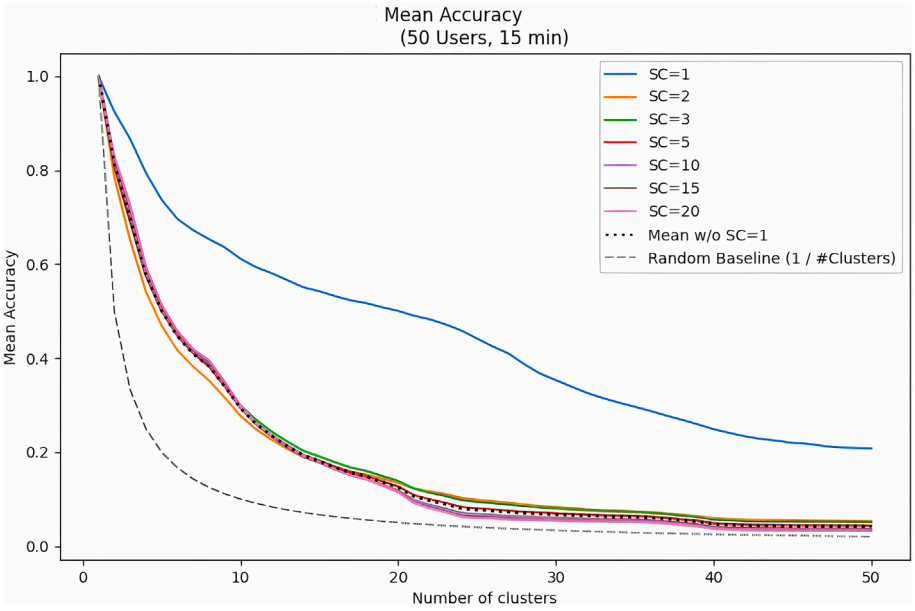


Fig. 9. Adversary accuracy for different numbers of clusters and smart contracts. The dashed line represents the theoretical baseline of $\frac{1}{|\Gamma|}$.

contracts per group, and the dashed line represents the expected success rate of $\frac{1}{|\Gamma|}$ under random guessing. The accuracy values observed are low (tending towards 0.1 or lower), indicating that the adversary's ability to correctly match anonymized segments to original profiles is only marginally better than random guessing, particularly as the number of clusters and smart contracts increases. This demonstrates the effectiveness of the proposed randomization method in maintaining privacy against profile reconstruction attempts.

The simulation results demonstrate the effectiveness of using multiple smart contracts to enhance privacy protection. The adversary's ability to correctly infer user profiles is highly dependent on the number of smart contracts used per group. When a limited number of smart contracts is employed, particularly a single contract per group, the adversary's accuracy remains significantly above the baseline, indicating poor privacy protection. Notably, all the blockchain-based billing methods analyzed in the current state-of-the-art employ a single smart contract per group and directly upload real consumption values, which, as illustrated in the figure, leads to a substantially increased risk of re-identification.

The proposed system, with a higher number of smart contracts, causes the adversary's accuracy to decline, approaching the theoretical probability of random guessing. These findings highlight the importance of distributing electricity profiles across multiple smart contracts to prevent accurate re-identification.

The results also highlight the impact of the number of clusters on the adversary's ability to identify user profiles. As the number of clusters increases, the complexity of classification grows, making it inherently more difficult for the adversary to correctly infer user identities. This is reflected in a noticeable decline in identification accuracy. In a more realistic scenario with a larger dataset and a greater number of users, the number of clusters would naturally increase, further reducing the adversary's success rate. This suggests that increasing the granularity of user profiles serves as an additional layer of privacy protection, making re-identification significantly more challenging.

Table 2
Side-by-side comparison between the proposed solution and related work.

Solution	Distributed approach	Public blockchain	No proof of work	Profiling resistance	Dishonest user detection
[15]	✓				
[19,23–25]	✓		✓		
[10,12]	✓	✓			
[16]	✓			✓	
[14]	✓				✓
[13]	✓	✓		✓	
[10,18,20]	✓	✓	✓		
[16]	✓	✓	✓		✓
[16,28–32]	✓	✓	✓	✓	
Our proposal	✓	✓	✓	✓	✓

The results confirm that multiple smart contracts effectively obfuscate user profiles but introduce a trade-off between privacy and computational cost. As shown in Section 6, increasing the number of smart contracts per group reduces the adversary's success rate but also leads to a linear increase in blockchain gas consumption and response time.

Balancing privacy protection and system efficiency is essential. A higher number of smart contracts improves privacy by lowering adversary accuracy towards the baseline ($\frac{1}{|\Gamma|}$) but also adds computational overhead. The evaluation suggests that using around 10 smart contracts per group provides strong privacy protection while keeping computational costs manageable, as demonstrated in Section 6.

7.3. Punishing misbehaving users

The proposed ledger-based system incorporates mechanisms that ensure accountability and enable the identification and punishment of misbehaving users.

Firstly, the system guarantees data integrity, non-repudiation, and access control through a secure distributed ledger, cryptographic hash functions, digital signatures, and consensus mechanisms. Unauthorized modifications are prevented, ensuring that all transactions remain immutable and verifiable. Digital signatures confirm the legitimacy of messages, preventing users from tampering with records or denying their past actions. Furthermore, smart contracts enforce predefined rules to restrict access to authorized users, ensuring that only legitimate actions are processed. These properties make it possible to detect anomalies and prevent fraudulent activities such as unauthorized modifications or false claims regarding electricity consumption or production.

Additionally, the system includes fraud detection mechanisms to identify and respond to users who attempt to manipulate electricity records. Coordinators (Co_x) oversee all transactions within their respective groups, verifying that no fraudulent activity, such as double spending or false energy reporting, occurs. By monitoring all transactions associated with a particular user, coordinators can detect inconsistencies and flag suspicious behavior. Once fraudulent actions are identified, the system allows for appropriate penalties, ensuring that misbehaving users are held accountable.

By combining these integrity, verification, and accountability mechanisms, the proposed system effectively detects and punishes misbehaving users, preventing energy fraud and reinforcing trust in the marketplace.

8. Comparative evaluation and SWOT analysis

This section provides an extended evaluation of the proposed system through two complementary dimensions. First, it presents a comparative analysis against selected state-of-the-art solutions. Then, it introduces a SWOT analysis to reflect on the system's main strengths and limitations.

8.1. Comparison with state-of-the-art solutions

This subsection compares the proposed system with state-of-the-art approaches discussed in Section 1.1. The comparison is based on five key properties derived from the functional and security requirements outlined in Sections 1.1, 3.3, and 3.4: (i) Distributed approach; (ii) Public blockchain; (iii) No reliance on energy-intensive Proof-of-Work; (iv) Profiling-attack resistance; and (v) Dishonest-user detection.

Table 2 summarizes how these properties are addressed by each solution. A checkmark (✓) indicates that the property is explicitly satisfied. Cells are left blank when the original work does not provide sufficient information to assess the feature. For profiling resistance, a checkmark is included if energy values are hidden from external viewers, even in cases where the authors do not explicitly evaluate the system against profiling attacks.

As shown in Table 2, the proposed system satisfies all five evaluated properties. This is expected, as these properties directly informed the system's design and implementation. Notably, very few existing solutions explicitly address both profiling resistance and dishonest user detection, which are essential for ensuring privacy and integrity in decentralized energy trading environments.

In summary, the comparison shows that the proposed system effectively addresses key functional and security requirements, demonstrating its relevance and improvements over existing approaches.

8.2. SWOT analysis

Based on the functional, security, and privacy evaluations presented in Sections 6 and 7, and the comparative criteria discussed previously, the following SWOT analysis summarizes the strengths, weaknesses, opportunities, and threats associated with the proposed hierarchical smart contract system.

- *Strengths:*
  - Strong privacy protection through hierarchical distribution of electricity data across multiple smart contracts, effectively preventing unauthorized user profiling.
  - Robust fraud detection and accountability mechanisms, enabled by cryptographic verification, digital signatures, and smart contract-enforced rules.
  - Practical feasibility demonstrated through implementation and testing on actual IoT devices with limited computational resources.
  - Good scalability, supported by the fee-less IOTA ledger, which maintains low transaction costs and acceptable response times as the number of transactions increases.
- *Weaknesses:*
  - Experimental validation was limited to a small number of Raspberry Pi devices, which constrains the ability to simulate large-scale deployments.
  - Legal and regulatory barriers may delay or restrict real-world deployment of the system.
- *Opportunities:*
  - Use of the IOTA blockchain, which avoids energy-intensive Proof-of-Work, while still offering the security benefits of a public distributed ledger.

  - Superior privacy protection compared to current state-of-the-art methods, due to the use of randomized electricity consumption segments that reduce re-identification risks.
  - Cost-effectiveness enabled by IOTA's fee-less transaction model, supporting financially viable large-scale deployments.
  - High scalability potential through support for parallel transactions and integration with layer-2 chains, making the system suitable for extensive IoT applications in smart grids.
- *Threats*
  - As the system scales, both gas consumption and gas prices may increase, resulting in reduced economic viability.
  - High upfront costs for installing and maintaining smart meters and IoT infrastructure may hinder large-scale deployment.

9. Conclusions and future work

This work explores the effectiveness of hierarchical smart contracts in protecting user privacy within decentralized energy distribution systems. By structuring smart contracts in a hierarchical manner, the system preserves privacy while maintaining efficient energy trading in smart grids. Building on previous research [33], this study addresses privacy concerns in energy networks reliant on distributed energy resources and IoT devices.

The proposed tree-structured architecture significantly reduces the risk of user profiling by adversaries. Theoretical analysis demonstrated that employing multiple contracts per group lowers the success rate of internal attacks, enhancing privacy protection. Simulation results validated this approach, showing that when more than one smart contract is used per group, adversary accuracy in identifying users decreases. Additionally, the system's computational cost scales linearly with the number of smart contracts, remaining within acceptable limits, ensuring feasibility for large-scale deployment.

A comparative analysis with selected state-of-the-art solutions was also conducted, focusing on five properties: decentralization, use of a public blockchain, energy efficiency, resistance to profiling attacks, and detection of dishonest users. The proposed system satisfies all five, demonstrating its comprehensive approach to privacy and security in decentralized energy trading.

Looking ahead, future work will focus on integrating a dynamic pricing scheme to foster a more competitive and fair market while further reducing the trust required in coordinators. Additionally, a privacy-preserving verifiable credentials system will be explored to enhance security and minimize reliance on trusted coordinators.

CRediT authorship contribution statement

Joan Ferré-Queralt: Writing – original draft, Visualization, Validation, Software, Investigation, Formal analysis. **Jordi Castellà-Roca:** Supervision, Methodology, Funding acquisition, Conceptualization. **Alexandre Viejo:** Writing – review & editing, Supervision, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Alexandre Viejo reports financial support was provided by Spanish Cybersecurity National Institute. Alexandre Viejo reports financial support was provided by Spanish Ministry of Science, Innovation and Universities. Alexandre Viejo reports financial support was provided by Government of Catalonia. Joan Ferre-Queralt reports financial support was provided by Spanish Government. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by Project PID2021-125962OB-C32 "SECURING/DATA" funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe"; by Project HERMES funded by the European Union NextGenerationEU/PRTR via INCIBE; and by Grant SGR2021-00115 funded by AGAUR, Generalitat de Catalunya. The first author is also supported by the Spanish Government under an FPU grant (ref. FPU23/01275).

Appendix A. Smart contracts

This appendix provides high-level pseudocode to support implementation and enhance the reproducibility of the designed smart contracts.

Algorithm 1 Main smart contract (mSC).

```
procedure MainSC_Init(date, expectedConsumption)
    Date ← date
    ProducerList ← {}, iSCList ← {}
    MainGridAuthority ← msg.sender
    ExpectedConsumption ← expectedConsumption
    ProductionCapacity[ ] ← {}, Price[0..24] ← {}
    Transactions ← {}, Debts[ ] ← {}
    timeSlot ← −1, tradingActive ← False
    scRegistrationActive ← True
end procedure
function newProducer(producerAddr)
    if scRegistrationActive then
        ProducerList.add(producerAddr)
    end if
end function
function newSC(iSC_addr)
    if scRegistrationActive then
        iSCList.add(iSC_addr)
    end if
end function
function addOffer(producer, slot, capacity, price)
Require: producer ∈ ProducerList
    ProductionCapacity[producer][slot] ← capacity
    PriceOffers[producer][slot] ← price
end function
procedure beginTrading
    tradingActive ← True
end procedure
procedure changeTime
    timeSlot ← timeSlot + 1
    computeClearingPrice(timeSlot)
end procedure
function buyElectricity(iSC, amount)
Require: tradingActive and timeSlot ≥ 0
    Transactions.add((iSC, amount, Price[timeSlot]))
    Debts[iSC] += amount × Price[timeSlot]
end function
procedure calculateDebts
    tradingActive ← False
    for all iSC in iSCList do
        record Debts[iSC]
    end for
end procedure
function payElectricity(iSC, payment)
Require: payment ≥ Debts[iSC]
    transfer payment to producers
    Debts[iSC] ← Debts[iSC] - payment
end function
```

Algorithm 2 Intermediate smart contract (iSC_n).

```
procedure ISC_INIT(date, parentAddr)
    Date ← date, parentSC ← parentAddr
    MainGridAuthority ← parentAddr.MainGridAuthority
    Children ← {}, Transactions ← {}, Debts ← {}
    Price[0..24] ← {}, timeSlot ← −1
    tradingActive ← False, scRegistrationActive ← True
end procedure
function NEWSC(childAddr)
    if scRegistrationActive then
        Children.add(childAddr)
    end if
end function
procedure BEGINTRADING
    tradingActive ← True
end procedure
procedure CHANGETIME
    timeSlot ← timeSlot + 1
    COMPUTEBALANCE(timeSlot - 1)
end procedure
function BUYELECTRICITY(gSC, amount)
Require: tradingActive
    Transactions.add((gSC, amount, Price[timeSlot]))
    Debts[gSC] += amount × Price[timeSlot]
end function
procedure CALCULATEDEBTS
    tradingActive ← False
    for all child in Children do
        record Debts[child]
    end for
end procedure
function PAYELECTRICITY(child, payment)
Require: payment ≥ Debts[child]
    transfer payment to parentSC
    Debts[child] ← Debts[child] - payment
end function
```

Algorithm 3 Group smart contract (gSC_m).

```
procedure GSC_INIT(date, parentAddr, coordinator)
    Date ← date, parentSC ← parentAddr
    Coordinator ← coordinator
    Operators ← {}, Users ← {}
    Price[0..24] ← {}, Transactions ← {}, Debts ← {}
    timeSlot ← −1, tradingActive ← False
end procedure
function NEWOPERATOR(opAddr)
Require: msg.sender = Coordinator
    Operators.add(opAddr)
end function
function NEWUSER(userAddr)
Require: msg.sender ∈ Operators
    Users.add(userAddr)
end function
procedure BEGINTRADING(priceSchedule)
Require: msg.sender = parentSC
    tradingActive ← True
    Price ← priceSchedule
end procedure
function BUYELECTRICITY(user, amount)
Require: tradingActive and user ∈ Users
    Transactions.add((user, amount, Price[timeSlot],
    verified = False))
end function
procedure VERIFYTRANSACTION(txID)
Require: msg.sender = Coordinator
    Transactions[txID].verified ← True
end procedure
procedure CHANGETIME
Require: msg.sender = Coordinator
    timeSlot ← timeSlot + 1
    report surplus/deficit to parentSC
end procedure
procedure CALCULATEDEBTS
Require: msg.sender = parentSC
    for all user in Users do
        Debts[user] ← sum_txns(user)
    end for
end procedure
function PAYELECTRICITY(user, payment)
Require: payment ≥ Debts[user]
    distribute payment among participants
    Debts[user] ← Debts[user] - payment
end function
```

Data availability

The data used in this study is publicly available in a repository cited within the paper.

References

[1] C.W. Gellings, The smart grid: enabling energy efficiency and demand response, CRC Press, 2009.
[2] X. Fang, S. Misra, G. Xue, D. Yang, Smart grid—the new and improved power grid: a survey, IEEE Commun. Surv. Tutor. 14 (4) (2012) 944–980.
[3] C. Liu, Y. Zhou, Z. Zhao, H. Tian, Y. Zhang, Z. Cai, Cyber security and privacy issues in smart grids, IEEE Commun. Surv. Tutor. 14 (4) (2012) 981–997.
[4] Y. Parag, B.K. Sovacool, Electricity markets design for the prosumer era, Nat. Energy 1 (4) (2016) 16032.
[5] W. Meng, Y. Zheng, V.W.S. Wong, Smart grid neighborhood area networks: a survey, IEEE Netw. 27 (1) (2013) 8–13.
[6] A.R. Metke, R.L. Ekl, Security technology for smart grid networks, in: Proceedings of the IEEE, Vol. 98, IEEE, 2010, pp. 2–5.
[7] M.B. Mollah, J. Zhao, D. Niyato, K.-Y. Lam, X. Zhang, A.M.Y.M. Ghias, L.H. Koh, L. Yang, Blockchain for future smart grid: a comprehensive survey, IEEE Internet. Things J. 8 (1) (2021) 18–43.
[8] I.A. Umoren, S.S.A. Jaffary, M.Z. Shakir, K. Katzis, H. Ahmadi, Blockchain-based energy trading in electric-vehicle-enabled microgrids, IEEE Consum. Electron. Mag. 9 (6) (2020) 66–71.
[9] W. Yang, Z. Guan, L. Wu, X. Du, Z. Lv, M. Guizani, Autonomous and privacy-preserving energy trading based on redactable blockchain in smart grid, in: GLOBECOM 2020–2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–6.
[10] M. Mihaylov, S. Jurado, N. Avellana, K. Van Moffaert, I.M. de Abril, A. Nowé, Nrgcoin: virtual currency for trading of renewable energy in smart grids, in: 11th International Conference on the European Energy Market (EEM14), 2014, pp. 1–6.
[11] S. Nakamoto, Bitcoin whitepaper, 2008. https://bitcoin.org/bitcoin.pdf. (17.07.2019).
[12] A.A.G. Agung, R. Handayani, Blockchain for smart grid, J. King Saud Univ. - Comput. Inf. Sci. 34 (3) (2022) 666–675.
[13] N.Z. Aitzhan, D. Svetinovic, Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams, IEEE Trans. Depend. Sec. Comput. 15 (5) (2018) 840–852.
[14] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, E. Hossain, Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains, IEEE Trans. Ind. Inf. 13 (6) (2017) 3154–3164.
[15] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, C. Weinhardt, A blockchain-based smart grid: towards sustainable local energy markets, Comput. Sci.-Res. Dev. 33 (2018) 207–214.
[16] Blockchain-Based Microgrids Energy Trading, Deal: differentially private auction for blockchain-based microgrids energy trading.
[17] A. de Vries, Bitcoin's energy consumption is underestimated: a market dynamics approach, Energy Res. Soc. Sci. 70 (2020) 101721.
[18] J. Hou, H. Wang, P. Liu, Applying the blockchain technology to promote the development of distributed photovoltaic in China, Int. J. Energy Res. 42 (6) (2018) 2050–2069.

[19] O. Samuel, N. Javaid, A secure blockchain-based demurrage mechanism for energy trading in smart communities, Int. J. Energy Res. 45 (1) (2021) 297–315.
[20] R. Khalid, N. Javaid, A. Almogren, M.U. Javed, S. Javaid, M. Zuair, A blockchain-based load balancing in decentralized hybrid P2P energy trading market in smart grid, IEEE. Access 8 (2020) 47047–47062.
[21] P. Zhuang, T. Zamir, H. Liang, Blockchain for cybersecurity in smart grid: a comprehensive survey, IEEE Trans. Ind. Inf. 17 (1) (2021) 3–19.
[22] A. Laszka, S. Eisele, A. Dubey, G. Karsai, K. Kvaternik, Transax: a blockchain-based decentralized forward-trading energy exchanged for transactive microgrids, in: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2018, pp. 918–927.
[23] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, K. Ren, A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks, IEEE Network 32 (6) (2018) 184–192.
[24] K. Gai, Y. Wu, L. Zhu, L. Xu, Y. Zhang, Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks, IEEE Internet. Things J. 6 (5) (2019) 7992–8004.
[25] Z. Guan, G. Si, X. Zhang, L. Wu, N. Guizani, X. Du, Y. Ma, Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities, IEEE Commun. Mag. 56 (7) (2018) 82–88.
[26] S. Khan, R. Khan, Multiple authorities attribute-based verification mechanism for blockchain mircogrid transactions, Energies 11 (5) (2018) 1154.
[27] Y.-B. Son, J.-H. Im, H.-Y. Kwon, S.-Y. Jeon, M.-K. Lee, Privacy-preserving peer-to-peer energy trading in blockchain-enabled smart grids using functional encryption, Energies 13 (6) (2020) 1321.
[28] D. Hou, J. Zhang, S. Huang, Z. Peng, J. Ma, X. Zhu, Privacy-preserving energy trading using blockchain and zero knowledge proof, in: 2022 IEEE International Conference on Blockchain (Blockchain), IEEE, 2022, pp. 412–418.
[29] A. Mahmood, A. Khan, A. Anjum, C. Maple, G. Jeon, An efficient and privacy-preserving blockchain-based secure data aggregation in smart grids, Sustain. Energy Technol. Assess. 60 (2023) 103414.
[30] C. Hu, Z. Liu, R. Li, P. Hu, T. Xiang, M. Han, Smart contract assisted privacy-preserving data aggregation and management scheme for smart grid, IEEE Trans. Depend. Sec. Comput. (2023).
[31] L. Lei, F. Wang, C. Zhao, L. Xu, Efficient blockchain-based data aggregation scheme with privacy-preserving on the smart grid, IEEE Trans. Smart Grid (2024).
[32] O. Samuel, N. Javaid, Garlichain: a privacy preserving system for smart grid consumers using blockchain, Int. J. Energy Res. 46 (15) (2022) 21643–21659.
[33] J. Ferré-Queralt, J. Castellà-Roca, A. Viejo, Smart contracts for a secure and privacy-preserving smart grid, in: International Conference on Risks and Security of Internet and Systems, Springer, 2023, pp. 103–118.
[34] G. Wood, et al, Ethereum: a secure decentralised generalised transaction ledger, Ethereum Proj. Yellow Pap. 151 (2014) (2014) 1–32.
[35] I. PETCU, How are EU electricity prices formed and why have they soared? 2022. https://www.eurelectric.org/in-detail/electricity_prices_explained/.
[36] C. Goncalves, R. Barreto, P. Faria, L. Gomes, Z. Vale, Dataset of an energy community's generation and consumption with appliance allocation, Data Br. 45 (2022) 108590.
[37] S.T. Muntaha, Q.Z. Ahmed, F.A. Khan, P.I. Lazaridis, Hybrid blockchain-based multi-operator resource sharing and SLA management, IEEE Open J. Commun. Soc. (2024).