UNIVERSITAT ROVIRA i VIRGILI

ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

FINAL DEGREE PROJECT

Smart Contracts and Blockchain for a Secure and Privacy-Preserving Smart Grid

By :
Joan Ferré Queralt

Project director :
Dr. Jordi Castellà Roca

June, 2023

Contents

Abstract

Electricity has been an integral part of human history, driving progress and improving quality of life. However, the current energy landscape faces challenges such as escalating demand, rising prices, and the depletion of traditional energy sources. Existing electricity grids face limitations and vulnerabilities, necessitating a transformative solution. To overcome these issues and transition to renewable energy, a new electricity model is needed—one that fosters proximity between producers and consumers, enhances production and efficiency, and includes prosumers who can both consume and produce electricity using Distributed Energy Resources (DER). Ensuring the security and privacy of this system is paramount. This work describes a smart grid based on blockchain technology and smart contracts, designed within a secure architecture. We demonstrate the effectiveness of our approach by implementing and testing the system in a controlled local environment. Our evaluation also includes measuring gas usage, yielding valuable insights into the scalability of the system. Leveraging the decentralized nature of blockchain and the automation capabilities of smart contracts, our system enables secure and efficient energy trading, promoting transparency, reliability, and trust among participants. Our implemented prototype validates the feasibility and effectiveness of our proposed solution. The system showcases robustness, scalability, and enhanced privacy while facilitating seamless energy trading between producers and consumers. This research presents a promising solution for the challenges confronting the electricity sector.

Resum

L'electricitat ha estat una part integral de la història humana, impulsant el progrés i millorant la qualitat de vida. No obstant això, la situació energètica actual s'enfronta a reptes com l'increment de la demanda, l'augment dels preus i l'esgotament de les fonts d'energia tradicionals. Les xarxes elèctriques existents s'enfronten a limitacions i vulnerabilitats, la qual cosa requereix una solució transformadora. Per superar aquests problemes i la transició a l'energia renovable, es necessita un nou model d'electricitat que fomenti la proximitat entre productors i consumidors, millori la producció i l'eficiència, i inclogui *prosumers* que puguin consumir i produir electricitat utilitzant Recursos d'Energia Distribuïts (DER). És primordial garantir la seguretat i la privacitat d'aquest sistema. Aquest treball descriu una xarxa intel·ligent basada en la tecnologia *blockchain* i contractes intel·ligents, dissenyats dins d'una arquitectura segura. Demostrem l'eficàcia del nostre enfocament aplicant i provant el sistema en un entorn local controlat. La nostra avaluació també inclou el mesurament de l'ús del gas, proporcionant informació valuosa sobre l'escalabilitat del sistema. El nostre sistema permet un comerç energètic segur i eficient, promovent la transparència, fiabilitat i confiança entre els participants. El nostre prototip implementat valida la viabilitat i l'eficàcia de la nostra solució proposada. El sistema mostra robustesa, escalabilitat i millora de la privacitat alhora que facilita un comerç energètic sense fissures entre productors i consumidors. Aquesta recerca presenta una solució prometedora per als reptes als quals s'enfronta el sector elèctric.

Resumen

La electricidad ha sido una parte integral de la historia humana, impulsando el progreso y mejorando la calidad de vida. Sin embargo, la situación energética actual enfrenta desafíos como el aumento de la demanda, el incremento de los precios y el agotamiento de las fuentes de energía tradicionales. Las redes eléctricas existentes se enfrentan a limitaciones y vulnerabilidades, lo que requiere una solución transformadora. Para superar estos problemas y lograr la transición hacia la energía renovable, se necesita un nuevo modelo de electricidad que fomente la proximidad entre productores y consumidores, mejore la producción y la eficiencia, e incluya a los *prosumers* que pueden consumir y producir electricidad utilizando Recursos Energéticos Distribuidos (DER, por sus siglas en inglés). Es primordial garantizar la seguridad y la privacidad de este sistema. En este trabajo, presentamos una red inteligente basada en la tecnología *blockchain* y contratos inteligentes, diseñados dentro de una arquitectura segura. Demostramos la eficacia de nuestro enfoque mediante la implementación y prueba del sistema en un entorno local controlado. Nuestra evaluación también incluye la medición del consumo de gas, lo que proporciona información valiosa sobre la escalabilidad del sistema. Nuestro sistema permite un comercio energético seguro y eficiente, promoviendo la transparencia, la confiabilidad y la confianza entre los participantes. Nuestro prototipo implementado valida la viabilidad y eficacia de nuestra solución propuesta. El sistema muestra robustez, escalabilidad y mejora de la privacidad al facilitar un comercio energético sin problemas entre productores y consumidores. Esta investigación presenta una solución prometedora para los desafíos que enfrenta el sector eléctrico.

1 Introduction

In the early 1970s, global electricity consumption was around 2,000 terawatt-hours (TWh) per year [37]. Over the next few decades, electricity consumption grew at a steady pace, as more people gained access to electricity and as economies around the world grew and developed. By 2020, global electricity consumption had reached around 23,000 TWh per year, more than eleven times the level of 50 years earlier [8].

This growth associated with electricity is highly visible when comparing the current situation of human development among the countries where electricity is still not an usual asset. We can see explained in [27] how the Human Development Index (HDI) is attached to the Per Capital Energy Consumption variable (PCEC) and how different types of energy (traditional and renewables) affect it differently as shown in figures 1 and 2. In [4] is also studied how it affects the economical growth.

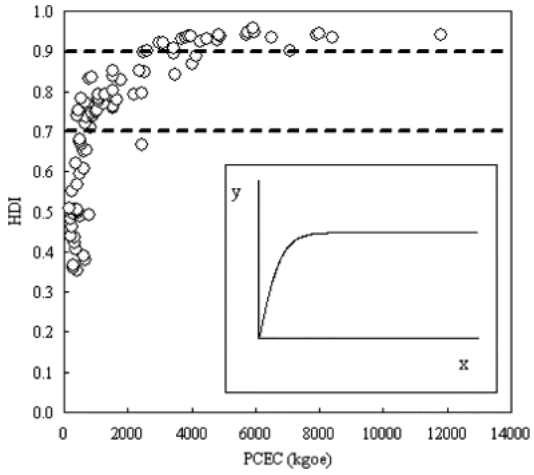

Figure 1: Saturation curve between HDI and PCEC

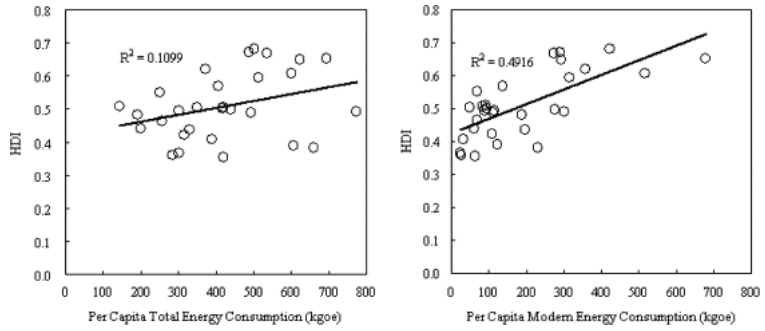

Figure 2: Difference between use of traditional energy sources and modern

As previously mentioned, the demand for energy has been increasing and is expected to continue to do so in the future [20]. The heightened demand for energy could potentially overburden a country's energy supply, resulting in the possibility of power shortages and disturbances in everyday life. Therefore, it is essential for a country to address this trend. This can not only ensure a stable supply of energy for its citizens, but, as mentioned above, also support the country's overall economic growth and competitiveness in the global market. There should be a way to regulate the balance between supply and demand to ensure the energy needs of a country.

The balance of supply and demand is upheld by means of a pricing process that involves several calculations, taking into account factors such as transmission, distribution, taxes, and other fees. The cost of generation includes the cost of fuel, (coal, natural gas, nuclear fuel...) and the expenses involved in the construction and maintenance of power plants. The focus in this work is on the cost of generation, as it is most affected by increasing demand.

The wholesale energy price, which only includes the cost of energy generation and not transmission or distribution, is usually determined the day before actual consumption. In this process participate the Transmission Operators (TSOs). Forecast total energy consumption for the following day is determined by a series of previous data and other factors such as special events or festivities, TSOs make the prediction and share this information with generators. Based on this information, generators provide estimates of how much energy they can produce and when [33].

When selecting power plants to generate electricity, the cheapest options, such as renewable energy sources, are chosen first. Nuclear energy is typically chosen next, followed by fossil fuels as the most expensive option. The final power plant chosen, typically the most expensive one, determines the price of electricity. The cost of generation and the availability of different energy sources therefore influence the final price [42].

Last year, coal met half of the increase in the post-pandemic high global demand, meaning an increase of the electricity prices and CO_2 emissions attached to the coal burning [20], which means that coal and fossil fuels are most likely not a viable option due to the environmental issues and their growing scarcity [9].

One alternative to the most traditional fossil fuels is the gas. However, the start of the Russia's invasion of Ukraine made a huge impact on the gas prices due to all the sanctions imposed by the EU and Russia on the gas trade, one of the main fossil fuels energy sources, making it much more expensive [31] and scarce. Cost of generation is not only affected by its source but also by geopolitics and other factors.

The European Commission has implemented several measures in response to high energy prices and concerns about energy availability. These measures include a cap on gas prices and a "solidarity" contribution from companies that have profited from the situation. Additionally, efforts have been made to reduce peak electricity demand in order to prevent blackouts [43]. However, it is possible that these measures may not be sufficient, and more significant changes may be necessary.

Apart from the fossil fuels, nuclear energy also plays an important role when looking at the most frequent energy sources. Nuclear fission is often cheaper and more efficient than fossil fuels. However, it also has some drawbacks [11], such as limited flexibility during periods of low demand, a limited lifespan for nuclear reactors, a limited supply of uranium, and the challenging disposal of radioactive waste. There has also been some significant fatal incidents in some nuclear plants such as Chernobyl or Fukushima which raised concerns about its safety with citizens' health [16].

According to some studies, one promising alternative to nuclear fission is nuclear fusion, which may potentially be a long-term solution to many energy problems [21]. However, at this time, it is not considered a practical or feasible current solution.

Since the previous options seem to have a series of significant problems, it might be beneficial to move towards another main source of energy. In recent years, the cost of renewable energy sources, such as wind and solar, has fallen significantly [1], making them more competitive with traditional fossil fuels and increasing the cost of the latter in order for it to be profitable. This has led to a shift towards a more sustainable and decarbonized electricity sector in the European Union. Transitioning to more sustainable energy sources can potentially have both economic and environmental benefits.

However, the adoption of renewable energy has not been increasing at a sufficient pace [20]. This is due in part to the challenges of transitioning away from fossil fuels, as renewable sources may not be as reliable and flexible as traditional ones [40]. There would have to be more offer of renewables and improve their reliability and efficiency.

One way to potentially improve the availability and efficiency of the energy distribution system is to locate energy producers closer to consumers. This can help to reduce the cost and loss of distributing energy over long distances. The use of distributed renewable energy technologies, like solar panels, also opens up the possibility of local, decentralized production and consumption of electricity [18].

Some households are currently using distributed energy resource (DER) systems, like solar panels, to generate and consume their own electricity. If these homes produce more energy than they need, they can sell the excess back to the main grid.

However, the price at which this excess energy is sold is often not attractive enough to sell. One of the reasons is that there may be limitations on the compensation received for excess energy, such as a cap on the amount of excess energy that can be compensated in certain countries like Spain [12].

Given the challenges and limitations described above, it may be necessary to adopt a new energy model to help meeting growing demand by increasing the use of renewable energy. It would also be desired to fairly incorporate decentralized and locally-generated energy. This new model should encourage users to use DER. One potential item that this new model could use is a smart grid, which has been proposed as a way to modernize and optimize energy distribution [13].

The aim of this work is to propose a new energy model that:

- Ables the distribution of locally produced energy between users that can both consume and produce energy and are popularly called prosumers.
- Guarantees the fairness and honesty of all the parts.
- Protects the privacy of the users.

1.1 Related work

In the recent years, there have been lots of studies about different implementations of a smart grid system, each of those focusing in different aspects of it. This work is focused on the overall implementations of a smart grid which consider the trading of electricity between prosumers.

The first definition of a smart grid was made in [7], which made a first list of requirements that this new energy model should have. Later on, in [36], they discussed some different communication standards for the smart grid, all of which were focused on the traditional centralized structure of the main grid.

However, a centralized structure in a smart grid is considered to have several security issues such as a Denial of Service attack, as explained in [5]. Some of those security concerns could be avoided using a distributed system, avoiding the single point of failure problem [30].

One of the first solutions to this problem was the introduction to an energy currency in [29], which was based on the blockchain technology proposed in [32]. The proposal opened up the possibility of using a blockchain as the base for the new grid.

Following the latter trend, there has been an increasing interest in utilizing blockchain technology to establish smart grids, particularly with the introduction of smart contracts. This technological advancement has enabled the creation of comprehensive and operational applications on the blockchain platform. Numerous works have been conducted on this topic, including those proposed in [3], [22], [28], [2], and [29], which present various blockchain architectures for implementing a smart grid.

However, it is important to note that each of these works has employed a blockchain that uses Proof of Work consensus mechanism. This approach has been shown to be highly energy-intensive, as it requires substantial computing resources to verify transactions [10]. Such energy consumption is a major challenge to the adoption of Proof of Work blockchain systems in large-scale applications, such as smart grids.

It is, therefore, imperative to explore alternative consensus mechanisms that can minimize energy consumption while maintaining the integrity and security of the system.

There have been other works that have explored the integration of blockchain technology into energy systems, but some of these works do not give adequate consideration to the privacy and anonymity of users. For instance, in [19], [22], and [38], there is limited focus on the concealment of users' identities or their energy consumption patterns.

It is essential to address these privacy concerns when designing blockchain-based energy systems to ensure that the data shared on the network is secure and not susceptible to malicious attacks. A lack of privacy measures can lead to various issues, such as unauthorized access to sensitive information or profiling of user behavior [45].

In [23] they consider anonymity and privacy, however, from the transaction it is only verified the identity of the user, but not if the production or consumption truly is the one specified, they don't contemplate the possibility that a user might send a fake production of energy.

While the identity of the user is verified through the blockchain transaction, there is no assurance that the production or consumption of energy specified in the transaction is genuine. In other words, a user could potentially submit a fake energy production record, which can mislead the system and other users, leading to undesirable outcomes. To mitigate this risk, it is crucial to have mechanisms in place that can verify the accuracy and authenticity of the energy production records.

Furthermore, it is also essential to consider the impact of false energy production records on the overall performance of the system. If fake records are allowed to propagate, they can lead to imbalances in the supply and demand of energy, which can result in system instability and inefficiencies.

In [15] it is also considered both privacy and anonymity, but there should be a way to determine a user's actions in case of dishonesty. In this work, the pseudonyms used makes the traceability a difficult task. It also uses a bloom filter to authorize users, which can produce false positives (it is probabilistic algorithm), in a system which could be used by a large quantity of users, this possibility should be none.

Other solutions such as [39] also take into account privacy and anonymity but have a reputation system in case of bad behavior, no one possesses the relationship between the smart grid accounts and real identity of the users. Although some measures such as preventing fraudulent users from participating in the trade can be applied in that case, the identification of the user should be possible for the authorities, since the measures should be taken directly to the user responsible.

Overall, our solution contemplates an energetically efficient transaction architecture which keeps the privacy of the users and their anonymity, which should be revocable in case of bad behavior. Furthermore, the information sent by the user should be always verified, not only the transaction but the amount of electricity produced/consumed.

1.2 Plan of this work

First of all, Section 2 provides a background of the main technologies utilized in this work.

In Section 3 we provide an overview of the protocol, outlining the key actors and stages involved in our proposal. We also identify a set of requirements that the system must fulfill to ensure its success.

Subsequently, in 4 we delve into the details of our proposal, offering comprehensive information about each component and procedure necessary for the operation of a functional smart grid. We carefully explain each step and stage of the system.

To assess the security requirements of the proposed energy trading system, we evaluate the system's architecture in Section 5. We demonstrate how the utilization of blockchain technology and the proposed architecture contribute to fulfilling the requirements discussed in Section 3.

To test the system's functional requirements, in Section 6 we implement a working prototype in a local environment.

Finally, in Section 7, the implementation is tested and evaluated so it we can conclude its correct functioning.

2 Background

In this section, we will introduce the technologies employed in our work. The purpose of this introduction is to provide a foundational understanding of the key systems that have played a significant role in our research.

2.1 Smart grid

A smart grid is an updated version of the current energy grid. Its main feature that differs from the traditional grid is the two-way flow of information and electricity, which means that electricity does not only go from main producers to final consumers but can also flow between consumers who can also be producers. The name given to those users who both consume and produce energy is *prosumers*.

The initial concept of SG started with the idea of advanced metering infrastructure (AMI) with the aim of improving demand-side management and energy efficiency, and constructing self-healing reliable grid protection against malicious sabotage and natural disasters [35].

Some other benefits that a smart grid could bring according to the National Institute of Standards and Technology [14] and listed in [13] would be :

- Improving power reliability and quality;
- Optimizing facility utilization and averting construction of back-up (peak load) power plants;
- Enhancing capacity and efficiency of existing electric power networks;
- Improving resilience to disruption;
- Enabling predictive maintenance and self-healing responses to system disturbances;
- Facilitating expanded deployment of renewable energy sources;
- Accommodating distributed power sources;
- Automating maintenance and operation;
- Reducing greenhouse gas emissions by enabling electric vehicles and new power sources;
- Reducing oil consumption by reducing the need for inefficient generation during peak usage periods;
- Presenting opportunities to improve grid security;
- Enabling transition to plug-in electric vehicles and new energy storage options;
- Increasing consumer choice;
- Enabling new products, services, and markets.

The benefits of a smart grid system could potentially address current issues with the energy system. In recent years, there have been numerous proposals for smart grid architectures that aim to bring these benefits to reality.

2.2 Blockchain

Blockchain technology emerged as a groundbreaking innovation, offering decentralized, secure, and transparent solutions across various industries. Its foundations lie in cryptography and computer science, and the first practical implementation of blockchain was Bitcoin, a digital currency introduced in 2008 by the pseudonymous Satoshi Nakamoto [32].

A blockchain is essentially a distributed, decentralized ledger that records transactions in a secure, tamper-resistant, and verifiable manner. The architecture of a blockchain consists of several key components [26] [41]:

- **Blocks**: Each block contains a list of transactions, a timestamp, a reference to the previous block (called the parent block), and a unique identifier known as the block hash. The block hash is generated using cryptographic algorithms and ensures the integrity of the block's contents.
- **Chain**: The blocks are linked together in a linear, chronological order, forming a chain. The chain's structure ensures that any alteration to a block's data would require modifying all subsequent blocks, which is computationally impractical, providing security against tampering.
- **Consensus Mechanism**: A set of rules and protocols, called the consensus mechanism, governs how new blocks are added to the blockchain. Common consensus mechanisms include Proof of Work (PoW) [32], Proof of Stake (PoS) [24], and Delegated Proof of Stake (DPoS) [25].
- **Nodes**: A blockchain network is composed of nodes, which are computers that store a copy of the blockchain and participate in the consensus process. The distributed nature of the network ensures no single point of failure, enhancing its security and reliability.

Blockchain technology employs various security features to ensure the trustworthiness and robustness of the system [46]:

- **Cryptographic Hash Functions**: These functions are used to generate unique block hashes and transaction identifiers, providing data integrity and security. Commonly used hash functions include SHA-256 (Bitcoin) and Ethash (Ethereum) [44].
- **Digital Signatures**: Blockchain transactions are secured using digital signatures, which are generated using asymmetric cryptography. This allows users to sign and verify transactions with their private and public keys, ensuring data authenticity and non-repudiation.
- **Immutability**: The inherent structure of the blockchain ensures that once data is recorded, it is nearly impossible to alter. This immutability is achieved through the

use of cryptographic hash functions, the linear chain structure, and the consensus mechanism.

- **Decentralization**: The distributed nature of blockchain systems eliminates the need for central authorities, reducing the risk of censorship, fraud, and single points of failure.

2.3 Smart contracts

One of the multiple applications blockchain has, and probably one of the most notorious one are the smart contracts. Smart contracts are computer programs executed automatically when predetermined conditions are met, eliminating the need for intermediaries and reducing the risk of fraud or disputes.

Some of the most notable features of smart contracts are:

- **Automation and Self-execution**: Smart contracts are self-executing and require minimal human intervention. They are coded with predetermined conditions, and once those conditions are met, the contract automatically executes the terms outlined in the agreement.
- **Trust and Security**: Smart contracts are built on blockchain technology, which provides a decentralized and transparent environment. This ensures that all parties involved have access to the same information, fostering trust and reducing the likelihood of disputes.
- **Cost-efficiency**: By eliminating the need for intermediaries and streamlining processes, smart contracts can significantly reduce transaction costs and administrative expenses.
- **Interoperability**: Smart contracts can interact with other blockchain-based systems and data sources, enabling seamless cross-platform integration and data exchange.
- **Customization**: Smart contracts can be tailored to meet the specific requirements of various industries and applications, ensuring flexibility and adaptability.

2.4 IOTA

IOTA, which stands for Internet of Things Application, is a distributed ledger technology (DLT) specifically designed for the Internet of Things (IoT) and machine-to-machine (M2M) communication. It aims to address the unique challenges and requirements of IoT systems by providing a secure, scalable, and feeless solution [34].

At the core of IOTA's infrastructure lies a Directed Acyclic Graph (DAG) called the Tangle. The Tangle is a decentralized data structure that enables the processing of transactions in parallel. Unlike traditional blockchain architectures, where transactions are organized into blocks and processed sequentially, the Tangle allows multiple transactions to be confirmed simultaneously. This parallel processing capability enhances scalability and enables IOTA to handle a significant number of transactions even in environments with millions of interconnected IoT devices.

One of the key advantages of IOTA is its feeless nature. In the Tangle, users are required to validate two previous transactions before submitting their own. This validation process not

only ensures the security and integrity of the network but also eliminates the need for transaction fees. The absence of fees makes IOTA particularly well-suited for microtransactions and M2M communications, which are prevalent in IoT ecosystems.

Another notable feature of IOTA is its ability to enable real-time data sharing among various devices and stakeholders within IoT systems. By leveraging IOTA's DLT, sensors, energy meters, and energy management systems can seamlessly exchange information, facilitating efficient demand-response management, load balancing, and overall grid stability. This real-time data sharing capability is crucial for optimizing energy consumption, improving operational efficiency, and enabling intelligent decision-making in IoT environments.

Security is a paramount concern in IoT systems, where the integrity and confidentiality of data are of utmost importance. IOTA's distributed ledger technology ensures data integrity and security by employing cryptographic techniques and the inherent structure of the Tangle. Since each transaction is cryptographically linked to two previous transactions, any attempt to tamper with the data would require altering the entire transaction history of the Tangle. Such a feat is computationally infeasible, providing robust security against malicious actors and ensuring the reliability of the system.

Furthermore, IOTA serves as a standardized platform for data exchange and communication among diverse stakeholders in IoT systems. It promotes interoperability between utilities, energy suppliers, and consumers, enabling efficient coordination, energy trading, and the seamless integration of different energy sources. This standardized approach fosters collaboration and innovation within the IoT ecosystem, leading to enhanced efficiency, reduced costs, and improved sustainability.

3 Protocol overview and requirements

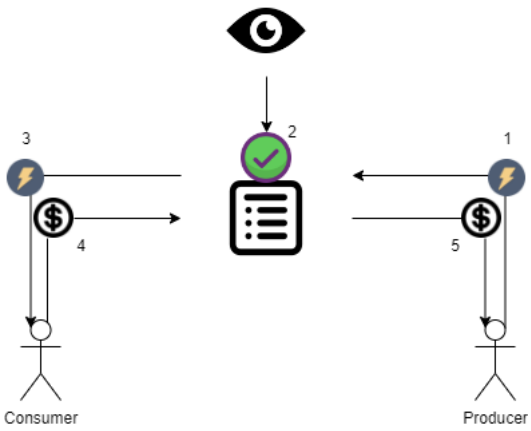

Figure 3: Local distribution of energy

As seen in Fig. 3, the main functionality of our proposal is the distribution of self-generated energy within a local consumers and producers group. Our system is in charge of: keeping all transactions of en energy trades, verifying them and distributing the paid amount by the consumers to the corresponding producers.

The proposed architecture considers the following actors/components:

- Smart meters: devices who keep track of a prosumer's usage of energy (production and consumption).
- Prosumers: each system user that at a certain time can both produce and consume energy, depending on the balance between the production/consumption that prosumer can be:
  - Consumer: if he/she consumes more energy than produced.
  - Producer: if he/she produces more energy than consumed.

  Each prosumer owns a smart meter and can have consumption, production facilities or both in their household.
- Main grid: connects the main energy sources to the smart grid and is in charge of electricity distribution. Is the one in charge of energy consumption prediction and providing the extra electricity needed.
- Smart grid: keeps transactions between prosumers and between prosumers and the main grid. Also coordinates all payments.
- Coordinators: responsible for verifying user's transactions, they also own a smart meter which verifies that the prosumers are not lying about their consumption/production amounts. Also, if a prosumer fails to pay, the coordinator takes the necessary measures, although those are outside the scope of this proposal.
- Operators: are tasked with registering prosumers on the smart grid layer, connecting them, and giving them the authorization to send messages with the relevant trading

information to their respective groups. They connect the prosumers to the smart grid physically (wiring) and digitally (giving them the necessary permissions).

The proposed architecture, as we can see in Fig. 4, consists of three main layers: prosumers layer, smart grid layer and main grid layer.

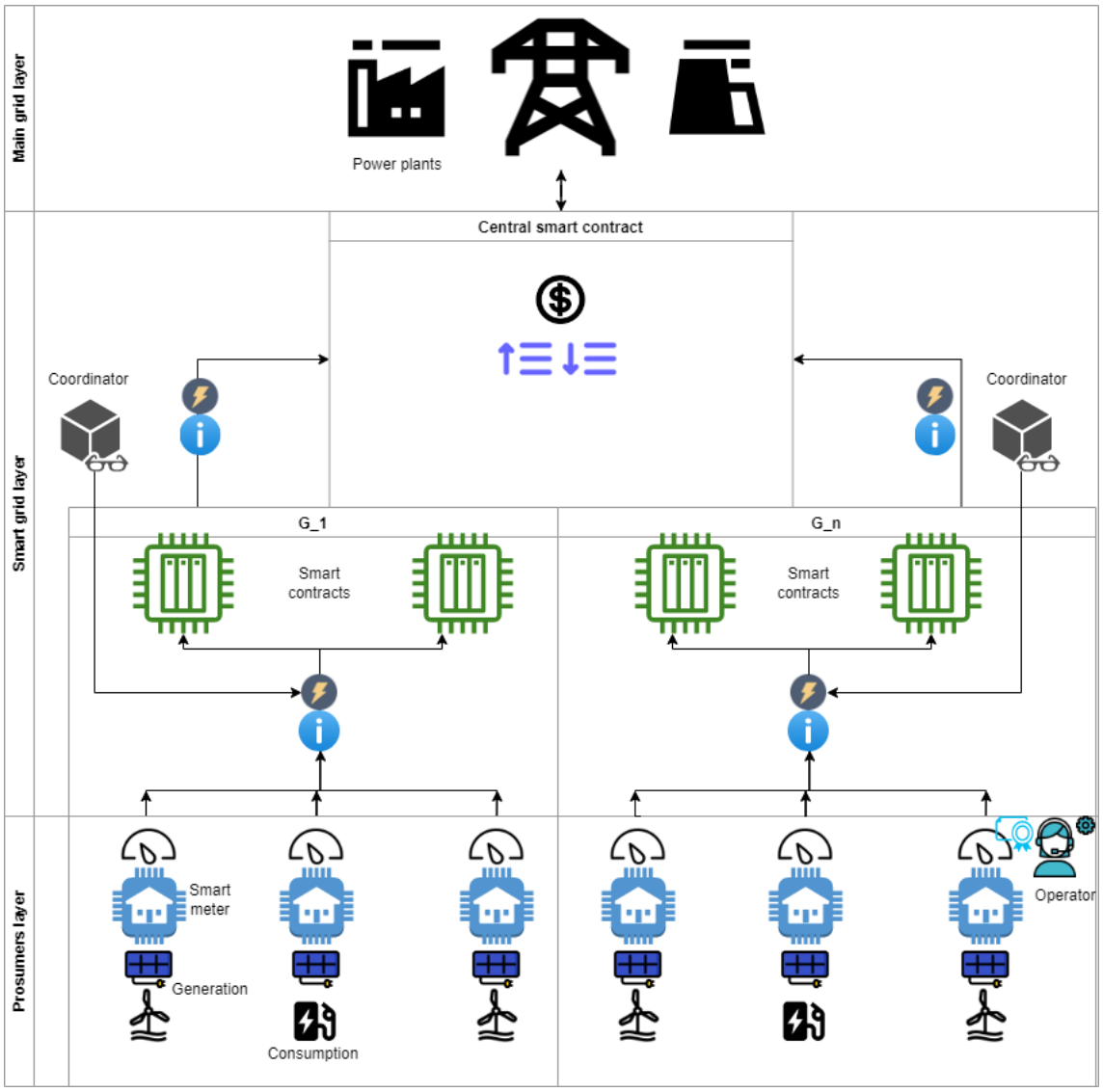

Figure 4: Architecture scheme

The prosumer layer is composed of several groups of prosumers who are geographically close to each other. In these groups, they trade electricity among themselves to increase efficiency.

The smart grid layer is located between the prosumers layer and the main grid layer. This layer also has a coordinator for each group of prosumers, each coordinator verifies their group members transactions.

This layer has the information of the energy usage of each group, which means that, in case a prosumer group needs more electricity or has a surplus, the smart grid layer communicates with the main grid level to buy r sell that electricity. In order to obtain that information, it records all transactions between prosumers in each group with the energy usage information.

Finally, the main grid layer is the system as we know of it today. It is composed of the main electrical companies with their bulk electricity generators and also control the distribution of energy.

3.1 Stages

The main functionality of the smart grid can be divided in three stages:

- Price calculation.
- Energy trading.
- Debt liquidation.

The first stage starts the day before energy trading, with a prediction of energy usage. That's when the main grid estimates the demand and communicates with the main electricity producers. The producers are ranked based on their prices, with the most expensive determining the trading price for electricity in each time slot. In this proposal, the price is determined in different time slots of the day.

The second stage, once the prices have been determined for each time slot of the day, consists of the communication between the prosumers' smart meters with the smart grid layer. The user's electricity usage and transactions is monitored by a smart meter located in their home. The smart meter periodically stores the electricity production/consumption information to the smart grid layer via transactions, allowing to track the user's electricity usage so that the the debt of the user can be calculated with the amount of energy produced/consumed and the price in each time slot.

The smart grid layer communicates with the main grid to register the usage of electricity between the groups and the main grid. The smart grid system monitors the energy needs of each group and can request additional electricity from the main grid or sell any surplus energy back to it. To facilitate these transactions, the main grid is treated as an additional prosumer in the system.

A secure storage system is used to store information about each address in order to identify users with fraudulent use of the system. This system can only be accessed by coordinators who have the key and can be used to cut off energy or users. Operators can introduce new user information by signing the information with a group signature in the registration phase.

In our proposal, coordinators verify each prosumer's message. They also have the responsibility of taking necessary measures if a prosumer fails to pay.

However, in order to be able to launch this function, coordinators must act as a trusted third party (TTP). This is because the main line of electricity cannot be fully anonymized, and coordinators must have access to the user's information and their addresses in order to verify the electricity transactions. As a result, coordinators must have a balance between ensuring user privacy and executing their responsibilities effectively.

The smart grid charges a fee to the users to pay the main distribution system. That is used to record every transaction between the groups and the main grid, so the main grid can act

as another seller/buyer in each group and also as a way to regulate the prices between the groups.

Finally, in the third stage of the proposal, the smart grid collects all of user's transactions and assigns each of them a debt to be paid (to them if they're producers of from them if they're consumers). This debt is communicated to the consumers and, once it's paid, the the corresponding amount is distributed to the producers. This way, there's no direct interaction between each consumer and producer.

The overall sequence can be seen in Fig. 5. Steps 1-5 correspond to the first stage, price calculation. Steps 6-8 correspond to the second stage, the trading. Finally, steps 9-11 correspond to the last stage, the debt liquidation.

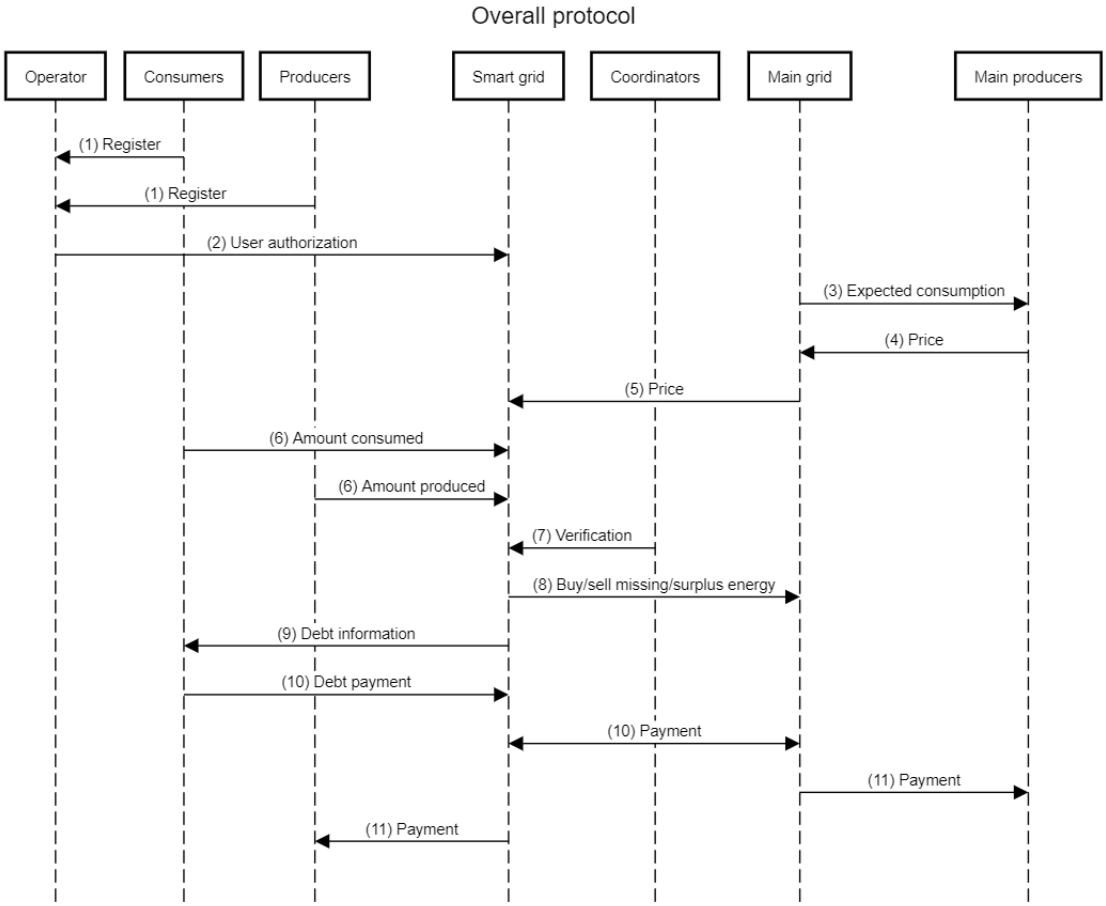

Figure 5: Overall sequence diagram

3.2 Notation

In order to describe our proposal, the following notation shown in Table 1 will be used:

Users	$U = \{U_1, U_2, ..., U_n\}$
Consumers	$C = \{C_1, C_2, ..., C_n\}$
Producers	$P = \{P_1, P_2, ..., P_n\}$
Coordinators	$Co = \{Co_1, Co_2, ..., Co_n\}$
Coordinator group	$G = \{G_1, G_2, ..., G_n\}$
Operators	$O = \{O_1, O_2, ..., O_n\}$
Group smart contracts	$SC = \{SC_1, SC_2, ..., SC_n\}$
Smart meters	$SM = \{SM_1, SM_2, ..., SM_n\}$
Energy consumed at time slot t	E_t^C
Energy produced at time slot t	E_t^P
Transactions	Tx
Main grid	MG
Main grid smart contract	SC_{MG}
Main grid producers	MG_P

Table 1: Notation

3.3 Requirements

From the proposal overview, and according to the previous works mentioned in the state of the art section, there are several key requirements that must be considered when designing the smart grid system:

3.3.1 Functional

In order for the previously described system to work properly, it has to achieve the following requirements:

- Energy trading: allow the sale and purchase of distributed generated energy.
- Scalability: the system should be able to handle a large number of U and SM and be capable of adapting to changing demand.
- Accuracy: the system Txs should be able to reflect energy demand and supply of each user in each time slot.
- Transparency: U should have easy access to their own energy usage and production information as well as their Tx.
- Price calculation: The smart grid system must be able to obtain the price of electricity the day before usage based on the demand and producer ranking.
- Debt calculation: the smart grid must be able to calculate each consumer's debt and determine the payment owed to each producer.
- Smart contract interaction: both users and the main grid must be able to interact directly with the smart contract, which must verify the identity of the user for each transaction.
- Low consumption: the system used shouldn't make an abusive use of electricity since the purpose of it is to win electricity efficiency.

3.3.2 Security

Given the base requirements of the previous subsection, the following requirements should be also implemented in order for the system to be secure and keep the privacy of the users:

- Integrity: the system should ensure the integrity of data and detect unauthorized modifications or tampering. Modified messages from P and the smart grid layer can't take part in the system.
- Availability: the Tx system should be highly available, able to function without interruption.
- Authentication: the system should implement robust authentication measures to ensure that only authorized users can access the system.

- Authorization: the system should have controls in place to ensure that users can only perform actions that they are authorized to perform. This includes measures to prevent unauthorized access to resources and functions.
- Non-repudiation: the system should have measures in place to ensure that users cannot deny having performed a particular action or Tx.
- Privacy: energy usage patterns from a user should be hidden when looking at the Tx of the SM, even having access to some of the Tx, the complete energy pattern from the user should not be extracted.
- Revocable anonymity: Tx shouldn't identify any user in a unique manner. They should be anonymous and free of any information about the user, its identity and location shouldn't be visible. Only in case of bad behavior (case mentioned above), a prosumer should be identifiable by Co.
- Storage immutability: stored successful Tx should be immutable. No one should be able to modify the records from previous Tx.
- Coordinator Verification: The smart grid must have coordinators to verify each prosumer's message to detect possible fraudulent behavior and ensure fair trade.

4 Proposal description

In this section it is explained in detail all the processes in our proposal. First, explaining the initialization of the whole system and the main components, followed by the registration of all the participating users. After that, the section will be divided into the three stages of the energy trading process.

Each section will include an explanation of all necessary concepts and also a step by step description of each process.

In the proposed solution for a smart grid, the blockchain and smart contracts are used as the main components. Smart contracts are self-executing programs with the terms of the agreement directly written into code.

4.1 Smart contracts

This section explains the main components of the smart grid system, which are the smart contracts. It will also enumerate the actions in each smart contract and which actors are able to perform them. It will also be mentioned how each smart contract is able to determine if a specific user is authorized to launch an operation.

4.1.1 Main smart contract

To establish a functioning energy marketplace, the main smart contract (SC_{MG}) must be the first to be deployed. This contract acts as an intermediary between each prosumer group (G) and the main grid. Its initial task is to obtain and store the necessary information about each MG_P so they can communicate their energy production capacity and price of production.

To achieve this, each MG_P must create a blockchain account and register it with the main SC. The SC then verifies the provided information and store the address of each MG_P. Using the information provided by each MG_P and the knowledge possessed by the main grid (MG), the SC_{MG} can calculate the price of electricity at any given time of the day.

In addition, the SC_{MG} needs to obtain the addresses of all other SC so that they can communicate any missing or surplus energy in each group. The registration process for each SC with the SC_{MG} will be explained in the SC section.

The MG_P keeps track of all transactions between each SC and the MG. Whenever a prosumer group needs energy from the MG or sells a surplus, the SC_{MG} will register the transaction with the quantity of energy and the price. This procedure is similar to the one followed between prosumers in each SC.

This main smart contract consists on a class which holds:

- **Date**: date of the trading day of the SM_{MG}
- **Producer identities**: public keys of MG_P which can interact with the SC_{MG} and sell electricity to the MG.
- **Coordinator identities**: public keys of Co which can register SC into the $SCMG$
- **Main Grid Authority**: public key of the user owner of the SC_{MG}.
- **Expected electricity consumption**: the predicted electricity consumption by the MG for the following day.
- **Production capacity**: list of the production capacity of the different MG_P and its price, divided by time slots.
- **Prices**: list of the price of electricity in each time slot.
- **Group smart contracts**: addresses of all SC which can interact with the SC_{MG} and sell or buy energy from the MG.
- **Transactions**: list of all transactions made between the MG and the SC. Each transaction contains the amount sold/bought and its price.
- **Debts**: list of all debts from each SC and the MG.
- **Time slot**: number of the current time slot. -1 before the trading and 24 when it's over.
- **Trading flag**: boolean which is True when the trading in the MG_{SC} is available.
- **Producers flag**: boolean which is True when MG_P are able to be registered into the MG_{SC} and make an offer.
- **Smart Contracts flag**: boolean which is True when SC are able to be registered into the MG_{SC}.

When a new SC_{MG} is deployed, the MG must provide the following arguments: i) *Date of the trading day*; ii) *Expected energy consumption*. By default, *Trading flag* is set to False and *Producers flag* is set to True, *Time slot* is set to -1.

Once the SC_{MG} is deployed, the different actors can interact with it with the following methods:

- *newProducer*() method can only be called by the MG authority when *Producers flag* is set to True. It introduces a new MG_P which will be able to sell electricity in the trading day. The producer is added to *identities*.
- *newSC*() method can only be called by the Co when *Smart Contracts Flag* is set to True. It registers all SC that will be able to communicate the need or surplus of electricity of each group. The new SC is added to *Group Smart Contracts*.
- *addOffer*() method which can be called by MG_P when *Producers flag* is set to True. It communicates the available production and its price at a certain time slot. The offer is added to *Production capacity*.
- *beginAuction*() method called by the MG which, once all MG_P have registered and added their offers or the time limit has come, it determines the electricity price for each time slot. It sets *Producers Flag* to False.
- *beginTrading*() method called by the MG when the trading day starts. It sets *Smart Contracts Flag* to False and *Trading Flag* to True. It also communicates all the SC that the trading begins.
- *changeTime*() method called by MG, it increases *Time slot* by 1.
- *buyElectricity*() and *sellElectricity*() methods are called by the SC during the trading day, when *Trading flag* is set to True. SC communicate the needed electricity or the surplus that has to be bought/sold to the MG. Both the amount and the price is stored in *Transactions*.
- *calculateDebts*() method called by the MG once the trading has ended. From all the registered transactions, it calculates the debt each SC and the MG has. Every debt is stored in *Debts*. It sets *Trading Flag* to False. It also calls the method *calculateDebts*() from each SC.
- *payElectricity*() method called by each SC which has a debt with the MG. The SC_{MG} distributes the funds to the corresponding sellers (MG_P or other SC).
- *getDebt*() method is called by any actor and, if it exists, it returns the debt he/she has to pay.

4.1.2 Group smart contracts

After deploying the SC_{MG}, the next step is to deploy the SCs. Each prosumer group (G) has several SCs that are responsible for tracking the energy usage and price for each user (U) within the G.

To ensure that each U can send Tx to their respective SC, they must be registered in the SC. It possess a table which contains the necessary information to authorize each user. Additionally, the SC must register with the SC_{MG} in order to receive information on the price at which electricity is sold during each time slot.

It's important to note that the authorization tables within each SC must be kept up to date. For example, if a new U joins the group or an existing U leaves, the table must be updated accordingly. This ensures that only authorized Tx are recorded and that the energy usage and billing remain accurate.

Moreover, each SC must possess the identity of the coordinator (Co) of the group. The coordinator is the only authority that can verify the Tx from each U.

Once the SCs have been deployed and authorized, they can start collecting Tx made by each U within the group. This data is important for ensuring fair and accurate billing according to the energy usage within the group.

SCs can communicate with with the SC_{MG}. For example, if one group has a surplus of energy, they can communicate with the SC_{MG} which can communicate with another group that needs more energy and sell their surplus.

Each SC consists on a class which holds:

- **Main smart contract**: address of the SC_{MG} in which the SC is registered.
- **Date**: date of the trading day of the SC.
- **Coordinator**: public key of the SC's Co. Able to verify transactions.
- **User Identities**: public keys of all U of SC's G. Those users are able to sell/buy electricity.
- **Operator Identities**: public key of operators who can register new U into the SC.
- **Prices**: list of the price of electricity in each time slot.
- **Transactions**: list of all transactions made between U. Each transaction contains the amount sold/bought and its price.
- **Debts**: list of all debts from each U.
- **Time slot**: number of the current time slot. -1 before the trading and 24 when it's over.
- **Trading flag**: boolean which is True when the trading in the SC is available.

When the SC is deployed, the Co must provide the following arguments: i) *Date of the trading day*; ii) *Main smart contract address*. By default, *Trading flag* is set to False, *Time slot* is set to -1.

Once the SC is deployed, the different actors can interact with it with the following methods:

- *newOperator*(): method called by the Co which adds a new O public key which will be able to register U.
- *newUser*(): method called by the O which adds a new U which will be able to sell/buy electricity.
- *beginTrading*(): method called by the SC_{MG} which sets the *Trading flag* to True and also communicates all prices.
- *buyElectricity*() and *sellElectricity*() methods are called by U during the trading day once every time slot, only when *Trading flag* is set to True. U communicates the consumed or produced electricity to be bought/sold to others U. Both the amount and the price of the time slot is stored in *Transactions*.
- *verifyTransaction*() method is called by Co when, at the end of each time slot, the Co verifies each U transaction.
- *calculateDebts*() method is called by the SC_{MG} once the trading day has ended. From all the registered transactions, it calculates the debt each U has.
- *payElectricity*() method called by each U which has a debt with the G. The SC distributes the funds to the corresponding sellers (SC_{MG} or other U).
- *changeTime*() method called by Co, it increases *Time slot* by 1. It also calculates the surplus energy or the needed energy that the G has had in the previous time slot and communicates it to the SC_{MG}.
- *getDebt*() method is called by any actor and, if it exists, it returns the debt he/she has to pay.

4.2 Architecture and procedures

In the proposed architecture, the implementation involves multiple procedures. This section aims to provide a comprehensive explanation of these procedures, beginning with the initialization phase and subsequently detailing the procedures employed in each of the three stages of our system.

4.2.1 Initialization

Fig. 6 depicts the procedure of initialization of the system. This procedure is made before the first stage of our proposal, which is the price calculation.

- **Main smart contract**: the MG authority does the following steps:
  1. Deploys the SC_{MG}, with the corresponding arguments explained in the previous section.
  2. Communicates the SC_{MG} address to the Co and MG_P.
  3. Registers MG_P to the SC_{MG} using *newProducer()*.

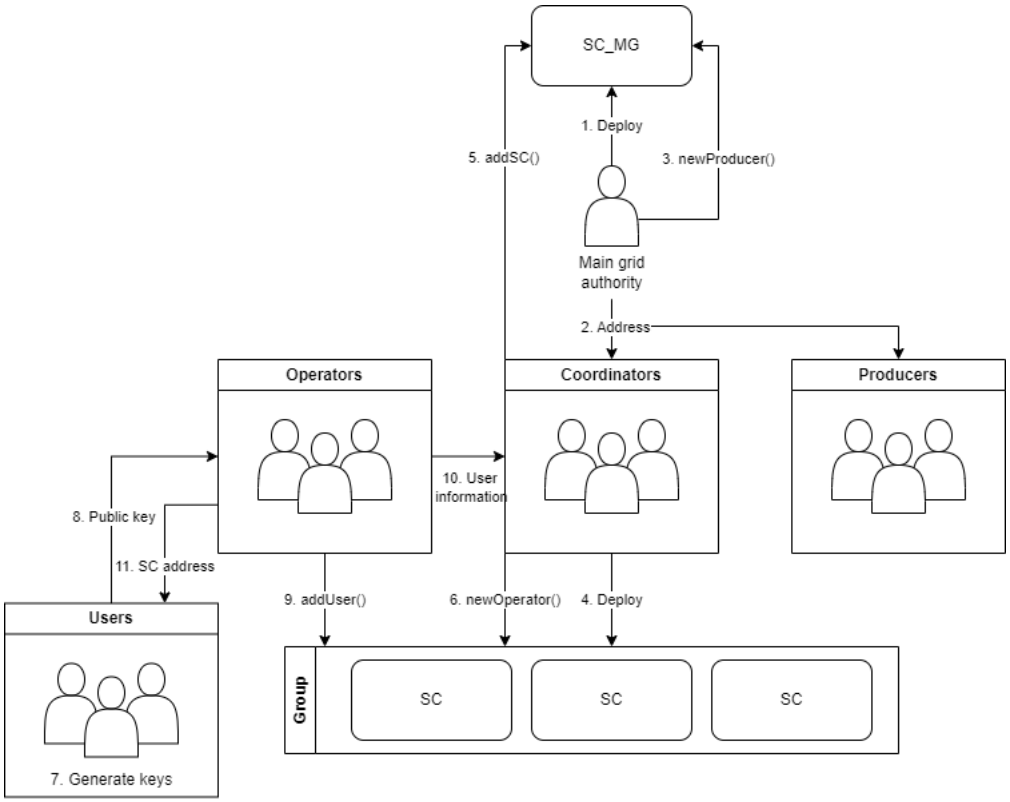

Figure 6: Initialization

- **Group smart contracts**: the Co of each group does the following steps:

  4. Deploys the each SC, with the corresponding arguments explained in the previous section.
  5. Communicates each SC address to the SC_{MG} using *newSC*().
  6. Registers O to the SC using *newOperator*().

- **User registration**:

  7. U is given a blockchain account and a private key and uses a Key Derivation Function to create n key pairs/addresses. The private keys are stored in a secure storage in which only U has access.
  8. U communicates every public key to the O, as well as their identification via a secure and private communication system.
  9. O registers U into each SC using *newUser*().
  10. O communicates U information to Co in order for the Co to be able to identify the U.
  11. O sends U each SC address.

Users of the smart grid need to have a wallet address. This address is generated from the public key and is where users can receive payments or deposits. The wallet address is also used to identify the user's account when sending transactions to other users or smart contracts.

To create a wallet address, the user must first generate their key pair. The private key should be kept secret and stored securely, while the public key can be shared with others. Using the public key, a wallet address can be generated through a process called hashing. This wallet address serves as a unique identifier for the user's account on the blockchain.

It's important for users to keep their private key secure and not share it with anyone, as it allows full access to their funds and account. If the private key is lost or stolen, the user may lose access to their funds forever. To ensure security, it's recommended to use a hardware wallet or store the private key in an encrypted format on a secure device.

With the key pair and wallet address, users can interact with the smart grid and participate in transactions with other users and smart contracts. The public key and wallet address also serve as a form of identification and verification, allowing other users and smart contracts to confirm the user's identity and ownership of funds.

The registration process of users is a crucial aspect of any smart grid system, as it ensures that only authorized individuals are able to access and perform transactions within the network. In this particular smart grid system, the registration of users is managed by the operators through the use of smart contracts. These contracts are designed to provide a secure and reliable mechanism for verifying the identity of users and controlling their access to the network.

During the registration process, users are required to provide certain information such as their name, contact information, and other relevant details. This information is then stored by the coordinator and it's linked to the user's blockchain address in a private storage.

Once a user is registered, the smart contract assigns them a unique identifier or address, which is used to track their transactions within the network. This address is also used to authorize their access to the network, ensuring that only authorized users are able to perform transactions.

The proposed solution utilizes group signatures for operators. Group signatures are a type of digital signature that allow a member of a group to sign a message on behalf of the group, while still maintaining the anonymity of the individual member who signed the message.

Before the step 7 in the Fig. 6, some steps must be followed by the O:

1. Installs a sealed smart meter SM_k at U_k's domicile.
2. Connects SM_k to U_k generation and consumption's electricity line in order for SM_k to read U_k's electricity usage information.
3. Connects SM_k to Co_k in order for Co_k to validate SM_k information.

4.3 Stages

Once the initialization process has ended, the system can start with the three defined stages. Starting the day before the trading with the price calculation.

4.3.1 *Price calculation*

In this stage, the only actors/components participating in it are: SC_{MG}, MG_P and the MG authority, Fig. 7 depicts the procedure.

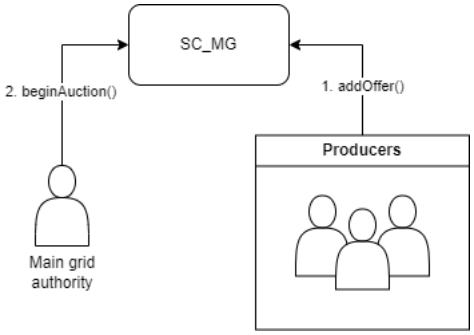

Figure 7: Price calculation

- **Calculate price**:
  1. All available P send their offer to the SC_{MG} using *addOffer*().
  2. When the deadline comes, MG authority ends the bidding time using *beginAuction*().
  3. SC_{MG} does the following:
     (a) In each time slot, it sorts the offers from the cheapest to the most expensive one.
     (b) Selects the n cheapest MG_P in each time slot so that the expected electricity consumption meets with the production.
     (c) The price in that time slot is the one from the most expensive MG_P offer from the n selected.

4.3.2 *Energy trading*

Once the price has been calculated, and the date of the SC and the SC_{MG} has arrived, the trading process starts. Fig. 8 depicts all the procedures involved in this stage, which lasts for the whole trading day.

- **Start trading**:
  1. The MG authority sends the method *beginTrading*() to the SC_{MG} when the trading day starts.
  2. When the SC_{MG} receives the order, it also sends the method to all the SC.
- **Trading**:
  3. In order for a U/SM to communicate the consumption/production of energy in order to sell or buy it, they have to go through the following steps:
     (a) Every n amount of time, at least once in every time slot, SM checks if the electricity usage is positive or negative. Depending on the result, it will use the method *sellEnergy*() or *buyEnergy*(), but the procedure is the same.

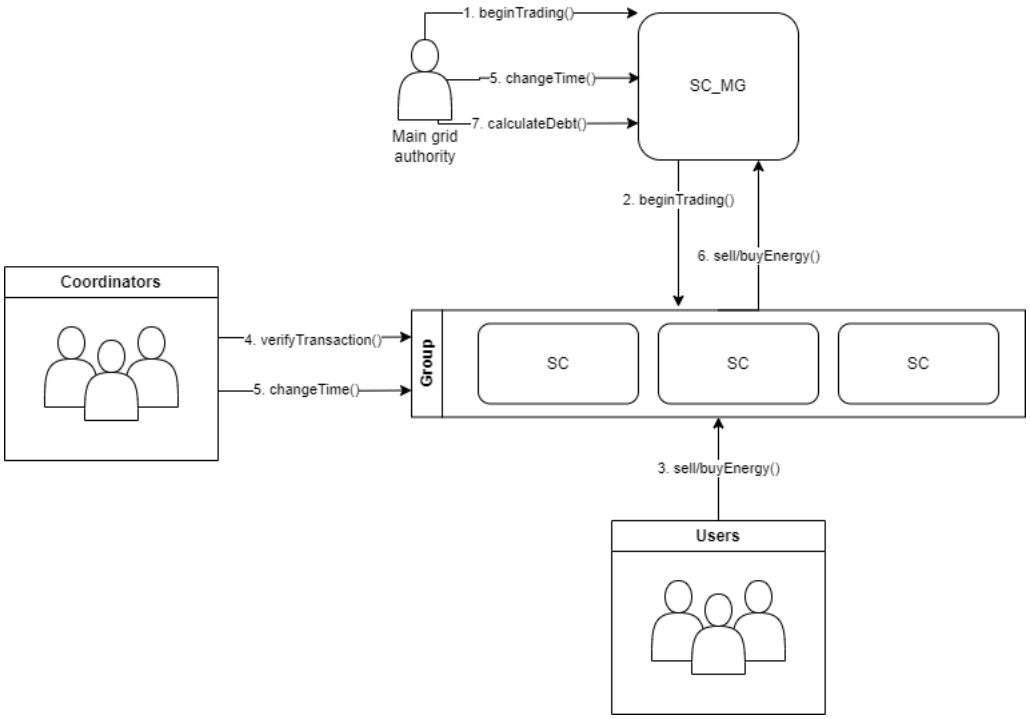

Figure 8: Initialization

(b) The energy usage is divided into m transactions.

(c) From k SC the G possesses, m SC are selected to receive each, one of the m transactions.

(d) Each SC, when it receives the transaction, it adds it to the *Transactions* list with the current price of electricity. However, it's still marked as unverified.

4. Before the end of the time slot, Co must verify U's transactions in order for them to be traded successfully. To do so, it follows the next steps:

   (a) During the time slot, it stores the energy usage information from each U of the G.

   (b) It checks the *Transactions* list and for each transaction it's able to verify, it sends the method *verifyTransaction*() with the transaction ID to the SC_{MG}.

   (c) SC_{MG} marks the transaction as verified.

5. Co now communicates the end of the current time slot to the SC. MG authority does the same with the SC_{MG}. They both use the method *changeTime*().

   (a) SC calculate the difference between the total electricity produced and consumed by G.

6. SC depending on the result of the difference, will call the methods *sellEnergy*() or *buyEnergy*() in the SC_{MG} with the total electricity produced/consumed.

   (a) If it's not the last time slot, the procedure returns to the point 3.

7. At the end of the trading day, the MG authority calls the method *calculateDebt()* which ends the trading phase.

4.3.3 Debt liquidation

Once the end of the trading day has come, it's time for the final stage, the calculation and liquidation of debts. Fig. 9

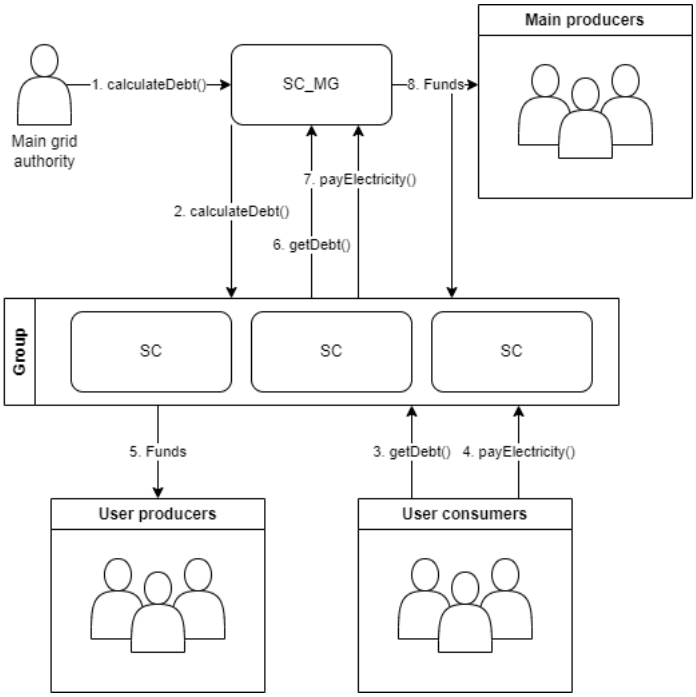

Figure 9: Initialization

- **Debt calculation**:

  1. The stage starts when the MG authority calls the method *calculateDebt()* in the SC_{MG}.
  2. SC_{MG} calls the method *calculateDebt()* of each SC.
  3. Each U who has consumed more than produced during the trading day, calls the method *getDebt()* from their G SCs.
  4. Once each C has their debt, they must pay it to each SC with the method *payElectricity()*.
  5. SC pays each U who produced more than consumed and has to be compensated.
  6. Each SC checks if G consumed more than produced by calling the method *getDebt()* in the SC_{MG}.
  7. If SC owes funds to the SC_{MG}, it calls the method *payElectricity()* to pay the MG for the extra electricity the G needed.
  8. Finally, SC_{MG} pays every MG_P for the energy produced, ending the third and final stage.

5 Security and Privacy study

In section 3.3.2 we defined the security requirements that the smart grid should fulfill in order for it to be viable. Next, some claims will be provided from the requirements.

Claim 1: the system can detect unauthorized modifications or tampering of data.

The immutability of transactions within a blockchain ensures the integrity and authenticity of each transaction, eliminating the possibility of double-spending. Given a secure and functioning blockchain, modifications or tampering of the ledger are not possible.

Claim 2: modified messages from P and the smart grid layer are prevented from participating in the system.

The system is secure due to the use of blockchain, which ensures that only legitimate transactions are allowed. This is due to the digital signature that each transaction possesses, if someone tried to modify a sent message to the smart contract, the signature would no longer be valid and thus, the smart contract would not accept it. This proof assumes that the blockchain and the digital signature are both secure.

Claim 3: the Tx system maintains high availability and is able to function without interruption.

Given a functioning blockchain with enough working nodes, the decentralized nature of a blockchain increases its availability, providing constant and reliable access to data.

Claim 4: the system's authentication measures effectively restrict access to only authorized users.

The smart contract verifies the identity of users who sign every transaction, ensuring that each request is authentic.

Claim 5: the system's controls restrict users from performing unauthorized actions and prevents unauthorized access to resources and functions.

The smart contract authorizes transactions only for registered users, maintaining the authorization requisites in the smart grid system.

Claim 6: the system has measures in place to prevent users from denying having performed a particular action or Tx.

The blockchain technology and the use of smart contracts ensure that transactions are traceable and tamper-proof, providing non-repudiation.

Claim 7: energy usage patterns cannot be derived from the Tx of the SC, even when having access to some of the Tx.

Each user is registered with multiple smart contracts. Each user is assigned a different address for each smart contract in the group, ensuring that only partial information about each user is available in each contract. This approach ensures that the smart contracts in each group and the corresponding addresses cannot be linked from the outside, providing an additional layer

of privacy protection for the users.

Smart contract (0x1234567890abcdef1234567890abcdef12345678)			
Transactions	User address	Energy	Price
	0x9FA4470983	4	4
	0xFAC5FC4112	-5	-
	0x7658CA5814	10	2
	...	...	...

Smart contract (0x0987654321fedcba0987654321fedcba09876543)			
Transactions	User address	Energy	Price
	0xB97797E553	-2	3
	0xEA931D1EC0	6	-
	0x93E34C1ACE	5	-
	...	...	...

Figure 10: Architecture scheme

Fig. 10 depicts an example of two smart contracts that appear to be unrelated when viewed from the outside. These two smart contracts could belong to the same group and even have transactions related to the same user. However, without the information held by the coordinator, it is not possible to establish any relationship between these two contracts.

Claim 8: Tx do not uniquely identify any user from an external point of view, ensuring anonymity.

The link between each U's address and their true identity is only known by the Co, a TTP. Blockchain technology keeps the anonymity of each transaction, which doesn't contain any personal information, only the partial energy usage.

To further enhance user anonymity, users periodically update their addresses. This helps to prevent anyone from obtaining the real identity of an address and tracking their transactions.

Claim 9: user identity and location are not visible within the system.

To maintain the anonymity of users and ensure that no one can trace which operator registered a specific user, the proposed solution utilizes group signatures for operators. Group signatures are a type of digital signature that allow a member of a group to sign a message on behalf of the group, while still maintaining the anonymity of the individual member who signed the message.

By using group signatures, the proposed solution ensures that the identity of the operator who registered a user is kept confidential. This is done because if someone knew which operator has which address, that someone could approximate the location of the registered user since an operator tends to work in the same relatively close area.

Claim 10: the smart grid has coordinators to verify prosumer messages and a prosumer can be identified by Co in cases of bad behavior.

Coordinators verify each prosumer's message to ensure that all transactions are ethical and free from any fraudulent activities, such as double spending. They also have the responsibility of taking necessary measures if a prosumer fails to pay. They are able to do that since they can see each U's Tx and have the knowledge if some of those Tx are fraudulent.

6 Implementation

In this section, it is explained the steps followed in order to implement our smart grid proposal as well as the design decisions taken.

First of all, the key elements of our system are discussed, which are: the blockchain, smart contracts, the developement framework and the evaluation environment. Next, it is also explained how the environment was set up.

6.1 Blockchain

In section 4 of our study, we highlighted the importance of incorporating a blockchain into our system. A blockchain provides a secure and transparent framework for recording and validating transactions, making it a desirable technology for various applications. However, not all blockchains are fully viable for a smart grid system due to specific limitations and considerations.

One example of a blockchain is the Bitcoin network. While Bitcoin has proven to be successful in the realm of digital currencies, it may not be the ideal choice for a smart grid system. Bitcoin's Proof of Work (PoW) consensus algorithm consumes a significant amount of computational power and energy, which can be a drawback for energy-efficient applications like the smart grid. The scalability of the Bitcoin network is also a concern, as it may struggle to handle the high transaction volumes and real-time data requirements of a smart grid system. [32]

Another popular blockchain is Ethereum, which offers programmable smart contracts and a more flexible framework [44]. Furthermore, Ethereum has currently transitioned to a Proof of Stake (PoS) consensus algorithm. However, evaluation of Ethereum 2.0's performance, security, and scalability in real-world scenarios are still in progress.

Additionally, other blockchain platforms such as Hyperledger Fabric [6] and R3 Corda [17] are designed for enterprise applications and offer greater privacy and control over data. However, these platforms may lack the scalability and real-time transaction processing capabilities required by a smart grid system.

Considering these limitations, we have identified IOTA as the most suitable option for our smart grid system. Unlike traditional blockchains, IOTA employs a Directed Acyclic Graph (DAG) called the Tangle as its underlying data structure. The Tangle enables parallel transaction processing, resulting in high scalability and the ability to handle a large number of transactions simultaneously. This scalability is crucial for accommodating the millions of connected devices and real-time data demands of a smart grid system.

Moreover, IOTA's feeless structure eliminates the need for transaction fees, making it particularly advantageous for microtransactions and machine-to-machine (M2M) communications, which are prevalent in the smart grid environment.

The Tangle's unique architecture also ensures data integrity and security. Each transaction is cryptographically linked to two previous transactions, making it computationally infeasible to tamper with the data. This robust security feature protects the system against malicious

attacks and ensures the reliability of the smart grid infrastructure.

Furthermore, IOTA provides a standardized platform for data exchange and communication among various stakeholders in the smart grid system, including utilities, energy suppliers, and consumers. This interoperability facilitates efficient coordination, energy trading, and the seamless integration of diverse energy sources.

6.2 Smart contracts

In addition to the requirement for a blockchain, our smart grid system also relies on the utilization of smart contracts.

IOTA allows two types of smart contracts to be deployed: the ones using WasmVM, an experimental project of IOTA, and the ones using EVM, the Ethereum Virtual Machine used in most of smart contract applications.

In the context of IOTA, there are two options for deploying smart contracts within Wasp's chains: using WasmVM, an experimental project of IOTA, or leveraging the Ethereum Virtual Machine (EVM), which is widely adopted in the realm of smart contract applications.

The decision to implement EVM support in IOTA was driven by several key factors:

1. **Developer Familiarity**: The EVM has become the de facto standard for developing smart contracts due to the popularity of Ethereum. Supporting the EVM within IOTA makes it easier for Ethereum developers to transition their existing applications or build new ones on the IOTA platform, leveraging their existing knowledge and tooling.
2. **Lack of development support**: WasmVM possesses scarce development support to build the tools needed in order to implement our smart grid proposal.
3. **Rich Ecosystem**: The Ethereum ecosystem offers a wide range of well-tested smart contract libraries, frameworks, and developer tools.
4. **Smart Contract Portability**: Supporting the EVM allows for the portability of smart contracts between other platforms and IOTA.
5. **Diverse Use Cases**: The adoption of the EVM in IOTA caters to a diverse range of use cases. While WasmVM provides a flexible and experimental approach, the EVM brings robustness and proven scalability to handle complex smart contract logic and high transaction volumes.

6.2.1 Developement framework

The development of a smart grid system, leveraging the private Tangle network, requires a robust and efficient approach to creating, deploying, and interacting with smart contracts.

When exploring different options, we considered a range of environments for smart contract development, including Truffle Suite, Remix, and Hardhat. While these environments have

their merits, Brownie emerged as the most favorable choice due to its unique combination of features and advantages.

The decision to use the Brownie framework in Python to create, deploy, and interact with smart contracts in the private Tangle network for the smart grid system was driven by several key factors:

1. **Python Ecosystem**: Python is a popular and widely-used programming language known for its simplicity, readability, and extensive ecosystem of libraries and frameworks. By choosing Brownie, a Python-based framework, the development process can leverage the rich Python ecosystem, allowing for easy integration with existing Python tools, libraries, and workflows.

2. **Smart Contract Development**: Brownie provides a developer-friendly environment for writing and testing smart contracts. Its intuitive and expressive syntax makes it easier to write complex smart contract logic, increasing development efficiency. Brownie also offers built-in testing and debugging capabilities, which are crucial for ensuring the correctness and reliability of smart contracts in the smart grid system.

3. **Deployment and Interaction**: Brownie simplifies the process of deploying smart contracts to the private Tangle network. It provides a convenient interface for compiling and deploying contracts, abstracting away the complexities of low-level interactions with the network. Brownie's integration with popular Ethereum clients, such as Ganache, allows for seamless deployment to local development networks for testing and debugging.

4. **Integration with Web3.py**: Brownie seamlessly integrates with Web3.py, a Python library for interacting with Ethereum-like networks. This integration enables developers to interact with the private Tangle network, send transactions, and retrieve data from the deployed smart contracts using familiar Python syntax and conventions.

5. **Community and Documentation**: Brownie benefits from an active and supportive community, making it easier to find resources, seek help, and collaborate with other developers. The availability of extensive documentation and tutorials further facilitates the learning curve and accelerates the development process for building the smart grid system.

6.3 Evaluation environment

The successful implementation of the systems we discussed, including IOTA, smart contracts, and the Brownie framework, relies on a robust and reliable infrastructure that supports their functionalities. Choosing the appropriate infrastructure is crucial for ensuring seamless operations and maximizing the efficiency of our smart grid system. Let's explore some examples of infrastructures and their respective advantages and disadvantages:

- **Public IOTA Network**: The public IOTA network offers a decentralized infrastructure for the Tangle and provides access to a realistic environment of a working ledger. However, the public network may introduce certain limitations, being the most notorious one the need for real funds.

- **Testnet**: Testnets are dedicated networks specifically designed for testing and experimentation purposes. They allow developers to simulate real-world scenarios and assess the performance of their applications without impacting the live environment. While testnets provide a controlled environment for development and testing, they may lack the scalability and stability required for production-grade systems.
- **Raspberry Grid**: using raspberries as a simulation for smart meters would be the most realistic environment. However, in order to make a first development and test the main functionalities thoroughly, it may be needed an easier setup.

A Linux-based personal computer will serve as the testing environment. This choice allows for flexibility and compatibility with the desired setup. In this setup, a private IOTA Tangle will be created, and all the essential components required for deploying and testing smart contracts will be implemented.

Creating a private IOTA Tangle on a personal computer offers several advantages and design considerations. Here are the main arguments to support this design decision:

1. **Testing and Development Environment**: Utilizing a personal computer as the testing environment provides a convenient and cost-effective solution. It allows developers to experiment with smart contracts, test different scenarios, and iterate quickly without relying on external resources or infrastructure. This flexibility is particularly beneficial during the development and prototyping stages, where a private Tangle on a personal computer can serve as a reliable and readily available platform.
2. **Resource Utilization**: Deploying a private Tangle on a personal computer optimizes resource utilization. Rather than relying on external servers or cloud infrastructure, which may incur additional costs, leveraging the personal computer's existing computing power and storage capacity is a cost-efficient approach. It also eliminates potential dependencies on external network connectivity, ensuring the availability of the Tangle for testing and development purposes, even in offline or restricted network environments.
3. **Easy Setup and Configuration**: Running a private Tangle on a personal computer simplifies the setup and configuration process. The availability of pre-configured scripts and repositories, such as the IOTA Hornet repository, streamlines the installation and deployment steps. This ease of setup empowers developers to quickly create a private Tangle without the need for specialized hardware or complex infrastructure deployments.

6.4 Environment setup

This section provides a detailed explanation of the various components of our setup and the deployment process. It focuses on three key elements: the private tangle, the Wasp node, and the deployer. Each of these components plays a vital role in the overall functionality of our system.

6.4.1 Private tangle

To establish a private IOTA Tangle, the key repository utilized is available at the following URL:

https://github.com/iotaledger/hornet.

The specific directory, "private_tangle," will be employed as it contains the necessary scripts to facilitate the local execution of a private Tangle on a personal computer.

To use this repository, the following steps were performed:

1. Clone the repository: Begin by cloning the IOTA repository using the provided URL. This will retrieve all the required files and scripts to set up and manage the private Tangle.
2. Navigate to the "private_tangle" directory: Once the repository is cloned, access the "private_tangle" directory, which holds the relevant scripts needed to execute the private Tangle on the local machine.
3. Execute the setup scripts: Run the setup scripts provided within the "private_tangle" directory. The scripts build the project using GoLang and deploy the necessary

Once these steps were completed, the private IOTA Tangle was fully operational on the Linux-based personal computer. The installation's structure is similar to the one depicted in Fig. 11.

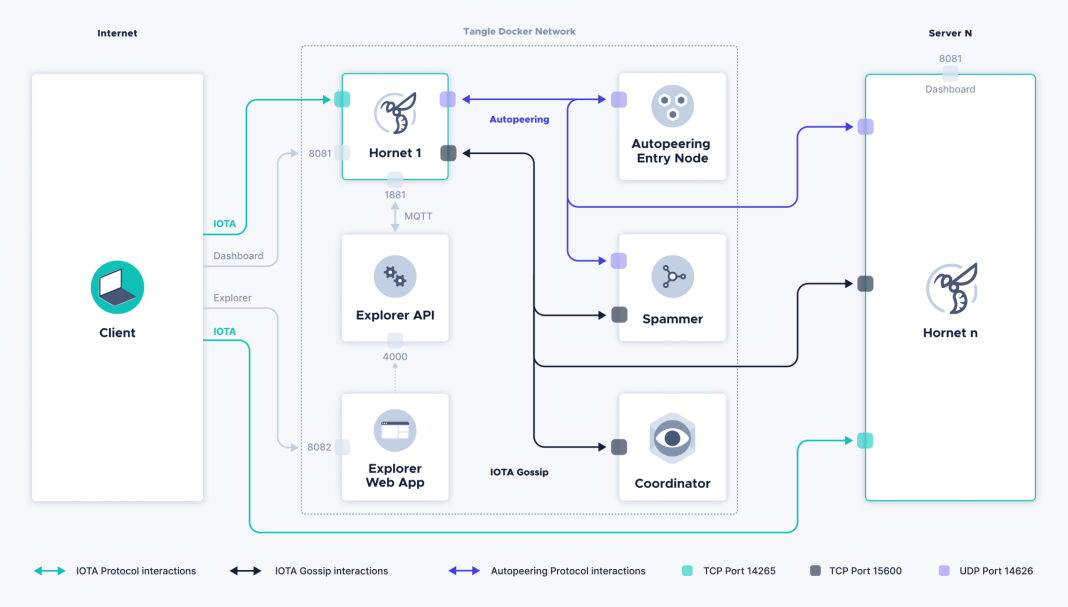

Figure 11: Private tangle structure

The elements shown in the figure are:

- **Client**: any user who wants to connect with the Tangle or its nodes, in this case, it will only be the personal computer.
- **Hornet nodes**: Hornet is an open-source IOTA node software that allows users to participate in the IOTA network. In the built system, two Hornet nodes are deployed to form a network of nodes. These nodes communicate with each other to maintain

the private Tangle and facilitate transactions. The Hornet nodes perform the following functions:

  - *Store and propagate transactions*: They receive new transactions and propagate them to other nodes in the network to achieve consensus on the validity of transactions.
  - *Validate transactions*: They verify the integrity and correctness of transactions by checking the signatures, balances, and other criteria defined by the consensus rules.
  - *Maintain the Tangle*: They store the history of transactions and build the Tangle structure by connecting transactions and referencing previous transactions.
  - *Participate in consensus*: They contribute to the consensus mechanism of the private Tangle by reaching agreement on the valid set of transactions.

- **Coordinator**: The Coordinator is a centralized component within the IOTA network that plays a significant role in maintaining the security of the network during its early stages of development. It's important to note that the Coordinator is not present in a fully decentralized IOTA network but is utilized in the initial phases to provide an extra layer of protection. The primary functions of the Coordinator are as follows:

  - *Issuing milestones*: The Coordinator periodically issues milestones, which are specific transactions that are considered as trusted reference points by the network. These milestones help establish a linear ordering of transactions in the Tangle and serve as a basis for determining the overall network consensus.
  - *Promoting transactions*: The Coordinator also assists in promoting and confirming transactions by indirectly approving them through the issuance of milestones. This helps to prevent certain types of attacks, such as double-spending or transaction manipulation, during the early stages of the network.
  - *Protection against attacks*: By having a centralized entity issuing milestones, the Coordinator provides an additional layer of security to the network, mitigating the risk of attacks and ensuring the reliability of transaction confirmations.

- **Spammer**: A node that periodically sends messages to your Tangle, thus enabling a minimal message load to support transaction approval as per the IOTA protocol.

- **Autopeering node**: A node that helps extra hornet nodes to be peered easily in a private tangle.

6.4.2 Wasp node

Once the private Tangle was successfully deployed on the personal computer, the subsequent step involved adding the essential nodes to support smart contracts within the IOTA ecosystem. These specialized nodes are known as *Wasp* nodes.

The primary role of the Wasp nodes is to enable the creation of smart contract chains connected to the Tangle. These chains serve as a means to record and manage the transactions associated with each specific smart contract. By attaching these chains to the Tangle, the Wasp nodes facilitate the tracking and execution of smart contract-related activities within the private Tangle environment.

The next step, once the Wasp node was connected to the private Tangle, was to deploy a chain to store the smart contracts. This chain also serves the purpose of connecting EVM accounts to the private Tangle, enabling them to interact with assets and smart contracts.

To deploy the chain, the *wasp-cli* was utilized. This is a bash program that facilitates the interaction with the running Wasp node. The following steps were followed using this tool:

1. Initialize *wasp-cli* to create an account from which the chain can be deployed.
2. Add the Wasp node information to *wasp-cli*.
3. In the present scenario, only one Wasp node was operational due to the development environment. However, in a real-world environment with multiple Wasp nodes, the subsequent step would involve peering them and establishing mutual trust.
4. Request funds from the testnet. As the Tangle is private, funds can be acquired on-demand. These funds are necessary for deploying both chains and smart contracts.
5. Deploy the chain with a designated ID and a list of Wasp nodes that will serve as validators. In this particular case, only one Wasp node was considered.

6.4.3 MG authority - deployer

Once the platform for deploying smart contracts was established, the next step involved creating the MG authority, which is responsible for deploying the SC_{MG}.

To achieve this, a wallet account needed to be created. Metamask was utilized to visualize the balance of this initial account. However, it's important to note that for individuals who will be interacting with the smart contracts, only a pair of public/private addresses are required, and therefore, for the other actors participating in the system, no Metamask account will be created.

To connect Metamask with the private Tangle network, the EVM-JSONRPC server integrated with the deployed chain was utilized. This integration facilitated the seamless interaction between Metamask and the private Tangle network..

By leveraging the EVM-JSONRPC server, the platform ensures a smooth user experience by bridging the gap between the Ethereum ecosystem and the private Tangle network.

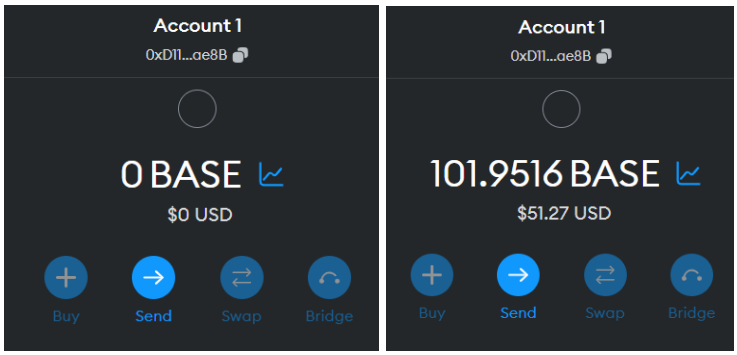

Figure 12: Metamask account before and after adding funds

In Fig. 12 can be seen the created account before and after adding the funds via *wasp-cli*. This test demonstrates the successful link between the EVM account and the private tangle.

7 Evaluation

To thoroughly test and evaluate the functionality of the implemented smart contracts, a comprehensive suite of Python scripts has been developed. These scripts serve the purpose of simulating a realistic environment that closely resembles a complete execution of the three stages proposed in our system.

The Python scripts were designed to replicate the various aspects and interactions within each stage of the system. By simulating real-world scenarios, the scripts enable comprehensive testing and evaluation of the smart contracts' capabilities and their integration with other system components.

For each step of each price calculation and energy trading stage, a test will be performed changing the number of actors participating in the actions.

Although implemented and fully functional, the debt liquidation stage evaluation is not performed in this work.

It's important to highlight that in the presented tests, the energy amounts and prices have been randomly determined. This deliberate choice to use random values serves the purpose of testing the system with a wide range of scenarios, including non-realistic ones. By incorporating a diverse set of values, the system can be thoroughly examined and evaluated for robustness and adaptability.

Additionally, the tests have been carefully designed to ensure that the expected consumed electricity is always met. This design decision contributes to the reliability and accuracy of the system. By validating that the expected energy consumption aligns with the actual energy consumed, the tests help to verify the correctness of the implemented mechanisms.

7.1 Price calculation stage

The first step evaluated is the price calculation stage, this section provides the results of each procedure.

7.1.1 Deployment

In this initial test, the first step is to deploy the SC_{MG}, which is performed by the account mentioned in the previous section. To verify the successful deployment of the contract, we can observe the output that confirms the transaction was processed without any errors. This indicates that the contract was successfully deployed.

```
1 Running 'scripts/price_calculation_stage.py::main'...
2 Transaction sent: 0
    x38f3e3311dda357d77ec58836f224ebe936a614c26d58c8b0932ff6ce3bce723
3   Gas price: 10.0 gwei   Gas limit: 2716432   Nonce: 751
4   MainSmartGrid.constructor confirmed   Block: 5963   Gas used: 2469220
      (90.90%)
5   MainSmartGrid deployed at: 0xfcB51711A1B359C0af49BABeE121eA54e1BC19B7
```

By observing the output and confirming the successful processing of the deployment trans-

action, we can proceed with confidence to the next steps of the testing process, knowing that the SC_{MG} has been deployed correctly and is ready for further interaction.

7.1.2 Set expected energy consumption

In the subsequent test, the objective was to randomly set the expected energy for each of the 24 time slots. The range of values chosen for the expected energy was between 60 and 100. This decision was made based on the supposition that, despite the presence of prosumers, the overall energy consumption would generally surpass the energy production among the users.

To ensure the accuracy and correctness of the function being tested, a validation step was included in the testing process. Upon completion of the test, the result of a getter function from the smart contract was printed to verify the stored expected energy values. This additional step provides a means to validate the functionality and integrity of the smart contract, as demonstrated in the following output.

```
1 Expected Energy Consumption: [63, 72, 96, 92, 88, 98, 67, 77, 65, 93, 67, 80,
      93, 83, 94, 68, 81, 65, 85, 69, 94, 68, 71, 94]
```

7.1.3 Add producers

The subsequent step in the testing process involved evaluating the addition of producers to the system. To facilitate this, a file named "producers.txt" was created, containing 50 key pairs for the producers. It is important to note that generating and storing private keys in plain text is suitable only for a testing environment. In real-world scenarios, private keys should never be stored in such an insecure manner.

To enable the interaction of the producers with the smart contract, funds were allocated to their respective accounts. This allocation of funds ensures that the producers have the necessary resources to participate in the system and perform transactions within the smart contract.

The responsibility for adding each of these producers to the smart contract lies with the MG authority account. This account, designated as the governing entity of the MainSmart-Grid system, possesses the necessary privileges and permissions to carry out administrative operations, including the addition of producers.

The evaluation of this specific test is carried out during the step of adding offers. It is at this stage that the success of adding the producers becomes evident, providing proof that they have been successfully added and are authorized to contribute their offers to the system.

7.1.4 Add offers

In this comprehensive test, both the addition of producers and the subsequent addition of offers will be evaluated. The objective is to assess the system's functionality in handling these operations. Each producer will be tasked with adding an offer in each of the 24 time slots, with random amounts ranging from 12 to 40 and random prices ranging from 1 to 10.

These specific value ranges have been carefully selected to ensure that, for each time slot, multiple producers are represented.

To assess the correctness of the output, the results of a getter function, which returns all the offers in each time slot, are printed to a text file. It is expected that the offers within each time slot are ordered by their respective prices. This ordering facilitates efficient decision-making processes when selecting offers based on price considerations.

To provide a clearer example, the following output showcases the correctness of the system's behavior with a scenario involving five producers and their respective offers. The output demonstrates the ordered offers in the first two time slots, reflecting the system's ability to correctly process and organize the offers based on price. It also reflects the capability of producers to add their offers, which demonstrates the correctness of the authorization provided by the add producer function.

```
1 Time slot: 0, Amount: 18, Price: 3, Producer: 0
      xBa60Bcd7bd5dafB5Ce45b93DD621e0368d1A5f0A
2 Time slot: 0, Amount: 40, Price: 3, Producer: 0
      xA38bB38Bf6c640aC18a2A02533C9d7B10229d557
3 Time slot: 0, Amount: 36, Price: 7, Producer: 0
      x6FE5E8d38C1624a121D96Ac59Ed1c986B0435731
4 Time slot: 0, Amount: 16, Price: 9, Producer: 0
      xe049F2302D5b80720a5bd7e7dB97dfA25858e1fF
5 Time slot: 0, Amount: 37, Price: 10, Producer: 0
      x353CeA25e024543FC0034fB2f6F5c646E1a1C21b
6 Time slot: 1, Amount: 20, Price: 2, Producer: 0
      x353CeA25e024543FC0034fB2f6F5c646E1a1C21b
7 Time slot: 1, Amount: 24, Price: 3, Producer: 0
      xe049F2302D5b80720a5bd7e7dB97dfA25858e1fF
8 Time slot: 1, Amount: 40, Price: 9, Producer: 0
      x6FE5E8d38C1624a121D96Ac59Ed1c986B0435731
9 Time slot: 1, Amount: 16, Price: 9, Producer: 0
      xA38bB38Bf6c640aC18a2A02533C9d7B10229d557
10 Time slot: 1, Amount: 26, Price: 10, Producer: 0
      xBa60Bcd7bd5dafB5Ce45b93DD621e0368d1A5f0A
```

7.1.5 Gas usage

Gas usage analysis plays a crucial role in understanding the resource consumption of smart contracts within a smart grid system. In a smart grid, various functions and operations are executed through smart contracts to facilitate efficient energy management, billing, and transactions. Gas usage analysis allows us to evaluate the computational cost of these functions and assess their scalability as the system expands.

In this analysis, we will examine the gas usage of different functions in a smart contract designed for a smart grid system. Specifically, in this first analysis concerning the price calculation stage, we will focus on the gas usage in the SC_{MG} in relation to the number of producers involved in the system. By studying the gas consumption patterns, we can gain insights into the efficiency and scalability of the smart contract.

To conduct the analysis, we have collected data representing the gas usage of specific functions in the smart contract at different producer counts. We have recorded the gas usage for different numbers of producers ranging from 1 to 50.

The range of producers was decided in order to be enough able to find a trend in the gas usage, but without surpassing the available computational capacity of the evaluation environment.

To understand the relationship between the gas usage and the number of producers, we will examine the trends exhibited by each variable. We will look for patterns such as constant gas costs, increasing or decreasing gas costs, or any other noteworthy observations. By comparing these variables with the number of producers, we can identify potential correlations and draw insights into the scalability of the smart contract functions.

7.1.5.1 Set expected

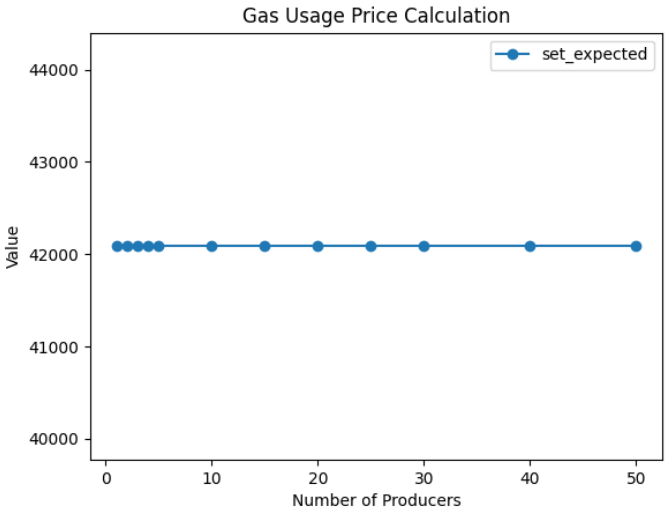

Figure 13: Set expected

This function just sets each of the 24 positions of an array to the expected number of energy consumption.

Since the size of the array is fixed at 24, the gas cost of executing the set_expected function would remain constant regardless of the number of producers. This is because the gas cost for assigning a value to an array element does not depend on the array's size.

The gas cost of assigning a value to an array element in a smart contract typically involves a fixed amount of gas, which covers the basic operations required for updating the element. This includes operations like accessing the memory location of the array element, copying the new value to that location, and updating the storage state.

As a result, the gas usage for the set_expected function remains the same at 42,085, irrespective of the number of producers. The gas cost is independent of the array size since it is fixed at 24 elements.

Therefore, the gas usage of the set_expected function does not change with the number of producers, indicating that the gas cost for assigning values to the array elements is consistent throughout the execution of this function.

7.1.5.2 Add producer

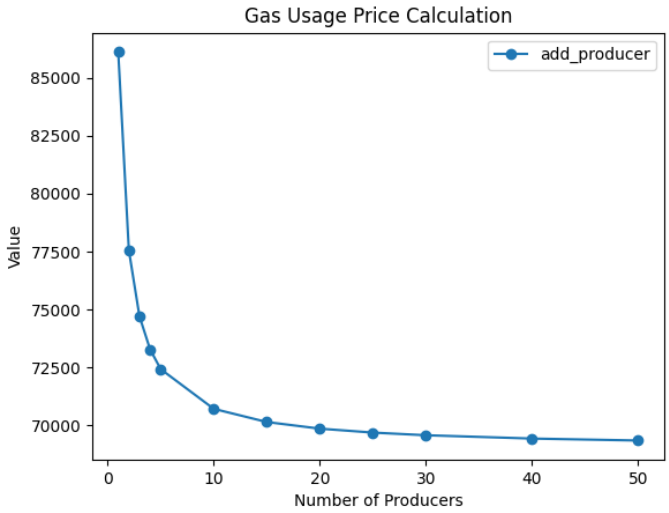

Figure 14: Add producer

In the gas usage analysis, we will now delve into the add_producer function, considering its role in adding a producer to a dynamic array within the smart contract designed for the smart grid system.

The add_producer function is responsible for appending a new producer to the dynamic array. It is important to note that adding elements to a dynamic array incurs a gas cost, and the cost can vary depending on the size of the array and whether the array needs to be resized to accommodate the new element.

In this analysis, we have recorded the gas usage for the add_producer function at different numbers of producers. The data provided includes the gas usage for adding producers when the number of producers is 1, 2, 3, and so on, up to 50.

Upon examining the data, we observe that the gas usage of the add_producer function decreases as the number of producers increases. This can be attributed to the way dynamic arrays work in Solidity, the programming language for smart contracts.

When the dynamic array is initially empty, adding the first element incurs an additional gas cost due to the internal resizing and allocation of memory for the array. This initial cost is typically higher than the subsequent additions. As more elements are added, the dynamic array is already allocated with sufficient memory, resulting in lower gas costs for appending additional elements.

To provide a comprehensive analysis, the gas usage values for the add_producer function are a mean of each addition. This means that the reported gas usage reflects an average cost considering multiple additions of producers to the dynamic array.

Based on these observations, we can conclude that the mean gas usage of the add_producer

function decreases as the number of producers increases. This indicates that the cost of creating the dynamic array becomes more insignificant the more producer are added, since the addition of new users is a constant cost.

Understanding the gas usage trends of the add_producer function is valuable for optimizing the smart contract within the smart grid system. It suggests that the smart contract design is efficient in terms of appending producers to the dynamic array, and as the system scales with more producers, the gas costs for adding new producers become more manageable.

7.1.5.3 Add offers

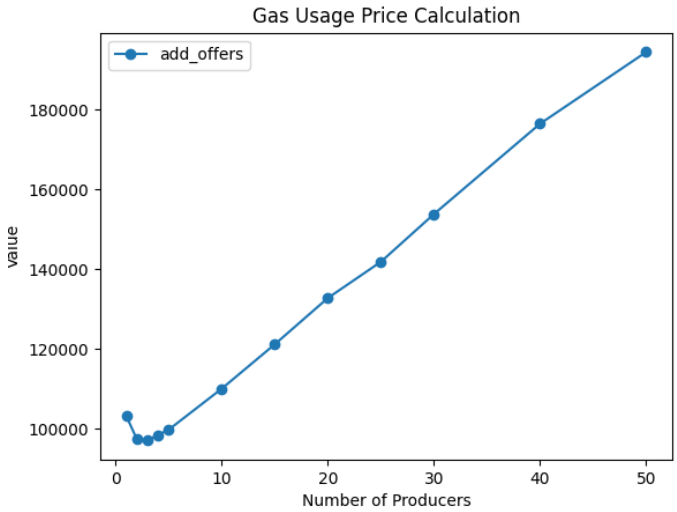

Figure 15: Add offers

In the gas usage analysis, let's now focus on the add_offer function, which adds an offer to a dynamic array in the smart contract designed for the smart grid system. It is important to note that this function maintains the order of the elements within the array.

The add_offer function performs two key operations: appending the offer to the dynamic array and ensuring that the array remains ordered. The gas usage for this function will be influenced by these two operations.

Similar to the previous analysis, we have collected data representing the gas usage of the add_offer function at different numbers of producers. The data provided includes the gas usage for adding offers when the number of producers is 1, 2, 3, and so on, up to 50.

Upon examining the data, we can observe the following trends and considerations:

- **Dynamic Array Resizing**: As mentioned earlier, when adding elements to a dynamic array, the initial addition might incur a higher gas cost due to resizing and memory allocation. Subsequent additions typically have lower gas costs since the dynamic array

has already been allocated sufficient memory. That's the reason why the gas usage with few producers is higher.

- **Ordered Array Maintenance**: The gas usage of the add_offer function could be influenced by the operations required to maintain the order of the elements in the array. This could involve shifting elements and comparing the values of the new offer with existing offers to determine the correct position.

To provide a comprehensive analysis, the reported gas usage values for the add_offer function represent the mean gas cost for each addition, considering multiple additions of offers to the dynamic array. This average cost accounts for both the resizing and memory allocation as well as the additional operations to maintain the order of the array.

Considering the above factors, we can draw conclusions from the analysis. The cost of the function is linear with the number of producers.

7.1.5.4 Begin auction

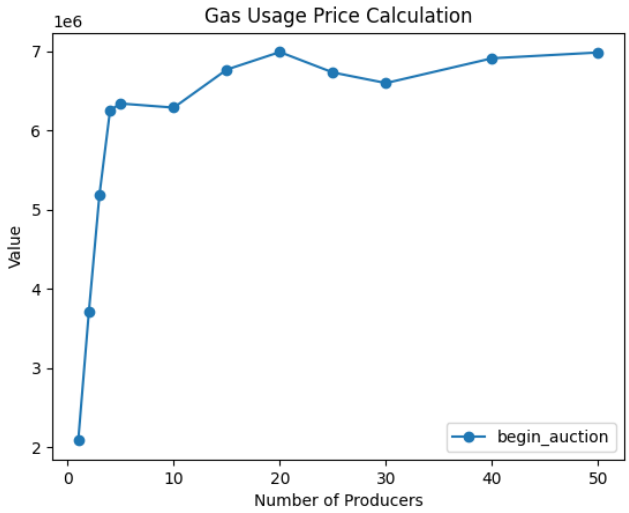

Figure 16: Begin auction

In the gas usage analysis, let's dive into the begin_auction function, which plays a crucial role in selecting the cheapest offers from the add_offer process to meet the expected energy consumption set at the set_expected process. It is important to note that the offers are already sorted, and the expected energy consumption remains consistent across all tests from 1 to 50 producers. Additionally, it's worth considering that the expected energy consumption might have an effect by capping the gas usage of this function.

The begin_auction function has the responsibility of selecting the "n" cheapest offers, where "n" represents the number of offers required to meet the expected energy consumption. The function operates based on the assumption that the offers are already sorted in ascending order, ensuring that the cheapest offers are at the beginning of the array.

Considering the above context, we can analyze the gas usage of the begin_auction function in the following manner:

Fixed Expected Energy Consumption: Since the expected energy consumption remains consistent across all tests, the number of offers required to meet the expected energy consumption will also remain the same. This aspect has the potential to cap the gas usage of the begin_auction function, as it will only need to select a fixed number of cheapest offers from the sorted array.

- **Gas Usage Efficiency**: The fact that the offers are already sorted in ascending order allows the begin_auction function to efficiently select the cheapest offers. This efficiency results from the ability to access and extract the desired offers directly from the beginning of the sorted array. Consequently, the computational steps required to select the offers and calculate the gas usage may be optimized.
- **Constant or Near-Constant Gas Usage**: Given that the expected energy consumption remains the same, and the offers are already sorted, we can expect the gas usage of the begin_auction function to remain relatively constant across different numbers of producers. The gas usage might fluctuate slightly due to computational overheads, but it should not experience significant increases.
- **Potential Capping Effect**: The capped expected energy consumption could potentially limit the number of offers that need to be selected. This capping effect further contributes to stabilizing the gas usage of the begin_auction function, as it reduces the computational complexity and the number of iterations required to meet the expected energy consumption.

Considering these factors, we can conclude that the gas usage of the begin_auction function is expected to remain constant or exhibit minimal fluctuations across different numbers of producers. The fixed expected energy consumption and the sorted nature of the offers facilitate an efficient selection process with limited computational overheads.

7.1.5.5 Producer gas usage

In Fig. 17, we can observe a clear similarity in the behavior of the producer variable with that of the add_offer function. This resemblance arises from the fact that the primary function executed by producers is indeed the add_offer function. Consequently, the gas usage exhibited by individual producers will also exhibit a linear relationship with the total number of producers involved.

Since producers solely perform the add_offer function, their individual gas usage closely mirrors the overall trend observed in the add_offer function. As the number of producers increases, the gas usage of each producer grows proportionally.

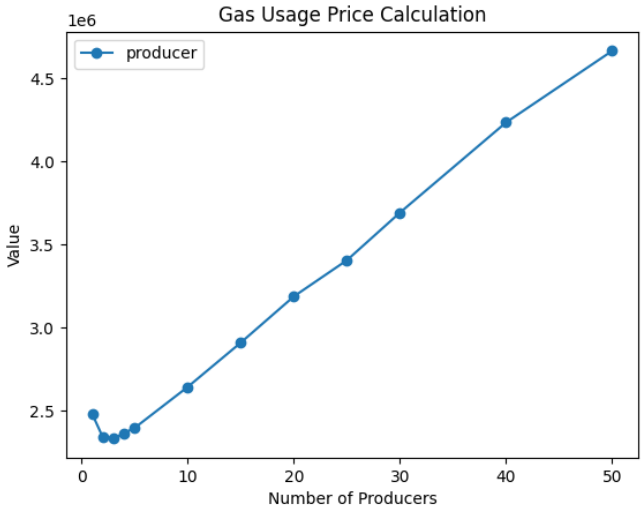

Figure 17: Producer gas usage

7.1.5.6 Main grid authority gas usage

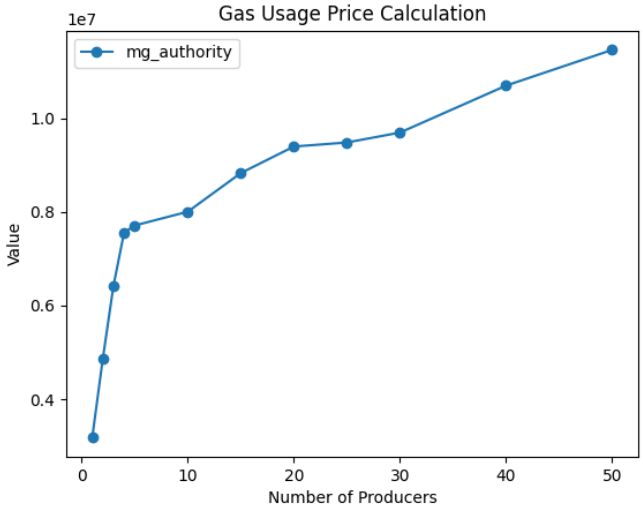

Figure 18: Main grid authority gas usage

The total gas usage by the main grid authority, as depicted in Fig. 18, exhibits interesting similarities with the combined gas usage of the "set_expected," "add_producer," and "begin_auction" operations. The main grid authority is responsible for performing these essential functions within the smart grid system.

The gas usage trend of the main grid authority aligns with the cumulative effect of executing the aforementioned operations. As the number of producers increases, the gas usage by the main grid authority actor also experiences a corresponding increase. This relationship stems from the fact that the gas cost of each operation performed by the main grid authority actor contributes to the overall gas usage.

7.1.5.7 Total gas usage

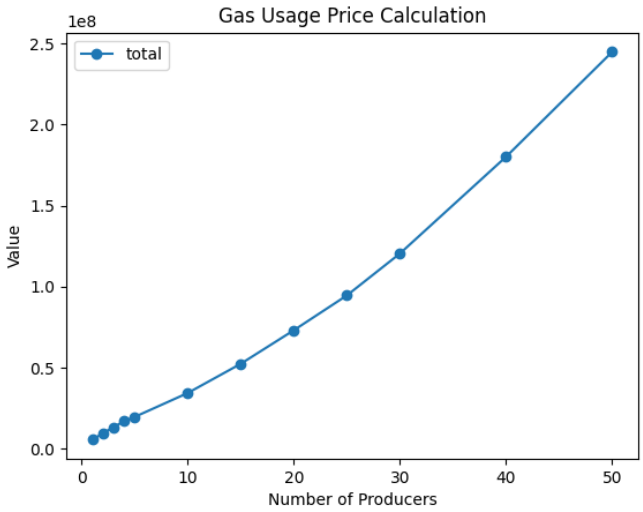

Figure 19: Total gas usage

The total gas usage, as depicted in Fig. 19, represents the cumulative gas consumption of all the operations performed within the smart contract for the smart grid system. It serves as a critical metric to assess the overall computational cost and efficiency of the system.

Upon analyzing the total gas usage data, we can conclude that as the number of producers increases, the total gas usage exhibits a consistent upward trend. This trend is expected as more operations are being executed with the addition of each producer. The gas consumption rises proportionally to accommodate the increased computational load.

7.2 Energy trading stage

In order to thoroughly validate this stage, a comprehensive simulation is conducted to emulate a 24-hour period of production and consumption involving multiple users within a group. This simulation aims to assess the system's performance under various parameters for smart contracts per group (from 1 to 10) and users per smart contract (from 1 to 25).

To ensure accurate results, the test has been meticulously designed to align the total consumption/production of users with the expected energy consumption defined in the main smart contract. It's worth noting that the main smart contract was deployed in the previous section, and it serves as a pivotal component of this validation process.

In order to verify the test's accuracy, all transactions executed by each user within their respective contracts are recorded and stored in a text file. Subsequently, these recorded transactions are compared against the transaction data stored in the main smart contract by the group smart contracts.

The primary objective of this comparison is twofold. Firstly, it aims to confirm that the sum of energy production/consumption across all the group smart contracts matches the expected consumption defined in the main contract. This step ensures that the collective energy exchange within the group aligns with the overall energy requirements set by the system.

Secondly, the transactions of each individual contract are cross-referenced to verify that they correspond to the total energy production/consumption of the users within that contract during each specific time slot. This meticulous analysis helps ensure the consistency and accuracy of energy exchanges at the granular level, further validating the behavior of the smart contracts.

By performing these rigorous tests and meticulous comparisons, we can ascertain the correct functioning and reliability of the system. It provides valuable insights into the system's ability to manage and coordinate energy production and consumption, ensuring that the contracts and transactions are executed accurately and in accordance with the defined parameters and expected outcomes.

7.2.1 Deployment

First of all, in order to test our system during the trading day, we use one of the previously deployed and initialized SC_{MG}, with all its parameters ready (expected energy consumption, producer offers, prices...).

Next, we add the coordinator to the SC_{MG} so the account is able to add the SC to the SC_{MG}. After that, the next step is to deploy the SC, add an operator to the SC so that it is able to add users (in this test, the operator will be the same account as the MG authority, in reality this would need a group signature) and add the SC to the SC_{MG} so it can communicate the energy needs. This is done for every SC to be deployed.

The fact that the coordinator is able to add the SC to the SC_{MG} tests the correctness of the authorization system provided by the add_coordinator function.

7.2.2 Add users

To test the addition of users, for each SC, we add users from the account defined as the operator. This process serves as a crucial test to verify the correct functioning of the add operator function, ensuring that users can be successfully added to the SC by the designated operator. The correct functioning of add users will be tested once the users are able to upload their offers.

7.2.3 Begin trading

In this test, the trading flags are set to true in each SC solely by invoking the respective function from the SC_{MG}. This step allows us to evaluate the system's capability to enable trading by activating the trading flags in each SC. The ability of the users to upload their offers in each SC will test the correctness of the function.

7.2.4 Trading

The "trading" test aims to simulate trading operations within an electricity trading system involving smart contracts, coordinators, and users. Its purpose is to evaluate the system's functionality by generating consumption values, facilitating electricity buying and selling based on these values, and verifying transactions.

To begin the test, a consumption array is generated to match the expected energy consumption provided by the main smart contract (SC_{MG}). This array ensures that the system operates based on the anticipated energy consumption for each time slot.

Once the consumption or production values for each smart contract (SC) and each user (U) within the SC have been determined for each time slot, the trading process commences.

During each time slot, users upload their offers using the sellElectricity and buyElectricity functions. This step tests the correct functionality of the addUser function within the smart contract. By allowing users to submit their offers, the system verifies that the addUser function properly adds new users to the smart contract.

Subsequently, the coordinator undertakes the vital task of verifying each transaction. In this test scenario, for simplicity and testing purposes, all transactions are assumed to be honest. The transaction verification step ensures the accuracy and integrity of the trading process.

Upon successful upload and verification of all transactions within a time slot, the coordinator proceeds to change the time slot for each individual smart contract (SC). Additionally, the main smart grid (MG) authority changes the time slot for the main smart contract (SC_{MG}).

By changing the time slots for the smart contracts, the system progresses to the next time period, enabling the trading operations to continue seamlessly.

7.2.5 Gas usage

In this section, a comprehensive study has been conducted to assess the gas usage in the energy trading stage. Similar to the evaluation section for price calculation, this study provides valuable insights into the gas consumption associated with various procedures involved in energy trading.

Within this section, a detailed breakdown of gas usage is presented, shedding light on the consumption patterns of each individual actor participating in the energy trading process. By examining the gas usage of different actors, we gain a comprehensive understanding of the resource demands at play.

Moreover, this study delves deeper into the gas usage associated with each type of smart contract utilized within the energy trading framework. By analyzing the gas consumption specific to different types of smart contracts, we can identify potential areas for optimization and efficiency enhancements.

Each subsection within the study will be accompanied by informative plots that visually depict the impact of altering the number of smart contracts per group and the number of users per smart contract on gas usage. These plots serve as valuable tools for analyzing the relationship between these variables and resource consumption.

7.2.5.1 New operator and new coordinator

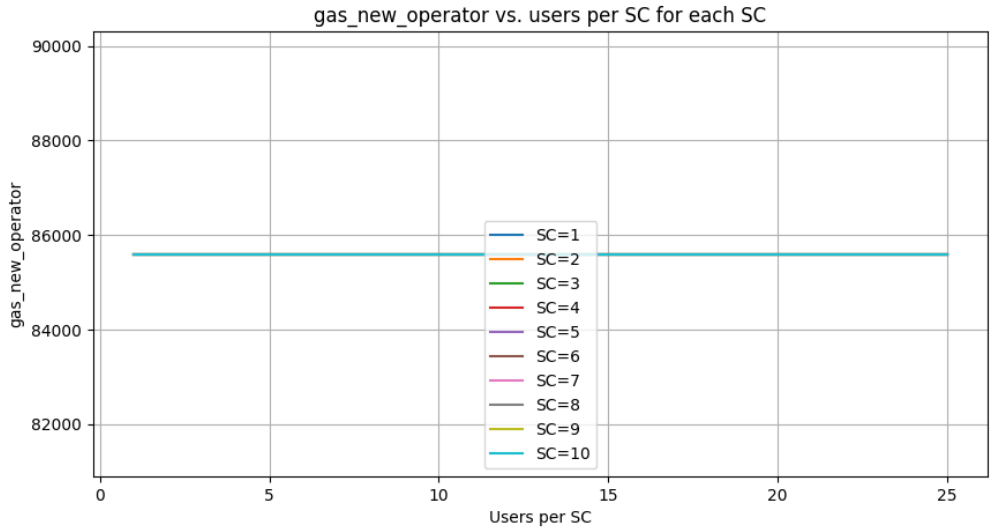

Figure 20: New operator

The cost of these procedures, as depicted in Fig. 20 and 21, appears to be constant in all executions, varying the number of SC per group and U per SC doesn't affect the gas usage. This constancy arises from the fact that these functions are only called once. However, it is important to note that in reality, the cost of adding new coordinators may vary depending on the number of groups present in the system. The evaluation test, however, focuses on a single group. Given that operators share a common group signature, it is sufficient to add only one operator to each SC. Therefore, the depicted plot of adding an operator accurately represents

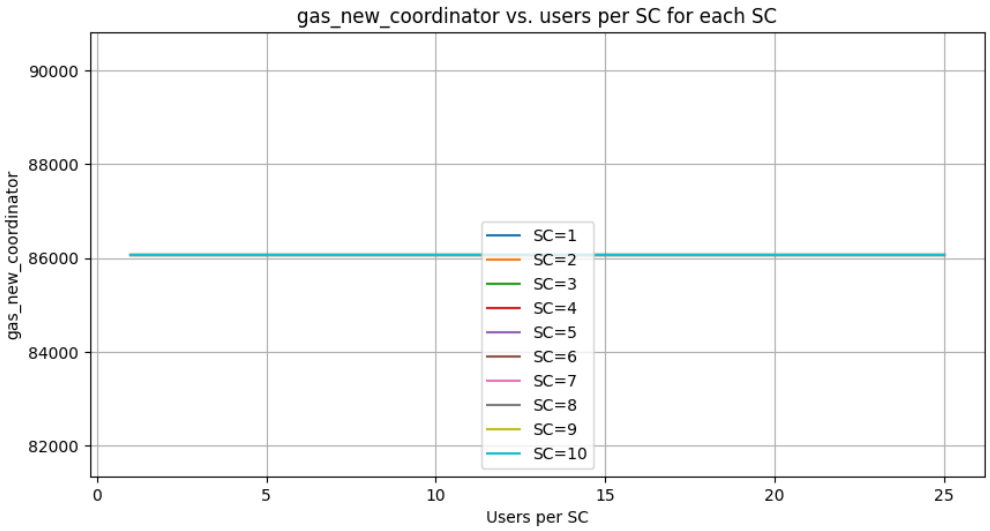

Figure 21: New coordinator

the expected behavior in a real-world scenario.

In any case, these procedures simply involve appending a new address to an array, indicating that the cost should still exhibit a linear behavior or a behavior similar to adding actors in the price calculation stage. In that stage, it was observed that the average cost decreased as more actors were added.

7.2.5.2 Add user

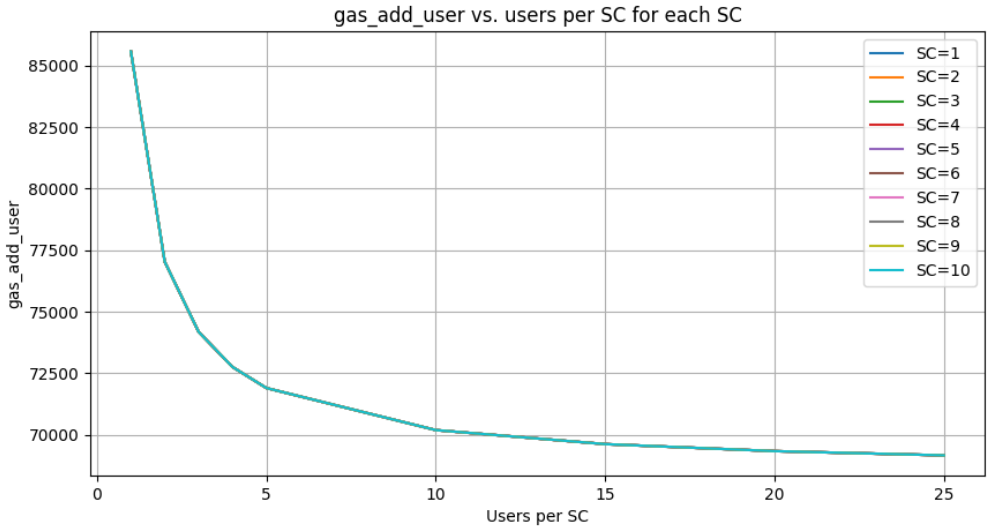

Figure 22: Add user

Figure 22 illustrates the gas usage associated with adding a new user to a smart contract (SC). This procedure involves simply adding the user to the U array, which exhibits a gas usage behavior similar to adding users in the price calculation stage. As mentioned in earlier sections, the decreasing gas usage can be attributed to the dynamic array behavior in Solidity.

As depicted in the figure, the gas usage depends only on the number of users, not the number of SC.

7.2.5.3 Begin trading

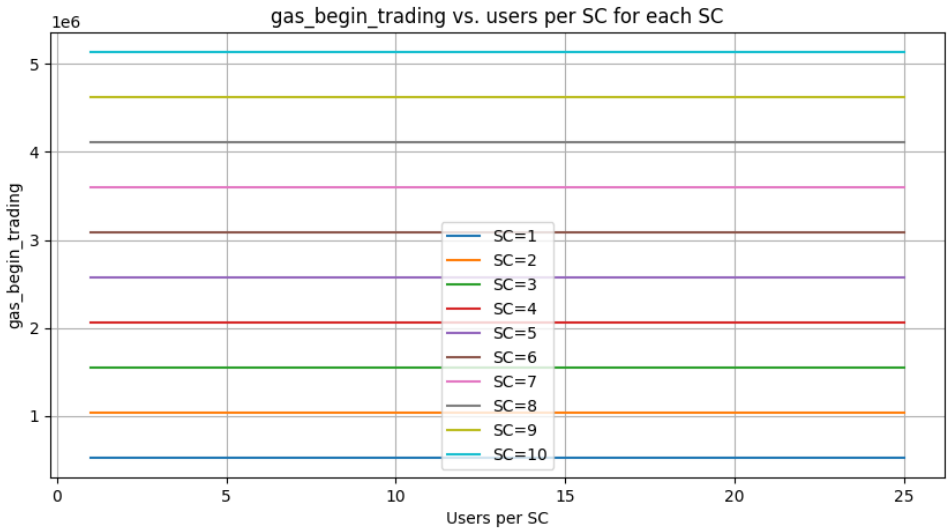

Figure 23: Begin trading

As observed in Figure 23, the gas usage of this procedure remains constant as the number of users (U) increases, but exhibits a linear trend when the number of smart contracts (SC) increases. This behavior can be attributed to the fact that this function invokes the "begin_trading" procedure in the SC_{MG}, which in turn calls the same procedure for each registered smart contract (SC). Consequently, this cascading effect leads to an increase in gas usage when the number of smart contracts is increased.

7.2.5.4 Trading and verification

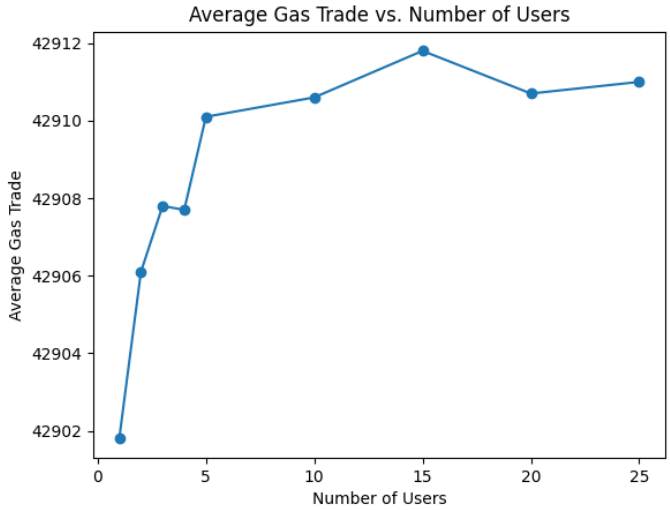

Figure 24: Trading

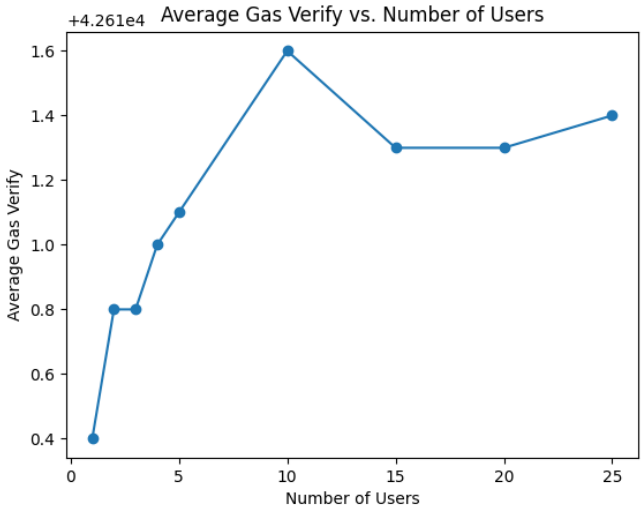

Figure 25: Verify

As illustrated in Fig. 24 and 25, both the trading and verification procedures exhibit a behavior analogous to the begin_auction phase in the price calculation stage. This similarity arises from the fact that, much like the price calculation stage, the energy consumption at each time slot remains consistent across all executions. Consequently, the number of trading transactions required to meet the expected energy consumption remains constant, regardless of the number of users involved.

7.2.5.5 Change time

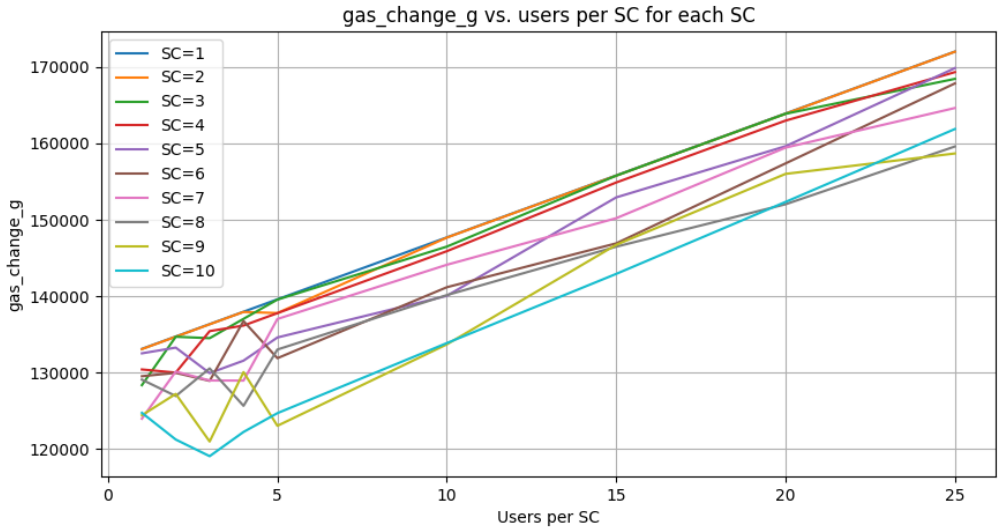

Figure 26: Change time group smart contract

Fig. 26 and Fig. 27 illustrate two distinct behaviors for the same procedure in the master contract (SC_{MG}) and individual smart contracts (SC). The discrepancy arises due to the differing functionality of the procedure in each contract.

In the main contract (SC_{MG}), the function simply increments the time slot variable by one.

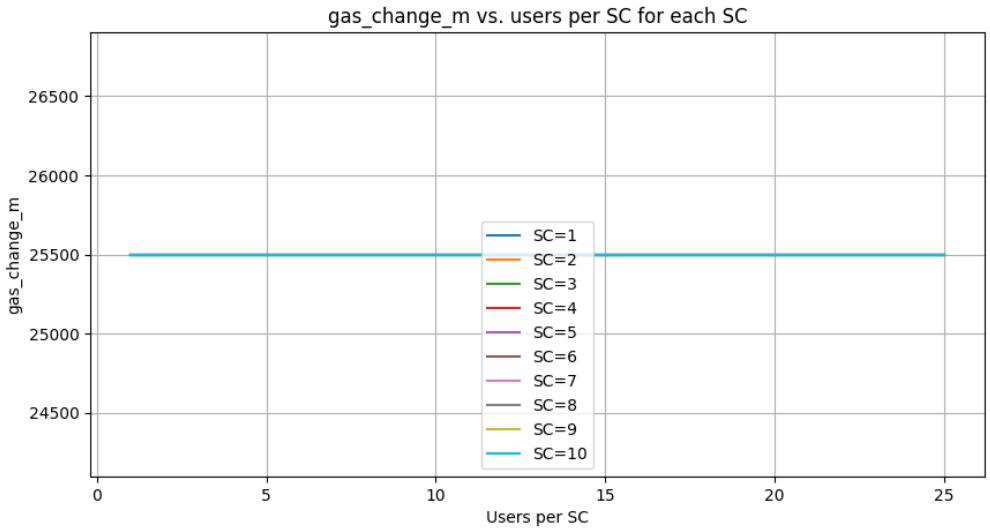

Figure 27: Change time main smart contract

As a result, the gas usage remains constant, regardless of the number of users or transactions. This behavior can be attributed to the simplicity of the operation performed within the function.

On the other hand, in each individual smart contract (SC), the procedure involves calculating the difference between the total energy consumed and produced. To accomplish this, each SC needs to iterate through each transaction, resulting in a linear relationship between gas usage and the number of users. This linear trend is expected because the computational workload increases proportionally with the number of users and transactions that need to be processed.

7.2.5.6 Individual actors

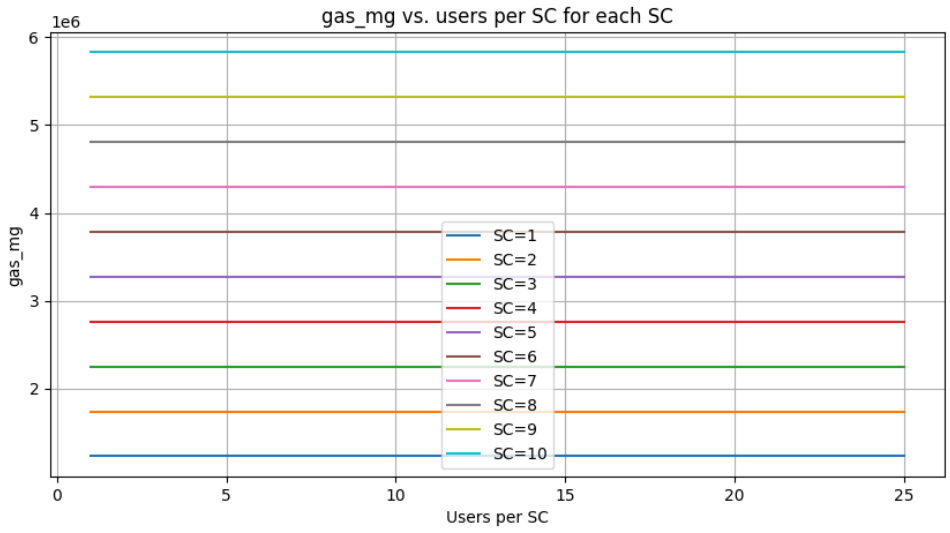

Figure 28: Smart grid authority total gas

Fig. 28 demonstrates that the total gas usage by the smart contract (SC) authority is solely dependent on the number of smart contracts (SC). This dependency is due to the fact that, as indicated in the previous plots, every procedure involving the authority of the master contract (MG) incurs a constant cost with respect to the number of users. The only procedure that exhibits an increase in gas usage as the number of smart contracts increases is the "begin_auction" procedure.

The constant cost observed for procedures involving the MG authority, irrespective of the number of users, indicates that the computational workload remains stable in relation to the user count. On the other hand, the "begin_auction" procedure, which is responsible for initiating auctions, experiences a linear increase in gas usage as the number of smart contracts grows.

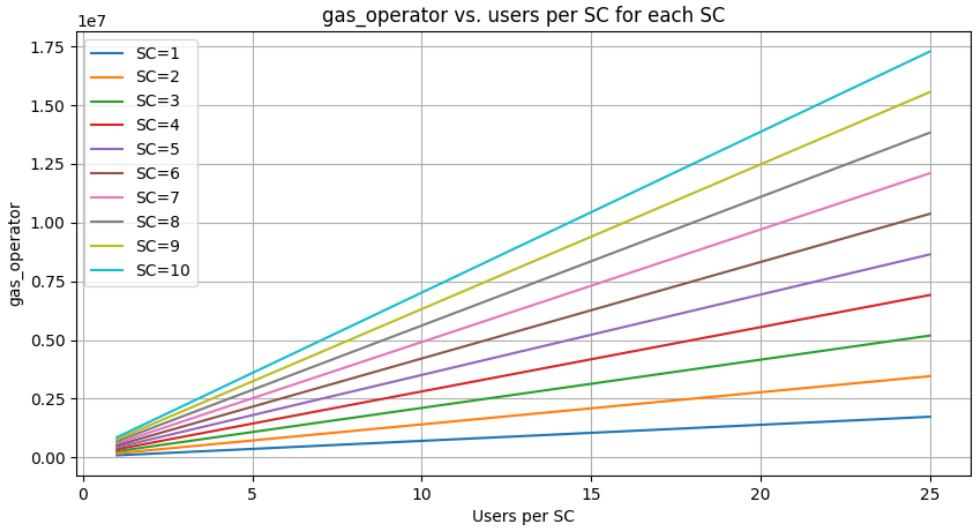

Figure 29: Operator total gas

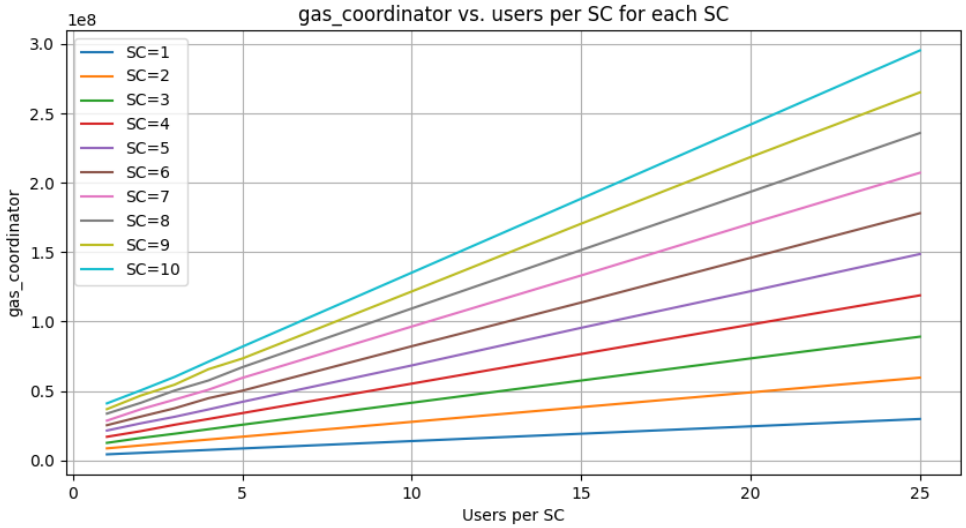

Figure 30: Coordinator total gas

Both Fig. 29 and Fig. 30 exhibit similar behaviors, demonstrating that the gas usage by both operators and coordinators is linear with respect to both the number of smart contracts (SC)

and the number of users (U). However, a noticeable difference can be observed: when the number of smart contracts is increased, the gas usage line has a steeper slope compared to when the number of users is increased. This indicates that the gas usage is more significantly influenced by the number of smart contracts rather than the number of users.

The steeper slope for gas usage with increasing smart contracts implies that the computational workload and associated gas costs for operators and coordinators are more strongly impacted by the number of smart contracts in the system. As the number of smart contracts grows, the gas usage increases at a faster rate compared to the incremental increase in gas usage caused by a larger user count.

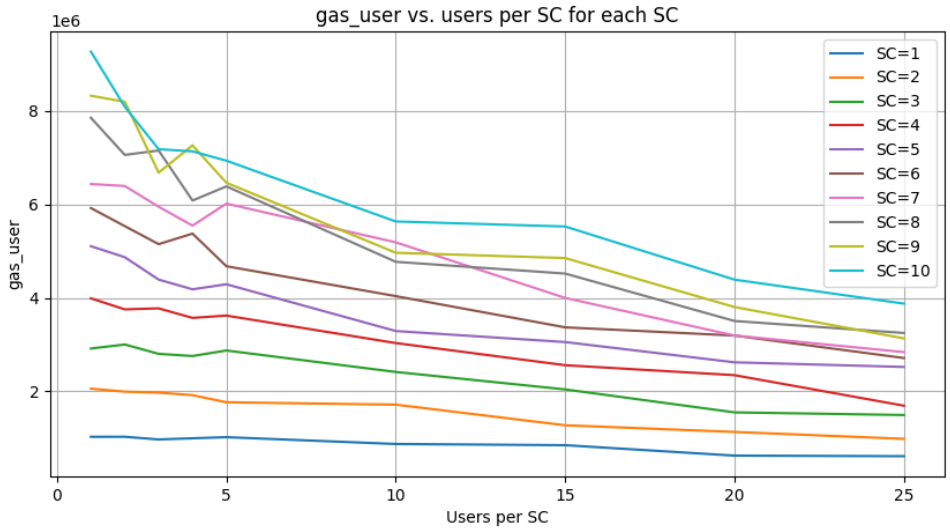

Figure 31: User total gas

As evident from Figure 31, the gas usage of users appears to decrease when the number of smart contracts (SC) per group increases and the number of users also increases. However, this observation can be attributed to the fact that, as previously mentioned, the number of trade transactions remains constant. Therefore, when divided by a larger number of users, the apparent gas usage appears to decrease. It is important to note that in a real-world scenario with a higher number of trade transactions, the gas usage should remain constant. This is because the operation performed by users is simply adding the transaction to a dynamic array.

7.2.5.7 Contract type

Similar to the MG authority, the gas usage in the main contract (SC_{MG}) is solely influenced by the number of smart contracts (SC) as seen in Fig. 32. This means that the computational workload and associated gas costs in the SC_{MG} remain constant regardless of the number of users.

Fig. 33 illustrates the gas usage in each individual smart contract (SC), and it demonstrates that the gas usage is primarily influenced by the number of users (U) rather than other factors such as the number of smart contracts.

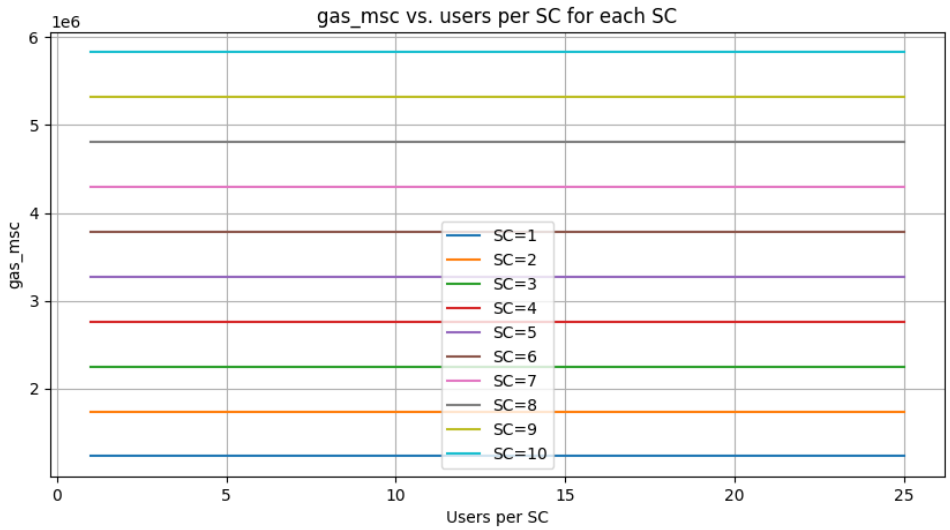

Figure 32: Main smart contract total gas

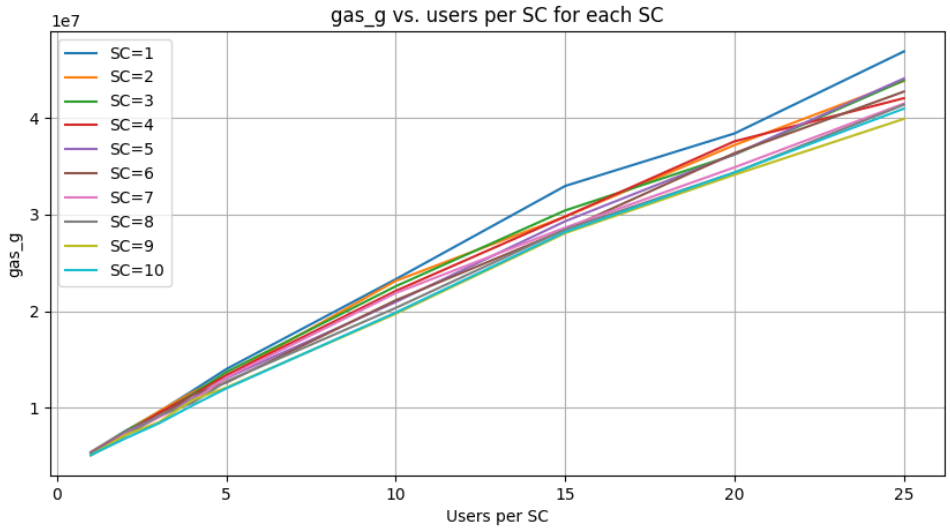

Figure 33: Group smart contract total gas

7.2.5.8 Total

From Fig. 34, we can analyze the total gas usage during a trading day and observe its dependency on both the number of smart contracts (SC) and the number of users (U). However, similar to the gas usage by the coordinator and the MG authority, it is evident that an increase in the number of smart contracts leads to a steeper increase in gas usage when the number of users is also increased.

This gas usage analysis reveals that while adding more smart contracts can enhance user privacy, it also results in a significant increase in the computational cost of the overall system.

Furthermore, it can be inferred that an increase in the number of users primarily affects the individual smart contracts (SC), reducing the likelihood of a bottleneck in the main contract

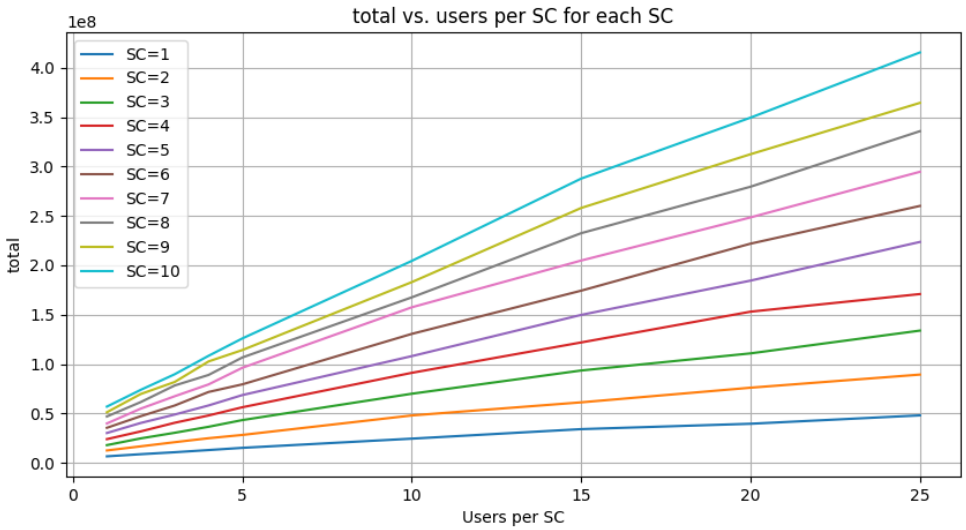

Figure 34: Total gas usage

(SC_{MG}) when the number of users becomes significantly higher. This observation suggests that the workload is effectively distributed among the smart contracts, which aligns with real-world scenarios where the number of users is typically much higher.

However, it is important to consider that an increase in the number of users would also require a larger number of groups (G) and consequently more smart contracts (SC). As a result, there may come a point where the number of smart contracts becomes too large, potentially causing challenges within the master contract (SC_{MG}). To address this, one potential solution is to introduce intermediate levels of smart contracts between the individual smart contracts (SC) and the master contract (SC_{MG}) to distribute the computational workload among a greater number of smart contracts.

Overall, the evaluation of the gas usage in both the price calculation and energy trading stages has yielded satisfactory results. None of the procedures exhibit exponential increases with any parameter, indicating a good level of scalability within the system. The system appears to handle an increasing number of users effectively, with gas usage growing linearly or with a manageable rate.

It is important to note that the scalability is contingent on the number of smart contracts (SC) being relatively smaller than the number of users (U). As long as this balance is maintained, the system can handle the increasing load efficiently.

8 Conclusions

In this work, we propose an innovative energy trading system that serves as a bridge between the traditional grid and a smart grid, addressing the challenges associated with increasing demand, rising prices, and the limited availability of renewable energy sources. Our system empowers users to both consume and produce electricity using distributed energy resources (DERs), facilitating the transition towards a more sustainable energy landscape.

Our system facilitates electricity trading among users and with the main grid, employing a shared pricing mechanism that offers improved profitability compared to existing models for DER electricity production. To ensure fairness and protect user privacy, we have introduced a coordinator actor who validates each transaction within the system.

The implementation of our smart grid system involves a series of actors and is divided into three stages of the energy trading process. First, the price calculation stage involves producers making offers and an auction being performed. Next, the trading stage allows users to both produce and consume electricity, engaging in transactions with other prosumers or the main grid. Finally, the debt liquidation stage involves calculating debts and facilitating the transfer of corresponding funds.

To meet the requirements of a smart grid, our proposal leverages blockchain technology and utilizes smart contracts. This provides the system with enhanced reliability, security, privacy, and availability. Furthermore, to improve privacy and scalability, we organize users into groups, with each group interacting with multiple smart contracts to conduct electricity trading while also concealing individual energy usage.

To evaluate the effectiveness of our system, we implemented it in a controlled environment using a private IOTA tangle and EVM smart contracts. Through simulations involving varying parameters such as the number of producers, the number of smart contracts per group, and the number of users per smart contract, we were able to test the system's functionality and analyze its computational cost by examining gas usage.

Our comprehensive gas usage analysis demonstrated the system's relative scalability, as we examined different trading scenarios with varying numbers of smart contracts and users. We confirmed that the system operates efficiently without encountering computationally exponential procedures.

Furthermore, in the evaluation section, we provided evidence of the system's accuracy and transparency. The ability to review all transactions and validate their correctness enhances trust among participants. Additionally, we successfully validated the price calculation mechanism, further reinforcing the system's reliability.

In conclusion, our system successfully achieved and underwent rigorous testing to meet all functional requirements. The evaluation section further confirmed the system's accuracy, scalability, and reliability. Our proposed energy trading system paves the way for a more sustainable and efficient smart grid, facilitating the transition towards a greener and more reliable energy future.

8.1 Future work

The future work for this project encompasses several key areas of improvement and expansion. One crucial step involves conducting a larger-scale evaluation using a raspberry grid, which closely resembles the computational power of a smart meter. By simulating parallel interactions with smart contracts and testing performance issues, the system's scalability and efficiency can be ensured. This evaluation will also include a thorough examination of the debt liquidation stage, with a focus on studying important parameters such as fees to optimize the overall process.

Another important aspect of future work will involve delving deeper into the privacy section. The objective will be to analyze information revelation and potential patterns that could be exploited by malicious users. By conducting an in-depth study on information entropy and system vulnerabilities, effective privacy mitigation strategies can be developed. This will enhance the system's ability to protect user data and maintain privacy in the context of energy trading transactions.

Additionally, it is crucial to evaluate and optimize the performance of the communication system of the blockchain in a non-local environment. This evaluation will help assess the efficiency of data transmission and synchronization under realistic conditions. By testing the system's robustness in a distributed network setup, resembling real-world scenarios, potential issues can be identified and addressed to ensure smooth communication and reliable operation of the blockchain network.

Bibliography

[1] IRENA (2021). *Renewable Power Generation Costs in 2020, International Renewable Energy Agency, Abu Dhabi*. URL: https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2021/Jun/IRENA_Power_Generation_Costs_2020.pdf?rev=c9e8dfcd1b2048e2b4d30fef671a5b84.

[2] Anak Agung Gde Agung and Rini Handayani. "Blockchain for smart grid". In: *Journal of King Saud University - Computer and Information Sciences* 34.3 (2022), pp. 666–675. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2020.01.002. URL: https://www.sciencedirect.com/science/article/pii/S1319157819309000.

[3] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams". In: *IEEE Transactions on Dependable and Secure Computing* 15.5 (2018), pp. 840–852. DOI: 10.1109/TDSC.2016.2616861.

[4] Ansgar Belke, Frauke Dobnik, and Christian Dreger. "Energy consumption and economic growth: New insights into the cointegration relationship". In: *Energy Economics* 33.5 (2011), pp. 782–789. ISSN: 0140-9883. DOI: https://doi.org/10.1016/j.eneco.2011.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0140988311000417.

[5] Kishan Bhat et al. "IEEE Cyber Security for the Smart Grid". In: *IEEE Cyber Security for the Smart Grid* (2013), pp. 1–122. DOI: 10.1109/IEEESTD.2013.6613505.

[6] Christian Cachin et al. "Architecture of the hyperledger blockchain fabric". In: *Workshop on distributed cryptocurrencies and consensus ledgers*. Vol. 310. 4. Chicago, IL. 2016, pp. 1–4.

[7] US Congress. "Energy independence and security act of 2007". In: *Public law* 2 (2007), pp. 110–140.

[8] ME Laura Cozzi and Timothy Goodson. "Global Energy Review 2021". In: *International Energy Agency* (2021).

[9] Herman Daly. "Fossil fuels". In: *Applied Energy* 47.2 (1994). Therapy for the Earth, pp. 101–121. ISSN: 0306-2619. DOI: https://doi.org/10.1016/0306-2619(94)90074-4. URL: https://www.sciencedirect.com/science/article/pii/0306261994900744.

[10] Alex de Vries. "Bitcoin's energy consumption is underestimated: A market dynamics approach". In: *Energy Research & Social Science* 70 (2020), p. 101721. ISSN: 2214-6296. DOI: https://doi.org/10.1016/j.erss.2020.101721. URL: https://www.sciencedirect.com/science/article/pii/S2214629620302966.

[11] Michael Dittmar. "Nuclear energy: Status and future limitations". In: *Energy* 37.1 (2012). 7th Biennial International Workshop "Advances in Energy Studies", pp. 35–40. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2011.05.040. URL: https://www.sciencedirect.com/science/article/pii/S0360544211003653.

[12] Endesa. *Self-Consumption Surpluses: sell the solar energy you have left over*. URL: https://www.endesa.com/en/catalog/endesa-self-consumption/compensation-surpluses.

[13] Xi Fang et al. "Smart Grid — The New and Improved Power Grid: A Survey". In: *IEEE Communications Surveys & Tutorials* 14.4 (2012), pp. 944–980. DOI: 10.1109/SURV.2011.101911.00087.

[14] Avi Gopstein et al. *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 4.0*. en. Feb. 2021. DOI: https://doi.org/10.6028/NIST.SP.1108r4. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=931882.

[15] Zhitao Guan et al. "Privacy-Preserving and Efficient Aggregation Based on Blockchain for Power Grid Communications in Smart Communities". In: *IEEE Communications Magazine* 56.7 (2018), pp. 82–88. DOI: 10.1109/MCOM.2018.1700401.

[16] Arifumi Hasegawa et al. "Health effects of radiation and other health problems in the aftermath of nuclear accidents, with an emphasis on Fukushima". In: *The Lancet* 386.9992 (2015), pp. 479–488.

[17] Mike Hearn and Richard Gendal Brown. "Corda: A distributed ledger". In: *Corda Technical White Paper* 2016 (2016).

[18] Madison K. Hoffacker and Rebecca R. Hernandez. "Local Energy: Spatial Proximity of Energy Providers to Their Power Resources". In: *Frontiers in Sustainability* 1 (2020). ISSN: 2673-4524. DOI: 10.3389/frsus.2020.585110. URL: https://www.frontiersin.org/articles/10.3389/frsus.2020.585110.

[19] Jianchao Hou, Haicheng Wang, and Pingkuo Liu. "Applying the blockchain technology to promote the development of distributed photovoltaic in China". In: *International Journal of Energy Research* 42.6 (2018), pp. 2050–2069. DOI: https://doi.org/10.1002/er.3984. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.3984. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.3984.

[20] IEA. *Wholesale energy costs made simple*. URL: https://www.edfenergy.com/large-business/talk-power/newsletter/wholesale-energy-costs-made-simple.

[21] K Habibul Kabir et al. "Energy Generation from Nuclear Fusion in Progressive Developing Countries: A Comprehensive Survey". In: *Energy Generation from Nuclear Fusion in Progressive Developing Countries: A Comprehensive Survey (October 25, 2022)* (2022).

[22] Jiawen Kang et al. "Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains". In: *IEEE Transactions on Industrial Informatics* 13.6 (2017), pp. 3154–3164. DOI: 10.1109/TII.2017.2709784.

[23] Rabiya Khalid et al. "A Blockchain-Based Load Balancing in Decentralized Hybrid P2P Energy Trading Market in Smart Grid". In: *IEEE Access* 8 (2020), pp. 47047–47062. DOI: 10.1109/ACCESS.2020.2979051.

[24] Sunny King and Scott Nadal. "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake". In: *self-published paper, August* 19.1 (2012).

[25] Daniel Larimer. "Delegated proof-of-stake (dpos)". In: *Bitshare whitepaper* 81 (2014), p. 85.

[26] Sarah Manski. "Building the blockchain world: Technological commonwealth or just more of the same?" In: *Strategic Change* 26.5 (2017), pp. 511–522.

[27] Daniel M. Martínez and Ben W. Ebenhack. "Understanding the role of energy consumption in human development through the use of saturation phenomena". In: *Energy Policy* 36 (2008), pp. 1430–1435.

[28] Esther Mengelkamp et al. "A blockchain-based smart grid: towards sustainable local energy markets". In: *Computer Science-Research and Development* 33 (2018), pp. 207–214.

[29] Mihail Mihaylov et al. “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. DOI: 10.1109/EEM.2014.6861213.

[30] Muhammad Baqer Mollah et al. “Blockchain for Future Smart Grid: A Comprehensive Survey”. In: *IEEE Internet of Things Journal* 8.1 (2021), pp. 18–43. DOI: 10.1109/JIOT.2020.2993601.

[31] J.P. Morgan. *What’s Next For Oil And Gas Prices As Sanctions On Russia Intensify*. URL: https://www.jpmorgan.com/insights/research/oil-gas-energy-prices.

[32] Satoshi Nakamoto. “Bitcoin whitepaper”. In: *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

[33] Ioana PETCU. *How are EU electricity prices formed and why have they soared?* URL: https://www.eurelectric.org/in-detail/electricity_prices_explained/.

[34] Serguei Popov. “The tangle”. In: *White paper* 1.3 (2018), p. 30.

[35] Farrokh Rahimi and Ali Ipakchi. “Demand Response as a Market Resource Under the Smart Grid Paradigm”. In: *IEEE Transactions on Smart Grid* 1.1 (2010), pp. 82–88. DOI: 10.1109/TSG.2010.2045906.

[36] K.S. Reddy et al. “A review of Integration, Control, Communication and Metering (ICCM) of renewable energy based smart grid”. In: *Renewable and Sustainable Energy Reviews* 38 (2014), pp. 180–192. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2014.05.049. URL: https://www.sciencedirect.com/science/article/pii/S1364032114003748.

[37] Hannah Ritchie, Max Roser, and Pablo Rosado. “Energy”. In: *Our World in Data* (2022). https://ourworldindata.org/energy.

[38] Omaji Samuel and Nadeem Javaid. “A secure blockchain-based demurrage mechanism for energy trading in smart communities”. In: *International Journal of Energy Research* 45.1 (2021), pp. 297–315. DOI: https://doi.org/10.1002/er.5424. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.5424. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.5424.

[39] Omaji Samuel and Nadeem Javaid. “GarliChain: A privacy preserving system for smart grid consumers using blockchain”. In: *International Journal of Energy Research* 46.15 (2022), pp. 21643–21659.

[40] Bruce N. Stram. “Key challenges to expanding renewable energy”. In: *Energy Policy* 96 (2016), pp. 728–734. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2016.05.034. URL: https://www.sciencedirect.com/science/article/pii/S0301421516302646.

[41] Melanie Swan. *Blockchain: Blueprint for a new economy*. " O’Reilly Media, Inc.", 2015.

[42] Talk Power Team. *Electricity Market Report - January 2022, IEA, Paris*. URL: https://www.iea.org/reports/electricity-market-report-january-2022.

[43] Stephen Thomas. *European Commission response to the energy crisis of 2022*. Discussion Paper. London: Public Services International Research Unit (PSIRU), University of Greenwich, Oct. 2022. URL: http://gala.gre.ac.uk/id/eprint/37893/.

[44] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[45] Peng Zhuang, Talha Zamir, and Hao Liang. “Blockchain for Cybersecurity in Smart Grid: A Comprehensive Survey”. In: *IEEE Transactions on Industrial Informatics* 17.1 (2021), pp. 3–19. DOI: 10.1109/TII.2020.2998479.

[46] Aviv Zohar. “Bitcoin: under the hood”. In: *Communications of the ACM* 58.9 (2015), pp. 104–113.

Appendix

A Code

A.1 Main smart contract

```
contract GroupSmartGrid is AccessControl {
    bytes32 public constant COORDINATOR = keccak256("COORDINATOR");
    bytes32 public constant OPERATOR = keccak256("OPERATOR");
    bytes32 public constant USER = keccak256("USER");
    bytes32 public constant MG_SC = keccak256("MG_SC");

    struct Transaction {
        int256 amount;
        bool verified;
    }

    address public mainGridSC;
    address public coordinator;
    address[] public userIdentities;
    address[] public operatorIdentities;
    int256[24] public prices;
    mapping(address => Transaction[24]) public transactions;
    mapping(address => int256) public debts;
    uint256 public timeSlot;
    bool public tradingFlag;
    string public date;
    uint256 id;

    constructor(address _mainGridSC) {
        mainGridSC = _mainGridSC;
        coordinator = msg.sender;
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        grantRole(MG_SC, _mainGridSC);
        grantRole(COORDINATOR, msg.sender);
        _setRoleAdmin(OPERATOR, USER);
        _setRoleAdmin(COORDINATOR, OPERATOR);

        tradingFlag = false;
        timeSlot = 25;
        id = 0;
    }

    function newOperator(address _operator) public {
        require(hasRole(COORDINATOR, msg.sender));
        grantRole(OPERATOR, _operator);
        operatorIdentities.push(_operator);
    }

    function newUser(address _user) public {
        require(hasRole(OPERATOR, msg.sender));
        grantRole(USER, _user);
        userIdentities.push(_user);
    }

    function beginTrading(int256[24] memory _prices) public {
        require(hasRole(MG_SC, msg.sender));
        for (uint256 i = 0; i < 24; i++) {
            prices[i] = _prices[i];
        }
        tradingFlag = true;
    }
```

```
57
58      function buyElectricity(int256 _amount) public {
59          require(
60              hasRole(USER, msg.sender) && tradingFlag,
61              "Caller doesn't have authorization"
62          );
63          transactions[msg.sender][timeSlot].amount -= _amount;
64          transactions[msg.sender][timeSlot].verified = false;
65      }
66
67      function sellElectricity(int256 _amount) public {
68          require(
69              hasRole(USER, msg.sender) && tradingFlag,
70              "Caller doesn't have authorization"
71          );
72          transactions[msg.sender][timeSlot].amount += _amount;
73          transactions[msg.sender][timeSlot].verified = false;
74      }
75
76      function verifyTransaction(address _user, uint256 _timeSlot) public {
77          require(hasRole(COORDINATOR, msg.sender));
78          transactions[_user][_timeSlot].verified = true;
79      }
80
81      function calculateDebts() public {
82          require(
83              hasRole(COORDINATOR, msg.sender),
84              "Caller doesn't have authorization"
85          );
86          address user;
87          userIdentities.push(mainGridSC);
88
89          for (uint256 i = 0; i < userIdentities.length; i++) {
90              user = userIdentities[i];
91              for (uint256 j = 0; j < 24; j++)
92                  if (transactions[user][j].verified) {
93                      debts[user] += transactions[user][j].amount * prices[j];
94                  }
95          }
96          tradingFlag = false;
97          timeSlot = 24;
98      }
99
100     function payElectricity() external payable {
101         require(
102             hasRole(USER, msg.sender) &&
103                 debts[msg.sender] < 0 &&
104                 int256(msg.value) == -debts[msg.sender]
105         );
106         debts[msg.sender] = 0;
107     }
108
109     function returnDebts() public {
110         require(hasRole(COORDINATOR, msg.sender));
111         for (uint256 i = 0; i < userIdentities.length; i++) {
112             if (debts[userIdentities[i]] > 0) {
113                 payable(userIdentities[i]).transfer(
114                     uint256(debts[userIdentities[i]])
115                 );
116                 debts[userIdentities[i]] = 0;
117             }
118         }
119     }
```

```
120
121     function changeTime() public {
122         require(hasRole(COORDINATOR, msg.sender));
123         int256 totalDiff = 0;
124         for (uint256 i = 0; i < userIdentities.length; i++) {
125             if (transactions[userIdentities[i]][timeSlot].verified) {
126                 totalDiff += transactions[userIdentities[i]][timeSlot].amount
    ;
127             }
128         }
129         transactions[mainGridSC][timeSlot].amount = -totalDiff;
130         transactions[mainGridSC][timeSlot].verified = true;
131     }
132 }
```

A.2 Group smart contract