University of Hradec Králové
Faculty of Informatics and Management

Use of Blockchain for Guarantees of Origin
Master's Thesis

Author: Tirić Amar

Branch of Study: Information Management

Advisor: Ing. Tereza Otčenášková, BA, Ph.D.

Consultants:
Ing. Stanislav Mikulecký, Ph.D.
Production Director at Unicorn Systems
Ing. Petr Svoboda
Senior Consultant at Unicorn Systems

Hradec Králové

May 2021

Declaration:

I declare that this thesis was composed solely by myself, and that the work contained herein is my own except where explicitly stated otherwise.

In Hradec Králové, 14th May, 2021 Amar Tirić

Acknowledgments

Firstly, my deep and sincere appreciation goes to my family for their continuous and unparalleled love, help and support. I owe my deepest gratitude to my supervisor Ing. Tereza Otčenášková for her enthusiasm, encouragement and offering helpful comments for improving my work. I would also like to extend my gratitude to Ing. Petr Svoboda and Ing. Stanislav Mikulecký of Unicorn Systems for their invaluable supervision, continuous assistance and constructive feedback towards the completion of the thesis. Lastly, I am thankful to the University of Hradec Králové and, especially Faculty of Informatics and Management for giving me the opportunity to study there and gain valuable knowledge.

Annotace

Tato práce analyzuje potenciál využití technologie blockchain pro Guarantees of Origin, tedy nástroj používaný pro sledování a prokázání původu obnovitelné energie. V práci je uveden přehled Guarantees of Origin, dále jsou zkoumány stávající implementace jejich registračních systémů a jsou uvedeny hlavní případy použití těchto systémů. Práce představuje komparativní analýzu stávajících blockchainových technologií a hodnotí jejich vhodnost z pohledu dílčích případů užití. Jsou identifikovány klíčové body, ve kterých může technologie blockchain podporovat nástroj Guarantees of Origin, a to zejména z hlediska zvýšení transparentnosti, auditovatelnosti a integrace stávajícího systému certifikátů. Praktická část diplomové práce popisuje dokumentaci procesu vývoje prototypu, který využívá technologii blockchain Ethereum a programovací jazyk Solidity využitelný pro Smart Contracts. Na základě prototypového řešení je vyhodnocena a diskutována vhodnost technologie blockchain z hlediska propustnosti transakcí, soukromí a škálovatelnosti pro její aplikaci v nástroji Guarantees of Origin.

Keywords: Blockchain, Energy Attribute Certificates, Ethereum, Guarantees of Origin, Obnovitelné zdroje, Smart Contracts, Solidity

Abstract

This thesis analyzes the potential of using blockchain technology for Guarantees of Origin, a tracking instrument used to prove the origin of renewable energy. An overview of the Guarantees of Origin tracking instrument is presented, existing registry system implementations are investigated, and main use cases for such a system are determined. The thesis presents a comparative analysis of the existing blockchain technologies and assesses the suitability of each from the perspective of the selected use case. Key points in which blockchain can support Guarantees of Origin tracking instrument are identified as potentially improving transparency, auditability and integration possibilities of the current certificate system. The practical part of the thesis describes documentation of the prototype development process that employs Ethereum blockchain technology and smart contracts programming language called Solidity. Based on the prototype solution, the appropriateness of the blockchain technology in terms of transaction throughput, privacy, and scalability for the application in the Guarantees of Origin tracking instrument is evaluated and discussed.

Keywords: Blockchain, Energy Attribute Certificates, Ethereum, Guarantees of Origin, Renewable Energy, Smart Contracts, Solidity

Content

List of Figures

List of Tables

List of Abbreviations

AIB – Association of Issuing Bodies
AK – Acknowledgment
CLI – Command Line Interface
DPoS – Delegated Proof of Stake
EAC – Energy Attribute Certificate
EECS – European Energy Certificate System
EIP – Ethereum Improvement Proposals
EU – European Union
EVM – Ethereum Virtual Machine
EW – Energy Web
EWASM – Ethereum WebAssembly
EWF – Energy Web Foundation
GO – Guarantee of Origin
GUI – Graphical User Interface
IB – Issuing Body
IoT – Internet of Things
MWh – Megawatt–Hour
PBFT – Practical Byzantine Fault Tolerance
PD – Production Device
PoA – Proof–of–Authority
PoC – Proof of Concept
PoS – Proof–of–Stake
PoW – Proof–of–Work
REC – Renewable Energy Certificate
SDK – Software Development Kit
SWOT – Strengths, Weaknesses, Opportunities and Threats
TPS – Transactions Per Second
TSO – Transmission System Operator
TWh – Terrawatt–Hour
ZSL – Zero–knowledge Security Layer
Zk-SNARK – Zero–Knowledge Succinct Non–Interactive Argument of Knowledge

1 Introduction

The popularity of green electricity has been growing significantly in last few years. Therefore, the demand of companies for this electricity originating from renewable sources is also increasing. This is mainly due to the increasing consumers demand for products that possess a certification that they have been produced using renewable energy. However, once renewable energy enters the grid of the Transmission system operators (TSOs), it can not be distinguished from gray energy originating in non-renewable sources. Since the energy that is provided through the grid is actually a mix of gray and green energy, it is not possible for companies that want to utilize only renewable energy to buy it directly from the grid unless the company did not set up its own facilities for generating renewable energy. Therefore, a tracking instrument called by a generic name Energy Attribute Certificates (EACs) was created. In Europe, a legal manifestation of EACs is known as Guarantees of Origin (GOs). By claiming renewable energy attribute certificates, companies can showcase to consumers and employees, their willingness to utilize clean green energy. The outlook for these certificates is quite optimistic, as the demand for renewable energy documented by Guarantees of Origin has risen by 15% in the first half of 2020, despite the negative effects of a global pandemic [1]. Blockchain appeared more than a decade ago, and ever since its appearance the proponents of the technology have argued that it has the potential to influence and change many aspects of how we conduct our daily lives and businesses. The Guarantees of Origin tracking systems can also be implemented using blockchain technology and some argue that, if implemented in a cooperative and supportive manner to the existing registry systems, may reduce processing costs and regulatory obstacles to attribute monitoring [2].

This thesis aims to analyze the prospects of utilizing blockchain technology for the Guarantees of Origin tracking instrument. It provides an overview of the blockchain technology itself, analyzes the advantages and disadvantages of blockchain technology in the context of this use case, and evaluates practical insights gained from developing a prototype for a Guarantees of Origin system based on Ethereum blockchain technology and Solidity programming language. The thesis content is organized as follows. Firstly, the Guarantees of Origin tracking instrument and the architecture of the existing registry systems is presented. Secondly, blockchain technology is introduced, its main concepts and features are elaborated, and potential use cases for the Guarantees of Origin system are described. In the Results and Discussion chapter, the implementation of the prototype is documented, and implications are evaluated in terms of the advantages and disadvantages it provides to the existing solution.

2 Objectives and Methodology

This chapter presents the objectives and the methodology of the thesis. Research Objectives section identifies the main objectives of the thesis. The Research Methodology section defines a methodological approach.

2.1 Research Objectives

This thesis systematically and comprehensively examines the potential for application of blockchain technology for the Guarantees of Origin tracking instrument. Based on the investigation of existing research literature related to utilizing blockchain technology for Guarantees of Origin or similar Energy Attribute Certificates systems, is examined. The thesis highlights the potential benefits that blockchain can bring to the Guarantees of Origin tracking instrument. Furthermore, this thesis documents the development of a prototype for a Guarantees of Origin registry system based on blockchain technology. In order to weigh in the trade-offs brought by the blockchain in terms of scalability, transaction throughput and privacy, the comparison of the possible blockchain implementations for this application is performed. The appropriateness of the blockchain technology for this specific use case is elaborated and its positive and negative aspects are investigated. The objectives of this thesis are specified as follows:

- Identify the main use cases that a Guarantees of Origin registry system implements.
- Understand the added value of utilizing blockchain technology as well as present and analyze various blockchain technology alternatives and the implications of their utilization for the Guarantees of Origin system.
- Examine the available academic literature and provide a comprehensive overview of the existing implementations and their advantages and disadvantages.
- Propose and evaluate a prototype for a traceability system for Guarantees of Origin tracking instrument using blockchain technology.

2.2 Research Methodology

The methodological approach based on the earlier noted objectives can be divided into the following steps:

- Step 1 – Briefly introduce and analyze the Guarantees of Origin tracking instrument and the existing registry systems. The overview of the information is gathered from the exploration of EU legislation and regulations, as well as various protocols and procedures specified by important organizations in the area such as the Association of Issuing Bodies (AIB).

- Step 2 – Introduce the blockchain technology and explain the main concepts necessary to understand the potential of its utilization. The information was gathered from the academic resources obtained through publication databases such as Web of Science, Scopus, JSTOR, etc. Scientific journals and articles were investigated in order to collect and compare the existing implementations of the blockchain for the specified purpose. Additionally, the official documentation websites and technology whitepapers were used as they contained extensive and detailed information on the subject. Based on the data collected, potential advantages of using blockchain technology were identified and explored. Furthermore, a technology selection process was documented and discussed.
- Step 3 – The technical solutions utilized for the development of the prototype are presented. A sequential approach in documenting and visually explaining the prototype features, through the perspective of each of the earlier specified Guarantees of Origin system use cases was employed. The relevant technical information is conveyed using diagrams and pseudo-code snippets. The original prototype code is provided in the Appendices.
- Step 4 – The usability of the prototype is evaluated, accompanied by a discussion about the benefits and drawbacks of utilizing blockchain technology for the Guarantees of Origin tracking instrument.

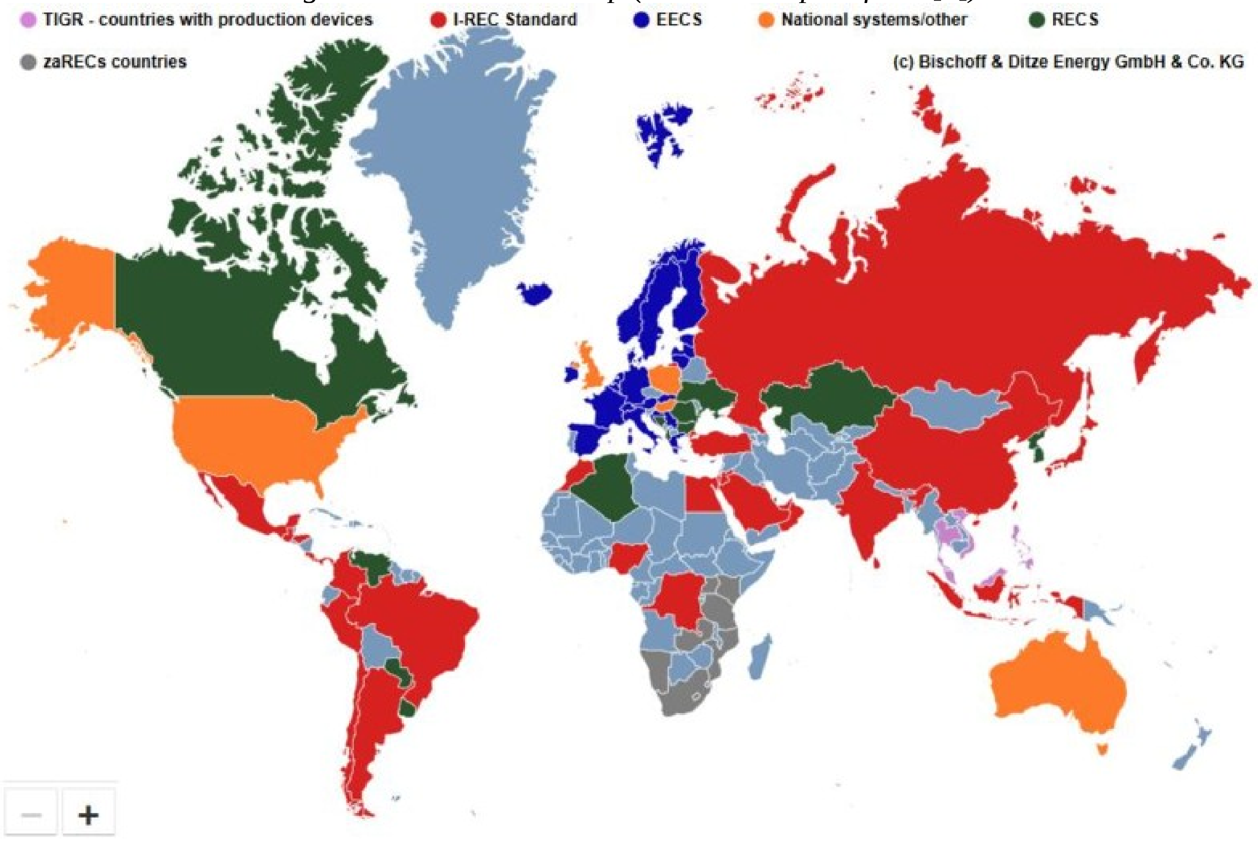
3 Overview of Guarantees of Origin Scheme

Energy Attribute Certificates (EACs) are electronic certificates by which the production, trade, and consumption of renewable energy can be documented and tracked. EACs do not represent the energy directly, rather they are contractual instruments that convey information about the electricity that has been produced, such as the type of power plant, the location where it was produced, the production period and amount.

The EAC usually represents one megawatt hour of electricity (1 MWh) and is the most trustworthy method to prove that a business is purchasing energy from legitimate renewable sources [3].

Based on the region in which a certain EAC is defined by the law, the following certificates are distinguishable: European Energy Certificate System Guarantee of Origin (EECS-GoO) prevalent in the European Union, Renewable Energy Certificates (RECs) in North America, Renewable Energy Guarantees of Origin (REGoOs) in the UK, and Tradeable Instruments for Global Renewables (TIGRs) as well as The International REC Standard (I-REC Standard) in the rest of the world [4]. This thesis shall focus on EECS-GO, a tracking system for EACs used in European Union and a few other countries in Europe, and will be referred to as the Guarantees of Origin for Renewable Energy. Figure 1 presents an overview of different EAC systems

Figure 1: EACs World map (Source: Adopted from [5]*)*


● TIGR - countries with production devices
● I-REC Standard
● EECS
● National systems/other
● RECS
● zaRECs countries
(c) Bischoff & Ditze Energy GmbH & Co. KG
−
+

Guarantee of Origin is defined in the Directive (EU) 2018/2001 [6] of the European Parliament and of the Council of 11 December 2018 on the promotion of the use of energy from renewable sources, and its predecessor Directive 2009/28/EC [7, p. 27], as “*an electronic document which has the sole function of providing evidence to a final customer that a given share or quantity of energy was produced from renewable sources*”. A Guarantee of Origin may be passed from one holder to another, regardless of the energy to which it relates. Double counting and double disclosure of Guarantees of Origin must be avoided in order to ensure that a unit of renewable energy is released to a consumer only once. Energy from renewable resources for which the supplier has sold the corresponding Guarantee of Origin certificate separately should not be disclosed or offered to the final consumer as renewable energy. The directive also states that Guarantees of Origin should be issued for all the units of renewable energy that have been produced, with certain exceptions in which it is decided that the producer has already received financial support for the production of the renewable energy. The directive further defines a standard size of a single Guarantee of Origin to 1 MWh. Guarantee of Origin (GO), does not have a specified fixed price for which it is sold, rather its price is determined by the market demand [8]. Each Guarantee of Origin certificate has a validity of 12 months after the energy unit has been produced. The directive (EU) 2018/2001 further specifies the information that each Guarantee of Origin is supposed to contain [6]:

- the energy source as well as the start and end dates of the production
- whether the GO is related to electricity, gas or heating/cooling
- the identity, location and type of the installation from which the energy originates
- whether the installation has received obtained benefits from investment support, in a manner different from the national support, and the specific type of that support scheme
- the date when the installation has become operational
- the country from which the GO was issued, the date and the unique identification number

In order for a company to use the GO certificates for energy disclosure, they must buy GO certificates in the amount of MWh that they want to use them for. In order for the GO to be used for proving the origin of renewable energy, they must be cancelled in the relevant certificate registry [9]. Upon cancellation, the GO can not be transferred anymore and the energy it relates to is considered as used. In order to develop the use of a standardized system, national GO tracking systems have established the Association of Issuing Bodies (AIB). AIB is an organization whose

goal is to enable the harmonized environment as well as structures and procedures that ensure the smooth operation of an international system that supports the electronic production, transfer, and cancellation of energy certificates [8]. For that purpose, they developed a standardized system called the European Energy Certificate System (EECS), a system for EECS certificates that may be based on Guarantees of Origin, or issued in relation to some other legislative schemes, or other voluntary arrangements [10]. In 2016, it was estimated that around 2/3 of GO issuing and transferring is done in compliance with EECS standard [11], [12]. Therefore, the Association of Issuing Bodies enables international transfers of the Guarantees of Origin, through the EECS, making it possible to import GO from other domains or export GO to other domains. A domain is a term used to refer to an area under which exists an authorized Issuing Body that issues an EECS product (GO) and usually refers to Member states of the EU [10].
Based on the relevant Guarantee of Origin directives mentioned earlier, the main functions that can be performed on a Guarantee of Origin certificate can be derived as follows:

- energy production device registration and auditing
- periodical production auditing
- registration of account holders
- issuing of certificates – provide the requested number of GO certificates to the energy producer corresponding to the amount of MWh that the requesting party installation has produced
- transferring of certificates – transfer GO certificates to another party within the same domain
- cancelling of certificates – the GO certificates that have been redeemed for their purpose, and therefore made non-transferable
- invalidating the expired certificates – the GO certificate that has been produced more than 12 months ago, is considered as expired, each Issuing Body should invalidate it within 6 months before the expiration date

These use cases should be supported by information systems and certificate databases of each issuing body. Furthermore, if the issuing body within a certain country needs to enable trading with other countries, it needs to be part of the AIB, and conform to the EECS standards. The system through which the transfer among AIB member registries is provided is called AIB Hub [13]. Therefore, two more use cases for an information system supporting the Guarantees of Origin tracking scheme can be defined:

- exporting certificates – GO certificates are transferred to a member registry of Issuing Body of another country

- importing certificates - GO certificates are transferred from a member registry of Issuing Body of another country

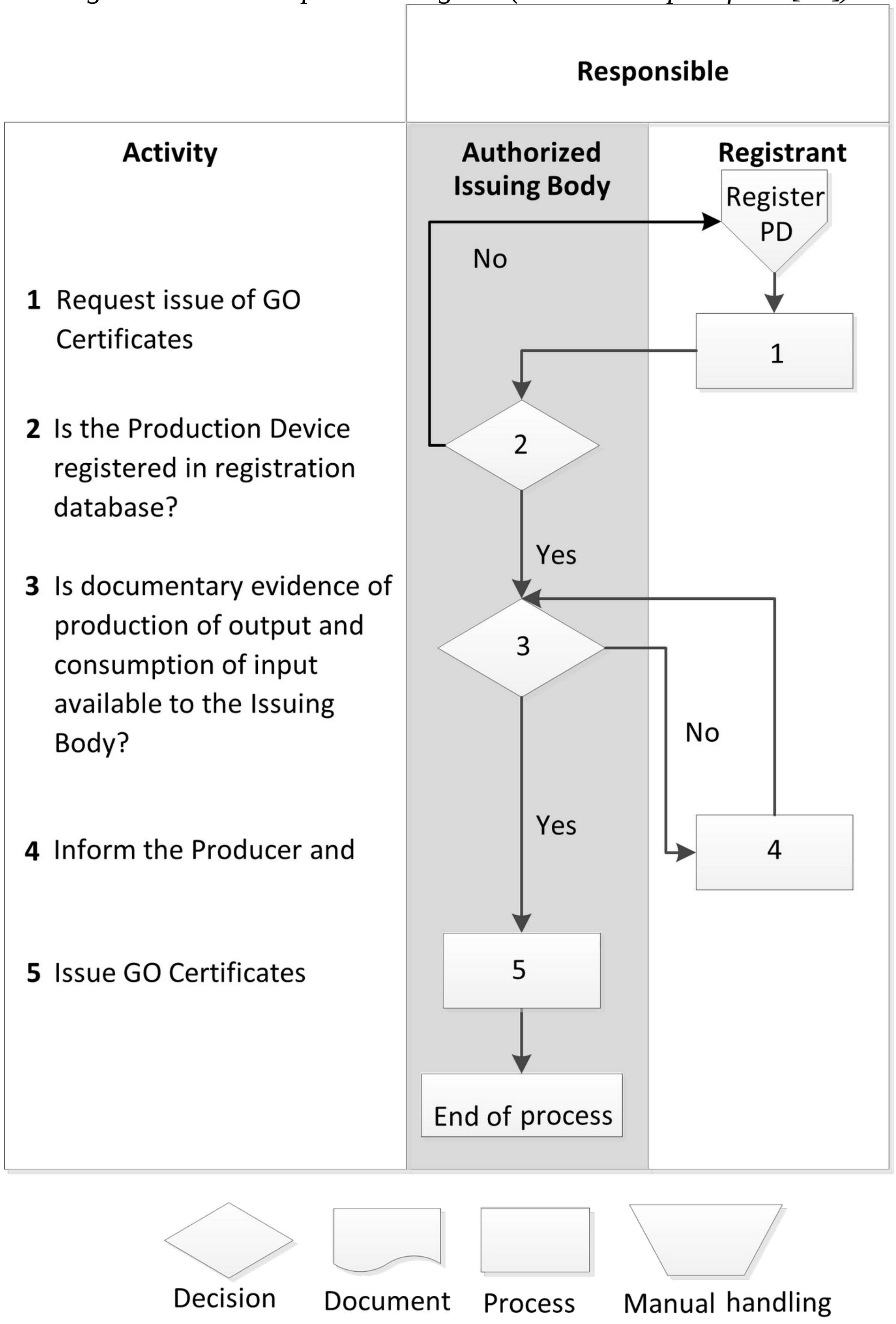
Since there exist different accounts in various registries, it is also necessary to export and import the accounts between the independent registry systems. The export and import use cases are generally governed by the AIB Hub, even though it may be possible for two Issuing Bodies to arrange transfer between themselves, without a middleman. However, this is not allowed for the members of AIB.

The first group of use cases should be regulated by the Guarantees of Origin Registry established by an Issuing Body within a certain domain. GO Registry refers to an information system with a supporting database, usually established on a national level, to support all the relevant GO use cases, within a specific country/domain. The Issuing Body will provide an information system to which the registered electricity provider should gain an access to, in order to request the GO certificates. For example, Czech Issuing Body OTE has a GO Registry information system called EZP and specifies the procedure necessary for an electricity producer or trader to gain access to the GO registry. Upon access approval renewable electricity producer is able to request GO certificates that correspond to the amount produced by an applicants production device. The same registry is used by the organizations that need to use GO for disclosure purposes. OTE requires that access to the EZP system is requested only by electricity producers or electricity traders that are licensed by Czech Energy Regulatory Office [14]. The EECS Electricity Domain Protocol for Czech Republic prepared by Issuing Body OTE states that upon the account activation on GO Registry for the producer, the account holder can perform the following operations [15]:

- apply for EECS-GO certificates,
- provide instructions for transferring EECS-GO certificates,
- provide instructions for cancelling EECS-GO certificates,
- provide suggestions for withdrawing EECS-GO certificates, and provide suggestions for data updates relevant to its registration in the EECS-GO Registration Database
- receive information and data about the account, as well as the EECS-GO certificates that have been registered.

Since the Guarantees of Origin issued by OTE are issued according to the EECS standard, they are referred to as EECS-GO certificates. The domain protocol document also specified procedures for maintenance of production device data, registration of a production device, de-registration of the device, as well as detailed description of certificate systems administration, and other relevant procedures within a domain, in this specific case, the domain of Czech Republic. Figure 2 displays the process flow diagram of issuing GO to a registered electricity producer.

Figure 2: Issue GO process diagram (Source: Adapted from [15]*)*



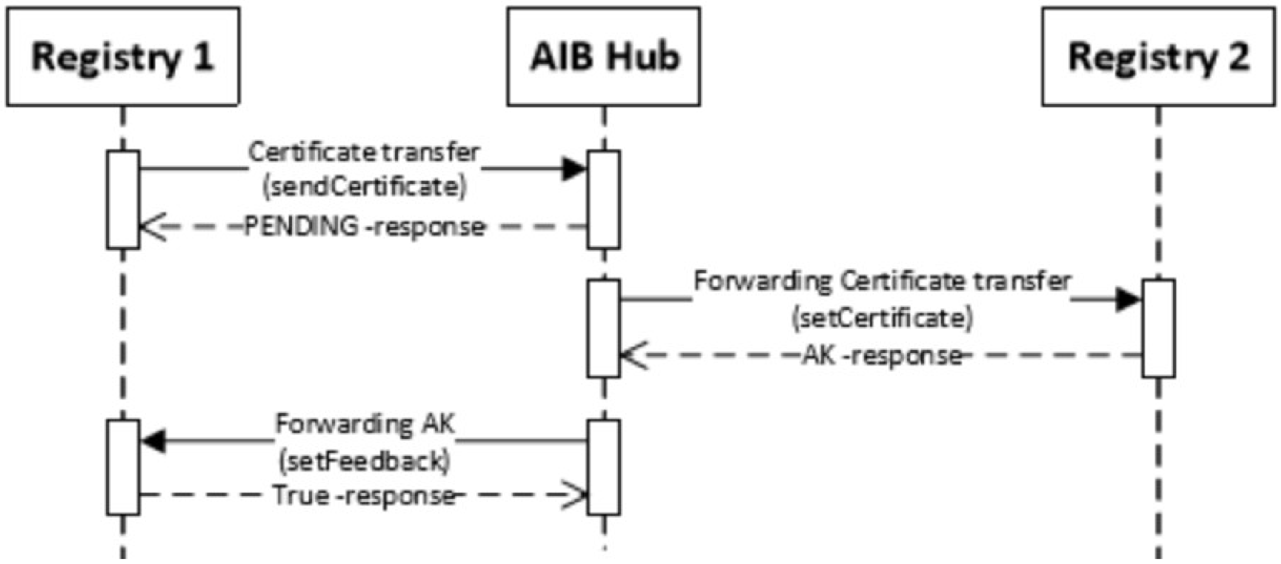
Responsible
Activity
Authorized Issuing Body
Registrant
Register PD
No
1 Request issue of GO Certificates
1
2 Is the Production Device registered in registration database?
2
Yes
3 Is documentary evidence of production of output and consumption of input available to the Issuing Body?
3
No
Yes
4 Inform the Producer and
4
5 Issue GO Certificates
5
End of process
Decision
Document
Process
Manual handling

The information system that supports issuing of GOs in the current centralized architecture will have the registered production device owner primarily requesting the issuance of the GO certificates, upon which the system checks the presence of the production device in the system. If the validation check is successful, the system will check whether the documentary evidence of the production of output and consumption of the input is available to the relevant Issuing Body. If available, the system issues the requested GO certificates in the value of the produced amount expressed in MWh.

Having multiple separate Issuing Bodies means that each Issuing Body has its own information system for the GO Registry, that supports some main use cases: issue, transfer, cancel, withdraw, expire. However, what if a certain participant of a GO certificate scheme wants to obtain GO certificates that are issued by an Issuing Body from another country/domain? The GO certificate would then have to be transferred between two Issuing Bodies before it is used for use cases that occur within a single Issuing Body. Since each Issuing Body has its own informational system representing the GO Registry, transferring certificates between them necessitates a single point of contact. This is necessary to enable inter-communication of the registries, and enable synchronization and coordination between them.

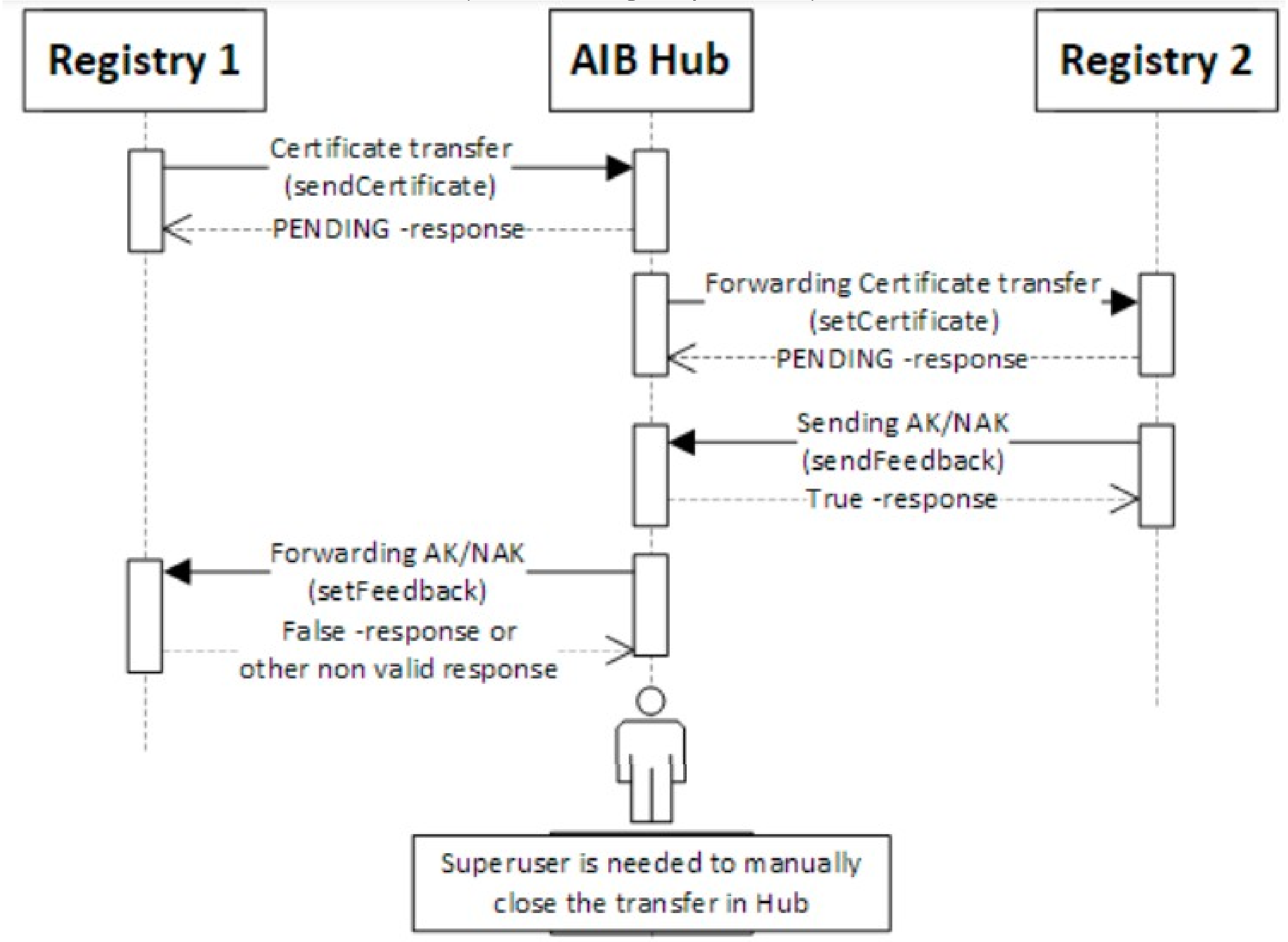
As mentioned earlier, this interface between Issuing Bodies and their registry systems is provided by the Association of Issuing Bodies, via the AIB Hub. The user of the Hub may or may not be a member of the AIB, who by signing Hub Participant Agreement, becomes a Hub Participant [10]. For an applicant Issuing Body and its registry database to be accepted into the AIB Hub, their information system needs to pass a number of tests performed by the AIB Hub, to make sure that it conforms to all the interfaces specified in the EECS Registration Databases document [16]. When a Registry of an Issuing Body has been registered within the AIB Hub system, it is able to perform transfer with other Registries owned by other Issuing Bodies. Figure 3 presents the basic data transmission protocol between two registries and AIB Hub.

Figure 3: Basic Data Transmission Protocol – asynchronous AK (Acknowledgment) (Source: Adopted from [16]*)*


Registry 1
AIB Hub
Registry 2
Certificate transfer (sendCertificate)
PENDING -response
Forwarding Certificate transfer (setCertificate)
AK -response
Forwarding AK (setFeedback)
True -response

The certificate transfer is initiated by the sending registry, which transmits a transfer request to the AIB Hub. Upon receiving the request, the AIB Hub forwards it to the receiving registry. When the receiving registry acknowledges the reception of the certificates, the feedback is sent to the AIB Hub, which ultimately returns the feedback to the sending registry. The AIB Hub acts as a centralized actor that takes care of coordination and synchronization of the transfers, as well as handling disputes related to the certificate transfers. In special cases, a manual intervention is necessary by a superuser of the AIB Hub to close the transfer, as can be seen in Figure 4.

Figure 4: Data transmission protocol between registries, with manual action from superuser (Source: Adopted from [16]*)*


Registry 1
AIB Hub
Registry 2
Certificate transfer (sendCertificate)
PENDING -response
Forwarding Certificate transfer (setCertificate)
PENDING -response
Sending AK/NAK (sendFeedback)
True -response
Forwarding AK/NAK (setFeedback)
False -response or other non valid response
Superuser is needed to manually close the transfer in Hub

The EECS Registration Databases document further specifies the following functional requirements that this transfer mechanism must support [16]:

- Transferable: Each message is assumed to be delivered from sender to recipient via the AIB Hub
- Transparency: Any transmission failure must be discovered;
- Attributable: message that came out of the intended sender should clearly be recognizable.
- Accurate: The message must come with great confidence that the transit was not altered;
- Private: The message must arrive with the assurance that no adequately trained third party can understand it.

The previously mentioned requirements specify essential aspects the AIB Hub and the GO Registries informational system must exhibit. Some of these will be quite relevant for further discussion, especially the aspect of the message privacy.

This section has presented and explained the GO tracking scheme and the current system design by which the main use cases related to the GO scheme are supported. The system is rather robust, well documented, and quite well integrated with the current European legal instrument. The GO registries of Issuing Bodies are quite resilient and reliable, and all the members of the AIB that participate in the EECS GO scheme, and that transfer certificates via the AIB Hub are frequently audited by the AIB itself. This ensures that GO registry systems that communicate with the AIB Hub fulfill all the necessary functional requirements. As the current system has been functioning successfully for quite a period of time, the question to ask is why to propose any change to it. Some claim that the Guarantees of Origin system is not dependent on technology, rather it is technology agnostic, so the "*industry should seek to utilize the best underlying technology available at any time to provide the best possible service/product*" [17]. In relation to that, it is then necessary to define the aspects of the mechanism that can be improved by using different technology, than the one currently being utilized.

Even though the robustness, security, resilience of the current system can not be questioned, utilizing blockchain technology could potentially lower the operational costs of the system as well as being more inclusive toward smaller producers [18]. Using blockchain technology might potentially lower the administrative burden, which would enable smaller production units to engage in the system, due to lower functional expenses. The inclusion of more small producers would result in an effective increase of the GO certificates in circulation. The following section introduces the concept of the blockchain, as well as providing an analysis of how a GO tracking system may be implemented within the frames of this technology.

4 Blockchain Technology and Potential Use for Guarantees of Origin

This chapter introduces the main features and characteristics of the blockchain technology. Furthermore, it identifies the key aspects in which blockchain technology can be utilized for Guarantees of Origin tracking instrument. The related work and existing implementations are presented and analyzed. Lastly, different blockchain platforms are evaluated and technology selection process is documented.

4.1 Blockchain Technology

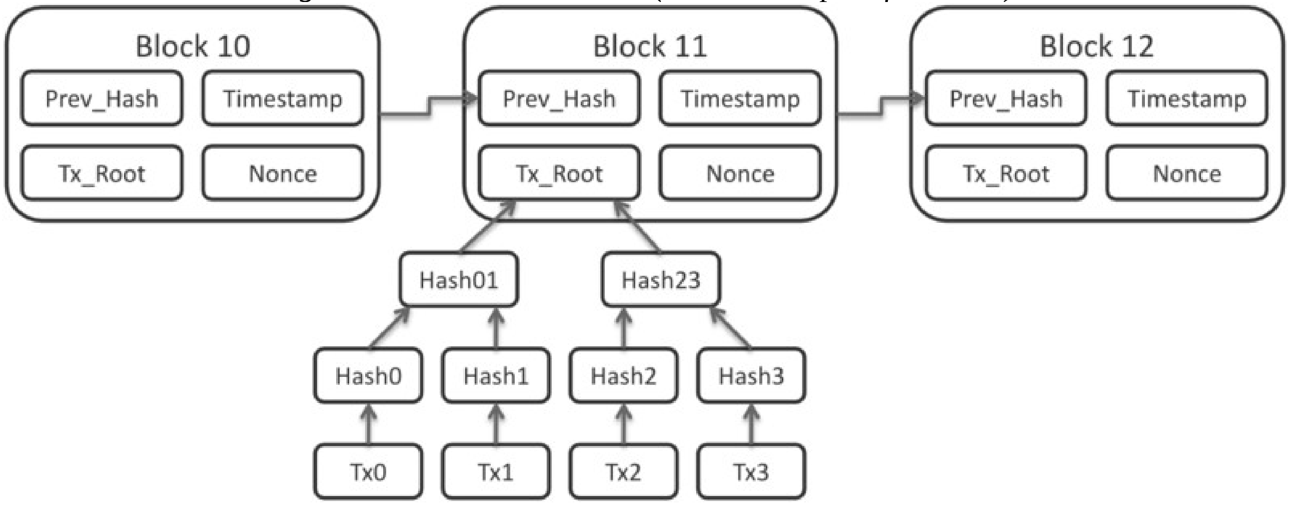
Blockchain is a technology that has seen a significant rise in popularity recently. The formal origin of the blockhain technology can be traced back to 2008 when the idea of Bitcoin cryptocurrency was introduced in the paper by Satoshi Nakamoto [19]. Blockchain technology can be defined as a public distributed ledger system that maintains the confidentiality of transaction data [20]. The public ledger is a record-keeping system that maintains participants' identities in a secure and anonymous manner, their respective cryptocurrency balances, and a record book of all the genuine transactions executed between network participants. A blockchain does not have to be public, as there exist variations that are not open for everyone and have an access control layer. These are called permissioned blockchains. All of the transactions that occur on the blockchain are stored in the list of blocks, which grows as more and more blocks are added. In order to make the ledger data consistent and enhance security, blockchain uses asymmetric cryptography and distributed consensus algorithms [21]. The transactions that occur on the blockchain are transparent and visible to everyone, they are timestamped and irreversibly stored, creating a fixed timeline of data that is not controlled by any central party. Blockchain is essentially a database that contains the whole history of all the transactions ever processed and whose integrity all users can trust [22].

Distributed peer-to-peer technology such as blockchain, suggests that there is no central party owning the infrastructure. No one can alter the data on the blockchain, thus it is characterized as being immutable. The concept of blockhain immutability was introduced with the cryptocurrency Bitcoin so that the problem of double spending is solved. Double-spending is a potential flow of the digital currency, and is defined as the risk that a digital currency can be spent twice [23]. Before each transaction is added to the blockchain, its authenticity is checked, and then it is stored on the blockchain as an immutable record. This is to prevent a malicious actor from spending the same cryptocurrency in two different transactions. The authenticity of transactions is established with the consensus protocol explained in the next chapters.

4.1.1 Technical Description of the Blockchain

Three important terms define a blockchain: blocks, nodes and miners [24]. As previously mentioned, blockchain is the distributed log of the transactions that are grouped in blocks. Blocks are timestamped and are identified by cryptographic hashes. Every block has a reference to the hash of the previous block, creating a chain of blocks, therefore called the blockchain. Each block contains at least two elements: a cryptographic hash and the block data [25]. Furthermore, each block contains a nonce which is a randomly generated number, used to generate a block header hash, making data in the block forever linked to the nonce and hash. Figure 5 shows the structure of the blocks in the Bitcoin blockchain.

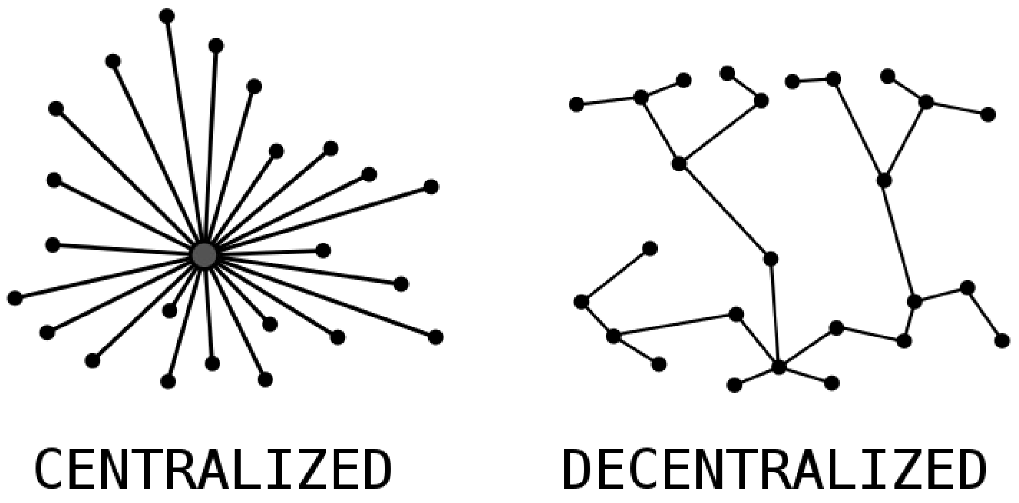
Figure 5: Bitcoin Block Data (Source: Adopted from [26]*)*



Nodes are one of the most important concepts in blockchain since they are what makes blockchain distributed and decentralized. Nodes are essentially a set of clients, and each store a copy of the entire blockchain [25]. However, one node can serve as a point of access for multiple clients. Blockchain nodes are the building blocks of the peer-to-peer network, they verify that the incoming transactions are valid, discarding invalid transactions. When a set of transactions is validated, they are ordered and bundled with a timestamp into a candidate block. Candidate block is the block that should be added to the blockchain in the process called mining [25].

The node that adds a new block to the blockchain is called the miner node. For the block to be added to the blockchain, nodes have to agree that the transactions contained in the block are valid and that the new block is properly referencing other blocks in the blockchain. This ensures that the integrity of the blockchain is maintained. Figure 6 shows the difference between a centralized system and a decentralized system composed of numerous interacting nodes.

Figure 6: Centralized and Decentralized system. (Source: Adopted from [27]*)*



The process by which nodes agree on the validity of the present state of blockchain is by various algorithms known as consensus protocols.

4.1.2 Consensus Protocols

Consensus is the concept in distributed computing *"wherein nodes within the system must reach an agreement given the presence of faulty processes or deceptive nodes" [28, p. 1545]*. Consensus protocols are necessary to exist in order to reach an agreement in the decentralized environment, as there is no centralized actor that may solve disputes and enforce consistency. Some of the most popular consensus protocols for establishing consensus on the blockchain are the following [21]:

a) Proof-of-Work (PoW) is the consensus protocol used in the Bitcoin blockchain [19]. For the blockchain participants to be able to add a new block, they have to solve computationally complex "cryptographic puzzles", in the process commonly known as mining [29]. As the number of participants increases over time, the computation becomes very complex, and requires stronger hardware resources. This results in high electricity consumption. The node that successfully solves the cryptographic problem appends the new block to the blockchain, and receives the reward for the work. In Bitcoin's case, the miner node is rewarded with a certain amount of Bitcoin. In terms of scalability, blockchains based on the Proof-of-Work consensus protocol offer good node scalability with poor performance [30].

b) Proof-of-Stake (PoS) is a consensus protocol that is considered as a more energy-efficient substitute to Proof-of-Work. In the Proof-of-Stake algorithm, nodes must prove that they possess a certain amount of cryptocurrency to participate in the transaction validation process. It is assumed that nodes that have a higher stake in the validation process are more trustworthy, and therefore would not profit from attacking the distributed network [21]. To validate a transaction, the validator must prove the ownership of his stake of cryptocurrency. The more cryptocurrency the validator holds, the bigger the chance to validate the next block [29]. The Proof-of-Stake algorithm has both advantages and disadvantages [31], however, compared to PoW it may provide higher scalability and better energy efficiency. The Proof-of-Stake algorithm has gained an increase in its popularity recently. This is enforced by the fact that one of the most popular blockchain technologies Ethereum is transitioning from Proof-of-Work to Proof-of-Stake algorithm [32].

c) Practical byzantine fault tolerance (pBFT) is a replication algorithm made for tolerating byzantine faults [33]. Byzantine fault is a problem in distributed computing, where components of the system may fail and there is imperfect information as to which component has failed. This algorithm requires that all nodes are known to the network [21]. The pBFT is an asynchronous model, in which all the nodes communicate with each other, and by the voting process, it is determined which node proceeds to the next voting round. The goal is to determine which nodes are acting as honest nodes, and which are faulty nodes either due to a node failure or malicious intent.

d) Proof-of-Authority (PoA) represents a new family of algorithms belonging to the Byzantine fault-tolerance consensus algorithms. The two most popular algorithms that belong to this group are Aura and Clique [34]. In the PoA algorithms the consensus process is performed in rounds. Each round has an elected party called mining leader, that has the obligation of proposing new blocks. PoA requires a smaller amount of exchanged messages between nodes, providing better performance than pBFT consensus algorithm [35].

Since pBFT requires that all nodes in the network are known to each other, it is better suited for permissioned and private blockchains, whereas PoW and PoS are more suitable for public blockchains. In addition to these, other known blockchain protocols are Delegated Proof-of-Stake (DPoS), Ripple, etc. [36]. PoA, Ripple and DPoS are more suitable for private and consortium blockchains [21].

4.1.3 Main Blockchain Characteristics

The main characteristics of blockchain and its categorization can be described as follows [37], [21]:

Public and permissioned blockchains describe the different levels of visibility of the ledger and openness of the network [38]. Public blockchains, such as Ethereum and Bitcoin, have a distributed ledger that is visible to everyone, meaning that anyone is allowed to participate in the network and observe its contents. However, this leaves access open to potentially malicious actors. For permissioned blockchains, only authorized parties can preview the ledger of all the transactions and can participate in the transaction validation and submission process. These are more common in business environments where a certain level of privacy is desired. Permissioned blockchain is a blockchain with access control procedures, in which the participants' roles and actions are preconfigured [39]. There is no general rule for the classification of the blockchains, but typically a private blockchain is classified as a type of permissioned blockchain. A private blockchain is generally used to refer to a blockchain with a closed access, validated and shared by a predefined group of nodes. On the other hand, consortium blockchain is seen as a hybrid solution between private and public blockchain, most suitable for semi-closed systems [37].

Decentralization is an another important characteristic of the blockchain. In a centralized system there exists a central party that is generally trusted, and that has control over the whole system. On the other hand, a decentralized system is not governed by any central entity. However, this is not to be confused with the word distributed, as blockchain is inherently distributed but it can be centralized [40]. Public blockchains operate in a fully decentralized environment, meaning that trust is established between unknown parties without a central authority. On the other hand, permissioned and private blockchains enjoy a lower degree of decentralization with the private blockchain being the most centralized.

Persistency is a characteristic signifying that all the transactions that occur on the blockchain are permanently stored in the ledger. This trait of the blockchain can be decomposed into two traits: transparency and immutability [37]. Immutability means that the data that has been accepted by the nodes of the network into the blockchain can not be modified, it is tamper-proof [41]. This is especially the case for public blockchain, since no node has control over the consensus process [21]. Transparency in the blockchain network means that the participants in the network can access the transactions, search through the blocks of blockchain and investigate its contents [42].

Anonymity is a feature enabled by the fact that users interact with the blockchain through a generated address. Therefore, usually it is not possible to identify the true identity of the blockchain user. It is also possible to use multiple different addresses in an effort to maintain anonymous identity [21]. On the other hand, for the blockchain systems such as private and permissioned, the knowledge of identity is usually required [37].

Auditability is a feature that is based on the blockchain transactions being persisted and easily previewable in the blockchain explorer systems. Also every transaction is timestamped, so the blockchain network is considered as a highly auditable system. The implementation of automatized verification processes in conjunction with blockchain could greatly increase the cost efficiencies in the audit environment [43].

4.1.4 Smart Contracts

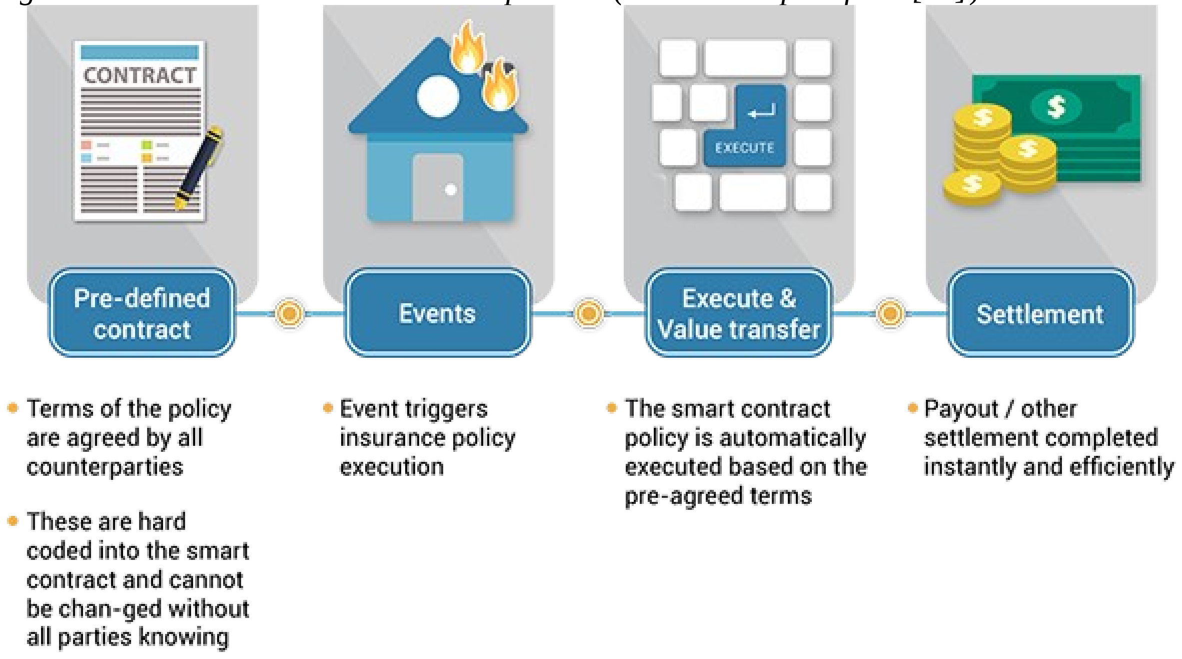
The idea of smart contracts has been conceptualized even before the first blockchain was developed. It was initially proposed in a paper by Nick Szabo in 1997, as a computerized transaction protocol that executes the contractual terms of agreement [44]. Namely, the parties engaged in a contractual relationship specify the necessary clauses which are executed automatically on the decentralized system. The idea of smart contracts has been revisited with the advancement of blockchain technology. Smart contracts can be executed on the blockchain and each contract statement is saved as an immutable transaction on a distributed ledger. Therefore, smart contracts do not need a trusted authority, since the correct execution is embedded into the coded executable statements smart contract is comprised of. These statements are defined and known by all the parties involved in a process. The pioneer blockchain platform to enable the execution of the smart contracts on the blockchain was Ethereum. This is enforced by a built-in Turing complete programming language, that enables anyone to create rules for ownership, transactions, and state transitions of a smart contract [45]. Zheng et al. specify some of the life-cycle phases of a smart contract [46]:

- Creation – in this phase, the involved parties negotiate the obligations and rules of the smart contract until an agreement is reached. After that, similar to the development of any other computer software, the business rules are translated to computer language.
- Deployment – after being deployed on a blockchain platform, the smart contract becomes immutable which means that usually it can not be modified, and even when modified all the involved parties are aware of the changes. Any alterations to the smart contract usually require a step back and deployment of a new smart contract.
- Execution – when the conditions and rules of the contract are satisfied, the contractual functions are called and validated by miners on the blockchain. All the transaction steps and changes of the smart contract state are recorded on the blockchain.

- Completion – after the execution of the transaction, all parties are notified of the state changes. At this stage, any asset transfers that were needed to be performed as a part of the smart contract logic should have been completed. The execution and completion phase can be repeated on the same smart contract.

Figure 7 specifies the phases of a smart contract that handles the business logic of insurance policies. As can be seen, a smart contract contains predefined rules and terms agreed upon by all the parties, that can not be changed without all the parties being aware of the changes. After a certain business event occurs the functions of smart contract are initiated. That can be, for example, an invocation of the smart contract function to make a claim request for an insurance policy by a customer involved in a car accident. The function is immediately executed on the blockchain, based on the terms defined in a smart contract. In the settlement phase, the payout or other settlements can be immediately executed on the blockchain and assets can be transferred to a requesting party. Some of the blockchain platforms that support deployment and development of smart contracts besides Ethereum are Hyperledger Fabric, Corda, Stellar, Rootstock, Lisk, and EOS [46], [47].

Figure 7: Smart Contracts in insurance policies (Source: Adopted from [48])


CONTRACT
EXECUTE
Pre-defined contract
Events
Execute & Value transfer
Settlement
Terms of the policy are agreed by all counterparties
These are hard coded into the smart contract and cannot be chan-ged without all parties knowing
Event triggers insurance policy execution
The smart contract policy is automatically executed based on the pre-agreed terms
Payout / other settlement completed instantly and efficiently

4.2 Blockchain and Guarantees of Origin

This section answers the question of why blockchain technology should be utilized for implementing the information system for tracking Guarantees of Origin certificates. It is important to note though, that the potential benefits brought to this business use case by use of blockchain technology should also apply to other types of Energy Attribute Certificates, no matter their legal environment.

Many doubt the idea of replacing the current system for Guarantees of Origin, posing the question of why the system that has been working for more than 15 years should be changed. In 2020, all-time-high record in terms of Guarantees of Origin cancelled via the AIB Hub has been reached, with the value of 735.1 TWh [49]. Some argue that the current system may lack simplicity and transparency, stating that the issue is not based on technology, but rather on some regulatory and policy constraints, and that blockchain should not be focusing on the technology aspect of proposed modifications [50]. They state that the blockchain should not aim at replacing the current infrastructure, rather provide value in conveying how the blockchain technology may create a more transparent, secure, efficient and cheaper infrastructure. If blockchain enthusiasts would propose an alternative solution to the existing system, there is a risk that a lot of hard work invested in the current system would be wasted, the replacement of which will create consumer uncertainty, undermine stakeholder integrity, and increase the possibility of double counting [17].

One of the arguments for using blockchain is that the current system in use excludes small energy producers, as the procedural costs are too high in the case of small energy amounts. Same argue that the current system lacks auditability, as it is managed by a central party and may be susceptible to errors as well as intentional malicious activity [51]. On the other hand, as mentioned in the previous section, blockchain technology provides a high level of auditability. The main areas in which blockchain technology could improve the systems for tracking Energy Attributes Certificates is to automate the issuing of the certificates, reduce transaction costs, create a global market and increase the transparency of the system [51]. On the downside, the total decentralization of the system would mean that it would be hard to validate certain processes. For example, if Energy Attribute Certificates are issued automatically by smart meters after sending production amount data to the blockchain, who is to prevent the act of smart meters being directly tampered with [52]. Furthermore, not all processes can be automated, such as production device registration and preregistration audit.

Spinnell and Zimberg analyze the benefits and drawbacks of utilizing blockchain for the Regional Transmission organization PJM EIS which operates a platform for trading RECs (Energy Attribute Certificates as defined in the United States) in the Northeastern United States [53]. They list the benefits of employing blockchain technology as decreasing operational costs, improving transparency among the market participants, easier auditing and guaranteeing authenticity by the immutability of data. They further perform a SWOT analysis on using blockchain for Renewable Energy Certificate trading, obtaining the following key points:

- As for strengths, they deem blockchain technology as being able to automate the transaction processes, as well as the system being more secure.
- As for weaknesses, the inability for blockchain technology to adapt to changing business environments as well as certain blockchain implementations not being fast enough were mentioned.
- As opportunities, the decentralization of the system and operational savings were seen as advantageous.
- As threats, in general, all points have specific blockchain security concerns in common.

In the study conducted by Castellanos et al., which performed a simulation of the Green Certificate Market on the Ethereum blockchain, the benefits obtained by using blockchain for this specific purpose are similar to what was concluded earlier [54]. Namely, using blockchain would ensure authenticity, transparency, as well as reduce transactional costs since there would be no need for a centralized party that would regulate the trading scheme.

In an interview with the European Energy Exchange, Alt and Wende talk about the potential of blockchain technology in the energy sector as being very high when using it for the Guarantees of Origin tracking instrument. They argue that using blockchain technology would enable smaller production device owners to more easily engage in the system. People with photovoltaic plants at their homes could be issued Guarantees of Origin by the Issuing Bodies, which they would be able to trade on the blockchain. The advantage of blockchain is seen as being able to provide a "*cost-effective connection of a large number of trading participants*" [55, p. 329].

Zhao et al. conducted a study by performing a simulation of the green certificates system developed on the blockchain. They argue that the traits of blockchain are perfectly suitable for the development of the green certificate market [56]. The advantage of the system is seen in the fact that the need for intermediaries is eliminated, prosumers and consumers establish direct interaction cost-effectively, and the transparency and immutability would create an authentic system necessary for a proper certificate market.

Hsiao analyzes the benefits of using blockchain for the EAC system used in the USA, called Renewable Energy Certificates (REC) [57]. The benefits of blockchain observed by Hsiao, are to standardize a fragmented regional registry structure, and provide an alternative that would handle the issue of fraud and double counting. In addition to these, the blockhain could also remove business middleman, since the validation processes can be integrated into the system. Lastly, blockchain would enable instant issuing of the certificates, with the verification process being performed automatically.

Zhang et al, analyze the disadvantages of the green energy certificate trading system in China [58]. They find that the main drawback of the existing system is the time-consuming process it takes energy producers to register into the system and prove their authenticity. They add that the requesting process is quite resource-intensive in terms of time and labor, lacking automation. Additionally, they identify that the system owner holds too much power over the process of issuing, monitoring, and revealing transactions, which leaves space for power abuse. However, it is not clear whether that would be a problem when a governmental organization is responsible for regulating the system. Lastly, they identify the system as being more vulnerable to attack, contrary to a decentralized blockchain system. They then go on to discuss the advantages of the blockchain system, mainly as having a better verification process with higher traceability and improved integrity. The higher degree of automation with less human intervention would improve efficiency and save cost. Since all the nodes would have access to the entire history of the transactions stored on the ledger, the information would be open and transparent.

Henderson et al. discuss the benefits of using blockchain for issuance and trading of Renewable Energy Certificates, and highlight that as reliable data is easily viewable on the encrypted ledger, there is no need for a central organization to validate the generation of data. They further argue that by streamlining trade authentication and data indexing, blockchain will help public bodies running RECs systems to cut costs [59].

4.2.1 Related Work

This section presents already existing studies and solutions that implement blockchain technology for the purpose of managing the tracking system for Guarantees of Origin and other types of Energy Attribute Certificates.

Energy Web Foundation (EWF) has developed a system under the name of Energy Web Origin (EW Origin) for tracking EACs. Energy Web Foundation is a nonprofit international organization, aiming to facilitate the transition to a low-carbon electricity system by leveraging blockchain and decentralized technologies [60]. The Energy Web Chain, an open-source blockchain network targeted to the industry's legislative, functional, and business needs, was introduced by the EWF in mid-2019. A very positive aspect of their system is the inclusion of grid operators, utilities, developers interested in renewable energy, and the community in their energy blockchain ecosystem. EW Origin is a system that was built upon the generic principles that are equally important for all the Energy Attribute Certificate systems, whether it be Guarantees of Origin (Europe), REC (North America), I-REC, etc. [61]. The system itself is a software development kit (SDK) that combines utilization of both on-chain and off-chain solutions to safely store private information off-chain, all the while maintaining its integrity and authenticity by utilizing blockchain technology. The EW Origin is composed of multiple modules, aimed at supporting the main EAC functionalities, as well as being in line with current legislation. The registry module stores data about users and devices, issuer module is used for issuing EACs and storing them on the blockchain, exchange module serves as a sort of a marketplace for assisting in the trade between buyers and sellers. The user interface module acts as a presentational layer to interconnect all the previously mentioned modules. EWF uses Ethereum as a basis of their technology stack, all the while tailoring the lacking aspects of Ethereum to the needs of the energy industry. The blockchain that EWF has developed is based on the Ethereum codebase, but it is not the Ethereum blockchain itself. EWF designs additional functionalities for the EW Chain, based on the communication with affiliates and participants of their ecosystem. Some of the adjustments that the EW Chain has made is improving scalability and transaction finality, by utilizing a different consensus algorithm than Ethereum blockchain which currently uses Proof-of-Work [32]. EWF launched a beta version of the public test network called Tobalaba, an Energy Web Blockchain for the energy sector, that utilizes Proof-of-Authority consensus protocol [51]. This specific Proof-of-Authority algorithm is called Aura, and it was developed by Parity, nowadays called OpenEthereum [62]. Aura stands for authority round since the generation of blocks is governed by a special type of nodes called authority nodes in a round-robin mode, where a block is verified in a time slot assigned to a validator node [51]. Each round one validator is assigned a primary validator role, responsible for proposing new blocks. When a validator fails to create a block due to hardware or other problems, the next validator is selected to process the remaining transactions. Other validator nodes verify that the transactions on the block are legitimate and the block is added to the chain.

Energy Web states that utilizing the PoA consensus algorithm can reach 30x greater throughput than the Ethereum mainnet, decrease the energy consumption by 54000x and lower the network costs by 500x, due to the PoA being a less resource-intensive algorithm, than PoW [63]. Energy Web further states that utilizing PoA enhances compliance with regulatory requirements since blocks are no longer mined by anonymous miners, rather they are mined by certified validator nodes. While Energy Web blockchain has an open access to all parties, not everyone can host a validator node and participate in the validation process. That means that the blockchain is owned by a small amount of authority nodes, that validate all the transactions on the network.

EWF specifies eligibility criteria by which it is defined which party can host a validator node [64]. Namely, all the validators must be registered organizations, be an official member of the EWF, and they must meet all the technical and security parameters specified by the EWF. To become an EWF member, it is further necessary to make a financial contribution or contribution via engaging in software development support to the Energy Web Chain. Furthermore, the onboarding process to the EWF requires know-your-customer (KYC) and anti-money-laundering (AML) procedures, to establish trustworthiness and authority among the validator nodes. Even though the Energy Web Chain promises much better performance and lesser resource utilization, it has met some criticism. Some have accused the EWF of being a "club for big utilities" [52]. This is primarily due to their utilization of Proof-of-Authority consensus protocol, which centralizes the ownership of the blockchain to the well-established parties that participate in the mining process as validator nodes. However, even though Energy Web Chain is not fully decentralized, it is not a black box solution either.

EWF provides extensive documentation of the Energy Web Chain on their open-source Wiki page, which proves their desire to provide transparent information to all the relevant parties. The eligibility criteria for joining the EWF and getting the status of a validator node is quite rational as they aim to include only those parties that have a stake in the energy business processes. Furthermore, EWF has already obtained a huge number of trusted affiliates, and this higher amount of validator nodes contributes to more decentralization and trustworthiness. From the perspective of the system for issuing EACs, this approach seems to be quite fitting for the use case. The existence of a lot of legislative and regulatory requirements in the process of issuing the EACs indicates that there are parties that must monitor and validate the legal execution of these processes. Therefore, a fully decentralized solution based on PoW or PoS may be a worse option than a more centralized blockchain solution based on PoA, especially when taking into consideration the performance gains.

Blockchain technology has not been utilized only in Europe for issuing Guarantees of Origin, but also for issuing Renewable Energy Certificates (RECs). IDEO CoLab, Filament, and Nasdaq have been testing blockchain solutions that allow power generators and others to sell EACs. For example, IDEO CoLab created a proof-of-concept system that used Nasdaq's Linq platform and Filament's firmware to issue RECs to photovoltaic suppliers to every kWH their panels generate, allowing even small solar producers to accurately control, claim, and exchange energy [65]. The prototype composes of the IoT modules that measure the creation of electricity by the solar panels, and then securely share the data with Nasdaq's platform for issuing and trading financial assets. When the data has arrived at the platform it is converted to a financial asset, as a way of reimbursing the producer for creating renewable energy. The asset can then be transferred to a utility that can perform a retirement process of the certificate. In the end, they conclude that while this would create a decentralized scale-free network, different issues would arise, such as the lack of accounting processes for tracking who put the energy on or off the grid, setting a proper price, determining the reimbursement amount and how much should a consumer be charged [66]. Consequently, though the decentralization and automation of the network would have their benefits, the development process would require a comprehensive approach in solving all the accounting issues that a fully decentralized solution would bring. That is why most of the solutions for issuing the EACs on blockchain retain some degree of centralization or build some off-chain solutions that work hand-in-hand with the blockchain solution.

Swytch, a non-profit foundation from Zug, Switzerland has developed a blockchain based decentralized platform, that rewards sustainability efforts by issuing Swytch tokens [67]. Their platform utilizes unique blockchain protocols for security and verification purposes, developing a system that is connected with smart meter data, IoT devices, and storage systems. The platform issues Swytch tokens to incentivize renewable energy production and create a system that can be used as an alternative to the existing voluntary system of tracking Energy Attribute Certificates. Swytch claims that they will not utilize the PoW consensus algorithm, rather more energy-efficient solutions will be considered. Recently, Swytch has become an affiliate to the EWF, whose open-source tools and blockchain will be used to refine the Swytch platform [68].

On a related note, Energy Blockchain Labs partnered with IBM Blockchain technology to create "*an efficient, transparent platform that allows high-emission organizations to meet quotas by buying carbon credits from low emitters*" [69]. The solution is based on IBM's Hyperledger blockchain technology, which is a permissioned blockchain in which transactions can be both private and public, to create a solution that incentivizes environmentally beneficial activities among the companies in China. The platform that was developed by Energy blockchain Labs provides an easier way for organizations to engage in Carbon Emissions Trading. Carbon emission trading is essentially, a type of policy allowing companies to buy or sell allocations of carbon dioxide output that is granted by the government. Governments issue a finite amount of CO2 credits to companies, and companies are allowed to release the amounts of CO2 equal to the amount of the credits they have. If company emits less than its limit, it can sell the surplus of CO2 credits to other companies, in an effort to reduce the total amount of emitted CO2 [70]. The assumed results of utilizing the blockchain technology were that the expected the average 10-month carbon asset development cycle was to be reduced by 20%-50%, improve efficiency and promote green technology. The expectations are that the long-term results of utilizing a more efficient way for carbon asset development and trading, would reduce the carbon emissions and motivate more investment in green energy solutions.

Another example worth mentioning relates to the Spanish utility company Iberdrola which used a platform developed by a Barcelona-based company FlexiDAO on the EWF platform. FlexiDAO used the EW Origin system to develop their own Spring framework, used as a proof of concept by Iberdrola and Spanish bank Kutxabank [71]. The system is composed of multiple layers. The metering module is connected to retailer’s system for tracking generation data. Smart contracts perform the automatic matching process as well as processes related to energy certificates, like importing the asset, issuing, trading and claiming. Furthermore, the system contains application layers such as an administration module for managing users, digital identities and private key storage. Lastly, there exists a user interface that enables easy interaction of a consumer or an energy producer with the system. The Spring system they have developed is claimed to be fully compatible with the existing GO mechanisms, while also providing more transparency and automating the processes related to tracking the certificates. Lastly, they claim that using blockchain would reduce the existing workloads by three Full-time-Equivalents for both retailer and the consumer [72]. However, certificates produced by Iberdrola’s platform can not be considered as real Guarantees of Origin, since they have not been issued by the Spain’s legally assigned Issuing Body CNMC [18]. This was acknowledged by Iberdrola, and was justified by the fact that this was only a proof of concept solution.

On a similar note, a grid operator in the United States known as PJM Interconnection has also collaborated with EWF and tested their system for tracking, trading and verifying renewable energy certificates (RECs), motivated by the fact that their current process of auditing and providing documentation is quite slow [71]. A US-based company EnLedger has developed Energy chain, an energy services platform based on the blockchain technology that supports a multitude of energy-related processes among which also the issuing of renewable certificates and credits [51]. EnergyChain is a private blockchain that supports smart contracts, and has a cryptocurrency called EECoin intended to be used for paying blockchain transaction fees [73]. EnergyChain is based on a Delegated Proof-of-Authority consensus protocol, with a high transaction throughput and private set of node validators. Furthermore, the EnergyChain itself was built upon Cosmos Tendermint SDK [74].

Other examples worth mentioning are the Spanish company Acciona that launched the system for tracking the origin of renewable energy as well as the Singaporean company SP Group that has launched a similar system in Asia [75]. Both of these projects were done in collaboration with EWF.

4.2.2 Technology Selection Process

This section presents a discussion and argumentation of the blockchain technology selection process for the development of the Guarantees of Origin prototype, the implementation of which is documented in the next chapter. For this selection, several criteria were considered. Firstly, the selected blockchain technology has to support smart contracts. Secondly, the consensus algorithm that the blockchain technology is using, is evaluated. Since blockchain performance is heavily dependent on the consensus algorithm, other criteria can be based on it, such as transaction throughput and scalability. Furthermore, criteria such as security, robustness, the size of the development community, and whether the blockchain is public or private, were considered.

Based on the discussion about the system for Guarantees of Origin in the previous chapters, one of the important features that a blockchain implementation would need to have is a good transaction throughput. This is based on the fact that the number of GOs traded keeps increasing every year, accompanied by an increase in the supply and demand for renewable energy. Therefore, the new blockchain system would also need to be developed with good scalability in mind. The GO tracking instrument is conditioned by a fairly complex background of legal requirements, so security is also a high priority feature to be expected from it. In addition, any new system would surely have to be developed in compatibility with the current systems, to prevent double-counting of the same MWh of electricity, which is an issue that a system for tracking GOs must avoid.

Another feature that such a system should have is the privacy of transactions, which is an aspect of blockchain that is rather hard to achieve, since it is generally known as an open and transparent network. However, in the system for tracking GOs and Energy Attribute Certificates in general, there may certainly exist a business use case, where a transaction is wanted to be kept completely or partially private.

The blockchain technologies that support smart contracts, and could be used for the development of the prototype are listed and briefly described below:

- Ethereum – is the first blockchain technology that made smart contracts available as a pioneer in the field. Ethereum is an open-source blockchain distributed computing platform that was initially proposed by Vitalik Buterin [45]. Ethereum blockchain is based on a virtual environment in which all the smart contracts are executed called Ethereum Virtual Machine (EVM) [76]. Even though a blockchain is composed of multiple nodes, EVM appears as a single entity being maintained by thousands of nodes [77]. Ethereum has a known issue with high transaction fees and sometimes very low transaction throughput. This is due to Ethereum using the PoW consensus algorithm, which tends to suffer from scalability issues. However, Ethereum is currently going through some modifications as it is transitioning the protocol to the Ethereum 2.0 version. A major change that Ethereum 2.0 is going to introduce is switching from PoW to PoS consensus protocol [78]. Additionally, in Ethereum 2.0, the EVM is being replaced with Ethereum WebAssembly (EWASM), and a big change in the architecture is being introduced with the concept of sharding. Sharding is the process of splitting database horizontally to spread the load, which in terms of the Ethereum blockchain comes down to spreading the load across 64 new chains in an effort to improve scalability of the network [79]. Currently, to develop smart contracts on Ethereum, it is possible to use programming languages such as Solidity, Vyper and Bamboo [80]. However, with the transition to Ethereum 2.0 and switching of EVM to EWASM (based on Web Assembly), it will be possible to code smart contracts on Ethereum using more popular programming languages such as C, C++ and Rust [81]. In terms of scalability and transaction throughput, Ethereum currently fairs quite poorly with a throughput of around 15-30 TPS (transactions per second) [82], however, it is assumed that the switch to Ethereum 2.0 protocol will be able to achieve 100 000 TPS [83].

- Quorum - is a blockchain platform that is a fork of Ethereum, initially developed by JP Morgan as a permissioned blockchain [84]. Blockchain fork is a split in a blockchain network, that creates an alternative chain. This is enabled by the open-source attribute of blockchains, by which anyone is available to create an alternative chain by forking the existing one. Forking is usually performed to mitigate the adverse effects of bugs or to create alternate blockchain with new features [85]. Quorum introduced changes to the Ethereum technology such as limiting participation to a known set of nodes, replacing PoW consensus protocol with the crash fault tolerant RAFT and IBFT protocols. These protocols result in a faster consensus process and it is possible to use them because Quorum is a permissioned network, and all nodes that are participants in the network are known to each other. In addition to these features, Quorum supports private transactions, which allows a subset of parties in consortia to privately transact, while the information about the transaction is hidden from the members of a larger consortia [86]. It is enabled by the ledger being split into a public and a private one, with a public ledger being available to anyone, and a private ledger only being available to a subset of parties who hold the key to decode it. Likewise, it is possible to deploy smart contracts privately and have the logic and transactions related to it being visible only to the parties involved in the process.

Quorum was recently acquired by ConsenSys from their original developer JPMorgan Chase [87].

- Hyperledger Fabric - is an enterprise-grade blockchain solution maintained by IBM and Linux Foundation [88]. Fabric is only one of the projects under the umbrella of Hyperledger blockchain solutions. Hyperledger Fabric is a permissioned blockchain that does not implement any cryptocurrencies. The consensus protocol utilized is the practical Byzantine fault tolerance (pBFT) algorithm. However, Fabric supports pluggable consensus mechanisms, meaning that it supports choosing which consensus protocol will be used based on the project requirements [89]. In terms of scalability and transaction throughput, Hyperledger Fabric is said to support about 1000 TPS [82]. Fabric's big advantage is with not using and Domain Specific Language (DSL), but rather smart contracts can be written in general-purpose programming languages such as Java, Go and Node.js [90]. Since Fabric blockchain platform is a permissioned system, it has known participants that may not entirely trust each other. A system can be built that is based on the existing trust between parties that may be based on legal agreements and dispute control mechanisms.
- R3 Corda – is a distributed ledger technology, that is not primarily blockchain-based since the layer of persistence is in the form of a Directed Acyclic Graph [91]. Corda is based around processing financial transactions more efficiently, therefore it is mainly centered around the financial industry [92]. Corda is also a type of permissioned blockchain network, being described as a system that enables business parties to transact privately and directly. It supports writing smart contracts in Java and Kotlin programming languages. Furthermore, similar to Hyperledger Fabric consensus algorithm is pluggable, therefore it can be adjusted to the use case. Since the Corda network is based around permissioned consensus protocols, it has a high transaction throughput and scalability. The consensus in Corda is achieved when the signatures of all the parties participating in the transaction are obtained, therefore it does not communicate with the parties not involved with the transaction to perform validation. This feature makes Corda transactions more efficient, however, some describe it more as a messaging protocol rather than a real blockchain. However, the process of creating the Corda network and configuring nodes is described as manual and cumbersome [93].
- EOS – is defined as a decentralized operating system based on blockchain. EOS is focused on providing a decentralized environment for creating and hosting secure decentralized applications [94]. In the EOS environment, all the nodes agree to a “constitution”, which is a set of rules specified by the EOS community. EOS is said to support all the features

for developing, hosting, providing security and authentication features. Furthermore, it is said to be a highly scalable network, with the maximum throughput achieved on the network being close to 4000 TPS [95]. EOS is based on Delegated Proof-of-Stake consensus protocol, which is one of the main reasons for its high scalability. Using this protocol enables the commodity of easily implementing high-level decisions regarding the technology modifications, like rolling back or fixing bugs instantaneously if the majority agreement among stakeholders is reached. However, based on some test evaluations of EOS system, the argument is that it would better act as a side chain to more secure networks, due to its inherent security issues [96].

- Tezos – is an open-source decentralized blockchain network, founded by Arthur Breitman in 2017 [97]. The idea behind the Tezos blockchain is to solve the issue with blockchain forks [98]. As mentioned earlier, blockchain forks occur when fixing major bugs or adding new features to the blockchain. It usually results in the creation of two separate blockchain networks, in which one blockchain diverges from a point in time at which a bug occurred or a major upgrade was added. Tezos introduced the concept of on-chain governance with self-amendments. This means that when an agreement of stakeholders is reached, blockchain protocol can be upgraded based on the approved developers' suggestions. Tezos is also focused on solving the smart contracts security issues. Tezos facilitates a method to improve the security of smart contracts using proving methods based on mathematical properties of the program [99]. This method is called formal verification. Tezos developed its smart contract language called Michelson, which is quite complex compared to other languages mostly due to improved security. An audit is performed on every contract before deployment, making Tezos smart contracts very secure. That is one of the reasons why Tezos is very popular in industries that require high security and precision such as aeronautics, healthcare, and the nuclear industry. Tezos utilizes a unique implementation of the Delegated Proof-of-Stake consensus algorithm. In order to become a delegate in the network, a participant needs to hold 8000 Tezos [100].

When choosing a blockchain platform to use for the prototype, the decision was to choose technology centered around Ethereum, therefore the prototype was written using Solidity language. The reasons for choosing Ethereum is that it is one of the biggest blockchain ecosystems for developing general use case decentralized applications. It is an open-source project extensively maintained by its community and is constantly receiving a lot of upgrades and improvements. Even though the smart contracts programming language Solidity did have some major security flaws in the early days of the network, these have nowadays been solved and are well documented. In addition to that, the Ethereum network has been quite often attacked, which resulted in major security upgrades, especially in terms of guidelines of how to write more secure contracts in Solidity. Therefore, the Ethereum ecosystem is currently quite resilient and robust. Lastly, the availability of a large community provides extensive resources on how to approach various challenges encountered in the development process. Furthermore, the implementation for the EAC tracking system on the blockchain implemented by the EWF called EW Origin was also developed based on the Ethereum technology stack. It served as a major aid when solving technical challenges such as choosing the token standard around which the certificate would be centered, or solving the privacy issue. Even though the Ethereum mainnet has become known as a very congested network with a small transaction throughput, it is constantly evolving and has a large group of project contributors and maintainers. In addition to that, the improvements that Ethereum is going to receive in the upgrade to the 2.0 protocol are going to solve many of the current issues. Table 1 presents a comparison of various blockchain platforms.

Table 1: Blockchain platforms comparison

	Ethereum	**Quorum**	Fabric	R3 Corda	EOS	Tezos
Permission type	Public, permissioned solutions based on Ethereum exist	Permissioned, private	Permissioned, private	Permissioned, private	Custom, built for both public and private use cases	Public
Execution environment	EVM	EVM	Docker	Java VM	EOS VM	Tezos VM
Smart contract language	Solidity, Vyper	Solidity, Vyper	Go, JavaScript, Java	Kotlin	C++	Michelson
Transactions per second (TPS)	15-30 TPS (with Ethereum 2.0 up to 100 000 TPS)	Cloud instances average around 240 TPS	Scalable to 20 000 TPS	15-1678 TPS	1000+ TPS, all time high 3996 TPS	Approx. 40 TPS
Consensus protocol	PoW, transitioning to PoS	Pluggable PoA protocols (IBFT and RAFT)	Crash Fault Protocol – a backend service	Based on a multilateral agreement between parties involved	Delegated Proof-of-Stake (DPoS)	A unique type of Delegated Proof-of-Stake (DPoS)

Source: Adapted from [82], [92] *Copyright © 2018-2019, IEEE, Extended with* [101-105]

4.2.3 Privacy considerations

One of the requirements for the Guarantees of Origin system is that not all transactions should be public and there should be a certain level of privacy for some transactions. However, achieving privacy on blockchain may be hard since, for most of the blockchain technologies, the distributed ledger means that everyone can see the parties involved in the transaction, as well as the transaction data. Therefore, transaction privacy is not inherent in the blockchain network. However, as discussed earlier, there are blockchain solutions that enable a degree of privacy on the blockchain. These solutions are discussed below.

Earlier sections have mentioned Quorum as a blockchain solution that supports privacy. Quorum enables privacy between the participants involved in the private transaction without other participants in the network being able to see the contents of the transaction. The support for the private transaction is enabled through the separation of public and private state. Quorum achieves this by adding a privacy layer to the existing core public Ethereum based layer. The public transactions are executed similarly to standard Ethereum transactions and they are visible to everyone. Private transactions in Quorum are enabled by a Private Transaction Manager called Tessera.

Tessera is a separate software component that is in charge of storing the private transaction and making sure that they are visible exclusively to the transaction participants. Quorum enables rather easy deployment of private contracts by specifying the account addresses that are to participate in the transaction and adding them to the *privateFor* field. When the *privateFor* field is specified, Quorum detects that private contract should be deployed. Private transaction manager Tessera works in conjunction with Enclave, which acts as a virtual Hardware Security module, and is responsible for managing the encryption and decryption of the data in isolation [106]. However, in the context of implementing an asset tracking system for Guarantees of Origin, Quorum’s private contracts may not be a feasible option because it maintains a private ledger separate from the public one. Since the transaction occurs privately, it does not prevent double-spending of digital assets exchanged as part of private transactions.

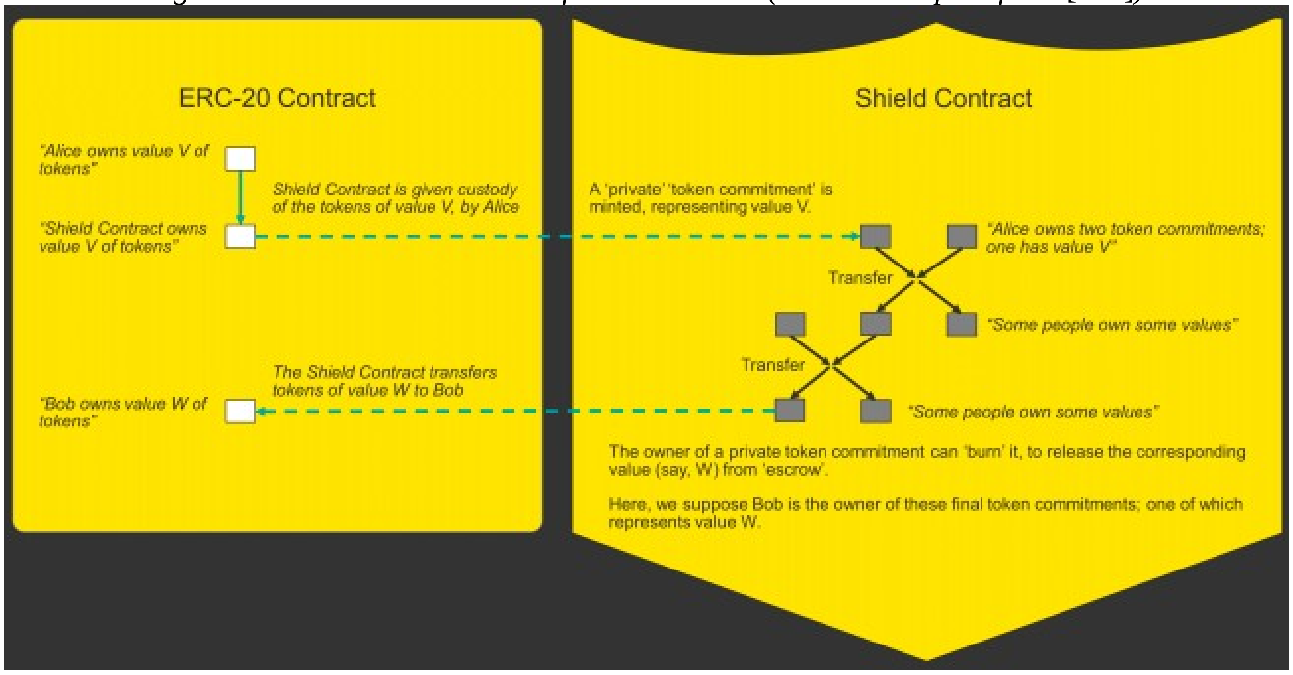
That is why Quorum developed a proof of concept integration of ZSL (zero-knowledge security layer) to enable the private transfer of digital assets in which the sender, recipient, and quantity of assets remain hidden. ZSL protocol was designed by the same people that are behind Zcash [107], a cryptocurrency that provides enhanced privacy options, leveraging the cryptographic proofs method called zk-SNARKs. Zk-SNARK stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge. It is a cryptographic method that enables one party to prove the knowledge of some information without actually revealing that information [108]. These digital assets, called by developers as “z-tokens”, are only transacted privately. When a private transaction occurs, a proof is presented to a private contract, by which the private contract can update its state based on the execution of the transaction. By utilizing private contracts enabled by Quorum, privacy and confidentiality are maintained, with the addition of fulfilling the necessary use case liabilities. However, this development of this solution has haltered and the developers warn that this solution is only a Proof-of-Concept and that it is not production-ready. Based on these arguments using it for the system for Guarantees of Origin would require further testing and security audit, as well as some potential development work to adjust it for this specific use case.

Another technology that is based on zk-SNARKs technology that enables privacy on the blockchain is Nigthfall, a project developed by Ernst & Young (EY) that was released in 2019 [109]. Nightfall provides a set of smart contracts and microservices that enable completely private transactions on the Ethereum blockchain. It is based on a zk-SNARK toolkit called Zokrates. Zokrates is a toolbox for zk-SNARKs on Ethereum that enables verifiable computation and generation of computational proofs in Solidity smart contract language [110]. Nigthfall enables privacy when transacting ERC-20 and ERC-721 Ethereum token standards [111]. These token standards are a common way to represent digital assets on the Ethereum network, and are explained below:

- ERC-20 is a standard for fungible tokens. Fungible token is a token which is same in type and value to other tokens of the same kind [112]. For example, 5 Ethereum tokens is the same as other 5 Ethereum tokens, 10 dollars bill is not different in type and value than another 10 dollar bill.
- ERC-721 is a standard for non-fungible tokens (NFT). Each NFT is unique in its way, and usually represents collectible items, access keys, lottery tickets [113].

Nightfall enables keeping the receiver and the amount of assets exchanged private. The sender of the transaction (known also as the Prover) will perform a computation privately off-chain, by using the set of private inputs and computing the set of public outputs that are stored on the blockchain. The public outputs are encrypted and unreadable to anyone other than the sender and the receiver. The sender of the assets will generate proof that shows that the public outputs were computed correctly, and will share them on the blockchain. The public outputs and the proof can then be used by any other party on the blockchain to verify that a pre-agreed calculation representing the transfer of the funds was initiated by the sender [114]. Nightfall uses smart contracts called Shield contracts that store the tokens of the sender as a private token commitment that can be further privately transacted. When the commitment is burned, the Shield contract transfers the tokens to the owner. Figure 8 shows a high-level diagram of how the shielded transaction of ERC-20 token is performed.

Figure 8: Shielded transaction of ERC-20 token (Source: Adopted from [114]*)*


ERC-20 Contract
Shield Contract
"Alice owns value V of tokens"
Shield Contract is given custody of the tokens of value V, by Alice
"Shield Contract owns value V of tokens"
A 'private' 'token commitment' is minted, representing value V.
"Alice owns two token commitments; one has value V"
Transfer
"Some people own some values"
Transfer
The Shield Contract transfers tokens of value W to Bob
"Bob owns value W of tokens"
"Some people own some values"
The owner of a private token commitment can 'burn' it, to release the corresponding value (say, W) from 'escrow'.
Here, we suppose Bob is the owner of these final token commitments; one of which represents value W.

Nightfall is a robust and complex solution for enabling privacy on the Ethereum blockchain. However, it has a few drawbacks. One of them is that the code security review has not yet been completed, and it is not suggested to use it in the production environment, especially for transferring assets with a physical value. In addition to that, Nightfall requires the existence of a trusted benefactor that would perform a setup of complex infrastructure, and deploy the Shield smart contract to the Ethereum network. The creation of the proofs on the client-side is quite a time-consuming process, as it may take up to 10 minutes. The last drawback to mention is inherent to the Zero-Knowledge algorithms themselves. Namely, a situation can occur when a party creates a large amount of tokens, and sends them directly to another party, that immediately converts them from private commitment to typical ERC-20 tokens. The appearance of that large amount on the account of the receiver party, can be linked directly to the disappearance of the same amount on the account of the sending party.

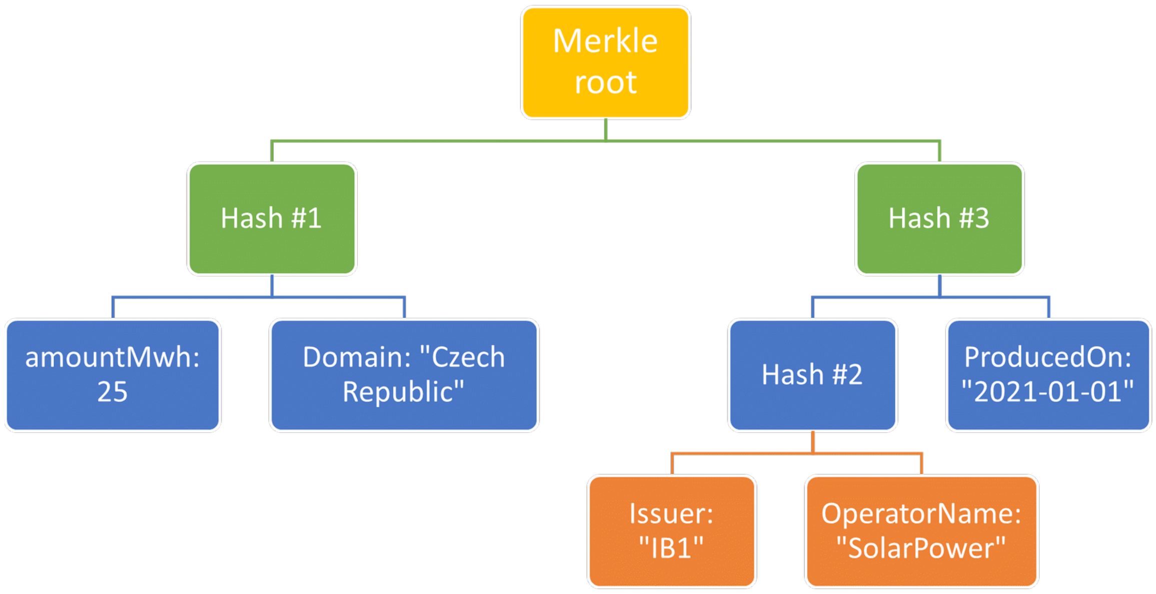
Aztec protocol – another privacy solution based on Zcash's zk-SNARKs cryptographic method that enables token transaction that keeps the amounts hidden. The Aztec protocol uses Zero-Knowledge proofs to enable confidential transactions for any type of Ethereum based digital asset. Just like the Nightfall protocol described earlier, Aztec protocol also requires performing a trusted setup [115]. Aztec protocol provides an open-source layer 2 network that aims at providing the Ethereum network with privacy and scalability [116]. The privacy is achieved by using Zero-Knowledge proofs, whereas the scalability is based on the PLONK technology, which performs a batching of transactions using the rollup zkSNARK, scaling the network by 300 transactions per second when the demand is high [117]. The Aztec protocol is a very promising project for privacy on Ethereum, and it recently deployed a beta version of their rollup service on the Ethereum mainnet, enabling shielding of Ethereum, privately transacting Ethereum, and then unshielding the shielded amounts [118]. There is no support for ERC-20 tokens yet, and it is uncertain if ERC-1155 multi-token standard that was used in the prototype described in the next section, would be supported.

Energy Web Foundation performed an analysis of the existing solutions for enabling privacy on the blockchain. One notable example is a method based on Merkle trees called Precise Proofs, which enables proving the authenticity of the document using a revealed subset of the document, while keeping the rest of the document private [119]. Merkle tree is a tree data structure that represents a binary tree in which the value of the inner node is the hash of the leaf nodes. The root node of the tree is called the Merkle root [120]. Precise proofs is a lighter approach to the Zero-Knowledge proofs mentioned earlier and can be used as an alternative to it if all the parties involved have had contact with the data at a certain moment. The method is performed on a key-value pair document, by which the key-pair value represents a leaf in a Merkle tree. Each leaf value is then hashed with its neighboring leaf value until the root of the tree has been calculated. The Merkle root can then be stored on a blockchain, and a verifying party can submit a proof of knowledge of the subset of the document that it wants to prove is authentic, along with the rest of the hashes of the document. The proof is then checked against the Merkle root stored on the blockchain. The subset of the document and the neighboring hashes constitute the proof, by which the root of the tree can be calculated. If the Merkle root that is calculated from the proof equals the Merkle root stored on the blockchain, then it is obvious that the subset of the document truly belongs to the original document, the full content of which remains private. Let us say for exemplary purposes that the Guarantee of Origin certificates has the following structure:

```
{
issuer: "CZIB",
domain: "Czech Republic",
productionPeriodStart: "2021-01-01",
productionPeriodEnd: "2021-01-10",
operatorName: "SolarPower",
dateRequested: new Date().toDateString(),
productionDeviceLocation: "Hradec Kralove, CZ",
technology: "T010011 / Solar",
productionDevice: "Solar Panel Device 4",
productionDeviceGSRN: "4400000440004444400444",
installedCapacity: "30Mw",
amount: "25Mwh"
}
```

Based on the data of the Guarantee of Origin certificate, the corresponding Merkle tree would have the structure as displayed in Figure 9.

Figure 9: Merkle tree example of GO certificate data (Source: Author's elaboration)



Then, if a certain party would want to prove that it knows the subset of the document, for example, that it knows the amount of MWh for a certain certificate they could generate the following proof:

```
Property: “amountMwh”,
Value: “25”,
Hashes: [“rightLeaf:”domainHash”, “rightLeaf”:”Hash#3”]
```

Then, based on the value and hashes provided, the proving party would be able to calculate the Merkle root by hashing the amount value with the domain hash value obtaining hash #1, and then hashing the obtained hash #1 with the provided hash #3. The calculated Merkle root would then be compared to the one that was published on the blockchain and if they were equal it would mean that the provided amount value was correct. Precise proofs as a concept was initially developed for Centrifuge OS, a protocol for decentralized asset finance, to enable proving the authenticity of data subsets in confidential business transactions [121]. The library for precise proofs that they developed was written in the Go programming language.

Energy Web utilizes precise proofs in their EW Origin system. They have developed a custom implementation of the precise proofs concept, enabling the possibility to reveal a certain subset of the document to a third party electricity market governing body, without revealing the whole information about the certificate [119]. Energy Web has also developed a JavaScript library for creating Merkle tree roots on the client-side, as well as creating the proofs for verifying data subsets. The JavaScript library that they have developed was used in the prototype described in the next chapter to showcase how certain data about the GO certificate can remain confidential.

5 Results and Discussion

This chapter is divided into two main sections: Implementation of a Prototype / Proof of Concept and Discussion. The Implementation of the Prototype section presents technical documentation of the prototype which is developed as a part of this thesis and its architectural overview. The Discussion section evaluates the prototype implementation, as well as assesses the suitability of utilizing blockchain technology for the Guarantees of Origin tracking instrument.

5.1 Implementation of a Prototype / Proof of Concept

This section provides documentation of the technology stack utilized for the development of the prototype. It also analyzes the data representation of the Guarantee of Origin certificates when storing it on the blockchain. Furthermore, the architectural overview of the prototype is presented and the smart contracts relationships and data attributes are documented. Lastly, the functionalities of the prototype implementation are documented in the form of pseudo-code snippets.

5.1.1 Technology stack

The technology stack utilized for the development of the prototype is composed of the following main technologies: Solidity, Truffle, Ganache and OpenZeppelin Contracts.

5.1.1.1 Solidity

The prototype for issuing the Guarantees of Origin using blockchain was developed using Solidity programming language for writing smart contracts. Solidity is an object-oriented, high-level language for writing smart contracts, influenced by C++, Python and JavaScript, tailored for developing contracts that run on Ethereum Virtual Machine (EVM) [122]. Writing smart contracts in Solidity does not necessitate deploying only to Ethereum blockchain, but also allows deploying on other blockchain platforms based on Ethereum Virtual Machine, such as permissioned blockchains Quorum, Parity (now OpenEthereum), Pantheon (nowadays known as HyperLedger Besu [91]), which are made for enterprise use and provide permissioning feature on top of Ethereum blockchain [123]. Therefore, writing code in Solidity enables easy migration to different blockchain platforms that support it. The prototype described below was compiled using Solidity compiler version 0.7.0. The language itself is statically typed, supports external libraries, multiple inheritance, as well as user-defined types. Since Solidity is a smart contract language, it defines the interaction between the Ethereum accounts with the procedures executed in the smart contracts, as well as the state and the persistence layer of the blockchain. A simple example of the Solidity syntax can be seen in the code snippet below.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.8.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

The code example in the figure above shows a simple smart contract that allows setting and getting an unsigned integer value from the storage. The *public* modifier means that the functions can be called both from outside as well as inside the contract. The first line in the example indicates the license of the contract. The second line specifies the version of the Solidity compiler which the smart contract was written for. The code inside the contract object contains functions and state variables of the smart contract. Modifier *uint* specifies an unsigned integer state variable. Function *set* with a public modifier is a public setter function for setting the state variable. Function *get* is a public getter function that returns the value of the state variable. The *view* modifier indicates that the function does not modify that state, but rather it is a read-only function that retrieves the state variable from the smart contract state. Solidity smart contract code is stored in files with .sol extension.

5.1.1.2 Truffle

Truffle is a development framework for blockchains based on Ethereum Virtual Machine (EVM) that provides features such as integrated smart contract compilation, automated testing, scriptable deployments, interactive console, external script runner, and many more [124]. Since Truffle is built for EVM based blockchains, it can work with deployments to live Ethereum networks, as well as other blockchain networks, and it has built-in support for Quorum as well as Hyperledger EVM. Truffle also comes with an integrated test framework for writing smart contract tests and supports tests written using Solidity or JavaScript programming languages. For JavaScript tests, it uses popular testing libraries Mocha and Chai framework for writing assertions.

5.1.1.3 Ganache

Ganache is a one-click personal local blockchain based on Ethereum that provides features for smart contract development throughout all of the development cycles [125]. Ganache is a part of the Truffle suite, so it is developed by the same team that is behind the Truffle development framework. Ganache provides features such as the environment for running tests, smart contract state inspection, blockchain log output preview, block explorer, as well as advanced mining controls to specify the methods in which the blocks are mined. Ganache can be used through two types of user interfaces, namely graphical user interface (GUI) and command-line interface (CLI). Apart from developing decentralized applications based on Ethereum, Ganache can be used as a development environment for Corda distributed application development as well.

5.1.1.4 OpenZeppelin Contracts

OpenZeppelin is a library of smart contracts, that developers can use to improve smart contract security, as the library implements security design patterns, has been tested and community reviewed [126]. OpenZeppelin provides multiple utility smart contract to abstract numerous general concepts, such as the implementation of standards (ERC-20 and ERC 721), extendable components for complex custom contracts, as well as role-based permissioning procedures. Furthermore, OpenZeppelin contracts have been security audited by prominent security companies [126]. OpenZeppelin library is organized in sub-modules based on use-cases. For example, the access sub-module provides smart contracts for managing the access control and ownership of smart contracts. The tokens sub-module provides extendable interfaces for the most frequently used tokens. The math sub-module provides smart contracts that define secure mathematical operations to prevent calculation bugs such as integer overflow, division by zero, etc. The smart contracts utilized in the implementation of the Guarantees of Origin prototype are *Ownable.sol* smart contract for managing the smart contract ownership, and ERC-1155 related contracts for managing the token operations based on the multi-token standard that is discussed in the following section.

5.1.2 Guarantee of Origin Blockchain Representation

One of the challenges encountered when using blockchain for Guarantees of Origin was the data representation of the certificate on the platform. The representation of Guarantee of Origin may be thought of as a digital asset that is created, issued, owned, transferred, and later on burned (cancelled). This idea is relatable to the already widely used concept of tokens on the blockchain. The tokens on Ethereum are used to represent a digital asset that can be mapped to a physical object such as gold, vouchers, cards, or anything else with an underlying economical or accounting value. Since the Ethereum community has already developed standards for creating, transferring, and burning tokens, and these standards are widely adopted, it made sense to use them for representing the Guarantees of Origin in the prototype. Furthermore, using a standardized implementation means that lot of potential security and compatibility issues have already been addressed by the community. Previous chapters mentioned the token standards such as fungible ERC-20 token and non-fungible ERC-721 token. The fungible ERC-20 token is a token standard in which all the tokens that conform to it are fungible, meaning that they are the same in type and value. Therefore, they are commonly used to represent some currency, e.g. a one-dollar note is equal to another dollar note in value and type, one Ethereum coin is equal in value and type to another Ethereum coin. On the other hand, a non-fungible standard ERC-721 represents a unique digital asset like a photo, video, a digital file, or a collectible item.

So, for a smart contract to be considered as an ERC-20 or ERC-721 token, it needs to implement methods and events defined by the standard. The methods and events defined for each standard can be previewed on Ethereum Improvement Proposals (EIP) page [112]. For example, some of the mandatory methods for ERC-20 token standard are *totalSupply* (method for getting the total supply of token), *balanceOf* (for checking the account balance), *transfer* (for transferring a certain amount of ERC-20 tokens from the address of the sender to the address of the receiver).

The mandatory methods for the ERC-721 non-fungible token standards are quite similar, with the addition of the *safeTransferFrom* method which calls a hook function on the receiver contract to ensure that it is aware of the ERC-721 standard. This is done to prevent tokens from being sent to a smart contract that is not aware of them after which the tokens can no longer be recovered. Based on the description of the token standards, it might seem that using the ERC-721 non-fungible token standard is a proper option to choose for the EAC use-case, since each Guarantee of Origin has some unique characteristics such as location, time, source of renewable energy production etc. This information distinguishes each certificate from one another and is quite significant to a party that wants to obtain them. Therefore, since the ERC-20 token is meant to represent the assets that have the same value and type, it would mean that the relevant information about the GO certificate would not be present in the token. On the other hand, even though each GO has unique production characteristics, it has a fungible aspect as well, represented by the amount of MWh produced. Implementing GOs using the ERC-721 token would mean that there would be a separate token contract created for each GO, which might result in a huge number of smart contracts created and complexity in managing the ownership of the huge number of these tokens. Therefore, the optimal solution would be the one that combines both of these token types.

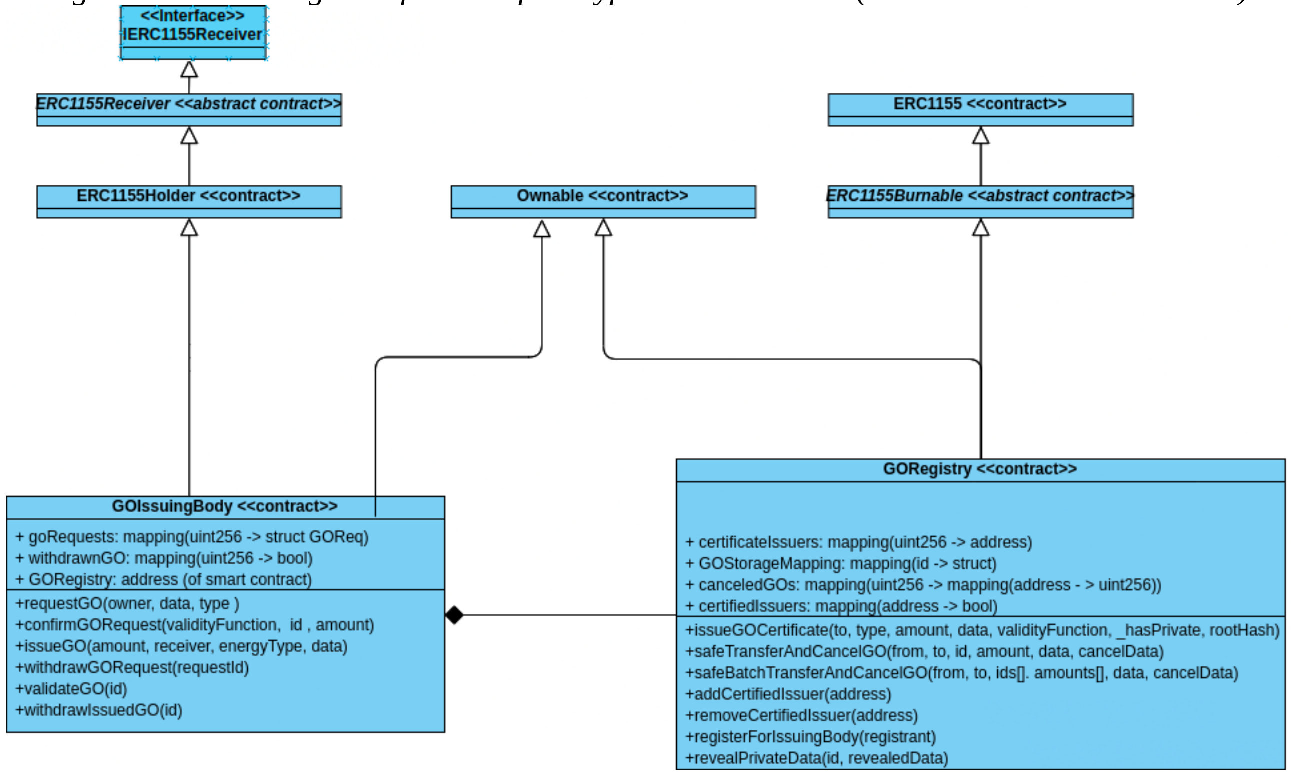
The ERC-1155 is a multi-token standard interface that represents the combination of fungible, non-fungible and semi-fungible tokens [127]. ERC-1155 is implemented by a single contract that stores all the created tokens in its smart contract state. It contains both the aspects of the ERC-20 token, meaning that the specific amount of tokens can be transferred to another party, as well as the non-fungible part that can be used to store GO production data. Energy Web Foundation project Origin used an extended version of the ERC-1155 multi-token standard, with the extension to ERC-1188 standard proposed by Energy Web Foundation token, and they also provide a good rationale about why using the ERC-1155 standard is optimal [128].

In terms of representing GO as an ERC-1155 token, when a certificate is created as this type of token, it is linked to both the fungible amount aspect represented in the MWh value, and the non-fungible part representing the specific certificate data. Therefore, the fungible part can be traded in any amount less or equal to the production value, but upon transfer it is still linked to the non-fungible part representing the production data. Furthermore, OpenZeppelin Contracts provides an extension of the ERC-1155 contract, namely the ERC-1155 Burnable contract that provides methods for burning tokens. The burning destroys the tokens so that they can no longer be transferred, which is a useful feature for cancelling the GOs.

5.1.3 Prototype Architecture Overview

The GO blockchain prototype is composed of the two main smart contracts and additional OpenZeppelin library smart contracts. The two main smart contracts are *GOIssuingBody.sol* and *GORegistry.sol.* The *GOIssuingBody* smart contract is deployed by an Issuing Body, which becomes the owner of the contract. The ownership of the contract logic is enabled by the OpenZeppelin *Ownable.sol* contract. The main feature of the ownership contract is that it provides a way to allow calling of certain functions of the smart contract only by the owner of the contract, which in this case would be the Issuing Body that deployed it. *GOIssuingBody* contract contains all the logic that is under the jurisdiction of the single Issuing Body for Guarantees of Origin. To point a few, it contains the functions and state variables related to requests for issuing GO certificates, it can confirm and issue a GO certificate, it can withdraw existing GO certificates, and it performs validation of the GO certificates. Issuing Body has a composition relationship with the *GORegistry.sol* contract. This relationship is initialized in the constructor of the *GOIssuingBody* smart contract. Therefore, before deploying *GOIssuingBody* smart contract, a *GORegistry* smart contract is deployed, and its address is used to instantiate the *GOIssuingBody* smart contract. The *GORegistry* smart contract contains the logic for representing the GO certificate as an ERC-1155 token, and it also stores the GO certificate data. Furthermore, it specifies the certified Issuing Bodies within this registry that have the privileges to issue GO certificates. *GORegistry* smart contract contains the functions for the creation of the GO certificate token, certificate token transfers, as well as cancellation of the GO certificate. Figure 10 presents a class diagram of the *GOIssuingBody* and *GORegistry* smart contracts as well as their relationship, and the smart contracts they are inherited from. Some of the utility and private functions are omitted in the class diagram for improved and immaculate preview.

Figure 10: Class diagram of the GO prototype smart contracts (Source: Author's elaboration)



<<Interface>>
IERC1155Receiver
ERC1155Receiver <<abstract contract>>
ERC1155 <<contract>>
ERC1155Holder <<contract>>
Ownable <<contract>>
ERC1155Burnable <<abstract contract>>
GOIssuingBody <<contract>>
+ goRequests: mapping(uint256 -> struct GOReq)
+ withdrawnGO: mapping(uint256 -> bool)
+ GORegistry: address (of smart contract)
+requestGO(owner, data, type)
+confirmGORequest(validityFunction, id , amount)
+issueGO(amount, receiver, energyType, data)
+withdrawGORequest(requestId)
+validateGO(id)
+withdrawIssuedGO(id)
GORegistry <<contract>>
+ certificateIssuers: mapping(uint256 -> address)
+ GOStorageMapping: mapping(id -> struct)
+ canceledGOs: mapping(uint256 -> mapping(address - > uint256))
+ certifiedIssuers: mapping(address -> bool)
+issueGOCertificate(to, type, amount, data, validityFunction, _hasPrivate, rootHash)
+safeTransferAndCancelGO(from, to, id, amount, data, cancelData)
+safeBatchTransferAndCancelGO(from, to, ids[]. amounts[], data, cancelData)
+addCertifiedIssuer(address)
+removeCertifiedIssuer(address)
+registerForIssuingBody(registrant)
+revealPrivateData(id, revealedData)

5.1.4 Implementations of the GO Use Cases

This section provides an explanation and in-depth analysis of how the use cases for Guarantees of Origin mentioned earlier were implemented in the prototype. For clarity purposes, the functions of the smart contract are represented using pseudo-code, instead of the Solidity code. The full smart contracts code in Solidity is provided in the Appendices.

As mentioned earlier, for a trustworthy GO tracking scheme operation, it is mandatory to perform a few steps before issuing or transferring the GO certificates. A prerequisite of including a production device in the trading scheme is for the device to be registered and audited. The audit of the production devices is also performed occasionally after the initial audit. Furthermore, the accounts for the producers and traders need to be established before they are able to request and transfer GO certificates. This prototype implementation provides a simplified on-chain approach for regulating the registration of the producer in the GO tracking scheme.

The initial step after the deployment of *GOIssuingBody* and *GORegistry* smart contracts is for the owner of the *GORegistry* contract to add the address of a deployed instance of a *GOIssuingBody* smart contract as a certified issuer. After the owner of the *GORegistry* adds a *GOIssuingBody* contract as a certified issuer, it enables that *GOIssuingBody* smart contract to issue the GO certificates within that *GORegistry* contract. In this way, the owner of the *GORegistry* can manage the privileges of which Issuing Body can issue the certificates within that specific registry. For a certain account to be issued GO certificates, it needs to be registered to a specific Issuing Body. One account can only be registered to one Issuing Body at a specific point in time. This action is performed by *GOIssuingBody* smart contract calling the function in the *GORegistry* contract, and sending the address of the account it wishes to register. These functions can be seen below in the form of pseudo-code. The owner of the Issuing Body initiates the call to the *GOIssuingBody* smart contract function.

```
//GOIssuingBody.sol
function  addUserToIBinRegistry(accountAddress, issuingBodySender)
        call GORegistry instance’s function issuingBodyMapping(accountAddress)
```

Then the *GOIssuingBody* smart contract executes the call to the *GORegistry* contract to the function that is described below.

```
//GORegistry.sol
function  registerForIssuingBody(accountAddress, issuingBodySender)
        issuingBodyMapping[issuingBodySender] ← store accountAddress to state
```

Upon the execution of this transaction on the blockchain, the account owner has been registered to belong to an Issuing Body and it can now request the issuing of the GO certificate. The approach for registering users to the Issuing Body and the Registry described above is simplistic, and the proper user and role management would probably require developing a separate off-chain software service, as EW Origin project did.

5.1.4.1 Issue GO

The process of issuing the GO is initiated with the request of the production device owner, which calls the smart contract function for requesting GO. The function for requesting GO is presented below.

```
//GOIssuingBody.sol
function  requestGO( ownerAddress, data, energyType)
        create GORequest object ← ownerAddress, data, energyType
        GORequestMapping[issuingBodySender] ← store GORequest object to mapping
        emit GORequested event ← with GOOwner and id data
```

As can be observed from the pseudo-code snippet above, the production device owner will send a request to a *GOIssuingBody* smart contract specifying the owner address, the GO certificate encoded data, as well as the energy type produced by the production device. The energy type is sent as an integer for performance purposes and it is based on the predefined enumerable object, where e.g. 0 – represents solar plant power, 1 – represents wind turbine power, 2 – represents hydropower. The smart contract will initialize a new GO request object and store it in the smart contract state, in the mapping array data structure. Mappings are key-value pair based data structures that resemble hash tables, and they are initialized in the way that every possible key is mapped to a value with byte-representation of all zeroes, which is the default value of a mapping key entry [122]. Due to that, mappings do not know the concept of length and can not be iterated by indices. The only way to access a value of the mapping is to know the specific key. However, they provide very fast execution times in terms of reads and writes of its entries, and are much more efficient than using an array. The last action to be executed is the emitting of the *GORequested* event. Events are special data types leveraged on EVM logging facilities, and when the event is emitted, it stores the logs provided as arguments on the blockchain. Therefore, the usability of events is twofold, firstly as logs of the transactions on the blockchain, and secondly as a way for client applications to subscribe to certain events and then perform actions based on that event.

The Issuing Body can preview all the requests for issuing the GO certificates, and if they deem the request valid they can issue the GO certificate. This is done by calling the *confirmGORequest* function or the *issueGO* function if the request has not yet been made. The function for issuing GO can be previewed in the pseudo-code snippet below.

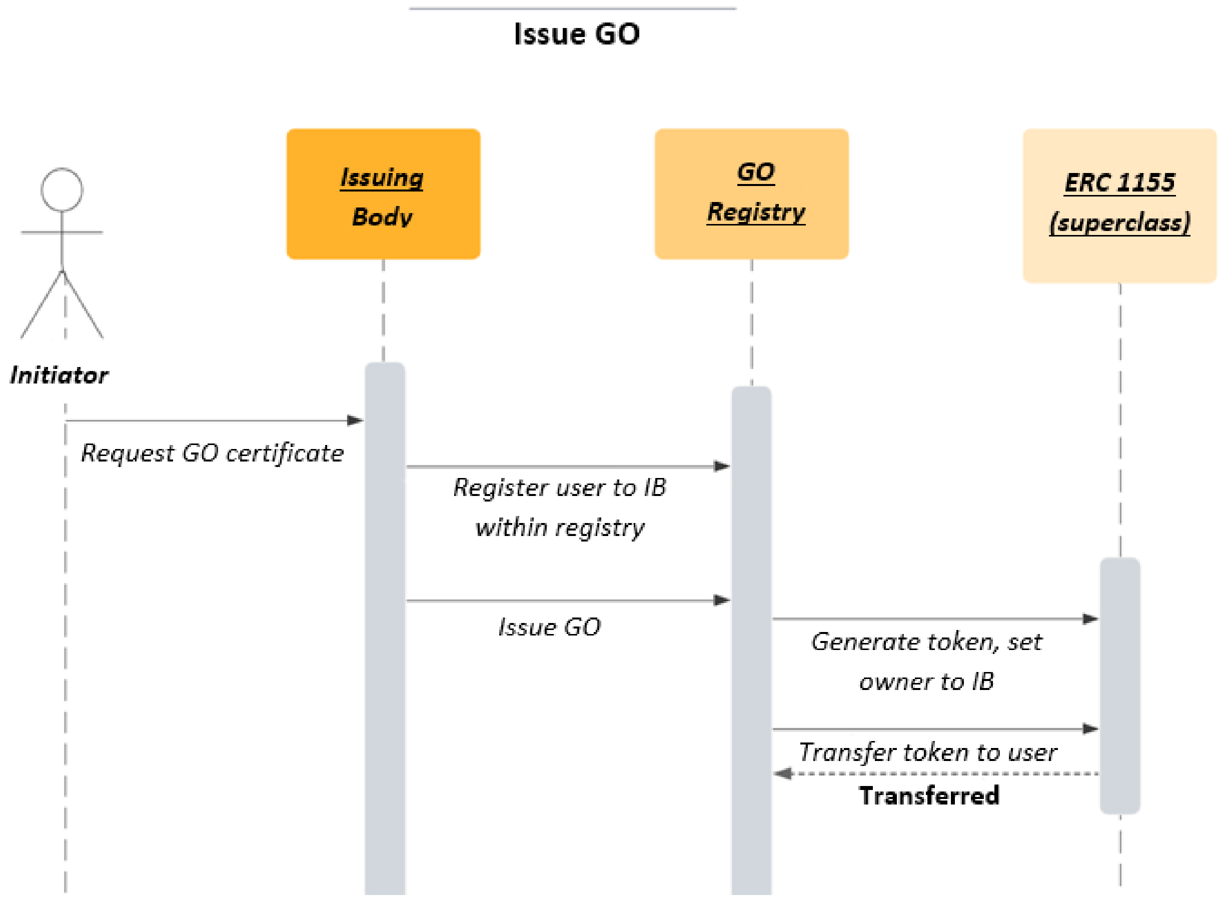
```
//GOIssuingBody.sol
function  issueGO( amount, GO_Receiver, energyType, GO_data)
        create GORequest object ← ownerAddress, data, energyType
        GORequestMapping[issuingBodySender] ← store GORequest object to mapping
        if GORequest already confirmed or withdrawn
                revert – this step reverts the state to the one before transaction
        set GORequest.confirmed ← true
        set validityFunction ← reference to current Issuing Body function for checking validity
                               of the GO certificate token
        call GORegistry instance’s function issueGOCertificate(GO_Receiver, energyType,
                                                    amount, data, validityFunction)
```

As can be seen from the snippet above, the *GORequestObject* is created, after which the conditional check is performed to see if the request has been withdrawn or already confirmed. If that is the case the transaction is reverted to the previous state of the smart contract, that existed before the function was called. If not, the request status is set to confirmed, and the validity function is initialized to the one defined by the *GOIssuingBody* smart contract. This validity function will be encoded in the created GO certificate token, and upon each transaction of the token, the validity function will be called. This is done because blockchain is immutable so there needs to be a predetermined way for Issuing Bodies to manage token validity giving them the option to revoke invalid tokens. It may happen that an Issuing Body has performed an audit of a production device and realized that certain production parameters have not been satisfied, or that the GO certificate has expired. Based on that, it can call the withdraw function on the blockchain which will invalidate the existing GO certificate. The last statement is the call of the *GORegistry* function for issuing the GO certificate token. The pseudo-code of that function is available below.

```
//GORegistry.sol
function  issueGOCertificate( to, energyType, amount, data, validationFunction)
        if call validationFunction – calls the IssuingBody validation function
                        fails ← revert the transaction
        else
                proceed
        tokenId ← generateToken(messageSender, amount, data) - calls the ERC1155 mint
                function that creates token and sets the issuer to the sending Issuing Body
        if amount > 0
                transfer token amount to the owner (if amount is 0, there is nothing to transfer)
        GOTokenMapping[tokenID] ← create and store GOObject to the storage mapping
        emit OneGOIssuedEvent
```

As can be seen in the snippet above, the validation function that resides in the *GOIssuingBody* contract is called to check if the GO certificate request has been withdrawn by the issuer or due to invalid data by the IssuingBody. If the validation step has been passed, the GO certificate token is then minted. Minting is the process of increasing the total supply of the token, and it is a common term in the ERC-20 token community. The creator of the token is set to be the Issuing Body. This is important since only the IssuingBody can mint additional tokens. Issuing Body can mint additional tokens if it decides that the initial supply did not correspond to the real production value. After the creation of the token, the corresponding token amount is then transferred to the owner of the production device. The owner can then trade these tokens with other parties. Figure 11 shows a sequence diagram of how the GO issuing process is executed in the prototype.

Figure 11: Issue GO certificate sequence diagram (Source: Author's elaboration)



5.1.4.2 Transfer GO

After the GO certificate tokens have been issued, they can also be transferred by the accounts that own them. However, since accounts may be stored in different Issuing Bodies, it is questionable whether a certain Issuing Body wants to trade with another Issuing Body. In the current implementation, all the transactions occur through the AIB Hub system, and there may be an additional step involved in migrating the accounts between the two independent registry system.

This blockchain implementation provides a single token registry through which all the transactions occur between multiple Issuing Bodies. The reason for using a single *GORegistry* contract is that it would provide a central standardized interface for defining all the logic behind issuing, transferring and cancelling tokens, in which Issuing Bodies maintain the privilege of withdrawing the invalid certificates. Additionally, this prototype implementation specifies rules for which Issuing

Bodies trade among each other in the *GORegistry* contract, as well as in which Issuing Body is a certain user registered. This represents another reason why it was more convenient to have a single instance of *GORegistry* contract deployed. In an ideal scenario, there would be a user registration module developed off-chain and therefore smart contracts would not need to contain any logic regarding the account management.

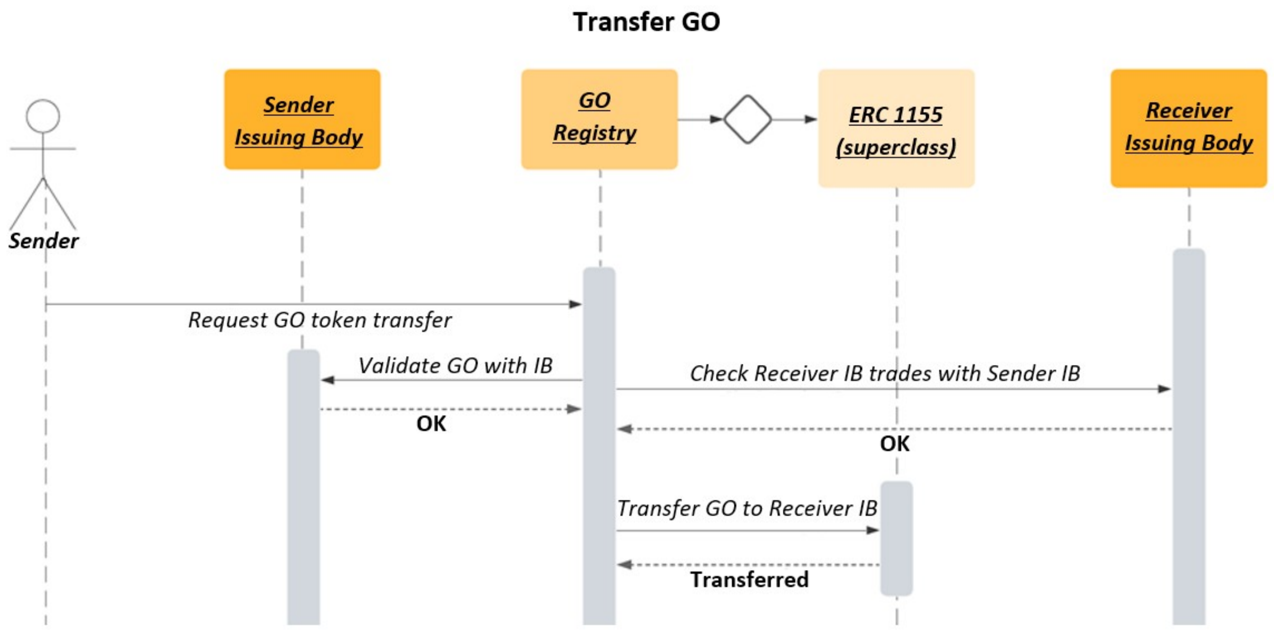
Having *GORegistry* contracts deployed in multiple instances would also require further effort in tracking the addresses of all the separate *GORegistry* contract addresses. However, if the need occurs, there may be multiple *GORegistry* smart contracts deployed, the implementation would not involve any major changes. Therefore, in the prototype, there is no need for account export or import, since they exist as addresses on a single blockchain. Since certain Issuing Bodies may not want to trade with one another, there must be a way for an *GOIssuingBody* contract owner to specify which Issuing Body it trades with. In this implementation, this is done in a convenient manner such that each *GOIssuingBody* smart contract stores a mapping of the address of other *GOIssuingBody* smart contracts it trades with.

When a transfer is initiated in the *GORegistry* smart contract, a function is called on the side of the receiving *GOIssuingBody*. If the receiving *GOIssuingBody* has specified that it trades with the sending *GOIssuingBody* the transaction will go through. If not, the transaction will fail. Therefore, before a GO token transfer occurs, the receiving *GOIssuingBody* smart contract must add the address of the sending *GOIssuingBody* smart contract. The transfer function is an ERC-1155 *safeTransferFrom* function that performs a safety check to ensure that a receiving side is aware of the ERC-1155 tokens so that they would not be locked in a non-aware smart contract. The summarized logic of the function in the form of pseudocode is provided below.

```
//GORegistry.sol
function  safeTransferFrom( from, to, tokenId, amount, certificateData)
        if sender not token owner– caller must own the tokens they trade with
                fails ← revert the transaction
        if receivingIssuingBody not trades with sendingIssuingBody
                fails ← revert the transaction
        if call validationFunction – calls the IssuingBody validation function
                fails ← revert the transaction
        receiverBalance ← receiverBalance + amount
        senderBalance ← senderBalance – amount
        emit Transfer event
```

As can be seen from the provided snippet, the sender will initiate the token transfer providing the receiver, id of the token, amount of GO to be transferred, and transfer data. The sender must be the owner of the tokens and their issuing bodies must be trading with one another. When the transaction has finished, the event is emitted, and the ERC-1155 transfer transaction is logged on the blockchain. Figure 12 provides a sequence diagram of the GO certificate token transfer.

Figure 12: Transfer GO sequence diagram (Source: Author's elaboration)



It is important to note that this prototype does not implement the market system for trading GOs, creating orders, and establishing the matching mechanism between the buyer and the seller. Usually, this would require a separate off-chain service, where the buyer and seller would be matched, and the transfer price would be established, after which the transfer process would occur on the blockchain. As mentioned earlier, the EW Origin project implements a separate module for managing the trading aspect. This module is responsible for order creation, matching services, and initiating the token transfer [129].

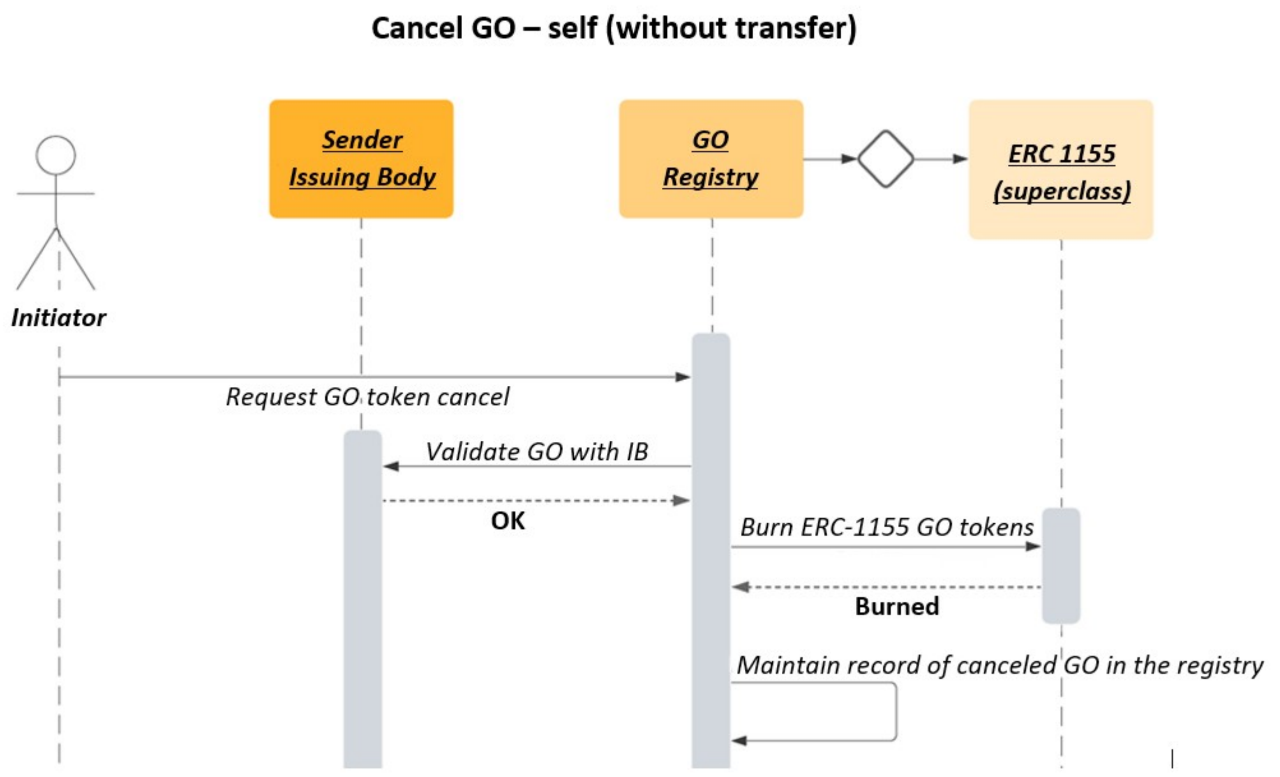
5.1.4.3 *Cancel* GO

The cancellation of the GO requires that a certain amount of the token is destroyed and can no longer be transferred and used. This renders the GO token as successfully used for energy disclosure or other purposes. To cancel a certain amount of GOs, a cancellation function on the *GORegistry* smart contract is called. This function also enables the certificate tokens to be transferred to other recipient and then have them cancelled. The cancellation process logic is provided in the pseudo-code snippet below.

```
//GORegistry.sol
function  safeTransferAndCancelGO( sender, receiver, tokenId, amount, certificateData,
                                        cancellationData)
        if call validationFunction – calls the IssuingBody validation function
                        fails ← revert the transaction
        if sender not token owner– caller must own the tokens they wish to cancel
                        fails ← revert the transaction
        if sender not equals receiver
                        call safeTransferFrom function

        canceledGOs[tokenID][receiver] ← increment by the amount of canceled tokens
        emit OneGOCanceledEvent ← with cancellation data and certificate data
```

The caller of this smart contract function will request the cancellation, provide the amount of tokens to be cancelled, the token id of the certificate, and any additional certificate and cancellation data. The sender must be the owner or have approval for the tokens. If the sender address is different from the receiver address, the transfer function will be invoked, and if all the conditions are valid the tokens will be transferred to the receiver. Before the cancellation occurs, a validation check is performed. The smart contract also supports performing a batch cancellation by providing an array of token ids and an array of amounts, ordered sequentially in which the first token id array entry corresponds to the first amount array entry. Figure 13 shows a sequence diagram of the GO cancellation process.

Figure 13: Cancel GO sequence diagram (Source: Author's elaboration)



5.1.4.4 Issuing GO with Private Data

The prototype enables issuing GO certificates with private data, using the JavaScript library for Precise Proofs, which was implemented as part of the EWF Origin solution, described earlier. It does not however enable keeping the amount of the certificates private, only making additional GO data confidential. The issuing process is similar to the non-private one, with the exception that the Merkle root hash is provided. The Merkle root hash is generated from the private data off-chain using the precise-proof JavaScript library. This Merkle root is stored as a way to verify the private data. When a proving party is required to reveal the private data related to a certain GO certificate, it can publish the data on the blockchain, by calling the *revealPrivateData* function in the *GORegistry* smart contract. The verifying party can then obtain the revealed private data, generate a Merkle tree root hash, and compare it with the original Merkle root hash that was initially published on the blockchain. If the root hashes are equal the data is verified, if not, the data does not correspond to the original one, and it means that the proving party did not provide the correct data.

EW Origin implements Precise Proofs for issuing private GO certificates. The amount and production device data remain hidden, they are stored as private commitments on the blockchain which can be transferred. However, the confidential part must be publicly revealed before the GO token certificate is cancelled. They also mention a few limitations of this approach, such as that the document schema needs to be predetermined and that there needs to exist a degree of trust between the participants. Furthermore, the solution requires interaction, unlike the Zero-Knowledge verification methods. Lastly, it requires the development of an off-chain verifying tool and off-chain data storage [119].

5.2 Discussion

This thesis introduced the analysis of the utilization of blockchain technology for the Guarantees of Origin tracking instrument. The prototype implementation described in the previous section explores the technology available for implementing the GO system using the blockchain. The technology stack based on the Ethereum blockchain technology can be evaluated as quite sufficient and robust technology for such a purpose. Furthermore, the Truffle framework streamlines main development processes by enabling easy deployment, simple installation process, detailed documentation, automated tests, and easy integration with existing front end frameworks. Additionally, the availability of the OpenZeppelin Contracts library removes a lot of repetitive work, and provides community audited and modular smart contract code based on the best industry practices. Apart from that, the Ethereum community is quite large, active, and growing. Therefore, there are a lot of resources and discussions, which aid in the development process. On the other hand, Solidity being quite a new programming language has some major disadvantages. One of the biggest problems that are not mainly the issue of Solidity but rather blockchain in general, is the fact that once the smart contract is deployed on the network, it becomes part of an immutable ledger. Therefore, it can not be upgraded as easily as a centralized software application would be. That is why smart contract programming requires extensive security testing before the production version of a smart contract is deployed to the network. Concerning that, there are certain development patterns and libraries that enable upgradeable contracts such as the OpenZeppelin contracts library [130]. Furthermore, since it is quite a young language, there have been issues with anti-patterns in Solidity that resulted in a 55 million dollar heist due to a Solidity bug [131]. However, Solidity as a language has matured since then, and a lot of these bugs have been resolved. Furthermore, the transition from Ethereum Virtual Machine to Ethereum Web Assembly (EWASM) will enable the utilization of more traditional and general-purpose programming languages such as C, C++ and Rust.

The prototype implementation described in the previous section only analyzes the aspects of the GO scheme related to the issuing and tracing of the GO certificates on the blockchain network.

However, it does not provide a complete solution that handles all of the aspects that the GO system requires. It does not implement the system for account management and registration, the market for trading GOs, and user interface for previewing the GO issuing, transferring, and cancelling. For a complete and robust solution, these off-chain services would also have to be implemented to account for all of the use cases required by the GO system. EW Origin is a good example of such a complete system. The account management module would be necessary to enable the Issuing Bodies to have control over registering the production devices and their owners in the system, as well as performing a preregistration audit and a periodical production audit. The market system would also be necessary to establish a platform for buyers and sellers, as well as an order matching system.

One of the advantages of using blockchain is that it provides an immutable and entirely traceable audit log for all the issued Guarantees of Origin certificates. The traceability indicates that every transaction is stored on the blockchain, therefore the history of every transaction related to a Guarantee of Origin certificate can be previewed. Furthermore, it decentralizes the ownership of the certificates giving users more freedom to perform transfers, without depending only on a single system or a single registry. Of course, the regulatory power of the Issuing Bodies can be retained, so that they can invalidate any faulty certificates. Since blockchain is inherently immutable, it can be argued that the blockchain network is more secure. Possible attack vectors would include having a malicious node majority, most commonly 51% of all nodes, or exploiting the design vulnerabilities in the programming language of smart contract or security flaws not addressed by the developers. Lastly, blockchain technology would enable easier integration with other applications and compatibility with other Energy Attribute Certificate systems. That would mean that there could be multiple market platforms for Guarantees of Origin certificates that could easily be integrated with the blockchain technology, decentralizing the trading aspect of the scheme. Furthermore, the coordination of the transfers between different certificate standards would be much easier, if these certificates schemes were using blockchain issued certificates [132].

On the other hand, the most noted issue in the implementation of the prototype was enabling privacy in terms of certificate issuing and transferring. Certain privacy enabling solutions do exist, but there is no perfect option. And even though there are solutions that enable privacy such as Zero-Knowledge Proofs or Precise Proofs utilized by the EW for the Origin system, they also come with certain drawbacks. Namely, few of these solutions are production-ready and each establishes certain premises to the privacy framework, such as a trusted setup and mechanisms for off-chain proofs and storage mechanisms. The solution utilized by EW Origin enables issuing and trading of the private EACs on the blockchain. However, a private certificate can not be cancelled and each private certificate needs to be converted to a public one to perform cancellation. Therefore, confidential information will have to be revealed at some point.

An important question to consider is whether to use public or permissioned blockchain platforms for the Guarantees of Origin system. On one hand, public blockchain platforms are decentralized, so no one is in control of the network. In addition to that, the network is accessible by anyone. Public blockchains introduce the issue of low throughput and transaction pricing which is generally high. This issue is caused usually due to using a computationally intensive PoW consensus algorithm. For example, it would be quite inefficient to use Ethereum mainnet to support the system for tracking Guarantees of Origin, as currently it supports around 15-30 transactions per second. This issue is being solved by a lot of blockchain technologies transitioning to different consensus algorithms, and improving the blockchain architecture in an effort to make it more scalable. The upgrade to Ethereum 2.0 protocol is presumed to extend the support to around 10 000 transactions per second [133]. On the other hand, permissioned blockchains come with certain nodes being in control of the entire network, lacking the decentralization aspect. Furthermore, some implementations like Quorum limit access only to known nodes. Permissioned blockchains are generally faster and transaction pricing does not exist. However, the issue at hand is defining which nodes will be in control of the network. Permissioned blockchains are considered a more vulnerable option in terms of security. Namely, it might be the case that a flawed security environment of just one node could lead to leaks of confidential data and Denial of Service attacks [134]. The Energy Web Foundation found a solution in creating a public blockchain accessible by anyone, in which the control and validation of the blockchain are in the hands of validator nodes, using the Aurora Proof-of-Authority consensus protocol. As mentioned earlier, the validator nodes are the affiliates of the Energy Web Foundation, and these are usually energy sector companies or software companies involved in the development process. Since these companies have a high stake in the business processes being performed legitimately, it is assumed that there is no motivation for them to collaboratively do malicious actions on the network. Additionally, the higher number of validator nodes also lowers the possibility of the blockchain platform being compromised. Table 2 shows the comparison of the current GO system and the GO system on the Ethereum blockchain-based on selected criteria.

Table 2: Current GO system and blockchain GO system comparison

	Current GO system	GO system on Ethereum blockchain
Transaction speed	Few dozens of seconds, intercommunication between the registry services and AIB Hub	Transaction speed depends on the blockchain technology, in the range of 15-100 000 TPS.
Privacy	The participants are not aware of the transactions, that other Issuing Bodies are involved in.	Blockchain is inherently public so all of the transactions are public. Some semi-privacy solutions exist, few or none are production-ready. Might require a few more years for a robust and well-adopted privacy solution.
Statistics	Only through AIB Hub, volumes exchanged between registries are public, can not see which Issuing Bodies are trading, that information is private	Visible on blockchain explorer, all transactions public, recipient and sender known, unless privacy solutions are used.
Public or private	The registry systems are known to each other.	Depends on the technology used, each account on the blockchain is identified by the address, however even though the address is unknown it does not guarantee full confidentiality. Generally, all the participants in the network are publicly known or can be traced to an identity.

Source: Author's elaboration

In an interview with the representatives of the European Energy Exchange, they mention that blockchain technology enables securely tracing of which party purchased which certificate, however it does not answer the question of whether the product purchased in the blockchain is pertaining to an equivalent real-world amount [55]. That is why fully decentralized solutions are not possible for the Guarantees of Origin system. The blockchain system for Guarantees of Origin would need to be developed in coordination with the current systems. It would also require the development of separate off-chain modules responsible for managing the registration of the production devices and production audits.

The Guarantees of Origin system requires the involvement of regulating bodies, rendering full decentralization of the system hardly achievable. EW Origin project takes this into account and provides a separate registration module for managing organizations and users. Each organization is mapped to a specific entity, in which each organization defines users with different permissions. If the system was to be integrated into the registry system, each organization would have to certify that they are active members of the system. If the organization in the registry stops being active they would no longer be able to interact with the system. Furthermore, all the production devices that are already registered in the registry would have to be imported into the Origin system. This is quite a positive characteristic of the Origin solution, since it does not try to optimistically replace the existing system, rather it aims at coordinating their blockchain-based system with the existing systems for issuing EACs. The Origin system enables everyone to access their marketplace, to preview registered device, and the market supply and demand. However, to interact with the system, a user must have a registered account [135].

Another big criticism on utilizing blockchain for the renewable energy sector is the contradicting fact that blockchains are incurring huge hardware and energy costs, e.g. Ethereum reached an all-time-high electricity consumption in 2021 to 32 TWh [136]. However, it is also important to note that Ethereum is aiming to gradually have a complete transition to a Proof-of-Stake consensus protocol which would reduce the energy costs by 99% [137]. On the other hand, Energy Web Chain utilizes the Proof-of-Authority consensus protocol, and it is estimated that the average demand of a validator node is 78 Watts, equivalent to a normal household bulb [67].

The coordination of the proposed blockchain alternatives with the existing systems is a crucial requirement in an effort to include blockchain technology. Especially, since current systems are supported by governments, regulators and stakeholders, as well as the fact that they are quite well-performing and have been extensively used over the years, so there is no need to disrupt the existing system with overly optimistic replacement technologies [50]. The Guarantee of Origin trading parties utilize existing trading platforms and registry systems. Blockchain technology should seek its role in the GO environment, as potentially integrating with the existing systems, since replacing current solutions would not be a sensible approach from both a financial and a technical perspective.

6 Conclusion

Blockchain has been widely promoted as a technology that is aiming to introduce decentralization to the system of Energy Attribute certificates, improve trading efficiency and enable better inclusion of the market participants. This thesis provides an overview of the advantages and disadvantages as well as a practical demonstration of the aspects in which blockchain technology has the potential to add value to the existing Guarantees of Origin systems. A description of the current Guarantees of Origin registry systems and their intercommunication via the AIB Hub is explained. Major points and concepts of blockchain technology and its potential usage in the GO system are elaborated. Existing solutions and related work using blockchain technology for Energy Attribute Certificates and other similar schemes are analyzed. Most importantly, the thesis documents the implementation of the prototype for issuing Guarantees of Origin using the blockchain technology, based on Ethereum blockchain technology and Solidity smart contract programming language.

The implementation of the prototype demonstrates that blockchain technology can be used for the Guarantee Of Origin system, potentially decentralizing certificate markets. Utilizing blockchain technology will enable transparency of the certificate transactions and the immutability of the certificate records. Compared to the existing system, certain aspects of blockchain technology need to be improved such as privacy features, as well as transaction throughput and scalability. However, there are a lot of promising projects, such as Ethereum sharding and Polkadot multi-chain technology, that might solve the low transaction throughput and poor scalability of the blockchain network.

Nevertheless, any blockchain solution for the Guarantees of Origin tracking instrument would need to be developed and integrated in coordination with the existing systems. Blockchain technology is not a standalone solution. It only represents an additional layer that can be used for logging the certificate information and all the transactions related to it on an immutable ledger. Other aspects such as certificate market and user management, would require integration of the blockchain solution with other off-chain solutions. Blockchain technology can add value to the existing Guarantees of Origin systems if realistic goals are set as well as if the accompanying solutions are developed in accordance with the regulations and standards defined for the Guarantees of Origin tracking instrument. Future work and research may include investigating the usage of emerging paradigms in blockchain technology, such as multi-chains and newly developed privacy solutions.

7 Bibliography

[1] B. Radowitz, “Demand for renewable power with guarantee of origin surges despite Covid | Recharge,” *Recharge | Latest renewable energy news*, Oct. 08, 2020. https://www.rechargenews.com/transition/demand-for-renewable-power-with-guarantee-of-origin-surges-despite-covid/2-1-890164 (accessed May 10, 2021).

[2] RECS International, “The blockchain and energy attribute tracking.” Sep. 2019. Accessed: May 10, 2021. [Online]. Available: https://recs.org/download/?file=Blockchain-paper_FINAL.pdf&file_type=documents&file_type=documents

[3] Natural Capital Partners, “Energy Attribute Certificates,” London. Accessed: Mar. 24, 2021. [Online]. Available: https://assets.naturalcapitalpartners.com/downloads/Energy_Attribute_Certificate_Factsheet.pdf

[4] “Energy Attribute Certificates (EACs).” https://energy-attribute-certificates.com/ (accessed Mar. 23, 2021).

[5] Bischoff & Ditze Energy, *English: The follwing map gives an overview of the different systems worldwide. Please note: some countries don’t have a national system in place yet, therefore sometimes one or more “external” EACs systems can apply. Furhermore, countries which are not colour coded can fall under the regime of I-RECS, NECS and other.* 2020. Accessed: May 12, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:EACs_World_map_02_2020_Bischoff_%26_Ditze_Energy.png

[6] “Directive (EU) 2018/2001 of the European Parliament and of the Council of 11 December 2018 on the promotion of the use of energy from renewable sources,” *PE/48/2018/REV/1*, pp. 82–209, Dec. 2018.

[7] “Directive 2009/28/EC of the European Parliament and of the Council of 23 April 2009 on the promotion of the use of energy from renewable sources and amending and subsequently repealing Directives 2001/77/EC and 2003/30/EC,” *OJ L 140*, pp. 16–62.

[8] “Renewable Energy Guarantees of Origin | AIB,” *AIB*. https://www.aib-net.org/certification/certificates-supported/renewable-energy-guarantees-origin (accessed Mar. 24, 2021).

[9] “How to document renewable electricity consumption in Europe?,” *ECOHZ*. https://www.ecohz.com/renewable-energy-solutions/guarantees-of-origin/ (accessed Mar. 25, 2021).

[10] “EECS Rules.” The Association of Issuing Bodies, Mar. 01, 2021. Accessed: Mar. 25, 2021. [Online]. Available: https://www.aib-net.org/sites/default/files/assets/eecs/EECS%20Rules%20Release%207%20v14.pdf

[11] Jaap Jansen, Eleanor Drabik and Christian Egenhofer1, “The Disclosure of Guarantees of Origin: Interactions with the 2030 Climate and Energy Framework,” CEPS special report 149, Nov. 2016. Accessed: Mar. 25, 2021. [Online]. Available: https://core.ac.uk/download/pdf/76839029.pdf

[12] “CEN - EN 16325 - Guarantees of Origin related to energy - Guarantees of Origin for Electricity | Engineering360.” https://standards.globalspec.com/std/9969735/EN%2016325 (accessed May 07, 2021).

[13] “AIB Hub | AIB.” https://www.aib-net.org/facts/eecs-registries/aib-hub (accessed Mar. 25, 2021).

[14] “Establishing an access to the Registry of Guarantees of Origin,” *OTE, a.s.* https://www.ote-cr.cz/en/poze-en/guarantees-of-origin/establishing-an-access-to-the-registry-of-guarantees-of-origin (accessed Mar. 25, 2021).

[15] “EECS Electricity Domain Protocol for Czech Republic.” OTE, a.s., Aug. 2018. Accessed: Mar. 26, 2021. [Online]. Available: https://www.ote-cr.cz/en/poze-en/guarantees-of-origin/files/domain-protocol-ote.pdf

[16] “EECS RULES Subsidiary Document AIB-EECS-SD03: EECS Registration Databases, version 7.5.” Association of Issuing Bodies, Jul. 17, 2020. [Online]. Available: https://www.aib-net.org/sites/default/files/assets/eecs/subsidiary-documents/AIB-EECS-SD03%20HubCom %20EECS%20Registration%20Databases%20-%20Release%207.5_0.pdf

[17] T. Lindberg, “Blockchain and Guarantees of Origin - recommendations REDII Article 19.8,” *ECOHZ*, May 23, 2018. https://www.ecohz.com/news/blockchain-and-guarantees-of-origin-recommendations-redii-article-19-8/ (accessed Mar. 30, 2021).

[18] D. Van Evercooren, “Will Blockchain replace the guarantee of origin? Or improve it?,” *Linkedin*, Feb. 27, 2019. https://www.linkedin.com/pulse/blockchain-replace-guarantee-origin-improve-dirk-van-evercooren/ (accessed Mar. 30, 2021).

[19] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” p. 9, 2008.

[20] O’Reilly Media, Inc., *Swan M. Blockchain: Blueprint for a New Economy2015*.

[21] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” in *2017 IEEE International Congress on Big Data (BigData Congress)*, Honolulu, HI, USA, Jun. 2017, pp. 557–564. doi: 10.1109/BigDataCongress.2017.85.

[22] N. Houy, “The Bitcoin Mining Game,” *Ledger*, vol. 1, pp. 53–68, Dec. 2016, doi: 10.5195/LEDGER.2016.13.

[23] J. Frankenfield, “Double-Spending,” *Investopedia*. https://www.investopedia.com/terms/d/doublespending.asp (accessed Apr. 01, 2021).

[24] “What Is Blockchain Technology? How Does It Work? | Built In.” https://builtin.com/blockchain (accessed Apr. 01, 2021).

[25] K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016, doi: 10.1109/ACCESS.2016.2566339.

[26] M. Wander, *English: Graphic of data fields in Bitcoin block chain.* 2013. Accessed: Apr. 01, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png

[27] Kes47, *English: Graphical comparison of centralized (A) and decentralized (B) system.* 2014. Accessed: Apr. 02, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Decentralization_diagram.svg

[28] L. M. Bach, B. Mihaljevic, and M. Zagar, “Comparative analysis of blockchain consensus algorithms,” in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, May 2018, pp. 1545–1550. doi: 10.23919/MIPRO.2018.8400278.

[29] D. R. Houben and A. Snyers, “Cryptocurrencies and blockchain,” p. 103.

[30] M. Vukolić, “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication,” in *Open Problems in Network Security*, vol. 9591, J. Camenisch and D. Kesdoğan, Eds. Cham: Springer International Publishing, 2016, pp. 112–125. doi: 10.1007/978-3-319-39028-4_9.

[31] B. Vitalik, “On Stake,” *Ethereum foundation blog*, Jul. 05, 2014. https://blog.ethereum.org/2014/07/05/stake/ (accessed Apr. 02, 2021).

[32] “Ethereum Could Turn On Proof of Stake Sooner Than We Anticipate,” *ConsenSys*. https://consensys.net/blog/ethereum-2-0/proof-of-stake-is-coming-to-ethereum-sooner-than-we-think/ (accessed Apr. 02, 2021).

[33] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, pp. 173–186, Feb. 1999.

[34] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, “PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain,” presented at the Italian Conference on Cyber Security (06/02/18), Jan. 2018. Accessed: Apr. 13, 2021. [Online]. Available: https://eprints.soton.ac.uk/415083/

[35] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, “BLOCKBENCH: A Framework for Analyzing Private Blockchains,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, New York, NY, USA, May 2017, pp. 1085–1100. doi: 10.1145/3035918.3064033.

[36] S. Zhang and J.-H. Lee, “Analysis of the main consensus protocols of blockchain,” *ICT Express*, vol. 6, no. 2, pp. 93–97, Jun. 2020, doi: 10.1016/j.icte.2019.08.001.

[37] W. Viriyasitavat and D. Hoonsopon, “Blockchain characteristics and consensus in modern business processes,” *Journal of Industrial Information Integration*, vol. 13, pp. 32–39, Mar. 2019, doi: 10.1016/j.jii.2018.07.004.

[38] M. D. Sheldon, “A Primer for Information Technology General Control Considerations on a Private and Permissioned Blockchain Audit,” *Current Issues in Auditing*, vol. 13, no. 1, pp. A15–A29, Mar. 2019, doi: 10.2308/ciia-52356.

[39] M. Liu, K. Wu, and J. J. Xu, “How Will Blockchain Technology Impact Auditing and Accounting: Permissionless versus Permissioned Blockchain,” *Current Issues in Auditing*, vol. 13, no. 2, pp. A19–A29, Sep. 2019, doi: 10.2308/ciia-52540.

[40] R. Emily, “What is the distinction between a blockchain and a distributed ledger?,” *Blockchain Byte*, p. 28.

[41] F. Hofmann, S. Wurster, E. Ron, and M. Bohmecke-Schwafert, “The immutability concept of blockchains and benefits of early standardization,” in *2017 ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K)*, Nanjing, Nov. 2017, pp. 1–8. doi: 10.23919/ITU-WT.2017.8247004.

[42] T. Jung, “How transparency through blockchain helps the cybersecurity community,” *Blockchain Pulse: IBM Blockchain Blog*, Apr. 15, 2019. https://www.ibm.com/blogs/blockchain/2019/04/how-transparency-through-blockchain-helps-the-cybersecurity-community/ (accessed Apr. 04, 2021).

[43] S. Psaila, “Blockchain: A game changer for audit processes? | Deloitte Malta | Audit & Assurance,” *Deloitte Malta*, Sep. 2017, Accessed: Apr. 04, 2021. [Online]. Available: https://www2.deloitte.com/mt/en/pages/audit/articles/mt-blockchain-a-game-changer-for-audit.html

[44] N. Szabo, “Formalizing and Securing Relationships on Public Networks,” *First Monday*, Sep. 1997, doi: 10.5210/fm.v2i9.548.

[45] V. Buterin, “Ethereum Whitepaper,” *ethereum.org*, 2013. https://ethereum.org (accessed Apr. 16, 2021).

[46] Z. Zheng *et al.*, “An overview on smart contracts: Challenges, advances and platforms,” *Future Generation Computer Systems*, vol. 105, pp. 475–491, Apr. 2020, doi: 10.1016/j.future.2019.12.019.

[47] M. Bartoletti and L. Pompianu, “An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns,” in *Financial Cryptography and Data Security*, Cham, 2017, pp. 494–509.

[48] GMBH draglet, *A simplified example of how a smart contract would work in an insurance environment.* 2017. Accessed: Apr. 16, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Smart_contracts_in_insurance_policies.png

[49] V. Spasic, "Record high demand for guarantees of origin in 2020 in EU," *Balkan Green Energy News*, Feb. 17, 2021. https://balkangreenenergynews.com/record-high-demand-for-guarantees-of-origin-in-2020-in-eu/ (accessed Apr. 04, 2021).

[50] T. Lindberg, "Blockchain and Energy Tracking," *ECOHZ*, May 24, 2018. https://www.ecohz.com/news/blockchain-and-energy-tracking/ (accessed Apr. 04, 2021).

[51] M. Andoni *et al.*, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 100, pp. 143–174, Feb. 2019, doi: 10.1016/j.rser.2018.10.014.

[52] "Will blockchain solve the problem of Renewables Guarantees of Origin?," *Everoze*, Mar. 2018. https://everoze.com/will-blockchain-solve-the-problem-of-renewables-guarantees-of-origin/ (accessed Apr. 05, 2021).

[53] D. Zimberg and J. Spinnell, *Renewable Energy Certificate Markets: Blockchain Applied*. 2018. doi: 10.13140/RG.2.2.26140.64643.

[54] A. Castellanos, D. Coll-Mayor, and A. Notholt, "Cryptocurrency as guarantees of origin: simulating a green certificate market with the ethereum blockchain," in *2017 the 5th IEEE International Conference on Smart Energy Grid Engineering (SEGE) : August 14-17, 2017, UOIT Oshawa, Canada*, 2017, pp. 367–372.

[55] R. Alt and E. Wende, "Blockchain technology in energy markets – An interview with the European Energy Exchange," *Electron Markets*, vol. 30, no. 2, pp. 325–330, Jun. 2020, doi: 10.1007/s12525-020-00423-6.

[56] F. Zhao, X. Guo, and W. K. (Victor) Chan, "Individual Green Certificates on Blockchain: A Simulation Approach," *Sustainability*, vol. 12, no. 9, p. 3942, May 2020, doi: 10.3390/su12093942.

[57] Jerry I.-H. Hsiao, "Blockchain for Corporate Renewable Energy Procurement-Potential for Verification of Renewable Energy Certificates," *UCLR*, vol. 15, no. 2, Feb. 2018, doi: 10.17265/1548-6605/2018.02.002.

[58] S. Zhang, J. Xuan, Z. Lyu, and Y. Fu, "Application Prospect of Blockchain in Renewable Energy Certificates," in *Proceedings of the 4th International Conference on Computer Science and Application Engineering*, New York, NY, USA, Oct. 2020, pp. 1–5. doi: 10.1145/3424978.3425016.

[59] Henderson Kimberly, E. Knoll, and M. Rogers, "What every utility CEO should know about blockchain | McKinsey," Mar. 23, 2018. https://www.mckinsey.com/industries/electric-power-and-natural-gas/our-insights/what-every-utility-ceo-should-know-about-blockchain# (accessed Apr. 15, 2021).

[60] "The Energy Web Chain,Accelerating the Energy Transition with an Open-Source, Decentralized Blockchain Platform, Version 2.0." Energy Web Foundation, Jul. 2019. Accessed: Apr. 08, 2021. [Online]. Available: https://energyweb.org/wp-content/uploads/2019/05/EWF-Paper-TheEnergyWebChain-v2-201907-FINAL.pdf

[61] "Origin Documentation - Energy Web Foundation." https://energyweb.atlassian.net/wiki/spaces/OD/overview?homepageId=883359997 (accessed Apr. 08, 2021).

[62] "OpenEthereum Documentation - Aura - Authority Round." https://openethereum.github.io/Aura.html (accessed Apr. 09, 2021).

[63] “Proof of Authority Consensus - EWF - Energy Web Foundation.” https://energyweb.atlassian.net/wiki/spaces/EWF/pages/717881446/Proof+of+Authority+Consensus (accessed Apr. 13, 2021).
[64] “Validator FAQ - EWF - Energy Web Foundation.” https://energyweb.atlassian.net/wiki/spaces/EWF/pages/712540161/Validator+FAQ (accessed Apr. 13, 2021).
[65] J. Wu and N. Tran, “Application of Blockchain Technology in Sustainable Energy Systems: An Overview,” *Sustainability*, vol. 10, no. 9, p. 3067, Aug. 2018, doi: 10.3390/su10093067.
[66] R. Williams, “How and why we built an internet connected solar panel,” *Medium*, Jul. 22, 2016. https://medium.com/ideo-colab/how-and-why-we-built-an-internet-connected-solar-panel-727d720d3803 (accessed Apr. 15, 2021).
[67] A. Rao, “Accelerating Energy Transition with Blockchain Technology,” *IFPEN*. https://www.ifpenergiesnouvelles.com/article/accelerating-energy-transition-blockchain-technology (accessed Apr. 15, 2021).
[68] HSV, “The Energy Web Foundation Partners With Swytch Who Become An Affiliate To Accelerate Blockchain…,” *Medium*, Apr. 25, 2020. https://hsvgts.medium.com/the-energy-web-foundation-partners-with-swytch-who-become-an-affiliate-to-accelerate-blockchain-b88c96c841f3 (accessed Apr. 15, 2021).
[69] Creating a more efficient green energy marketplace with IBM Blockchain technology, “Energy Blockchain Labs Inc.,” Oct. 02, 2018. https://www.ibm.com/case-studies/energy-blockchain-labs-inc (accessed Apr. 15, 2021).
[70] K. Amadeo, “Could Carbon Emissions Trading Become the New Bitcoin?,” *The Balance*. https://www.thebalance.com/carbon-emissions-trading-3305652 (accessed Apr. 15, 2021).
[71] J. Kosowatz, “Blockchain Enables New Power Markets,” *Mechanical Engineering*, vol. 141, no. 06, pp. 26–31, Jun. 2019, doi: 10.1115/1.2019-JUN1.
[72] A. Burger, “Iberdrola, Kutxabank Put FlexiDAO’s EWF-based Spring Renewable Energy Blockchain to the Test,” *Solar Magazine*, Feb. 18, 2019. https://solarmagazine.com/iberdrola-kutxabank-put-flexidao-ewf-based-spring-renewable-energy-blockchain-to-the-test/ (accessed Apr. 15, 2021).
[73] J. Dispenza, C. Garcia, and R. Molecke, “Energy Efficiency Coin (EECoin) A Blockchain Asset Class Pegged to Renewable Energy Markets,” p. 6, Jul. 2017.
[74] “Cosmos SDK | Tendermint.” https://tendermint.com/sdk/ (accessed Apr. 15, 2021).
[75] G. Mooney, “Blockchain for Utilities – reality checkpoint | SAP Blogs,” Apr. 24, 2019. https://blogs.sap.com/2019/04/24/blockchain-for-utilities-reality-checkpoint/ (accessed Apr. 15, 2021).
[76] Y. Hirai, “Defining the Ethereum Virtual Machine for Interactive Theorem Provers,” in *Financial Cryptography and Data Security*, vol. 10323, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. A. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds. Cham: Springer International Publishing, 2017, pp. 520–535. doi: 10.1007/978-3-319-70278-0_33.
[77] “Ethereum Virtual Machine (EVM),” *ethereum.org*. https://ethereum.org (accessed Apr. 20, 2021).
[78] I. Spectrum, “Ethereum Plans to Cut Its Absurd Energy Consumption by 99%.” https://cacm.acm.org/news/233826-ethereum-plans-to-cut-its-absurd-energy-consumption-by-99/fulltext (accessed Apr. 21, 2021).
[79] “Shard chains,” *ethereum.org*. https://ethereum.org (accessed Apr. 21, 2021).

[80] S. Bistarelli, G. Mazzante, M. Micheletti, L. Mostarda, D. Sestili, and F. Tiezzi, “Ethereum smart contracts: Analysis and statistics of their source code and opcodes,” *Internet of Things*, vol. 11, p. 100198, Sep. 2020, doi: 10.1016/j.iot.2020.100198.

[81] C. Schwarz, “Ethereum 2.0: A Complete Guide. Ewasm.,” *Medium*, Feb. 05, 2021. https://medium.com/chainsafe-systems/ethereum-2-0-a-complete-guide-ewasm-394cac756baf (accessed Apr. 21, 2021).

[82] V. Clincy and H. Shahriar, “Blockchain Development Platform Comparison,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Milwaukee, WI, USA, Jul. 2019, pp. 922–923. doi: 10.1109/COMPSAC.2019.00142.

[83] “What is Ethereum 2.0? Overview, Features and Price Implications,” *Diginex*, Nov. 27, 2020. https://www.diginex.com/insights/what-is-ethereum-2-0-overview-features-and-price-implications/ (accessed Apr. 22, 2021).

[84] “What Is Quorum Blockchain? A Platform for The Enterprise,” *Blockgeeks*, May 06, 2020. https://blockgeeks.com/guides/quorum-a-blockchain-platform-for-the-enterprise/ (accessed Apr. 22, 2021).

[85] “What is a blockchain fork?| CMC Markets.” https://www.cmcmarkets.com/en/learn-cryptocurrencies/what-is-a-blockchain-fork (accessed Apr. 21, 2021).

[86] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, “Performance Evaluation of the Quorum Blockchain Platform,” *arXiv:1809.03421 [cs]*, Jul. 2018, Accessed: Apr. 21, 2021. [Online]. Available: http://arxiv.org/abs/1809.03421

[87] I. Allison, “ConsenSys Acquires JPMorgan’s Quorum Blockchain,” *CoinDesk*, Aug. 25, 2020. https://www.coindesk.com/consensys-acquires-jp-morgan-quorum-blockchain (accessed Apr. 22, 2021).

[88] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, “Performance Analysis of Hyperledger Fabric Platforms,” *Security and Communication Networks*, vol. 2018, pp. 1–14, Sep. 2018, doi: 10.1155/2018/3976093.

[89] E. Androulaki *et al.*, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proceedings of the Thirteenth EuroSys Conference*, Porto Portugal, Apr. 2018, pp. 1–15. doi: 10.1145/3190508.3190538.

[90] “Introduction — hyperledger-fabricdocs master documentation.” https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html (accessed Apr. 22, 2021).

[91] D. Creer, “How to choose the best platform for your blockchain based system?,” *Medium*, Mar. 30, 2020. https://medium.com/gft-engineering/how-to-choose-the-best-platform-for-your-blockchain-based-system-8b9c57862225 (accessed Apr. 20, 2021).

[92] C. Saraf and S. Sabadra, “Blockchain platforms: A compendium,” in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, Bangkok, May 2018, pp. 1–6. doi: 10.1109/ICIRD.2018.8376323.

[93] A. LØNSETTEIG, “R3’s Corda uncovered: It’s not blockchain,” *Global Trade Review (GTR)*, Oct. 01, 2017. https://www.gtreview.com/magazine/volume-15issue-3/r3s-corda-uncovered-not-blockchain (accessed Apr. 22, 2021).

[94] J. Frankenfield, “EOS,” *Investopedia*. https://www.investopedia.com/terms/e/eos.asp (accessed Apr. 22, 2021).

[95] N. Kuznetsov, “Ethereum 2.0 and EOS Crossing Swords Over Scalability Supremacy,” *Cointelegraph*. https://cointelegraph.com/news/ethereum-20-and-eos-crossing-swords-over-scalability-supremacy (accessed Apr. 23, 2021).

[96] B. Xu, D. Luthra, Z. Cole, and N. Blakely, “EOS: An Architectural, Performance, and Economic Analysis,” p. 25.

[97] A. Makarov, “Top 6 smart contract platforms: a deep dive,” Jan. 04, 2021. https://www.itransition.com/blog/smart-contract-platforms (accessed Apr. 27, 2021).
[98] L. M. Goodman, “Tezos: A Self-Amending Crypto-Ledger Position Paper,” p. 18, Aug. 2014.
[99] “What Is Tezos? | Tezos.” https://tezos.com/docs/learn/what-is-tezos (accessed Apr. 27, 2021).
[100] “What Is Tezos (XTZ)?,” *Binance Academy*. https://academy.binance.com/en/articles/what-is-tezos-xtz (accessed Apr. 27, 2021).
[101] T. K. Sharma, “All You Need To Know About EOS,” Sep. 05, 2019. https://www.blockchain-council.org/blockchain/all-you-need-to-know-about-eos/ (accessed Apr. 27, 2021).
[102] D. P. Mera, “Quorum Blockchain Stress Evaluation in different environments,” p. 129.
[103] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second,” May 2019, pp. 455–463. doi: 10.1109/BLOC.2019.8751452.
[104] “Transactions Per Second (TPS),” *Corda*, Feb. 09, 2018. https://www.corda.net/blog/transactions-per-second-tps/ (accessed May 07, 2021).
[105] S. Avala, “Why we built our blockchain business on EOS instead of Ethereum,” *VentureBeat*, Apr. 13, 2019. https://venturebeat.com/2019/04/13/why-we-built-our-blockchain-business-on-eos-instead-of-ethereum/ (accessed Apr. 27, 2021).
[106] “Overview - GoQuorum.” https://docs.goquorum.consensys.net/en/stable/Concepts/Privacy/Privacy/ (accessed Apr. 28, 2021).
[107] “ConsenSys/quorum,” *GitHub*. https://github.com/ConsenSys/quorum (accessed Apr. 28, 2021).
[108] K. Peters, “What is zk-SNARK?,” *Investopedia*. https://www.investopedia.com/terms/z/zksnark.asp (accessed Apr. 28, 2021).
[109] C. Kim, “EY Open-Sources ‘Nightfall’ Code for Private Transactions on Ethereum,” *CoinDesk*, May 31, 2019. https://www.coindesk.com/ey-open-sources-nightfall-code-for-private-transactions-on-ethereum (accessed Apr. 28, 2021).
[110] “Introduction - ZoKrates.” https://zokrates.github.io/ (accessed Apr. 28, 2021).
[111] *EYBlockchain/nightfall*. EY Blockchain, 2021. Accessed: Apr. 28, 2021. [Online]. Available: https://github.com/EYBlockchain/nightfall
[112] “EIP-20: ERC-20 Token Standard,” *Ethereum Improvement Proposals*. https://eips.ethereum.org/EIPS/eip-20 (accessed May 05, 2021).
[113] “ERC-721 Non-Fungible Token Standard,” *ethereum.org*, Apr. 11, 2021. https://ethereum.org (accessed Apr. 28, 2021).
[114] C. Konda, M. Connor, D. Westland, Q. Drouot, and P. Brody, “Nightfall.” EY Global Blockchain R&D, May 2019. [Online]. Available: https://img.learnblockchain.cn/pdf/nightfall-v1.pdf
[115] A. Shevchenko, “Privacy on Ethereum: Aztec Protocol Launches on Mainnet,” *Cointelegraph*. https://cointelegraph.com/news/privacy-on-ethereum-aztec-protocol-launches-on-mainnet (accessed Apr. 29, 2021).
[116] “AZTEC Docs.” https://developers.aztec.network/ (accessed Apr. 29, 2021).
[117] J. Andrews, “Aztec: zkRollup Layer 2 + Privacy,” *Medium*, Oct. 12, 2020. https://medium.com/aztec-protocol/aztec-zkrollup-layer-2-privacy-1978e90ee3b6 (accessed Apr. 29, 2021).

[118] T. Walton-Pocock, “Launching Aztec 2.0 Rollup,” *Medium*, Mar. 19, 2021. https://medium.com/aztec-protocol/launching-aztec-2-0-rollup-ac7db8012f4b (accessed Apr. 29, 2021).
[119] A. Nagy, “Privacy solutions overview - EWF - Energy Web Foundation.” https://energyweb.atlassian.net/wiki/spaces/EWF/pages/610992129/Privacy+solutions+overvie w (accessed Apr. 29, 2021).
[120] M. Thoma, “Merkle Trees,” *Medium*, Jan. 28, 2021. https://levelup.gitconnected.com/merkle-trees-e4fdaeaa3094 (accessed Apr. 29, 2021).
[121] L. Vogelsang, “Introducing Precise-Proofs: Create & Validate Field-Level Merkle Proofs,” *Medium*, Apr. 26, 2018. https://medium.com/centrifuge/introducing-precise-proofs-create-validate-field-level-merkle-proofs-a31af9220df0 (accessed Apr. 29, 2021).
[122] “Solidity — Solidity 0.8.3 documentation.” https://docs.soliditylang.org/en/v0.8.3/ (accessed Apr. 20, 2021).
[123] E. Irannezhad, “The Architectural Design Requirements of a Blockchain-Based Port Community System,” *Logistics*, vol. 4, no. 4, Art. no. 4, Dec. 2020, doi: 10.3390/logistics4040030.
[124] “Truffle | Overview | Documentation,” *Truffle Suite*. https://trufflesuite.com/docs/truffle/overview (accessed Apr. 30, 2021).
[125] “Ganache | Overview | Documentation,” *Truffle Suite*. https://trufflesuite.com/docs/ganache/overview (accessed May 04, 2021).
[126] “OpenZeppelin | Contracts,” *OpenZeppelin*. https://openzeppelin.com/ (accessed May 04, 2021).
[127] “EIP-1155: ERC-1155 Multi Token Standard,” *Ethereum Improvement Proposals*. https://eips.ethereum.org/EIPS/eip-1155 (accessed May 05, 2021).
[128] J. Waldenfels, “Issuing certificates with the EW Origin SDK (Part I),” *Medium*, Mar. 19, 2020. https://medium.com/energy-web-insights/issuing-certificates-with-the-ew-origin-sdk-part-i-7630c14e13b (accessed May 05, 2021).
[129] J. Waldenfels, “Exchange Module - Origin Documentation - Energy Web Foundation.” https://energyweb.atlassian.net/wiki/spaces/OD/pages/1138884622/Exchange+Module (accessed May 06, 2021).
[130] “Writing Upgradeable Contracts - OpenZeppelin Docs.” https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable (accessed May 09, 2021).
[131] D. Kuhn, “Did Ethereum Learn Anything From the $55M DAO Attack?,” *CoinDesk*, Sep. 20, 2020. https://www.coindesk.com/ethereum-learn-dao-attack (accessed May 09, 2021).
[132] J. Waldenfels, “Origin FAQ - Origin Documentation - Energy Web Foundation,” Feb. 23, 2020. https://energyweb.atlassian.net/wiki/spaces/OD/pages/1571422212/Origin+FAQ (accessed May 09, 2021).
[133] EC-Council, “What is Ethereum 2.0 and Why Does It Matter? | EC-Council,” *EC-Council Official Blog*, Jan. 08, 2021. https://blog.eccouncil.org/what-is-ethereum-2-and-why-does-it-matter/ (accessed Apr. 22, 2021).
[134] A. Baydakova, “Enterprise Blockchains: Walled Off Yet Vulnerable,” *CoinDesk*, May 03, 2020. https://www.coindesk.com/enterprise-blockchains-walled-off-yet-vulnerable (accessed May 11, 2021).
[135] “Registration - Origin Documentation - Energy Web Foundation.” https://energyweb.atlassian.net/wiki/spaces/OD/pages/792264767/Registration (accessed May 07, 2021).

[136] L. Matthews and H. BA, “The 9 Most Sustainable Cryptocurrencies for 2021,” *LeafScore*, Apr. 02, 2021. https://www.leafscore.com/blog/the-9-most-sustainable-cryptocurrencies-for-2021/ (accessed May 07, 2021).

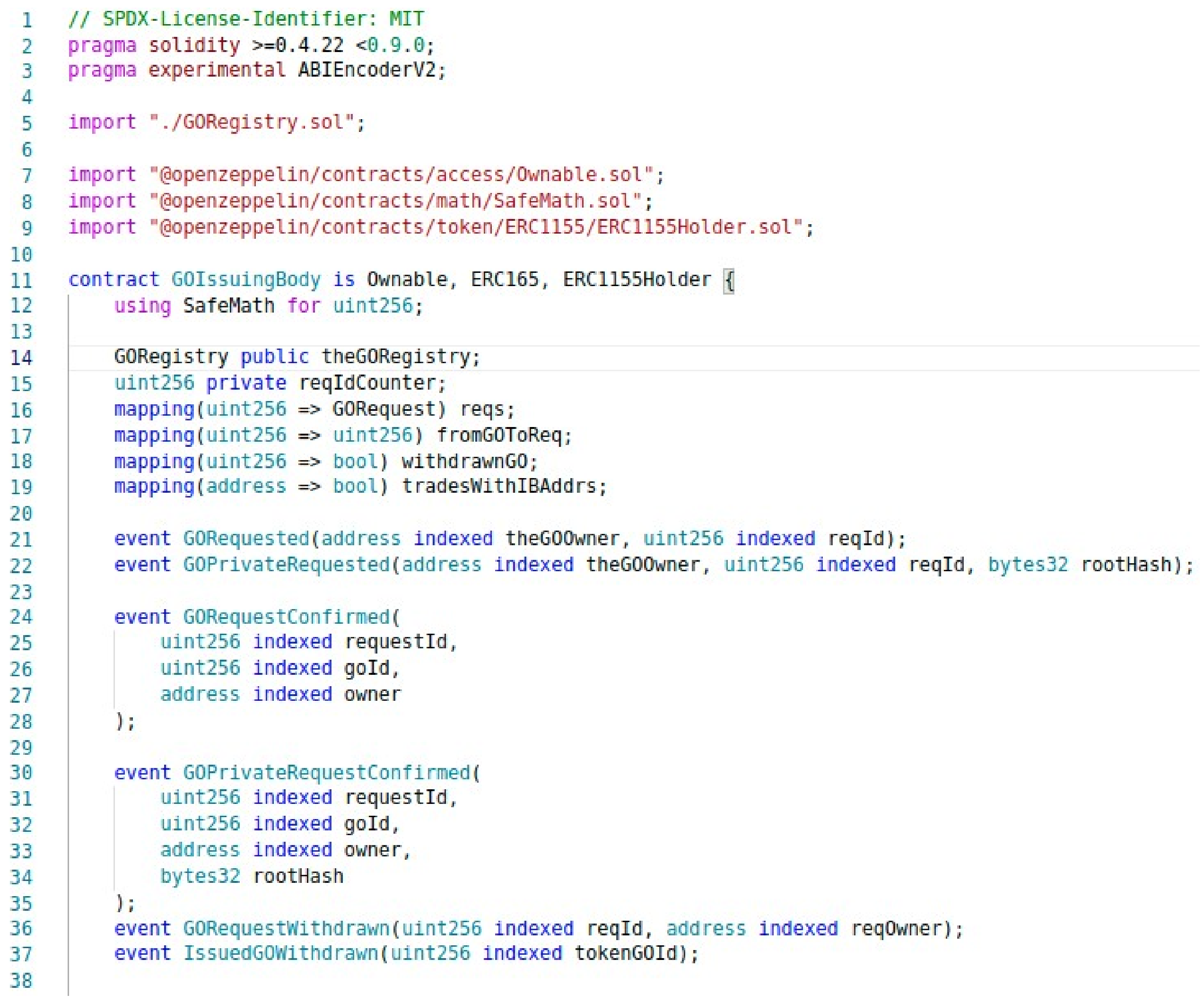
[137] P. Fairley, “Ethereum will cut back its absurd energy use,” *IEEE Spectr.*, vol. 56, no. 1, pp. 29–32, Jan. 2019, doi: 10.1109/MSPEC.2019.8594790.

[138] A. Tirić, *Prototype of Guarantees of Origin on Blockchain*. 2021. Accessed: May 13, 2021. [Online]. Available: https://github.com/Tyrx95/GoO-blockchain

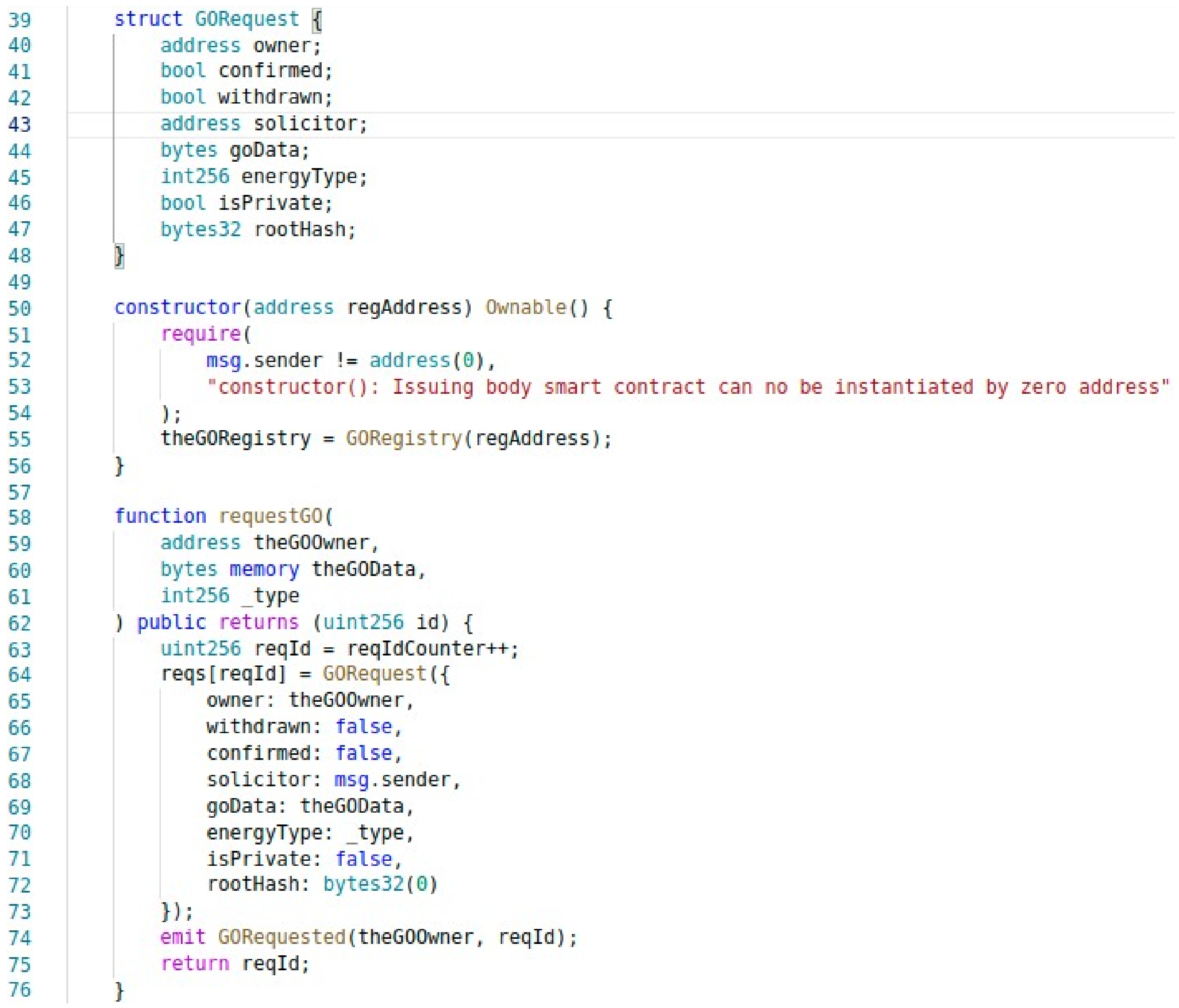
8 Appendices

The appendices chapter provides the source code of the two main prototype smart contracts written in the Solidity programming language. The first section provides a smart contract code of *GOIssuingBody.sol* smart contract. The second section provides the code of the *GORegistry.sol* smart contract. The source code can also be previewed on the author’s GitHub repository [138].

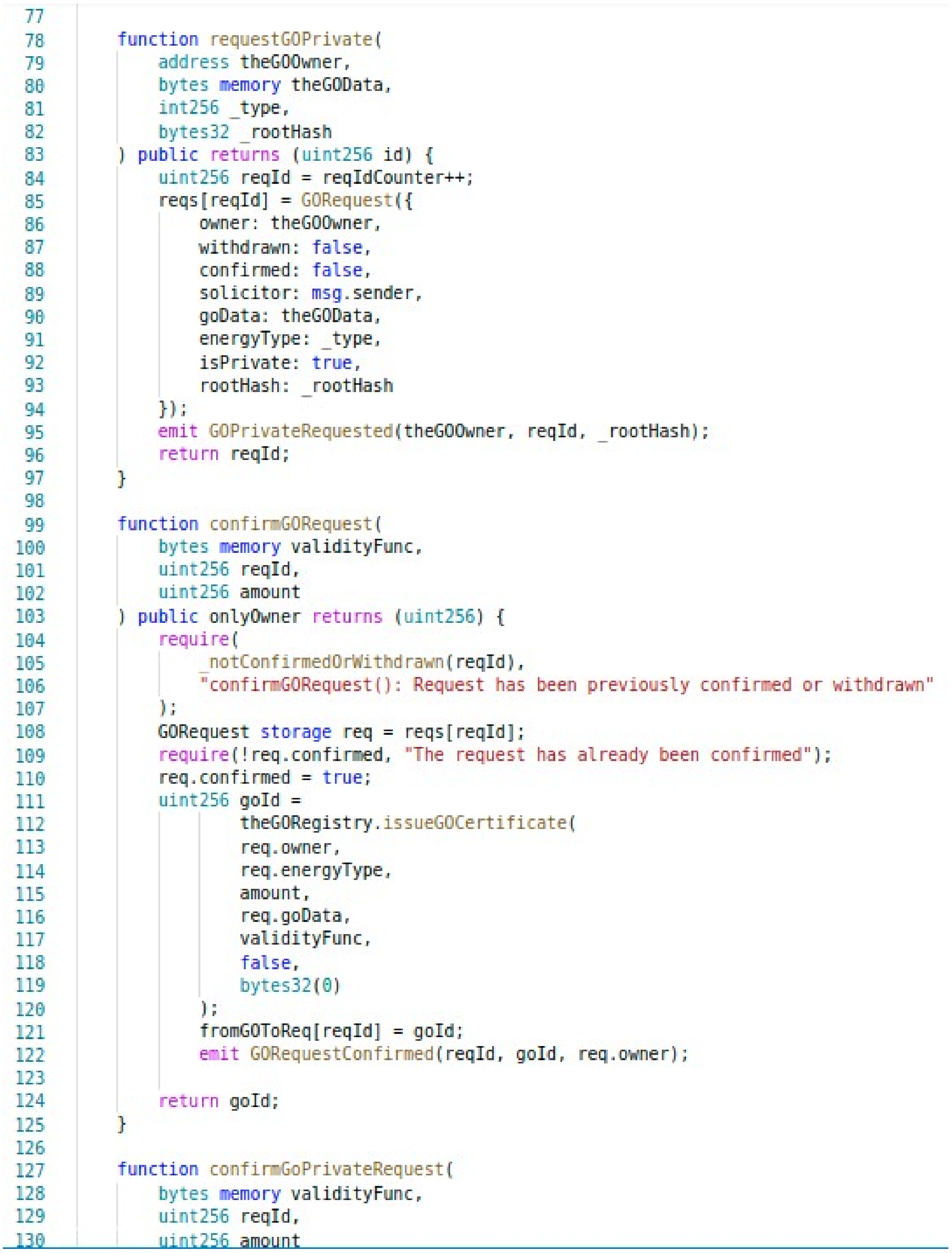
8.1 GOIssuingBody.sol smart contract code



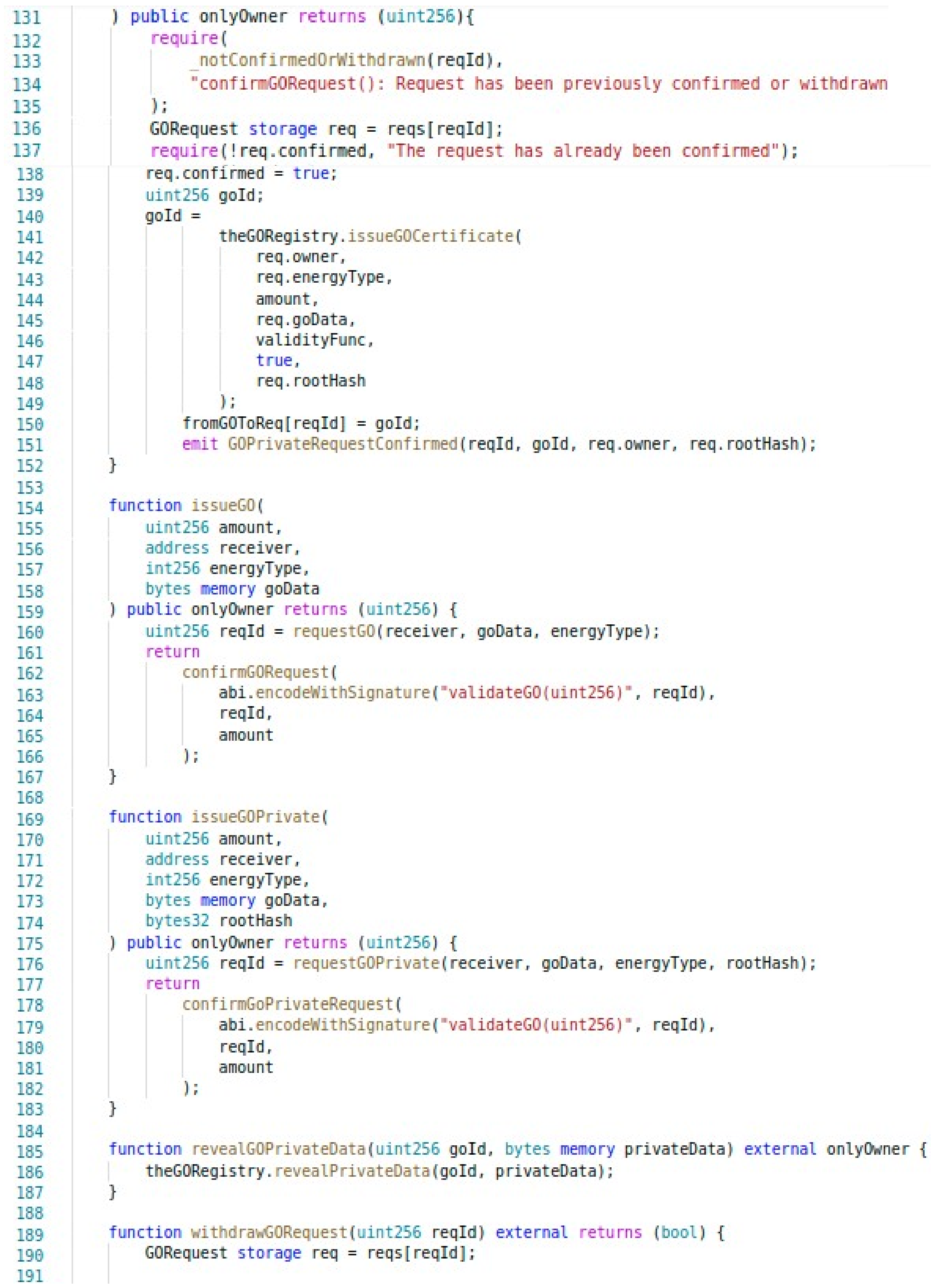
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;
pragma experimental ABIEncoderV2;

import "./GORegistry.sol";

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/math/SafeMath.sol";
import "@openzeppelin/contracts/token/ERC1155/ERC1155Holder.sol";

contract GOIssuingBody is Ownable, ERC165, ERC1155Holder {
    using SafeMath for uint256;

    GORegistry public theGORegistry;
    uint256 private reqIdCounter;
    mapping(uint256 => GORequest) reqs;
    mapping(uint256 => uint256) fromGOToReq;
    mapping(uint256 => bool) withdrawnGO;
    mapping(address => bool) tradesWithIBAddrs;

    event GORequested(address indexed theGOOwner, uint256 indexed reqId);
    event GOPrivateRequested(address indexed theGOOwner, uint256 indexed reqId, bytes32 rootHash);

    event GORequestConfirmed(
        uint256 indexed requestId,
        uint256 indexed goId,
        address indexed owner
    );

    event GOPrivateRequestConfirmed(
        uint256 indexed requestId,
        uint256 indexed goId,
        address indexed owner,
        bytes32 rootHash
    );
    event GORequestWithdrawn(uint256 indexed reqId, address indexed reqOwner);
    event IssuedGOWithdrawn(uint256 indexed tokenGOId);

```



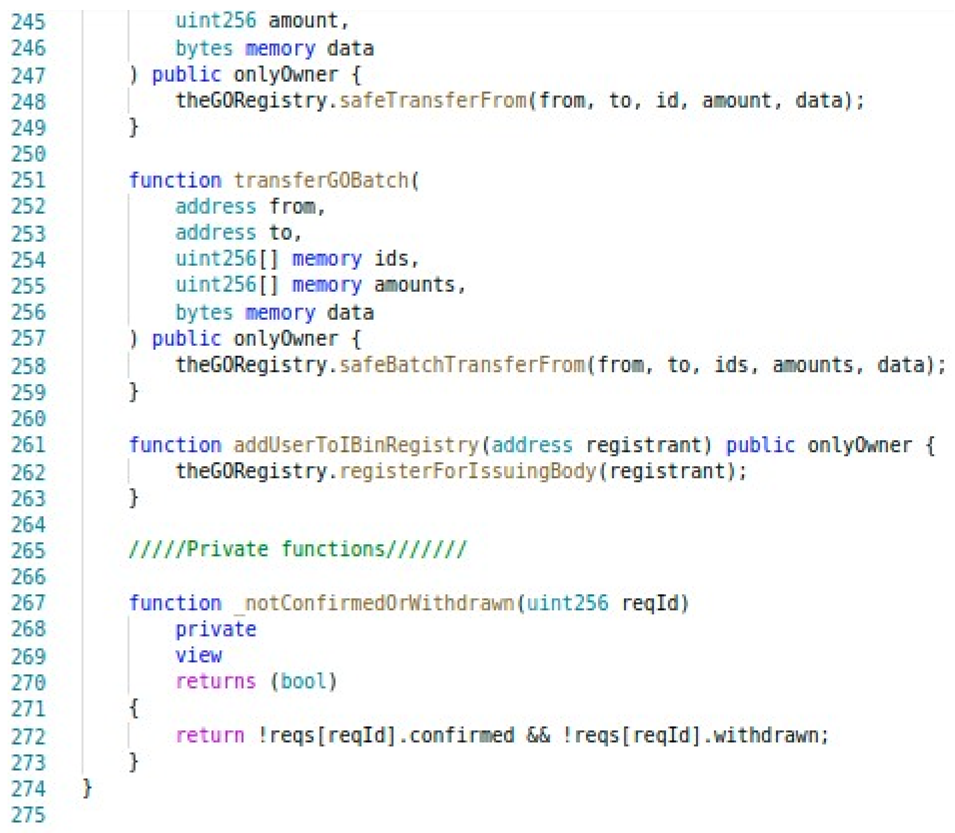
```
    struct GORequest {
        address owner;
        bool confirmed;
        bool withdrawn;
        address solicitor;
        bytes goData;
        int256 energyType;
        bool isPrivate;
        bytes32 rootHash;
    }

    constructor(address regAddress) Ownable() {
        require(
            msg.sender != address(0),
            "constructor(): Issuing body smart contract can no be instantiated by zero address"
        );
        theGORegistry = GORegistry(regAddress);
    }

    function requestGO(
        address theGOOwner,
        bytes memory theGOData,
        int256 _type
    ) public returns (uint256 id) {
        uint256 reqId = reqIdCounter++;
        reqs[reqId] = GORequest({
            owner: theGOOwner,
            withdrawn: false,
            confirmed: false,
            solicitor: msg.sender,
            goData: theGOData,
            energyType: _type,
            isPrivate: false,
            rootHash: bytes32(0)
        });
        emit GORequested(theGOOwner, reqId);
        return reqId;
    }
```



```
    function requestGOPrivate(
        address theGOOwner,
        bytes memory theGOData,
        int256 _type,
        bytes32 _rootHash
    ) public returns (uint256 id) {
        uint256 reqId = reqIdCounter++;
        reqs[reqId] = GORequest({
            owner: theGOOwner,
            withdrawn: false,
            confirmed: false,
            solicitor: msg.sender,
            goData: theGOData,
            energyType: _type,
            isPrivate: true,
            rootHash: _rootHash
        });
        emit GOPrivateRequested(theGOOwner, reqId, _rootHash);
        return reqId;
    }

    function confirmGORequest(
        bytes memory validityFunc,
        uint256 reqId,
        uint256 amount
    ) public onlyOwner returns (uint256) {
        require(
            _notConfirmedOrWithdrawn(reqId),
            "confirmGORequest(): Request has been previously confirmed or withdrawn"
        );
        GORequest storage req = reqs[reqId];
        require(!req.confirmed, "The request has already been confirmed");
        req.confirmed = true;
        uint256 goId =
                theGORegistry.issueGOCertificate(
                req.owner,
                req.energyType,
                amount,
                req.goData,
                validityFunc,
                false,
                bytes32(0)
            );
            fromGOToReq[reqId] = goId;
            emit GORequestConfirmed(reqId, goId, req.owner);

        return goId;
    }

    function confirmGoPrivateRequest(
        bytes memory validityFunc,
        uint256 reqId,
        uint256 amount
```

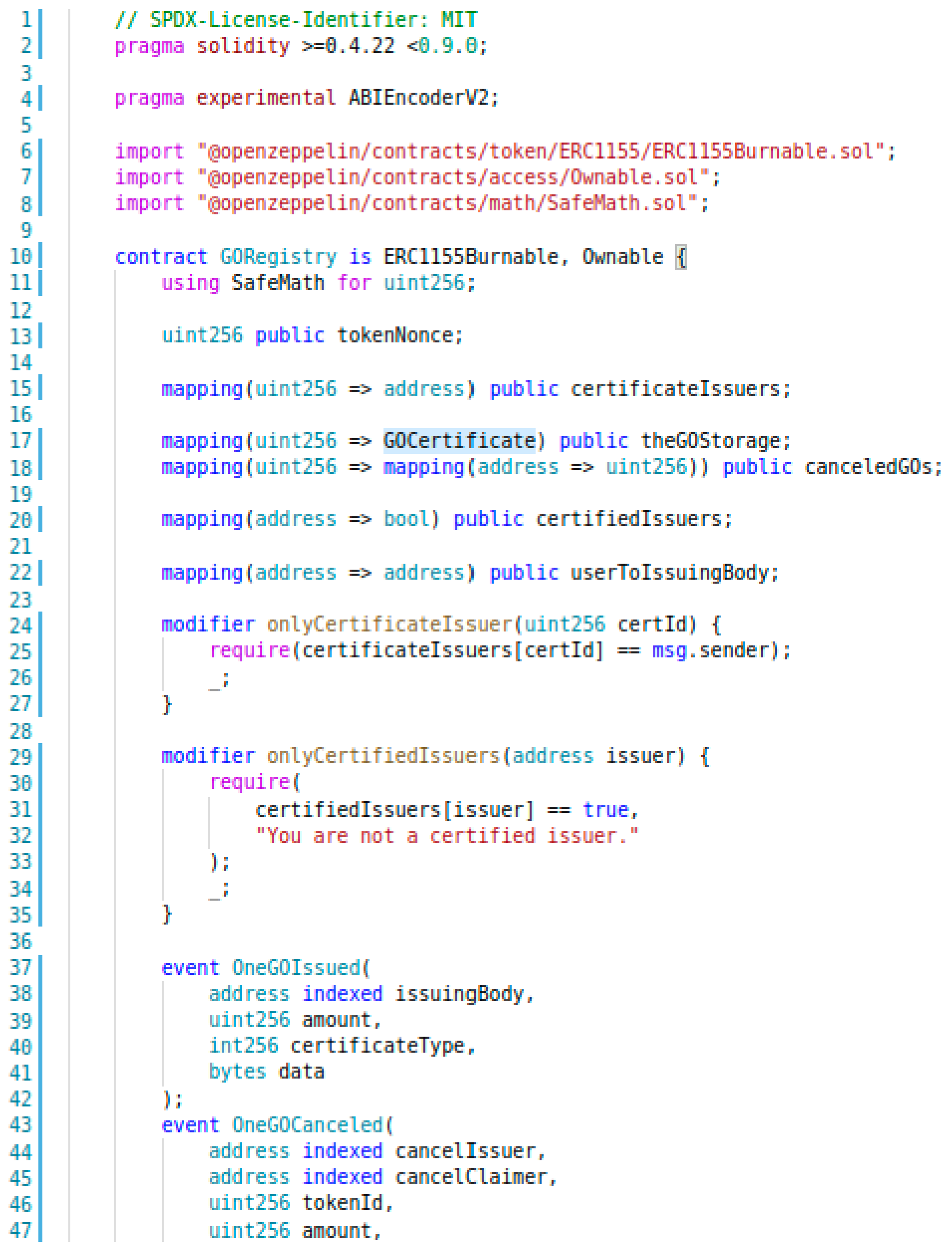


```
    ) public onlyOwner returns (uint256){
        require(
            _notConfirmedOrWithdrawn(reqId),
            "confirmGORequest(): Request has been previously confirmed or withdrawn
        );
        GORequest storage req = reqs[reqId];
        require(!req.confirmed, "The request has already been confirmed");
        req.confirmed = true;
        uint256 goId;
        goId =
                theGORegistry.issueGOCertificate(
                    req.owner,
                    req.energyType,
                    amount,
                    req.goData,
                    validityFunc,
                    true,
                    req.rootHash
                );
            fromGOToReq[reqId] = goId;
            emit GOPrivateRequestConfirmed(reqId, goId, req.owner, req.rootHash);
    }

    function issueGO(
        uint256 amount,
        address receiver,
        int256 energyType,
        bytes memory goData
    ) public onlyOwner returns (uint256) {
        uint256 reqId = requestGO(receiver, goData, energyType);
        return
            confirmGORequest(
                abi.encodeWithSignature("validateGO(uint256)", reqId),
                reqId,
                amount
            );
    }

    function issueGOPrivate(
        uint256 amount,
        address receiver,
        int256 energyType,
        bytes memory goData,
        bytes32 rootHash
    ) public onlyOwner returns (uint256) {
        uint256 reqId = requestGOPrivate(receiver, goData, energyType, rootHash);
        return
            confirmGoPrivateRequest(
                abi.encodeWithSignature("validateGO(uint256)", reqId),
                reqId,
                amount
            );
    }

    function revealGOPrivateData(uint256 goId, bytes memory privateData) external onlyOwner {
        theGORegistry.revealPrivateData(goId, privateData);
    }

    function withdrawGORequest(uint256 reqId) external returns (bool) {
        GORequest storage req = reqs[reqId];

        require(
            msg.sender == owner() || msg.sender == req.owner,
            "Request can be withdrawn only by Issuing Body or the request initiator"
        );
        require(
            !req.confirmed,
            "Confirmed GO requests can not be withdrawn"
        );
        require(!req.withdrawn, "The request has already been withdrawn ");

        req.withdrawn = true;
        emit GORequestWithdrawn(reqId, req.owner);
        return true;
    }

    function withdrawIssuedGO(uint256 goTokenId) external returns (bool) {
        require(
            !withdrawnGO[goTokenId],
            "GO has already been withdrawn by the Issuing Body Owner"
        );
        withdrawnGO[goTokenId] = true;

        emit IssuedGOWithdrawn(goTokenId);
        return true;
    }

    function validateGO(uint256 reqId) external view returns (bool) {
        GORequest memory req = reqs[reqId];
        uint256 tokenGOId = fromGOToReq[reqId];

        return
            req.confirmed &&
            !req.withdrawn &&
            !withdrawnGO[tokenGOId] &&
            reqId <= reqIdCounter;
    }

    function setTradesWith(address theIBAddress, bool value) public onlyOwner {
        tradesWithIBAddrs[theIBAddress] = value;
    }

    function isTradingWith(address issuingBodyAddress)
        public
        view
        returns (bool)
    {
        return tradesWithIBAddrs[issuingBodyAddress];
    }

    function transferGO(
        address from,
        address to,
        uint256 id,
        uint256 amount,
```

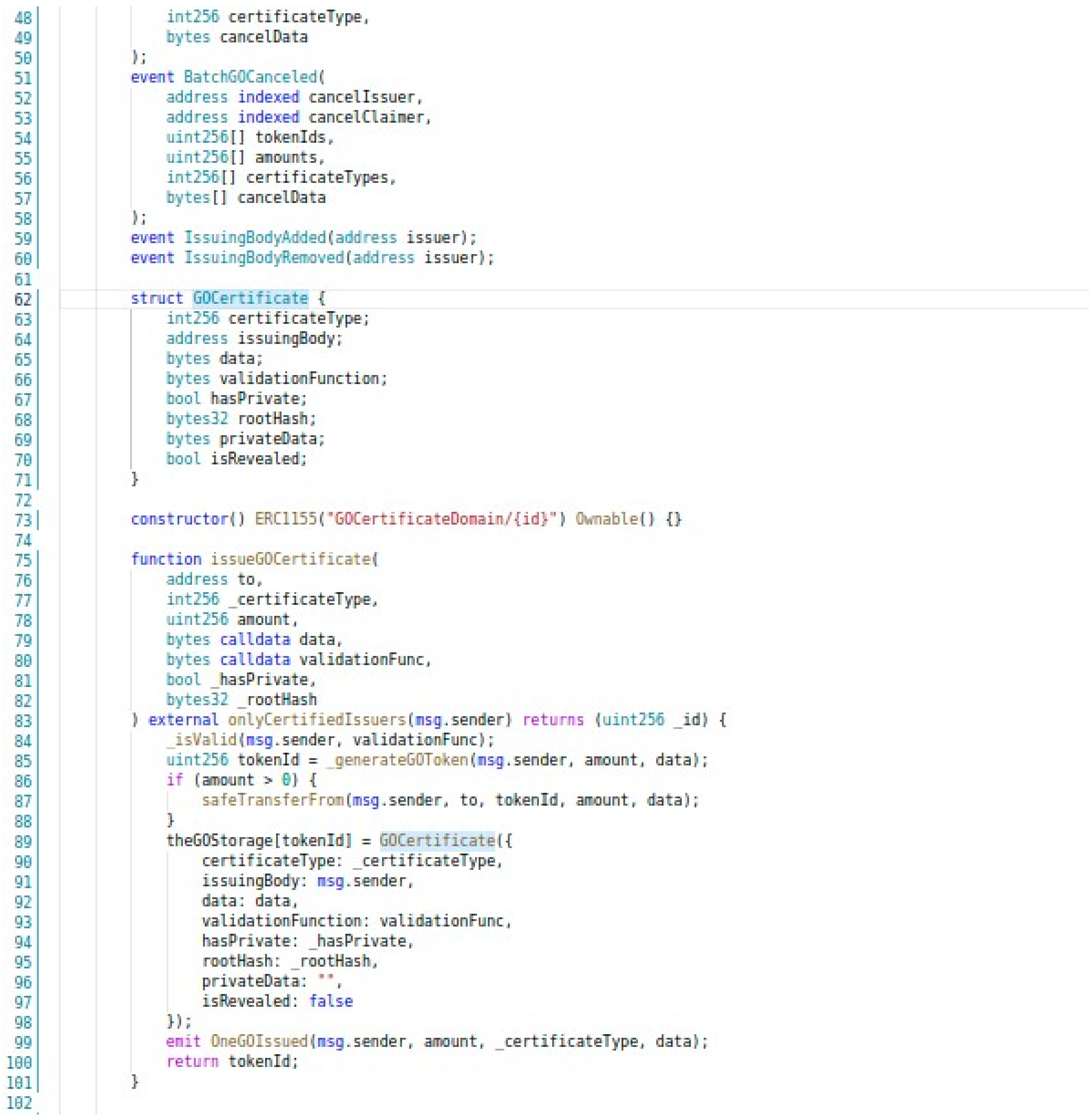


```
        uint256 amount,
        bytes memory data
    ) public onlyOwner {
        theGORegistry.safeTransferFrom(from, to, id, amount, data);
    }

    function transferGOBatch(
        address from,
        address to,
        uint256[] memory ids,
        uint256[] memory amounts,
        bytes memory data
    ) public onlyOwner {
        theGORegistry.safeBatchTransferFrom(from, to, ids, amounts, data);
    }

    function addUserToIBinRegistry(address registrant) public onlyOwner {
        theGORegistry.registerForIssuingBody(registrant);
    }

    /////Private functions///////

    function _notConfirmedOrWithdrawn(uint256 reqId)
        private
        view
        returns (bool)
    {
        return !reqs[reqId].confirmed && !reqs[reqId].withdrawn;
    }
}
```

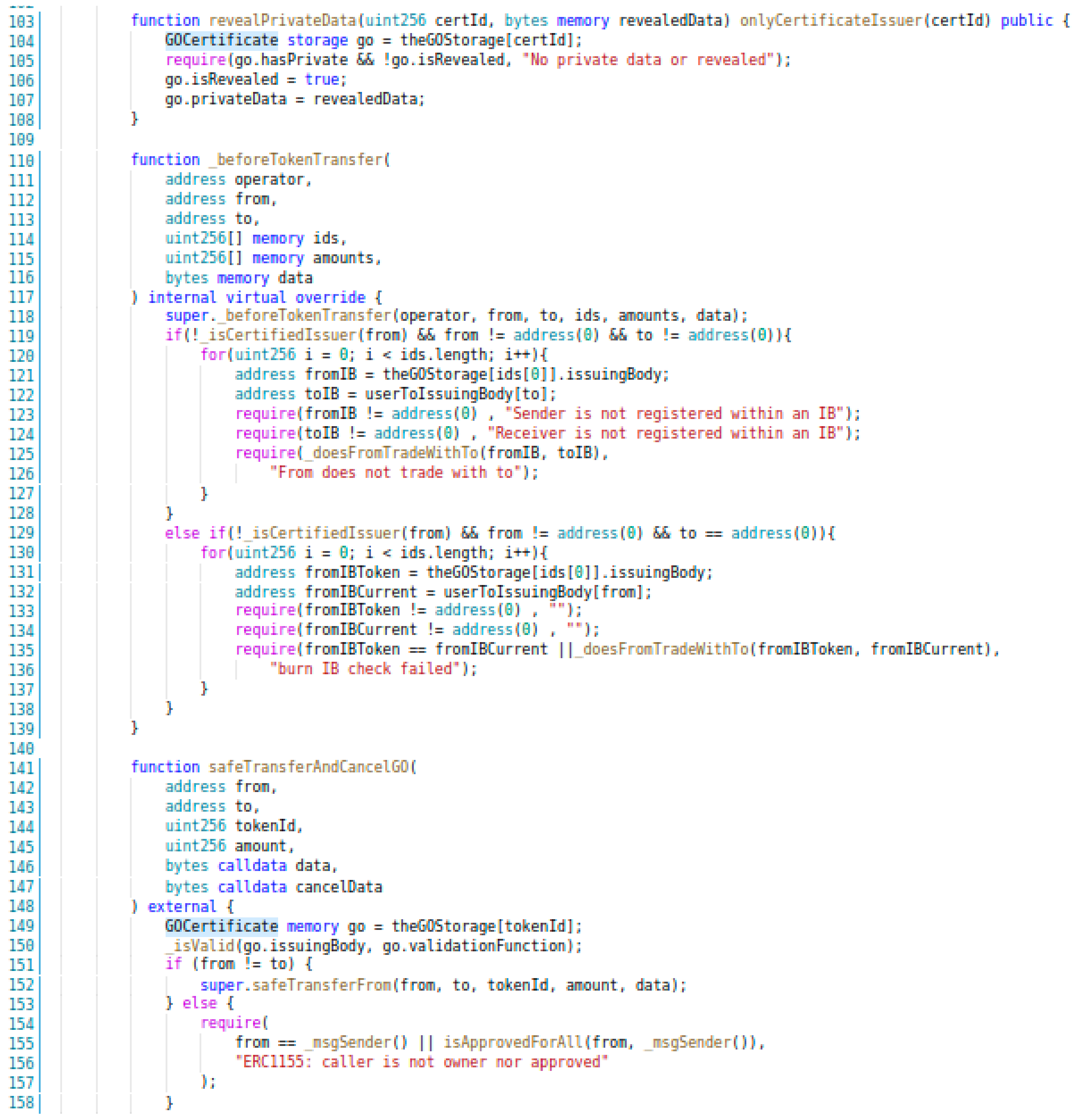
8.2 GORegistry.sol smart contract code



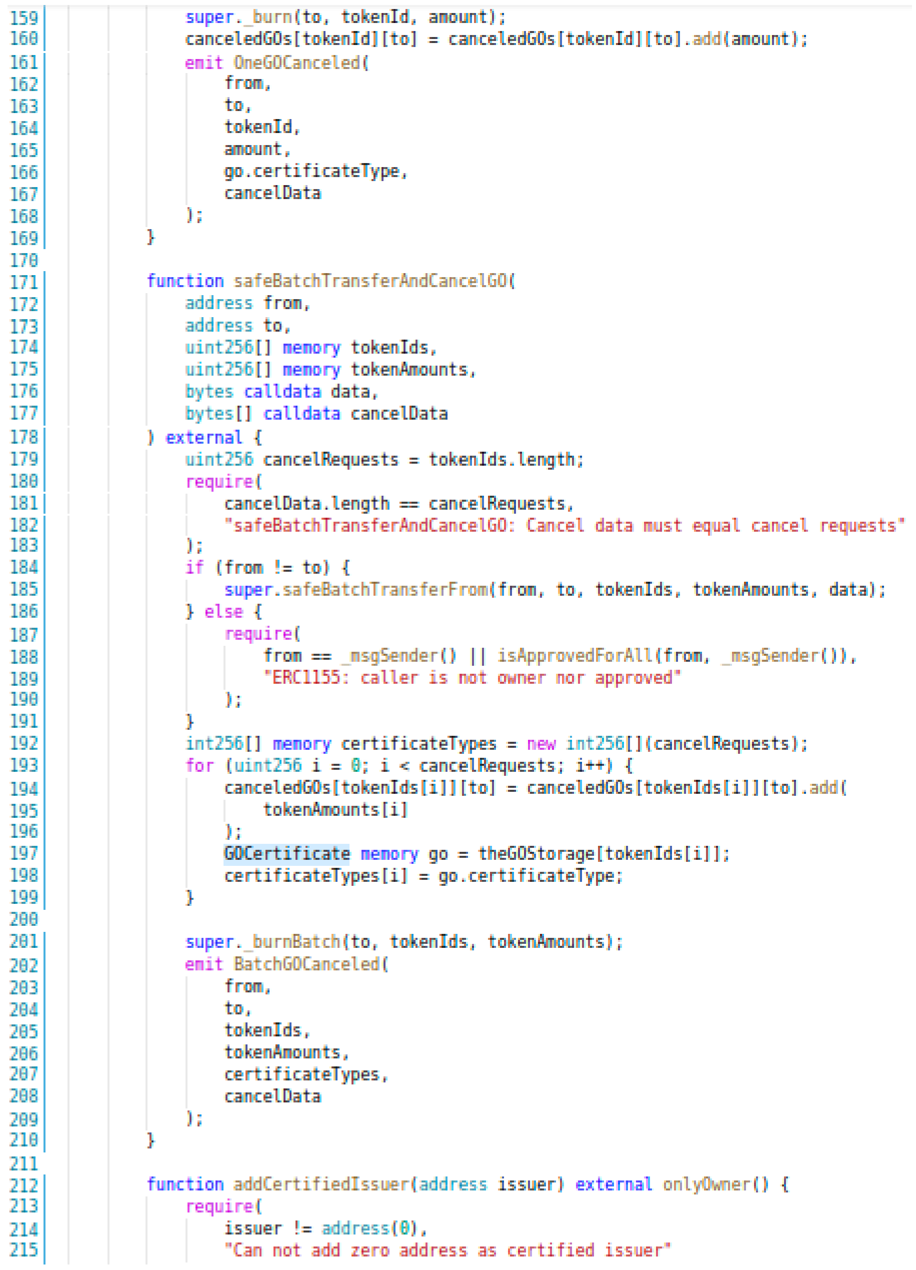
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

pragma experimental ABIEncoderV2;

import "@openzeppelin/contracts/token/ERC1155/ERC1155Burnable.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/math/SafeMath.sol";

contract GORegistry is ERC1155Burnable, Ownable {
    using SafeMath for uint256;

    uint256 public tokenNonce;

    mapping(uint256 => address) public certificateIssuers;

    mapping(uint256 => GOCertificate) public theGOStorage;
    mapping(uint256 => mapping(address => uint256)) public canceledGOs;

    mapping(address => bool) public certifiedIssuers;

    mapping(address => address) public userToIssuingBody;

    modifier onlyCertificateIssuer(uint256 certId) {
        require(certificateIssuers[certId] == msg.sender);
        _;
    }

    modifier onlyCertifiedIssuers(address issuer) {
        require(
            certifiedIssuers[issuer] == true,
            "You are not a certified issuer."
        );
        _;
    }

    event OneGOIssued(
        address indexed issuingBody,
        uint256 amount,
        int256 certificateType,
        bytes data
    );
    event OneGOCanceled(
        address indexed cancelIssuer,
        address indexed cancelClaimer,
        uint256 tokenId,
        uint256 amount,
```



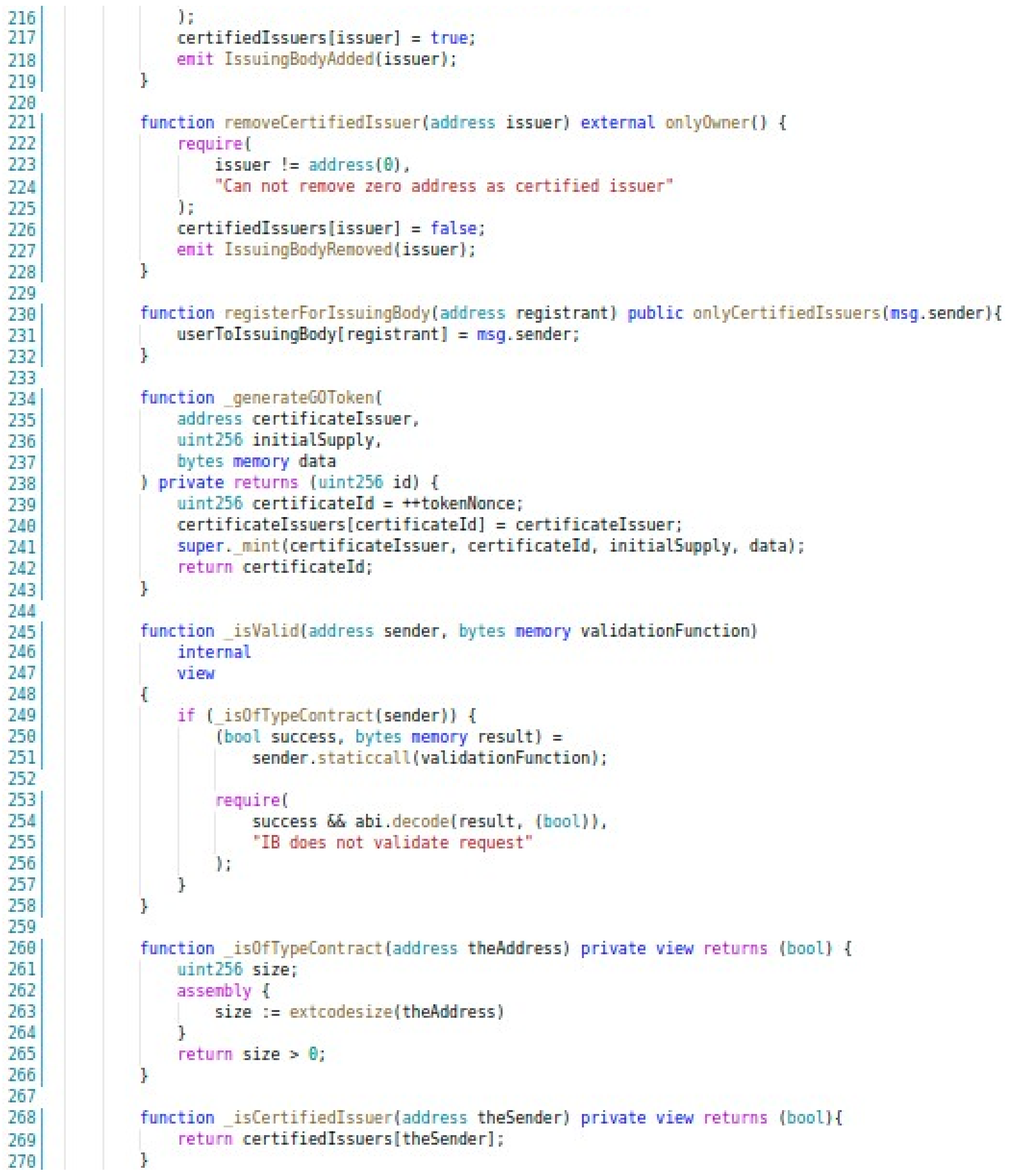
```
        int256 certificateType,
        bytes cancelData
    );
    event BatchGOCanceled(
        address indexed cancelIssuer,
        address indexed cancelClaimer,
        uint256[] tokenIds,
        uint256[] amounts,
        int256[] certificateTypes,
        bytes[] cancelData
    );
    event IssuingBodyAdded(address issuer);
    event IssuingBodyRemoved(address issuer);

    struct GOCertificate {
        int256 certificateType;
        address issuingBody;
        bytes data;
        bytes validationFunction;
        bool hasPrivate;
        bytes32 rootHash;
        bytes privateData;
        bool isRevealed;
    }

    constructor() ERC1155("GOCertificateDomain/{id}") Ownable() {}

    function issueGOCertificate(
        address to,
        int256 _certificateType,
        uint256 amount,
        bytes calldata data,
        bytes calldata validationFunc,
        bool _hasPrivate,
        bytes32 _rootHash
    ) external onlyCertifiedIssuers(msg.sender) returns (uint256 _id) {
        _isValid(msg.sender, validationFunc);
        uint256 tokenId = _generateGOToken(msg.sender, amount, data);
        if (amount > 0) {
            safeTransferFrom(msg.sender, to, tokenId, amount, data);
        }
        theGOStorage[tokenId] = GOCertificate({
            certificateType: _certificateType,
            issuingBody: msg.sender,
            data: data,
            validationFunction: validationFunc,
            hasPrivate: _hasPrivate,
            rootHash: _rootHash,
            privateData: "",
            isRevealed: false
        });
        emit OneGOIssued(msg.sender, amount, _certificateType, data);
        return tokenId;
    }

```



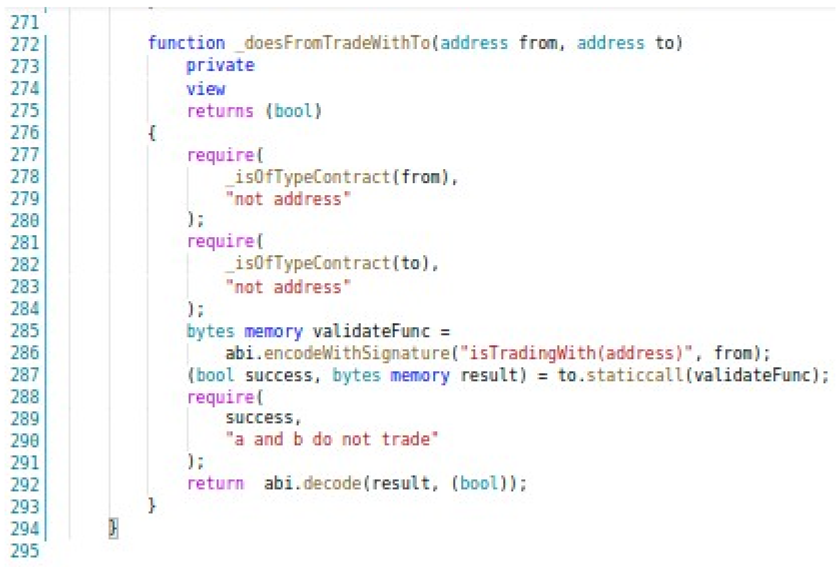
```
        function revealPrivateData(uint256 certId, bytes memory revealedData) onlyCertificateIssuer(certId) public {
            GOCertificate storage go = theGOStorage[certId];
            require(go.hasPrivate && !go.isRevealed, "No private data or revealed");
            go.isRevealed = true;
            go.privateData = revealedData;
        }

        function _beforeTokenTransfer(
            address operator,
            address from,
            address to,
            uint256[] memory ids,
            uint256[] memory amounts,
            bytes memory data
        ) internal virtual override {
            super._beforeTokenTransfer(operator, from, to, ids, amounts, data);
            if(!_isCertifiedIssuer(from) && from != address(0) && to != address(0)){
                for(uint256 i = 0; i < ids.length; i++){
                    address fromIB = theGOStorage[ids[0]].issuingBody;
                    address toIB = userToIssuingBody[to];
                    require(fromIB != address(0) , "Sender is not registered within an IB");
                    require(toIB != address(0) , "Receiver is not registered within an IB");
                    require(_doesFromTradeWithTo(fromIB, toIB),
                        "From does not trade with to");
                }
            }
            else if(!_isCertifiedIssuer(from) && from != address(0) && to == address(0)){
                for(uint256 i = 0; i < ids.length; i++){
                    address fromIBToken = theGOStorage[ids[0]].issuingBody;
                    address fromIBCurrent = userToIssuingBody[from];
                    require(fromIBToken != address(0) , "");
                    require(fromIBCurrent != address(0) , "");
                    require(fromIBToken == fromIBCurrent ||_doesFromTradeWithTo(fromIBToken, fromIBCurrent),
                        "burn IB check failed");
                }
            }
        }

        function safeTransferAndCancelGO(
            address from,
            address to,
            uint256 tokenId,
            uint256 amount,
            bytes calldata data,
            bytes calldata cancelData
        ) external {
            GOCertificate memory go = theGOStorage[tokenId];
            _isValid(go.issuingBody, go.validationFunction);
            if (from != to) {
                super.safeTransferFrom(from, to, tokenId, amount, data);
            } else {
                require(
                    from == _msgSender() || isApprovedForAll(from, _msgSender()),
                    "ERC1155: caller is not owner nor approved"
                );
            }
```



```
        super._burn(to, tokenId, amount);
        canceledGOs[tokenId][to] = canceledGOs[tokenId][to].add(amount);
        emit OneGOCanceled(
            from,
            to,
            tokenId,
            amount,
            go.certificateType,
            cancelData
        );
    }

    function safeBatchTransferAndCancelGO(
        address from,
        address to,
        uint256[] memory tokenIds,
        uint256[] memory tokenAmounts,
        bytes calldata data,
        bytes[] calldata cancelData
    ) external {
        uint256 cancelRequests = tokenIds.length;
        require(
            cancelData.length == cancelRequests,
            "safeBatchTransferAndCancelGO: Cancel data must equal cancel requests"
        );
        if (from != to) {
            super.safeBatchTransferFrom(from, to, tokenIds, tokenAmounts, data);
        } else {
            require(
                from == _msgSender() || isApprovedForAll(from, _msgSender()),
                "ERC1155: caller is not owner nor approved"
            );
        }
        int256[] memory certificateTypes = new int256[](cancelRequests);
        for (uint256 i = 0; i < cancelRequests; i++) {
            canceledGOs[tokenIds[i]][to] = canceledGOs[tokenIds[i]][to].add(
                tokenAmounts[i]
            );
            GOCertificate memory go = theGOStorage[tokenIds[i]];
            certificateTypes[i] = go.certificateType;
        }

        super._burnBatch(to, tokenIds, tokenAmounts);
        emit BatchGOCanceled(
            from,
            to,
            tokenIds,
            tokenAmounts,
            certificateTypes,
            cancelData
        );
    }

    function addCertifiedIssuer(address issuer) external onlyOwner() {
        require(
            issuer != address(0),
            "Can not add zero address as certified issuer"
```



```
        );
        certifiedIssuers[issuer] = true;
        emit IssuingBodyAdded(issuer);
    }

    function removeCertifiedIssuer(address issuer) external onlyOwner() {
        require(
            issuer != address(0),
            "Can not remove zero address as certified issuer"
        );
        certifiedIssuers[issuer] = false;
        emit IssuingBodyRemoved(issuer);
    }

    function registerForIssuingBody(address registrant) public onlyCertifiedIssuers(msg.sender){
        userToIssuingBody[registrant] = msg.sender;
    }

    function _generateGOToken(
        address certificateIssuer,
        uint256 initialSupply,
        bytes memory data
    ) private returns (uint256 id) {
        uint256 certificateId = ++tokenNonce;
        certificateIssuers[certificateId] = certificateIssuer;
        super._mint(certificateIssuer, certificateId, initialSupply, data);
        return certificateId;
    }

    function _isValid(address sender, bytes memory validationFunction)
        internal
        view
    {
        if (_isOfTypeContract(sender)) {
            (bool success, bytes memory result) =
                sender.staticcall(validationFunction);

            require(
                success && abi.decode(result, (bool)),
                "IB does not validate request"
            );
        }
    }

    function _isOfTypeContract(address theAddress) private view returns (bool) {
        uint256 size;
        assembly {
            size := extcodesize(theAddress)
        }
        return size > 0;
    }

    function _isCertifiedIssuer(address theSender) private view returns (bool){
        return certifiedIssuers[theSender];
    }
```



```
        function _doesFromTradeWithTo(address from, address to)
            private
            view
            returns (bool)
        {
            require(
                _isOfTypeContract(from),
                "not address"
            );
            require(
                _isOfTypeContract(to),
                "not address"
            );
            bytes memory validateFunc =
                abi.encodeWithSignature("isTradingWith(address)", from);
            (bool success, bytes memory result) = to.staticcall(validateFunc);
            require(
                success,
                "a and b do not trade"
            );
            return  abi.decode(result, (bool));
        }
    }
```

University of Hradec Králové
Faculty of Informatics and Management
Academic year: 2020/2021

Study programme: Information Management
Form of study: Full-time
Specialization/combination: Information Management (im2-p-an)

Document for registration DIPLOMA THESIS

Name and surname:	**BSc. Amar Tiric**
Personal number:	**I1900794**
Address:	Zije Dizdarevica 12B, Zenica, 72 000 Zenica, Bosna a Hercegovina
Work topic:	Využití technologie blockchain pro záruky původu
Work topic in English:	Use of Blockchain for Guarantees of Origin
Supervisor:	Ing. Tereza Otčenášková, BA, Ph.D. Department of Information Technologies

Theses guidelines:

The master thesis aims to create a prototype (or Proof of Concept) of Guarantees of
Origin agenda with the use of Blockchain technology. In addition, it provides rationale for using Blockchain technology for the Guarantees of Origin scheme as well as for potential drawbacks of such an application.
The thesis outline is as follows:

- Introduction
- Objective and Methodology
- Overview of Guarantees of Origin scheme
- Blockchain Technology and potential use case for Guarantees of Origin
- Selection of Blockchain technology and its Framework
- Implementation of a Prototype / Proof of Concept
- Results and Discussion
- Conclusion
- Bibliography
- Appendices

Recommended resources:

- Alt, R., Wende, E. Blockchain technology in energy markets – An interview with the European Energy Exchange. Electron Markets 30, 325–330 (2020). https://doi.org/10.1007/s12525-020-00423-6
- F. Imbault, M. Swiatek, R. de Beaufort and R. Plana, "The green blockchain: Managing decentralized energy production and consumption," 2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe), Milan, 2017, pp. 1-5, doi: 10.1109/EEEIC.2017.7977613.
- Zimberg, Dean & Spinnell, Jonathan. (2018). Renewable Energy Certificate Markets: Blockchain Applied. 10.13140/RG.2.2.26140.64643.
- K., Asma & Verma, Piyush & Southernwood, Joanna & Massey, Beth & Corcoran, Peter. (2019). Blockchain in Energy Efficiency: Potential Applications and Benefits. Energies.
- Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf