Blockchain-based Smart Contracts Application for Energy Trading

Joana Bugalho*
Instituto Superior Técnico
Universidade de Lisboa, Portugal
(Dated: December 15, 2022)

The world is far from achieving the goals set out by the 2030 Agenda for Sustainable Development. The energy sector is now responsible for almost three-quarters of the emissions causing climate change, proving that, to achieve these goals, the implementation of new solutions in this sector is essential.

It is imperative to restructure the energy sector to accommodate a bigger share of sustainable energy sources, more specifically in the residential sector, since it has one of the biggest electricity consumption in the world.

With the shift toward renewable energy, new agents that consume but can also produce - the prosumers - start to have a more important role in the market. For this reason, the traditional centralized structure of the market can no longer support the new challenges. It becomes more crucial to create new models for energy trading, that are decentralized, efficient and running on secure platforms.

This thesis proposes the development of an application, using blockchain technology, that implements smart contracts that automatically validate and audit energy transactions.

Three different smart contracts were implemented, using the Ethereum blockchain: trading exclusively with the Energy Service Company (ESCO), peer-to-peer (P2P) trading according to predefined rules and P2P trading through auctions.

While blockchain-based smart contracts proved to be a viable option for energy trading, the Ethereum blockchain is not the best blockchain to trade small amounts of energy, since the fees applied during the day were much higher compared to the price paid for the energy.

Keywords: Smart Contracts, Blockchain Technology, Decentralized Energy Trading, Transactions, Smart Energy Communities

I. INTRODUCTION

Despite significant progress over the last two decades, the world is still falling short in the 2030 Agenda for Sustainable Development and the 17 Sustainable Development Goals (SDGs) [1], adopted by all United Nations Member States in 2015, which address economic, social, and environmental challenges.

Amidst the increase in energy consumption, the energy sector is now responsible for almost three-quarters of the emissions causing climate change, proving that the energy sector has to be at the heart of the solution to climate change [2].

Given that the residential sector has one of the highest electricity consumption shares, this sector needs to change drastically, focusing primarily on increasing renewable energy generation and consumption, and expand energy efficiency and digitalization.

As the energy market evolves to integrate more sustainable energy sources, new agents that consume, but can also produce energy - the so called prosumers - start to participate in the market. Consequently, the traditional centralized structure solution can no longer support the challenges that the energy market brings today. It becomes crucial to create new business models, decentralized, efficient and running on secure platforms, to support energy interactions within a community, such that the use of sustainable energy becomes more affordable and reliable.

Smart contracts, programs used to automate the execution of an agreement, without the need for a third-party, along with blockchain technology, that stores all the information in a decentralized manner, can be ideal for the interactions that occur within an energy community. All participants would be able to trade energy between them, according to their own preferences. The blockchain can track, within the network, orders, transactions, payments, production, consumption data and much more. And since the participants share a common goal - a more sustainable energy system - new opportunities for the use of renewable energy and efficient consumption are within reach.

There has been a considerable development of studies and initiatives about the use of blockchain in the energy sector, predominantly in Peer-to-Peer (P2P) energy trading communities, since blockchain can connect and coordinate a large number of participants in the same area.

Several papers review the state-of-art of blockchain smart contracts applied to different energy communities [3–6].

Others study different approaches to the use of blockchain-based smart contracts, such as deploying it with an energy authority [7], using private blockchains [8, 9], and even designing entirely new platforms [10].

At the moment, some projects have been implemented in real communities, such as Grid+ [11], Power Ledger [12], and Brooklyn microgrid [13], developed in P2P energy grids.

The overall conclusion of these studies and real cases is that blockchains and smart contracts provide clear benefits to the energy system, markets and participants.

* joana.bugalho@tecnico.ulisboa.pt

II. OBJECTIVES AND OVERVIEW

This project proposes the development of a platform, using a blockchain-based solution, that implements smart contracts that automatically validate and audit energy transitions (see Figure 1), contributing in this way to a more sustainable society. With this system, several activities can be automated: defining electricity costs for specific periods, different payment policies, defining schedules for buying and selling electricity, settlements details, etc.. Apart from contributing to the reduction of carbon emissions and increase in energy efficiency, it may also have significant financial impacts, by shielding the market from fossil fuels market instability. Therefore, compared to the systems in place today, we can increase the efficiency, speed and scalability of the energy markets.

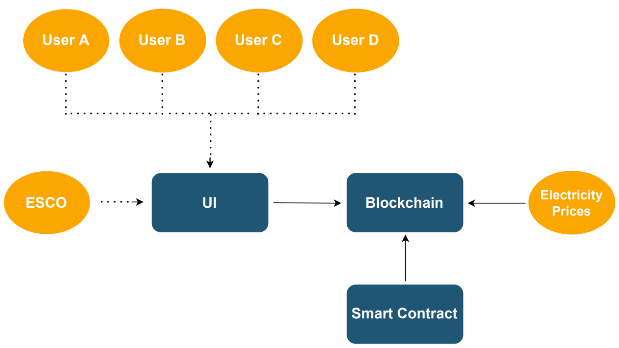

FIG. 1. Proposed conceptual diagram for the solution.

All participants (users) will be connected to the energy trading application through an user interface (UI), where they can interact with each other. An Energy Service Company (ESCO) will also be a part of this community, to guarantee that there is always enough energy inside the community. This ESCO will buy or sell energy to the community when needed, and it will provide all energy-related equipment to assure energy trading operations. Additionally, each user can decide if they wish to use smart meters and a home management system. The platform is then connected to a blockchain, which has a smart contract stored within. Since there is an ESCO in this community, that will acquire energy when needed, the blockchain is also connected to the present electricity costs.

In Figure 2, there is the representation of the architecture for the proposed solution. Users will inform the smart contract if they wish to buy or sell energy, then the smart contract will compute the price at which the electricity will be sold at that moment. The price will be exchanged between buyer and seller, in the form of tokens (cryptocurrency). After the transaction is complete, the electricity is sent through the infrastructure shared between the community.

The design of the smart grid infrastructure is not within the scope of this project, but note that this structure needs to connect all participants in this energy trading community, and it also needs to have devices connected to the application in each participant's house, to account for the flow of electricity.

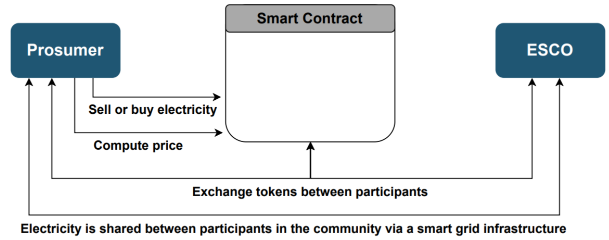

FIG. 2. Architecture of the proposed solution.

III. SMART CONTRACTS

Smart contracts are programs that run when predetermined conditions related to a contract between two agents are met [14]. They are typically used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement. These contracts are usually associated with cryptocurrencies or tokens, as a way to trade solely online.

Smart contracts work by making use of simple "if/when…then…" conditions that are written into software code. When those conditions are met and verified in a piece of information written in the network, the computers execute the action detailed in the contract. In a SC, there can be as many procedures as needed to satisfy the participants' conditions [14]. After the program associated with the activated condition has come to an end, and the final transaction is completed, the system will be updated. Once the overall process has ended, the transaction cannot be changed, and only parties who have permission can see the results [14].

One significant advantage of smart contracts, that distinguishes it from a traditional contract, is that, by setting tamper-proof execution of computer code, the records in the database cannot be modified unilaterally by either the parties involved [15]. Another advantage is linked to the use of smart meters. When we connect smart contracts with smart meter data, energy efficiency may be enhanced by enabling the automatic measurement of energy generated or consumed and automatically adjusting demand and supply, when described in the terms of the contract.

In the case of an energy community, to trade energy in an efficient and fast way, protocols that describe the conditions of the transactions, like the price of energy at that time, must be integrated with information from the network (e.g., smart meter data that describes the amount of energy that was generated, bought or sold) to conduct the energy transaction. These protocols are described in Smart Contracts (SC), which allow participants to trade based on their own preferences and all the available information.

IV. BLOCKCHAIN

A blockchain is a shared ledger (book-of-records) that facilitates the process of recording transactions and tracking assets in a network [16]. As the name indicates, it consists of blocks, that contain information, and are binded to each other, through a chain. In this particular network, the asset is energy, but it can also have other tokens associated with it, something that is exchanged for commodities, in order to make transactions simpler.

The main key concepts behind a blockchain are [17]:

- Distributed ledger technology (DLT): all participants in the network can have access to the ledger and its records of transactions. With this digital ledger, the information is recorded only once, eliminating duplication inside the system, common in traditional ledgers.
- Immutable records: No participant can tamper with a transaction after it has been added to the blockchain. If, for some reason, a record has an error, a new transaction can be added to the blockchain in order to reverse the error, and both transactions are visible to participants.
- Smart contracts: to speed up the system, smart contracts can be stored on the blockchain and executed automatically.
- Permissions: permissions are what protect the network, it ensures that the transactions are authenticated and verifiable. This also grants data protection and privacy.
- Consensus: through consensus algorithms, the network can verify the transactions. There are several consensus mechanisms, discussed ahead.

Blockchains are immutable, decentralized, and saved across several networks. The information is written in blocks, each connected to the one before and after it, via cryptography. Once the writing block process is complete, it becomes almost impossible to tamper with it. The transactions are recorded only once and are visible to all participants in the network. As a result, trust, accountability, and transparency can be expected.

In Figure 3, there is a representation of the structure of a block in the blockchain.

For a blockchain to work properly, all blocks must be checked to validate the information. The blockchain network must be able to work efficiently, even in the presence of dishonest information. For this purpose, consensus algorithms are created in decentralized systems, a common agreement between nodes, that ensures validity and prevents manipulation [3]. The consensus mechanism also assures synchronization between different blockchain nodes and a shared public ledger, and enables network nodes to reach a disputable free agreement without any third-party.

The most common types of consensus algorithms are Proof-of-Work (PoW), Proof-of-Stake (PoS), Proof-of-Authority (PoA) and Practical Byzantine Fault Tolerance (PBFT). When using PoW, the node with the highest computational power usually mines the block, given that is less likely that it will attack the network. In the case of PoS, the algorithm requires a stake, owners can offer their coins as collateral to become validators, and to attack you would need to own the majority of stakes. Unlike PoS, PoA uses the identity as a stake, it only allows approved accounts to validate the information. In this case, the authority must remain uncompromised. Lastly, PBFT provides a Byzantine fault tolerance algorithm, that can be achieved if the loyal nodes have a majority agreement on their strategy.

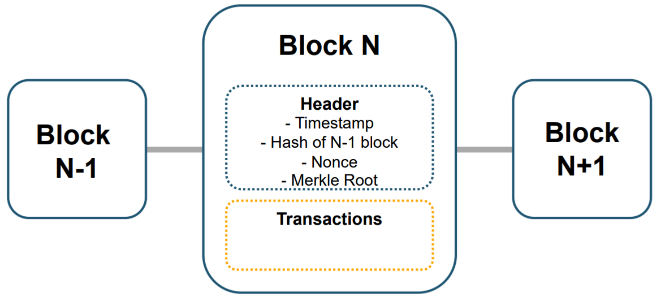

FIG. 3. Basic structure of a block inside the blockchain [18].

All consensus algorithms have different strengths and weaknesses. A comparison between the algorithms stated above is available in [3, 4].

In Figure 4, the process for validating a transaction is explained. First, a transaction must be requested and authenticated. Right after, a block that includes that transaction is created and sent to the network, so that selected nodes can validate the transaction. The validation depends on the consensus algorithm selected. After the validation is completed, the nodes receive an incentive to continue and confirm transactions. Then, the block is added to the blockchain, the updated blockchain is distributed across the network and finally the transaction is completed.

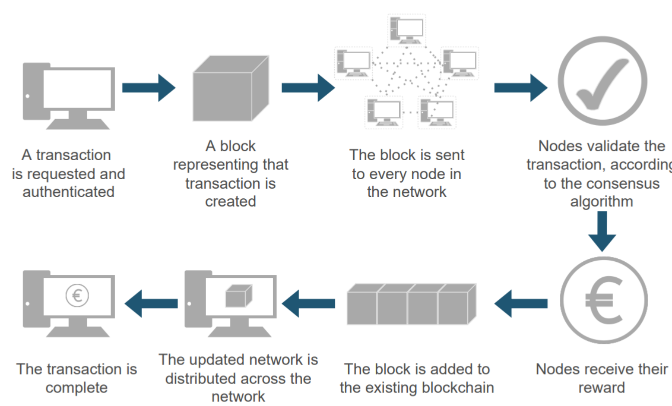

FIG. 4. General process to get a transaction into the blockchain [19].

The incentives paid to the nodes that validate the transactions keep the blockchain secure and decentralized.

V. IMPLEMENTATION

A. Smart Contracts

Smart contracts (SCs) are written in Solidity language, an object-oriented programming language used for implementing smart contracts. The blockchain technology used in the implementation is the Ethereum Blockchain, with ETH (ether) as its cryptocurrency.

Three different smart contracts were designed in this project, each one more complex than the other:

1. Trading exclusively with the ESCO - users insert their energy needs in a database, that is connected to the application. The program would then calculate the price to pay for the amount of energy requested and proceed with the transactions.

2. P2P trading according to generic contract - once again users insert their energy needs in a database, by time intervals. At the end of each interval, the program creates a list of transfers by giving priority to P2P trading and whoever requested energy first, and it would only go to the ESCO when there is no more possible P2P trade. Then, it would calculate the price and go through the transactions.

3. P2P trading through a blind auction - the users can create auctions if they wish to sell energy. Whoever wants to buy, can place bids anonymously. The highest bidder at the end of the auction wins, and the winner gets the full amount of energy that was auctioned. Additionally, it is still possible to go directly to the ESCO to trade energy.

Independent of the model, the smart contract will have the following components:

- Struct TransferStruct – allows for the creation of the transfer data, including the address of the sender and the receiver, the amount of energy requested, the number of ETH transferred and the timestamp. This will define the components that will be added to each block, displayed in the platform.

- Function addToBlockchain – where all the data is added to the blockchain of this smart contract. Inside this function, an event is emitted, i.e., the transfer of ETH is included inside this function.

- Function getAllTransactions – access all transactions done through the smart contract.

- Function getTransactionCount – get the number of transactions done through the smart contract.

After getting the structure of the smart contract, a generic smart contract for the ESCO is planned. The contract will follow the prices of the SPOT electricity market, accessed through an API [20] that is connected to the blockchain (using an oracle), where the electricity price for that day is specified.

The design of the generic smart contract for interactions with the ESCO is the following:

1. The smart contract reads the electricity price for the moment the transaction is meant to begin.

2. If the price is lower than a variable P, then the price that will be paid is the average of the hourly prices for that day.

3. If the price is higher than P, then the price will stay the same.

4. After defining the price, in €/MWh, the SC will calculate the number of ETH to pay for the electricity requested, by multiplying the amount of energy in MWh, the price of the electricity defined in steps 2 or 3 and the exchange rate of € to ETH. It is also applied an energy trading fee of 10% when the user wants to buy energy from the ESCO.

5. The number of ETH to transfer between buyer and seller will then be cut to 8 decimals, to assure that it is a finite number.

The variable P was defined as $P = 150$ €/MWh, which represents the average of the electricity price in the last few months, from June to August of 2022. This type of contract protects the ESCO, in case the electricity prices run much lower than the average. This means that, for the low-price scenarios, the SC will use the daily average for the computation, while for the high-price scenarios, the actual price is used.

The trading fee is only applied when the ESCO sells energy, to make sure that the P2P trading is profitable for all parties, in the next smart contracts.

This smart contract will be implemented in all smart contracts described below.

First Smart Contract

In the first smart contract, only trading with the ESCO was allowed. The participants in the energy trading community can buy energy from the ESCO or sell their energy to the ESCO, following the contract described above.

The application is connected to a database where all data regarding the energy trading is detailed: time, account, and balance. The time indicates at what time the trading is planned to happen, the user can program the energy trading to occur at a time of their preference; the account indicates the address of the user that wishes to do the trading; and the balance defines the amount of energy to trade, given the '+' signal for selling to the ESCO and the '-' for buying from the ESCO.

When the time in the database corresponds to the current time, the program will execute the smart contract for that user. After it calculates the amount of ETH that the sender needs to pay for the requested amount of energy at that time, it starts the process of the transaction. First, it will ask permission from the sender, then the blockchain creates the transaction. After the validation of the transaction is completed, a new block is added to the blockchain. Then the infrastructure can proceed to transfer the amount of energy agreed upon.

Second Smart Contract

Peer-to-Peer (P2P) trading is added in this contract, but in very a straight-forward way. The energy can be traded between all users in the community, yet, all trading still follows the same price-setting rules as before, the prices that were used for the ESCO trading in the previous model.

Here, all the participants can input the energy they want to buy or sell. The application will prioritize the trade between peers and users that requested a trade first. As a last resort, the system will go to the ESCO to finalize all pending trades.

Following the rules above, the program will go through the list of the balances until it finds the first user that asked for energy, that is, the balance is negative. Now it needs to find users that can sell energy. Again, it will go through the list of balances until it finds a user that wants to sell (positive balance), and then it will create a transfer between the selected users, and update their balances. Given that whoever comes first in the file has priority, the first user that wants to sell energy will sell to the first user that wants to buy energy. If the amount of energy that the seller has is not enough, then the program will move on in the list until it finds another positive balance and creates another transfer with the amount of energy that is lacking. After the first buyer that appears on the list has the total amount of energy that he requested, the program will move on to the next buyer and repeat the process, until it reaches the end of the list.

If, at the end of this loop, there are users that still do not have the balance at zero, that is, there are users that still need to buy or sell energy, then the program will create a transfer between those users and the ESCO directly.

With the list of transfers created, which now has the following data: sender, receiver and amount of energy to transfer, the program will run the generic smart contract to calculate the prices and complete the transaction.

Third Smart Contract

In the last SC, the P2P trade will be done through auctions. A first-price sealed-bid auction, also known as a blind auction, is used and the program will function as the auctioneer. In this type of auction, all bidders can submit sealed bids. In this way, no bidder can know the bid of other participants. The highest bidder wins and pays the exact price that he bid during the auction, while the losers don't have to pay anything.

In this smart contract, there are three main features:

- Auctions

The user can decide to start an auction, by inputting 3 values: the amount of energy he wants to sell in that auction, the amount of time he wishes for the auction to be open, and the minimum bid, the minimum amount of ETH the bidders can bid. Once the time of the auction has ended, the winner will be selected, and the transaction will be made. The ESCO cannot participate in the auctions.

Users can see the auctions that are available. Once an auction has started, the owner can also choose to cancel the auction. Only one auction per user at one time is allowed. And to avoid high values of minimum bids, if an auction has come to an end and there are no bids, the energy will be sold directly to the ESCO, at the price in the generic contract.

- Bids

The users can place bids anonymously in the open auctions, simply by giving the address of the auction's owner and their bid (in ETH). They can place as many bids as they wish on the available auctions. If an auction ends and the highest bid belongs to more than one user, then the winner will be the user that placed the bid first.

There is also the option to withdraw from an auction. When an user gives the auction's owner address, all bids made in that auction will be deleted.

- Trading with ESCO

At last, the users can also trade directly with the ESCO. If the user does not want to participate in auctions or the trade is time sensitive, they can trade directly with the ESCO, at the agreed price in the generic contract.

It is assumed that buying from the ESCO has an energy trading fee of 10%, this leaves the users with enough room to make their trading amongst peers without losing money. If a user wants to buy energy from the ESCO, it will pay the price at the moment (or the daily average) plus a 10% fee, having to pay 1.1 of the electricity price. If a user wishes to sell energy to the ESCO, it will only receive the electricity price. This means, that if the user wants to receive a bigger compensation for their electricity, it has that 10% of freedom to trade amongst peers, without any loss to either participant.

B. Blockchain

The smart contracts above will be implemented in a blockchain. As stated before, the blockchain Ethereum was used. Ethereum is an open-sourced (public) and decentralized blockchain, with ether (ETH) as the cryptocurrency, that powers thousands of decentralized applications, dApps.

Ethereum is a technology used for building apps, where users can own assets and transact between them without being controlled by a central authority. This blockchain is currently using proof-of-stake for the consensus mechanism.

The network is composed of nodes, and each node communicates with a small subset of the network – called 'peers'. Whenever a node wants to add a new transaction to the blockchain, it sends a copy of the transaction to its peers, who send a copy to their peers, propagating throughout the network. Then, validators, selected nodes in the network, participate in the consensus mechanism. Validators stake ETH into a smart contract on Ethereum to participate in the system, and this staked ETH acts as collateral that can be destroyed if a validator behaves dishonestly or lazily. They are chosen at random to check if new blocks are valid and can occasionally create and propagate new blocks. After the block has been checked, the validators send a vote, called attestation, in favor of that block across the network. The votes of all validators that checked the same block are then used to determine the validity of the block being proposed [21].

A transaction usually includes the following information [21]:

- Recipient - the receiving address,
- Signature - the sender's address,
- Nonce - transaction identifier,
- Value - the amount of ETH to transfer,
- Data - any additional data to include,
- Gas Limit - the maximum amount of gas units to be used in the transaction,
- Maximum priority fee per gas - the maximum amount of gas to be included as the tip,
- Maximum fee per gas - the maximum amount of gas the user is willing to pay for the transaction.

As anticipated, the receiver and sender addresses, and the value are identified in the SC. Only the gas limit was defined at 21 000 Gwei, a standard value for ETH transfers, while all other values for the gas are set to default. And the following data was added to each block:

- Timestamp - time of the transaction,
- Energy requested - the amount of energy to be transferred between sender and receiver.

C. Web Interface

To make the application user-friendly and as real as possible for testing, a website was created as the User Interface (UI). For this, Web3.0 is used – decentralized internet based on public blockchains, being built, operated and owned by its users.

Web3 uses blockchains, cryptocurrencies and NFTs (non-fungible tokens) to give users ownership. A collection of libraries in JavaScript, web3.js, is used with the main purpose of interacting with the Ethereum blockchain. This library facilitates the development of websites that connect clients with the blockchain, by allowing the users to create smart contracts and perform transactions.

Others programs were also connected to this application, for developing purposes. Such as Vite + React, that were used to create the UI. To connect the UI to the Ethereum blockchain, HardHat and Alchemy were installed. HardHat is an Ethereum development environment that allows to run solidity locally, i.e., test smart contracts before deploying them. And Alchemy is a web3 development tool, that deploys the blockchain smart contract.

Users of the application will need to interact with the website and actually be able to transfer funds between them, therefore a cryptocurrency wallet is necessary. Metamask is then selected to handle cryptocurrency. Metamask allows users to interact with the Ethereum blockchain dApp. It can be accessed through a browser extension or a mobile app. Its users can securely connect to decentralized applications, and send and receive Ethereum-based cryptocurrencies and tokens.

All participants in the community, including the ESCO, can create an account with Metamask and fund their wallets with ETH. This extension also provides a secure interface, since the user has total control over their own account. The user chooses if they wish to connect to a website, and once the smart contract requests a transaction from the user through Metamask, it is up to the user to approve it or reject it. A confirmation from the owner's accounts is always necessary to proceed with a transaction.

The application will be tested in a Testnet network. These networks allow developers to test their dApps before deploying them into the Ethereum mainnet, without having to worry about the costs of transactions while testing. The Goerli Testnet Network was chosen amongst other testnets due to its similarity to the Ethereum mainnet.

VI. RESULTS

For testing purposes, an energy community with 4 prosumers and one Energy Services Company (ESCO) is considered. For each of the users, the balance is defined in one moment in time. This balance determines if the user wishes to sell or buy energy, decided by the '+' or '-' respectively, and the amount of energy.

In Table I, the balance of all users is presented. These are the values that will be used when testing the application.

User	Balance (MWh)
A	- 0.007
B	+ 0.005
C	- 0.003
D	+ 0.006
TOTAL	+ 0.001

TABLE I. Energy balance of each user, in one moment in time.

The interaction between community and the application will be evaluated in 4 different scenarios, to compare the influence of the time and electricity market prices:

- During the day with low electricity prices,
- During the day with high electricity prices,
- During the night with low electricity prices,
- During the night with high electricity prices.

The smart contract with the ESCO will define its prices according to the SPOT market for electricity. To test the application, real data from the SPOT market is used. The prices for the 4 different scenarios are selected from the month of September 2022. For the days where the electricity price is low, it is also considered a low average price for that day, once it will also be a part of the smart contract. The electricity prices to be considered are presented in Table II.

Description	Price (€/MWh)	Average price (€/MWh)
Day – low prices	60.50	100.16
Day – high prices	221.29	175.42
Night – low prices	75.64	119.77
Night – high prices	248.80	193.04

TABLE II. Electricity prices for the four different scenarios to be tested.

A. First smart contract

The database (Table III) that is connected to the program contains the balances already discussed.

Time	Account	Balance (MWh)
t	A	- 0.007
t+5	B	+ 0.005
t+10	C	- 0.003
t+15	D	+ 0.006

TABLE III. Database loaded in the application for the first smart contract.

The time is dependent on the moment the test is being run. And an interval of 5 minutes is set between transactions, to make sure that there is enough time to approve the transactions and switch accounts.

The first smart contract was tested with the values for the different electricity scenarios. All transactions began at the time inserted in the database, and to the correct accounts.

With every transaction, there are two fees to be paid: the gas fee for the energy trading, and the contract interaction fee. Table IV describes the values paid for the energy trading: the price paid for the energy requested (the amount of ETH transferred between accounts in €) and the total cost of the transaction (including all fees).

The prices paid for the trade are according to the generic smart contract with the ESCO. At low prices, the smart contract used the daily average to calculate the price, whereas the electricity price for the moment was used for the high electricity prices.

Note that the total cost during the day is much higher than during the night. This proves that the hour of the day when the transaction happens has a major influence on the fees applied.

Scenario	From	To	Price (€)	Total cost (€)
Day - low prices	A	ESCO	0.7712	23.1592
	ESCO	B	0.5008	20.8146
	C	ESCO	0.3305	19.9046
	ESCO	D	0.6010	20.0876
Day - high prices	A	ESCO	1.7039	20.6805
	ESCO	B	1.1065	20.8731
	C	ESCO	0.7303	18.7798
	ESCO	D	1.3277	20.3297
Night - low prices	A	ESCO	0.9222	1.3435
	ESCO	B	0.5989	1.0161
	C	ESCO	0.3952	0.8071
	ESCO	D	0.7186	1.1382
Night - high prices	A	ESCO	1.9158	2.3272
	ESCO	B	1.2440	1.6498
	C	ESCO	0.8210	1.2226
	ESCO	D	1.4928	1.8922

TABLE IV. Amount of currency paid for each transaction made, while testing the first smart contract.

B. Second Smart contract

The same database (Table III) was used for this smart contract. Following the rules presented for the second smart contract, the following list of transfers was created.

Account from	Account To	Balance (MWh)
A	B	0.005
A	D	0.002
C	D	0.003
ESCO	D	0.001

TABLE V. List of transfers created by the program when executing the second smart contract, for the database presented in table III.

The smart contract was able to created the correct list of transfers to execute, for each of the scenarios. The transactions are presented in Table VI.

Scenario	From	To	Price (€)	Total cost (€)
Day - low prices	A	B	0.5008	24.3668
	A	D	0.2003	19.7916
	C	D	0.3005	19.1269
	ESCO	D	0.1002	18.8464
Day - high prices	A	B	1.1065	18.6353
	A	D	0.4426	13.6673
	C	D	0.6639	8.7342
	ESCO	D	0.2213	5.6206
Night - low prices	A	B	0.5989	0.4535
	A	D	0.2395	0.6326
	C	D	0.3593	0.7513
	ESCO	D	0.1198	0.5115
Night - high prices	A	B	1.2440	1.6355
	A	D	0.4976	0.8890
	C	D	0.7464	1.1378
	ESCO	D	0.2488	0.6402

TABLE VI. Amount of currency paid for each transaction made, while testing the second smart contract.

Note that this contract, while satisfying the existence of P2P trading, it is not the best logic to trade amongst peers. If a buyer is asking for a larger quantity than any offer available, which is the case of user A, he will have to trade twice as many (or more times) to get the energy he needs. Consequently, user A paid twice as many fees as the other peers. Analyzing the results, it would be more beneficial to trade the full amount directly with the ESCO, instead of paying twice as many fees.

C. Third Smart Contract

In this smart contract, there is an increased complexity. There are more possible options for energy trading. As a result, two different situations are considered, in order to test the most features of the application: 2 auctions available and both have bids, and 2 auctions available but only one has bids.

Situation 1

User B and user D have auctions available at reasonable prices, i.e., between 1 and 1.1 of the electricity price at that moment. Therefore, user A and user C place bids in both auctions, user A places the highest bid in the user D auction, and user C places the highest bid in the user B auction.

In Tables VII and VIII, the values of the auctions and bids used for testing are presented.

Account	Electricity (MWh)	Minimum bid (ETH)
Day – low prices		
B	0.005	0.000333
D	0.006	0.000397
Day – high prices		
B	0.005	0.000730
D	0.006	0.000880
Night – low prices		
B	0.005	0.000395
D	0.006	0.000480
Night – high prices		
B	0.005	0.000820
D	0.006	0.000980

TABLE VII. Values used for the auctions while testing the final smart contract, for the first situation.

Account	Auction B	Auction D
Day – low prices		
A	0.000339	0.000418
C	0.000340	0.000400
Day – high prices		
A	0.000740	0.000917
C	0.000755	0.000903
Night – low prices		
A	0.000405	0.000504
C	0.000409	0.000492
Night – high prices		
A	0.000843	0.001000
C	0.000860	0.000993

TABLE VIII. Values used for the bids, in ETH, while testing the third smart contract, for the first situation.

Situation 2

User B and user D have auctions, but user B is asking more than the ESCO at that time. Therefore, user A and user C only place bids in user D auction, where user A places the highest bid. Since the auction of user B does not have any bids, the energy will be sold directly to the ESCO. User C buys energy directly from the ESCO.

In Tables IX and X, the values for the auctions and bids are presented.

Account	Electricity (MWh)	Minimum bid (ETH)
Day – low prices		
B	0.005	0.000360
D	0.006	0.000397
Day – high prices		
B	0.005	0.000800
D	0.006	0.000880
Night – low prices		
B	0.005	0.000432
D	0.006	0.000480
Night – high prices		
B	0.005	0.000900
D	0.006	0.000980

TABLE IX. Values used for the auctions while testing the final smart contract, for the second situation.

Account	Auction B	Auction D
Day – low prices		
A	-	0.000418
C	-	0.000400
Day – high prices		
A	-	0.000917
C	-	0.000903
Night – low prices		
A	-	0.000504
C	-	0.000492
Night – high prices		
A	-	0.001000
C	-	0.000993

TABLE X. Values used for the bids, in ETH, while testing the third smart contract, for the second situation.

The third and last smart contract was implemented. Tables XI and XII describe the results for this smart contract, for the first and second situations, respectively.

Scenario	From	To	Price (€)	Total cost (€)
Day - low prices	A	D	0.6431	10.1079
	C	B	0.5231	7.9194
Day - high prices	A	D	1.4108	9.5064
	C	B	1.1615	8.9171
Night - low prices	A	D	0.7754	1.1668
	C	B	0.6292	1.0206
Night - high prices	A	D	1.5385	1.9299
	C	B	1.3231	1.7145

TABLE XI. Amount of currency paid for each transaction made, while testing the first situation of the third smart contract.

In the first situation, only two transactions per electricity scenario were made. The winner of the auctions paid the exact amount of ETH they bid. Even if user A still needed 0.001 MWh according to their balance, no adjustment was made.

Rather than the two transactions made in the previous situation, three transactions were made here for each electricity price scenario in the second situation, one for the auction of user D, where user A won and paid the exact amount of ETH he bid. Since user B auction did not receive any bids, the en-

Scenario	From	To	Price (€)	Total cost (€)
Day - low prices	A	D	0.6431	4.0966
	ESCO	B	0.5008	4.3653
	C	ESCO	0.3636	4.0608
Day - high prices	A	D	1.4108	4.7104
	ESCO	B	1.1064	4.5305
	C	ESCO	0.8033	3.9327
Night - low prices	A	D	0.7754	1.1668
	ESCO	B	0.5988	0.9902
	C	ESCO	0.4348	0.7866
Night - high prices	A	D	1.5385	1.9299
	ESCO	B	1.2440	1.6354
	C	ESCO	0.9032	1.2124

TABLE XII. Amount of currency paid for each transaction made, while testing the second situation of the third smart contract.

ergy was sold directly to the ESCO, and user C bought energy from the ESCO. When trading with the ESCO, the generic smart contract computed the prices.

VII. DISCUSSION

All smart contracts developed for this dissertation proved to be effective when testing them. All transactions had the correct sender and receiver, and the smart contract computed the correct price in all electricity price scenarios.

Comparing the three smart contracts deployed, it is obvious that the third one, where the trade is done through a blind auction, is the most favorable one. In the last smart contract, a seller can sell their energy at a higher price than the electricity price at that moment, which is all that the ESCO is willing to pay. And since the ESCO asks for 1.1 of the electricity price, it is more profitable for buyers to participate in the auctions. If a seller wants to ask more than the ESCO, it can do so, but with the risk of the energy being sold directly to the ESCO for the agreed price, at the end of the auction. This pushes the seller to sell their energy with a minimum bid between 1 and 1.1 of the electricity price, and buyers to bid in the auctions available, instead of automatically paying 1.1 of the electricity price to the ESCO.

Analyzing the tables with the results for all tests that were performed, the first aspect which is important to notice is the fees that the user must pay to operate with this application. This application uses the Ethereum blockchain, and every time the user is making a transaction using a smart contract, two fees must be paid:

- Transaction fee associated with the energy trading: the gas paid to the validators for the transaction.
- Contract interaction: the fee paid to interact the smart contract.

The transaction fees are linked to the congestion in the network. The more people transact, the more congested the network becomes, following higher gas fees to incentivize the validators to work faster. On the other hand, the contract interaction is directly connected to the complexity of the smart contract, as well as memory usage and other factors.

In this particular case, these compensations paid to the network in a single transaction are already higher than the amount paid for the energy trading, the user is paying more to use the application than for the energy he requests in each transaction.

This means that the moments where it is beneficial to trade in this application are constricted. It is only favorable to trade energy in this application when the gas fees are low (mainly at night) or when trading larger amounts of energy, so that the fees are lesser than the amount paid for the energy trading.

Overall, this application seems to have two critical elements that determine if the moment is convenient for energy trading: the electricity prices, where one needs to see when the prices are low (including the daily average) and gas prices, to make sure that the fees paid for the transaction do not overpower the actual price transacted.

VIII. CONCLUDING REMARKS

In the beginning of this thesis, it was proposed a new framework for energy trading systems: a solution to make transactions inside a local energy community. For that, a blockchain-based smart contract application was suggested. The main focus of this project was to develop smart contracts, using blockchain technology, that can be applied to P2P trading.

First, a generic smart contract was developed for all interactions with the ESCO, using the electricity prices from the SPOT market.

After, three smart contracts were designed in Chapter V A, each one more complex than the other.

As expected, the last smart contract is the most suited for P2P trading, since it is more advantageous for both sides. Sellers can receive a bigger payment for their energy than they would receive if they went directly to the ESCO. And buyers can pay less for the same amount of energy in auctions.

These smart contracts were run through the Ethereum Blockchain, a public blockchain with ETH (ether) as its cryptocurrency. Users can manage their ETH with the help of Metamask.

By analyzing the results attained, this blockchain proved to not be the most suitable platform for energy trading, users could be paying more for the fees applied to use the blockchain and interact with the contract, than the actual price of the energy.

While developing the application, a few limitations came to light, which prevents this solution to be a real implementation for P2P energy trading. These constraints should be looked at as a way to improve and not as limits to its applicability.

First, the legality of a smart contract is still not well defined. Certainly, a smart contract in these conditions is not obligated to be considered as a smart legal contract. However, if a person acts against the community, consequences must be discussed. It is also necessary to have systems in place to resolve disputes.

And even though the smart contracts deployed for this dissertation were quite simple, complex smart contracts can cost a lot to deploy. Additionally, with every interaction, there is also a fee, which would also be higher. Complexity and memory allocated would have to be a relevant concern when designing the smart contract.

One possible improvement for the third smart contract would be another type of auction. In the auction deployed, the bids were placed anonymously, and the winners took all the energy that was auctioned. This is not the only type of auction that can be applied to energy trading systems. An english auction or a dutch auction can also be enforced. And it is even possible for the partition of the auctioned electricity. Imagining that a bidder only wants part of the electricity that is being auctioned, it can be possible to bid just for a part of the electricity. Then the auctioned energy would go to several bidders.

Regarding the blockchain technology, Ethereum presented its own limitations. The Ethereum's cryptocurrency proved to be not the most appropriate when a small amount of ETH is being transacted, that result from small amounts of energy. The fees that are in place, such as the contract interaction, mentioned before, and the gas fee, due to network congestion, are higher than the price paid for the electricity. Consequently, making this platform not as inviting as it should be.

Nonetheless, there are strategies to help reduce the gas costs, such as predicting network congestion and even bundle transactions together.

Another concern is obviously linked to privacy and security. To participate in this energy community, its participants have to provide personal information, such as location. Since Ethereum is a public blockchain, any piece of information stored in the ledger would be visible to all, which then leads to an issue with data privacy.

Notwithstanding, the development of smart energy contracts, that enable users to actively participate in the energy market has to be a main concern in the present and near future. In this way, they can contribute to the increase of renewable energy production and consumption, and to a more efficient use of electricity and energy grids. And since blockchain technology was developed to implement distributed secure transactions, it is compatible with the implementation of smart energy contracts.

Clean energy technology is becoming a major new area for investment and employment – and a dynamic arena for international collaboration and competition [2]. These decentralized systems need to be designed and truly implemented, to assure consumers that this new way of handling energy is a reliable and affordable choice, hopefully turning consumers into prosumers, and consequently increase their renewable energy use and reduce their ecological footprint.

---

[1] The 17 goals — sustainable development (2022).
[2] IEA, World energy outlook 2021 (2021).
[3] A. Miglani, N. Kumar, V. Chamola, and S. Zeadally, Blockchain for internet of energy management: Review, solutions, and challenges, Computer Communications **151**, 395 (2020).
[4] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, Blockchain technology in the energy sector: A systematic review of challenges and opportunities, Renewable and Sustainable Energy Reviews **100**, 143 (2019).
[5] S. N. G. Gourisetti, D. J. Sebastian-Cardenas, B. Bhattarai, P. Wang, S. Widergren, M. Borkum, and A. Randall, Blockchain smart contract reference framework and program logic architecture for transactive energy systems, Applied Energy **304**, 117860 (2021).
[6] A. Hasankhani, S. Mehdi Hakimi, M. Bisheh-Niasar, M. Shafie-khah, and H. Asadolahi, Blockchain technology in the future smart grids: A comprehensive review and frameworks, International Journal of Electrical Power Energy Systems **129**, 106811 (2021).
[7] F. Buccafurri, G. Lax, L. Musarella, and A. Russo, An ethereum-based solution for energy trading in smart grids, Digital Communications and Networks https://doi.org/10.1016/j.dcan.2021.12.004 (2021).
[8] A. S. Yahaya, N. Javaid, F. A. Alzahrani, A. Rehman, I. Ullah, A. Shahid, and M. Shafiq, Blockchain based sustainable local energy trading considering home energy management and demurrage mechanism, Sustainability **12**, 3385 (2020).
[9] A. Dorri, F. Luo, S. S. Kanhere, R. Jurdak, and Z. Y. Dong, Spb: A secure private blockchain-based solution for distributed energy trading, IEEE Communications Magazine **57**, 120 (2019).
[10] G. Bansal, A. Dua, G. S. Aujla, M. Singh, and N. Kumar, Smartchain: A smart and scalable blockchain consortium for smart grid systems, 2019 IEEE International Conference on Communications Workshops (ICC Workshops) , 1 (2019).
[11] I. Losa, M. de Nigris, G. Bissel, V. Sanz, L. Sigrist, and S. Galant, Overview of the solutions developed by the grid+ project for supporting the european smart grids pilot projects in achieving the eegi goals: main findings and lessons learnt, Comillas.edu http://hdl.handle.net/11531/7950 (2015).
[12] Power ledger (2022).
[13] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, Designing microgrid energy markets: A case study: The brooklyn microgrid, Applied Energy **210**, 870 (2018).
[14] IBM, What are smart contracts on blockchain? (2022).
[15] D. Vieira Fernandes, Tokens, "smart contracts" and system governance, SSRN Electronic Journal 10.2139/ssrn.3492274 (2019).
[16] IBM, What is blockchain technology? (2022).
[17] IBM, Benefits of blockchain (2022).
[18] I. A. S. M. . B. R. Younes, Z., Blockchain applications and challenges in smart grid, (2021).
[19] How does a transaction get into the blockchain? (2016).
[20] Api - spot electricity market (2022).
[21] Ethereum, Ethereum development documentation (2022).
[22] U. Secretary-General, Progress towards the sustainable development goals (2021).