

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Department of Software Science

Francisco Javier Ortín Cervera 204066 IAPM

**PROOF OF FEEDBACK: A NOVEL BLOCKCHAIN
PROTOCOL FOR REWARDING VALUABLE FEEDBACK**

Master Thesis

Academic Supervisor

Innar Liiv

PhD

Tallinn 2022

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Francisco Javier Ortín Cervera
(signature)

Date: May 10, 2022

Annotatsioon

Tagasiside on alati vajalik, kuid mitte alati kätte saadud, eriti kui seda ei nõuta. Aga sageli on see väga väärtuslik, eriti kui tegemist on spontaanse ja aktiivse tagasisidega. See võib aidata igal üksusel ja ühiskonnal tervikuna kasvada ning seetõttu on mõistlik seda premeerida. Siin esitatud töös pakutakse välja uus plokiahela protokoll nimega Tagasiside tõendus, mida kliendid saavad kasutada tagasiside andmiseks, mida hindavad mitmed kohtunid ja selle eest kliendid saavad premeeritud.

Selles lõputöös on nutilepingute abiga loodud prototüüp EVM plokiahelas ja välja on töötatud DApp, et kasutajad saaksid protokolliga suhelda. Siin esitatud töö valideerib ka kavandatud protokolliga ja liidese, viies läbi auditi peamiselt nutilepingute kohta, mis näitavad, et sellist süsteemi on võimalik luua, ning saadud väärtus ületab oluliselt tegevuskulusid.

Esitatud analüüsis tuuakse välja, kuidas üldine süsteem ei ole täielikult detsentraliseeritud, kuna selle mõned osad tuleb hoida privaatsena, kuid mainitakse ka selles töös kasutatavaid läbipaistvuse, jälgitavuse ja tokeniseerimise võimalusi, mis teevad selle siiski väga väärtuslikuks protokolliks. Mõned ettepanekud tulevase arengu kohta, nagu süsteemi reguleeriva DAO rakendamine, on esitatud ka selle lõputöö viimases osas.

Abstract

Feedback is always needed but not always received, especially when it is not required. But often it is very valuable, most notably when it is spontaneous and active feedback. This can help any entity and the society as a whole grow, and that is why it makes sense to reward it. The work here presented proposes a new blockchain protocol called the Proof of Feedback, which customers can use to provide feedback that is evaluated by a set of jurors and rewarded.

In this master thesis a prototype has been build with the help of smart contracts in an EVM blockchain and a DApp has been developed for the users to interact with the protocol. This thesis also validates the proposed protocol and interface by conducting an audit mainly on the smart contracts showing that is possible to create such system and the value generated greatly surpasses the operational cost.

In the analysis presented is pointed out how the overall system is not totally decentralized since some parts of it must be kept private, but is also mentioned the transparency, traceability and tokenization possibilities used in this work that still make it a very valuable protocol. Some proposals for future development like the implementation of a DAO governing the system are also presented at the end of this thesis.

List of abbreviations and terms

| | |
|-------|---------------------------------------|
| API | Application Programming Interface |
| BSC | Binance Smart Chain |
| DAG | Direct Acyclic Graph |
| DAO | Decentralized Autonomous Organization |
| DApp | Decentralized Application |
| DDoS | Distributed Denial of Service |
| DB | Database |
| DONs | Decentralized Oracle Networks |
| DPoS | Delegated Proof of Stake |
| EVM | Ethereum Virtual Machine |
| EOA | Externally Owned Account |
| ERC | Ethereum Request for Comment |
| EUR | Euro |
| ICO | Initial Coin Offering |
| IoT | Internet of Things |
| IPFS | InterPlanetary File System |
| JSON | JavaScript Object Notation |
| JWT | JSON web token |
| KYC | Know Your Client |
| NFT | Non-Fungible Token |
| NPS | Net Promoter Score |
| PBFT | Practical Byzantine Fault Tolerance |
| PoA | Proof of Authority |
| PoF | Proof of Feedback |
| PoH | Proof of History |
| PoRep | Proof of Replication |
| PoS | Proof of Stake |
| PoSA | Proof of Stake Authority |
| PoW | Proof of Work |
| SSL | Secure Socket Layer |
| UI | User Interface |

| | |
|-------|--|
| UK | United Kingdom |
| UKCSI | United Kingdom Customer Satisfaction Index |
| USD | United States Dollar |

Table of Contents

| | |
|--|-------------|
| List of Figures | viii |
| List of Tables | ix |
| 1 Introduction | 1 |
| 2 Theoretical background | 3 |
| 2.1 Customer feedback studies | 3 |
| 2.1.1 Types of feedback | 4 |
| 2.1.2 How to define valuable feedback | 6 |
| 2.2 Blockchain | 7 |
| 2.2.1 Core concepts | 8 |
| 2.2.2 Consensus protocols and mechanisms; mining and other blockchain protocols | 10 |
| 2.2.3 Types of blockchains | 13 |
| 2.2.4 Blockchain issues | 14 |
| 2.3 Smart contracts | 14 |
| 2.3.1 EVM compatible blockchains | 15 |
| 2.3.2 ERC-20 tokens | 16 |
| 2.4 Oracles | 17 |
| 2.4.1 Types of blockchain oracles | 18 |
| 2.5 Decentralized autonomous organizations (DAO) | 18 |
| 3 Novel blockchain protocol: Proof of Feedback | 20 |
| 3.1 Existing related works | 20 |
| 3.2 Architecture of the protocol | 21 |
| 3.2.1 Original idea | 21 |
| 3.2.2 Possible architectures | 22 |
| 3.2.3 Feedback storage | 26 |
| 3.2.4 Jurors | 26 |
| 3.2.5 Regulator node | 27 |
| 3.2.6 Validators | 27 |
| 3.2.7 Feedback evaluation | 29 |
| 3.2.8 Oracles | 29 |
| 3.2.9 Possibility of a ruling DAO | 30 |
| 3.3 Implementation prototype | 31 |

| | | |
|----------|---|-----------|
| 3.3.1 | Blockchain Polygon | 31 |
| 3.3.2 | PoF contracts | 32 |
| 3.3.3 | PoF client | 35 |
| 3.3.4 | PoF server | 39 |
| 4 | Validation of the proposed system | 41 |
| 4.1 | Smart contracts audit | 41 |
| 4.1.1 | Gas costs analysis | 41 |
| 4.1.2 | Contract vulnerabilities | 43 |
| 4.1.3 | Platform security flaws | 47 |
| 4.2 | Decentralization analysis | 47 |
| 4.2.1 | Preserving feedback requirements | 48 |
| 4.3 | Possible improvements | 48 |
| 5 | Conclusions | 50 |
| | References | 52 |
| | Appendices | 64 |
| | Appendix 1 - "Proof of Feedback" contracts | 65 |

List of Figures

| | | |
|----|---|----|
| 1 | <i>PoF original concept from the Easy Feedback Token whitepaper [1]</i> . . . | 21 |
| 2 | <i>PoF architecture model 1.</i> | 23 |
| 3 | <i>PoF architecture model 2.</i> | 24 |
| 4 | <i>PoF architecture model 3.</i> | 25 |
| 5 | <i>PoF feedback registration smart contract call</i> | 32 |
| 6 | <i>PoF feedback evaluation smart contract calls</i> | 33 |
| 7 | <i>PoF random jurors draw algorithm</i> | 34 |
| 8 | <i>PoF interface. Landing page</i> | 35 |
| 9 | <i>PoF interface. Customer view for giving feedback</i> | 36 |
| 10 | <i>Flow chart for the feedback registration process</i> | 37 |
| 11 | <i>PoF interface. Juror view while evaluating evaluating feedback</i> | 38 |
| 12 | <i>PoF interface. Feedback evaluation data flow chart</i> | 39 |
| 13 | <i>PoF reentrancy vulnerability point</i> | 44 |

List of Tables

| | | |
|---|--|----|
| 1 | <i>Table showing corresponding consensus mechanism of the main blockchains</i> | 12 |
| 2 | <i>Main units of ether</i> | 16 |
| 3 | <i>Table showing the corresponding evaluation grades</i> | 29 |
| 4 | <i>Table the gas cost of the different smart contract functions</i> | 42 |
| 5 | <i>April 2021 - April 2022 historical MATIC price</i> | 42 |
| 6 | <i>Future cost estimation in euros</i> | 43 |
| 7 | <i>Generated value by feedback getting rewarded and jurors evaluating feedback</i> | 43 |
| 8 | <i>Relation of generated value versus cost</i> | 43 |

1. Introduction

Feedback is something that is always needed in any organization that wants to advance and serve well the society by fulfilling its mission. Feedback can be very valuable, depending on whether it is well-founded and whether it helps the feedback recipient get better. There are already several solutions in the business world to manage specific feedback such as order process evaluation, product reviews, etc. There is also other feedback that is not so easy to measure but that can be even more valuable than product feedback, especially when it comes to a company's reputation or business model.

Since feedback can have great value in many different aspects, it makes sense to reward feedback that can help entities grow their business or some other kind of activity and serve the society in a better way. But to do so using traditional economic tools and without an interest of conflicts is very difficult. For this purpose, a decentralized protocol called "Proof of Feedback" is designed to work in a decentralized manner with the help of blockchain technologies. This original concept was firstly devised by Easy Feedback Token OÜ, and sketched in its whitepaper [1], but research is needed to prove that it can be implemented and is worth doing it in the blockchain. The author of this master's thesis is a board member of the company. This work is meant to explore and research the technical possibilities of such protocol, and how the sketch idea can be implemented.

This master's thesis aims to answer the following questions:

- Is it possible to develop a protocol for rewarding valuable spontaneous feedback using blockchain technologies? Is it worth operating such system?
- Is it possible to create a decentralized application (DApp) that allows users getting rewarded?
- Could this protocol be ruled and maintained by a decentralized autonomous organization (DAO)? What kind of functions could it assume and how could it be governed?

The Proof of Feedback would be a protocol where customers can send feedback to a company in a private and non-anonymous manner and then get rewarded for it. For that a court of jurors evaluating the feedbacks needs to be designated and operate in a transparent

and external way to judge what is valuable feedback that needs to be rewarded.

Decentralization of such a rewarding system can help prevent conflict and promote transparency while at the same time fulfil one of the main purposes of blockchain. Also, deleting the middleman (entities managing all that process of rewarding economically) can help make it cheaper and faster to process. But for that this system needs to be feasible and this protocol idea needs research in various aspects, such as how feedback is transmitted from clients to oracles, whether that transfer is done in or out of the blockchain; there is also a need for research on how to use blockchain oracles that supply smart contracts with data from the outside world.

This master's thesis is meant to explore the possibilities of using these concepts involved in Web 3 for tokenizing processes that previously would have been impossible to be monetized. This explores further uses of blockchain technology other than decentralized finance (DeFi), that has become so popular in the recent years. The innovation of the blockchain and the value of feedback join in this work.

The outline of this work starts with an overview of the theoretical background about feedback, blockchain, smart contracts and oracles. The third chapter presents the novel blockchain protocol Proof of Feedback and a prototype developed to test it out. Then in the fourth chapter are carried out different analysis of the protocol and the prototype to validate the concept and its viability. And the thesis finishes with conclusions from the presented protocol and prototype along with its validation process.

2. Theoretical background

In this chapter has been compiled a review of the available literature regarding the different building blocks of this master's thesis. Starting from customer feedback in section 2.1 as the base point, following to the main base for all the technology involved in section 2.2 about blockchain, and going more specific into the different elements of the blockchain that can make a feedback rewarding system possible: the building blocks of interaction, smart contracts in section 2.3; the connections between the distributed system and the real-world data, oracles in section 2.4; and the possibility of a more independent organization on top of all that system, a decentralized autonomous organization (DAO) in section 2.5.

2.1 Customer feedback studies

Researching the available literature there are many different definitions of feedback, mainly depending on to whom it is addressed. The following definition gives an overall integrating view of what feedback is: "Feedback is personalized information based on direct observation crafted and delivered so receivers can use the information to achieve their best potential." [2].

One of the main ideas that motivates the development of this work is that feedback gives value to the entity receiving the feedback as well as to the society. The better the person receiving the feedback can act/perform/operate/serve the more it contributes to the overall good and growth of the society.

Different kinds of feedback have been researched and will be taken on account since they can be extrapolated, but this work will focus on customer feedback that is the main object of value to be rewarded.

One thing is clear, nowadays there is a common agreement on feedback playing a key role in the growth and prosperity of any company. And there are multiple sources of feedback as well as multiple kinds, each one has its nuances.

Already since the early 2000's it was clear that the customer service is a very powerful tool in the business world [3] and that starts from feedback.

Feedback is something that is always needed, even with approaches like having product owners advocating for the customers need that is not enough, research shows that up to half the features in IT products are never used [4]. At the same time there might be some features that a client would appreciate very much but they are never developed due to the lack of active communication between the client and the service provider. This is another motive for incentivising feedback.

The UK Customer Satisfaction Index (UKCSI) shows that the numbers of customers facing a problem is not even constant but has increased in the last year [5]. If anyone were to think there is a point where there are no new problems, well this report proves it wrong at least for the present time.

Another reason that shows feedback is always needed is the open loop problem [6], referring to the challenges for product management to receive accurate customer feedback to use as a basis in their decision-making processes.

Like Thomson said in its 2005 article "It costs more to retain a customer than to get a new one." [7]. Knowing what the customer thinks and needs is crucial for the growth of a company, this is one of the reasons why feedback is so valuable.

2.1.1 Types of feedback

We need to distinguish between different types of feedback depending on who is receiving and who is giving the feedback.

- Employees ↔ employees
- Employee ↔ employer
- Customer → company
- Teacher ↔ student
- Student ↔ student

As previously mentioned this work is focused on the feedback that is given by a customer to a company (what will be called customer feedback or client feedback).

Within this category of customer feedback, when we focus on the consumer company type there are also different types of feedback:

- *Complaints*, for instance where there has been some problem in the provision of a service, or a product does not fulfill the expectations of the customer.

- *Claims*, very similar to the complaints, some would call it a kind of complaint which regards a problem with ownership or lack of it, as well as deceptive services or products. Sometimes it requires legal advice to be solved.
- *Suggestions*, when a client has a proposal to improve a product or service.
- *Congratulations*, the client feels the need to congratulate the company for some product or service that has been really successful and that way inform that is a good path to keep in the future.
- *Queries*, often the customer does not know what is the intended use of a feature, or some part of a process is not clear. Feedback in this cases is about making some information more clear.
- *Requirements* [8], this is very often in software development when client and provider work tight together, or where often the pipelines depend directly on the requirements from the customer. This kind of feedback is usually an integral part of the product development, but it can also be regarded as customer feedback.

The first two types (complaints and claims) will be called *reactive feedback* while the other 4 (suggestions, congratulations, queries and requirements) will be designated as *active feedback*.

It can be said that the previously mentioned types of feedback would fall in the category of what is spontaneous feedback most of the time, though it could be part of programmatic surveys.

In an article from 2019, Onorel [9] proposes different the types of feedback dividing them in three categories:

1. *Informal feedback*, a superior, a subordinate, a customer, or a supplier may provide informal comments on working performance. Indeed, regular observation of how others behave when interacting with them is crucial to feedback. This necessitates direct engagement within the workplace.
2. *Formal feedback*, after a longer period of observing employee behavior, a performance appraisal processor, and workplace surveys, formal feedback is given.
3. *360° feedback*. Though this one might be considered part of the formal feedback. In this type of feedback, data is gathered from a variety of people involved in the process, and the receiver's attention is focused on its involvement as seen from many angles.

But feedback can also be classified as:

- **Structured feedback:** quantitative, qualitative (free text, questions). It comes from the active initiative of the one receiving the feedback, Kurt Schneider calls it *pulled* feedback [8] (from the article: “Response rates are typically low.”). Then here it can be programmatic, as something that is asked over time or focused on a given product or service that has just been provided.
- **Spontaneous feedback:** from the initiative of the consumer, or as Kurt Schneider would say *pushed feedback*. “No external motivation is involved. Usually, end-users do not bother to write a letter or email hours after the encounter.” [8].

Another division in types of feedback can be publicly available and private feedback. The first refers to feedback that anyone can read, either because it is published in the company’s website or because it is in any kind of blog or web for reviews.

They have different effects on the receiver of the feedback depending on how the feedback is written. We know from research that the chance of sales is influenced by seller ratings in sites like Ebay [10] and that it also has an important impact in Amazon book sales [11]. We also know that longer feedback usually is correlated with its quality [12] as well as it influences in the sales [11].

This master’s thesis is focused on the spontaneous customer feedback, even though the developed prototype can be also used in structured feedback, but would still have to be evaluated with the purpose of assessing the value that is giving to the receiver and the society.

2.1.2 How to define valuable feedback

Not all feedback is good or equally valuable. There are many research papers talking about the necessity of processing feedback [2, 13, 4] to get something valuable from it. As it can be seen in the Gerdes, Stringam and Brookshire article [14] many attempts have been made to develop an algorithm to assess quantitative and qualitative feedback, specifically text.

That is why, specially at the beginning of such a rewarding system, is very important the human evaluation by experts in the field regarding the feedback. Professionals that can assess how good it is. In the second chapter 3.2.7 would be proposed an standard system for the evaluation of feedback.

There exists one unofficial standard that some might claim it defines what is good feedback, if for "good" we take that which provokes economical growth. That is the Net Promoter Score.

Net Promoter Score (NPS)

The Net Promoter Score is a metric that assesses customer satisfaction and predicts business success. This tried-and-true statistic revolutionized the corporate world and is now the gold standard for customer experience management programs all around the world.

It's has been proclaimed since its introduction in 2003 as "the one number you need to grow" [15]. NPS can be even successfully used to measure social performance too[16].

But it also has lots of critics specially in the research field [17, 18]. It could be said that the main critic is that is not good feedback, since it does not help improve ones performance, but is just an indicator of how the entity is doing in its market. And it can be criticized citing Goodheart's law: "When a measure becomes a target, it ceases to be a good measure"[19].

2.2 Blockchain

Blockchain is referred as the technology that enables the user interaction with different applications and systems in a decentralized way by means of recording those interactions (generally called transactions) in a distributed ledger [20]. That information is kept untampered and unanimous by nodes in the network with the help of cryptographic hashes for asserting the authenticity of the data and a consensus mechanism to agree on the state of the blockchain at any time. This makes it tamper evident and tamper resistant [21].

One very important remark is the need to understand that there are many different blockchains, there is not such a thing as "the blockchain". Generally, every time blockchain is mentioned it refers to the distributed ledger when using verbs such as: storing, present in the blockchain, uploading to the blockchain; and to the network of nodes when talking about interacting with the blockchain.

Blockchain technology enhances transparency, traceability and accountability [22, 23, 24, 25] of different kinds of data publicly stored in a blockchain. It is a technology that falls in the category of distributed systems, and has many different products and other technologies built on top of it. That is why it needs to be defined what are the key components of blockchain, so it can be differentiated from other distributed technologies that, by being used around blockchain, might also be mistaken as it.

Blockchain is a trustless alternative to centralized systems (like classical banking, API service providers...) providing a system that instead of trust requires confidence in the technological robustness and integrity [26]. That is why it has also been said that "blockchains

are an instance of institutional evolution” [20].

Blockchain is notorious for its use in cryptocurrencies and different financial services related to it. But the previously mentioned features are all important and crucial features for possible applications in many other fields. When researching about these different use cases, applications and implementations of blockchain there are mainly 3 fields that have been extensively explored:

1. **Healthcare** [27] For the personal control of the patients data in a decentralized and secure way [28]. This field of research often is intertwined with IoT [29, 28] which is discussed in the next point.
2. **Internet of Things (IoT)** [30, 31, 32, 33] . Even connecting IoT and DeFi [34]. As it happened with the healthcare field, IoT is also used in supply chain systems which is the next big field [35].
3. **Supply chain**, several alternatives to the traditional tracking systems have been proposed using blockchain that help solve trust problems [36]. But there is still a lot of research needed on the adoption of it in this field [37].

Another topic that has been arising in the last year is the possibility of decentralized governance, especially in what are so called Decentralized Autonomous Organizations (DAO) that will be covered in section 2.5. But also in traditional institutions. There are studies comparing the past and current governing systems as well as future possibilities [25, 38].

Privacy is also a concern in blockchain technology and a whole set of technology tools and methods have arisen around that topic [39].

Some of the main blockchains [40] are: Bitcoin[41], Ethereum[42], IOTA[43] is intensively used for IoT systems[44], Polygon, Avalanche, Hyperledger, Tezos, Ripple, Litecoin.

2.2.1 Core concepts

There are many blockchains that are built up using different components. But there are certain concepts that are common to most of them, and therefore permit us to talk about the following core concepts:

- **Cryptographic hash functions:** they perform what is called *hashing*, a way of calculating a considerably unique output (called a message *digest*, or simply *digest*) given an input of practically any size (e.g., a file, text, or image) using a cryptographic

hash function . Individuals can separately take input data, hash that data, and derive the identical output, proving that the data has not changed. Even the tiniest modification in the input (such as a single bit) results in a completely different output digest.

- **Addresses:** blockchain networks utilize an address, which is a short alphanumeric string of characters produced from the blockchain network user's public key using a cryptographic hash function, as well as some other data (e.g., version number, checksums). Addresses are typically used as the receiver and sender endpoints of a transaction in most blockchain implementations.
- **Transactions:** it could be said that a transaction is the unit of information that is stored in the blockchain. A transaction is an exchange of information between two or more parties. When dealing with cryptocurrencies, for example, a transaction is a cryptocurrency transfer between blockchain network members. But a transaction can be any interaction between different parties that is recorded in the blockchain, it can be any information transfer between addresses, be those belonging to a person or a smart contract (smart contracts will be discussed in 2.3). Usually transactions need to be paid a fee to be processed.
- **Cryptographic nonce:** An arbitrary number that is only used once. A cryptographic nonce can be coupled with data to generate a variety of hash digests:
$$\text{hash}(\text{data} + \text{nonce}) = \text{digest}$$
- **Public and private key:** users in a blockchain make use of private and public keys to make transactions. With different cryptographic methods private keys are used to make digital signatures that can be later verified when validating transactions in a block using the public key of the user.
- **Signatures:** as mentioned in the previous point signatures are a critical component of blockchains since they are used all the time to verify the origin and validity of transactions.
- **Wallet:** users can record their private keys manually but it is often done with the help of software in a physically own device (called hardware wallets) or with the help of some service provider.
- **Blocks:** Users of the blockchain network submit potential transactions to the network using software. These transactions are sent to one or more nodes within the blockchain network. The submitted transactions are subsequently broadcast to the rest of the network's nodes, although this does not automatically add the transaction to the blockchain. Once a pending transaction has been sent among nodes, it must wait in a queue until it is published within a block (group of transactions) to the blockchain by a publishing node in many blockchain systems.
- **Distributed ledger:** is the main component of the blockchain, the ledger that all the nodes in the blockchain must agree in its state (consensus). Is the copy that all nodes

keep of the transactions made between users of the given blockchain.

- **Nodes:** a blockchain node is one of many devices that run the blockchain protocol software and, in some cases, record transaction history. In a decentralized peer-to-peer network, nodes communicate with one another. There are mainly three types of nodes:
 1. *Lightweight*, a node that does not keep or maintain a copy of the blockchain and must rely on full nodes to process transactions.
 2. *Full-node*, contains all current state of the blockchain, all the blocks that have been registered in the distributed ledger. Sends current information to other nodes, and verifies the validity and authenticity of freshly added blocks.
 3. *Archive node*, is like a full node but with all information but since the genesis block (first block of the blockchain).
- **Consensus mechanism:** is the main mechanism that nodes in a blockchain use to agree on the validity and the current status of the blockchain. It will be discussed more in detail in in the next subsection 2.2.2.

2.2.2 Consensus protocols and mechanisms; mining and other blockchain protocols

It will be discussed in this subsection how there are certain misleading terms around consensus protocol and consensus mechanism. When talking about blockchain consensus mechanisms Aggarwal and Kumar give the following definition: "A consensus mechanism is a fault-tolerant mechanism used in a blockchain to reach an agreement on a single state of the network among distributed nodes. These are protocols that make sure all nodes are synchronized with each other and agree on transactions, which are legitimate and are added to the blockchain. Their function is to ensure the validity and authenticity of the transactions." [45].

But there is also a different understanding that will be referred as consensus protocol within which we can include the consensus algorithm that is used during the mining process. For instance the consensus protocol used in Bitcoin as explain below.

Nakamoto consensus protocol

The Nakamoto consensus protocol can be summarized by the following rules, which correspond to the five components of a blockchain consensus protocol [46]:

1. *Proof of Work (PoW)*: for generating blocks there requires finding a preimage to a hash function so that hashing a block produces a hash satisfying a given difficulty

(for instance producing a hash starting with 8 zeros). This difficulty is updated every 2016 blocks depending on the network power so that the average block mining time stays around 10 minutes.

2. *Gossiping Rule*: whenever a node receives or generates a new transaction it needs to be broadcasted to the rest of the nodes in the network immediately.
3. *Validation Rule*: all blocks and transactions need to be validated before applying the gossiping rule. Several things need to be validated: digital signatures of the transaction, double-spending check, correctness of the transaction (e.g.: does the wallet have enough coins to send).
4. *Longest-Chain Rule*: when there is more than one chain generated by the process of different nodes adding transactions, the network takes as valid the longest one.
5. *Block Rewards and Transaction Fees*: in the form of a coinbase transaction to itself, the block generator can claim a certain amount of new tokens plus fees collected from all enclosed transactions.

If we deem Proof of Work as a consensus mechanism, is still part of the bigger picture here indicated as consensus protocol. Consensus mechanism are often referred as consensus algorithms too.

Main consensus mechanisms

Most blockchains have similar rules for their consensus protocols, and here are presented the consensus mechanisms used in the main blockchains (Bitcoin, Ethereum , Binance chains, Polygon...):

- *Proof of Work (PoW)*, that has been described above.
- *Proof of Stake (PoS)*, there is a group of so called validators in the network that stake the native token of the blockchain (there is a minimum required). Then these validators are randomly selected to create and mine a block. They also have the duty of checking and validating blocks created by other validators [47].
- *Delegated Proof of Stake (DPoS)*, is similar to PoS but here any stakeholder can delegate his/her voting power to a validator by staking tokens. Instead of constructing blocks themselves, the stakeholders grant the right to create blocks to the delegates they support, decreasing their computational power usage to zero [48].
- *Proof of Stake Authority (PoSA)*, is a combination of PoS and Proof of Authority (PoA). In this mechanism there is a set of validators previously selected (PoA) and then those validators stake their tokens to get elected for mining blocks (PoS).
- *Direct Acyclic Graph (DAG)*, Tangle, Any new arriving transaction in Tangle could join the blockchain network as a new vertex (or tip) as soon as they approve a number

of unconfirmed transactions (typically two with the random selection). When the cumulative weight (the total of its own weight and the weights of other transactions) reaches the predefined threshold, confirmation is obtained [49].

There are many more: Proof of Space [50], Proof of Elapsed Time [51], Practical Byzantine Fault Tolerance (PBFT) [52], but it is not possible to cover all of them in this work so only the most important ones are described, they are also presented in table 1.

Table 1. *Table showing corresponding consensus mechanism of the main blockchains*

| Blockchain | Consensus mechanism | Source |
|-------------------|--|--|
| Bitcoin | Proof of Work | Original paper [53] |
| Ethereum | Moving from PoW to PoS | Official documenta- tion[47] |
| IOTA | Tangle with the helpt of Direct Acyclic Graph | Official documenta- tion[54] |
| Polygon | Proof of Stake | Official documenta- tion[55] |
| BinanceSmartChain | Proof of Stake Authority | Official documenta- tion[56] |
| Avalanche | Snowball Algorithm (with the help of DAGs) | Official documenta- tion[57] |
| Solana | Proof of History (PoH) but it is also combined with PoS and the Proof of Replication (PoRep) | Solana whitepaper: "The combination of PoRep and PoH provides a defense against forgery of the ledger with respect to time (ordering) and storage." [58] |

Blockchain protocols

There is an extended understanding to consensus mechanisms. There are blockchain projects that used the noun "proof" to start naming their protocols that need different parties agreement on some part of their project. The author of this thesis has found appropriate to name them blockchain protocols, differentiating them from real blockchain consensus mechanism on the whole network level. Examples of this kind of protocols are: Proof of Disease [59], Proof of humanity [60] not to be mistaken with the Kleros one [61], Proof of Ownership [62] and many more that are constantly created. These type of

"consensus" protocols are deployed in a blockchain (or in several different blockchains, what could make them cross-chain protocols), they are not natively part of the blockchain.

2.2.3 Types of blockchains

Permissionless and *permissioned* approaches to blockchain have been characterized as two general high-level categories. Anyone can read and write to the blockchain without authorization in a *permissionless* blockchain network. *Permissioned* blockchain networks restrict participation to specified individuals or organizations and provide finer control. Understanding the variations between these two categories allows any entity to determine which subset of blockchain technologies is best suited to its requirements, when considering its adoption [21].

There is a more general categorization of blockchains:

- **Permissionless blockchains:**

- *Public blockchains*, anyone can communicate with another transacting party using public or open blockchains. The two parties' identities are either pseudonymous or completely anonymous. An open blockchain indicates that transactions have little to no privacy depending on the blockchain, as all participants can see all transactions. An open blockchain also necessitates a significant amount of computational resources to operate a large-scale distributed ledger [63].
- *Side blockchains*, a sidechain is a secondary blockchain that is linked to the main chain by a two-way peg. They are often developed to help solving certain blockchain problems like scalability, security, privacy, etc [64]. They can be considered a first step towards cross-chains.
- *Cross-chains*, these are specific blockchains build on top of their blockchains to interconnect them. They might be referred as layer 2 blockchains and they are used to access different blockchains (layer 1) in a decentralized manner. Cross-chain refers to the application of certain technologies to allow value to pass through barriers between chains, allowing value to be transferred from one blockchain to another, allowing for value circulation. The data synchronization between the ledgers must ensure that the changes in the two ledgers are consistent; otherwise, problems such as double payment or value loss would arise [65].

- **Permissioned blockchains:**

- *Private blockchains*, only pre-validated persons or groups of individuals can access the ledger as well as enter and see data using private or closed blockchain technologies. Others are aware of all users' identities before they transact [63].

- Quorum and Hyperledger projects are examples of private blockchains [66].
- *Consortium blockchains*, are a semi-decentralized kind in which the blockchain network is managed by multiple organizations. It differs from the private blockchain, which is controlled by a single entity. In this type of blockchain, more than one organization serves as the authority for mining and information exchange. Blockchains are employed in a variety of industries, including banking and government agencies. R3 is an example of a consortium blockchain [52].
 - *Hybrid blockchains*, is a merged solution of public and private blockchain networks. In this situation, the functionality of both blockchains are used (for example, users can choose between a "private permission-based system" and a "public permission-less system"). Users can regulate who has access to which data in the blockchain on the hybrid platform. Only a few of the blockchain's records are allowed to be made public, while the rest are kept hidden in the private network [52].

2.2.4 Blockchain issues

Clearly, there is a need to research and develop more blockchain related technologies. By the year 2016, even though the idea of blockchain can be claimed to have existed already for almost 30 years [21], research was focused only on Bitcoin [67], leaving many important related topics such as smart contracts and others covered in this master's thesis out of the scope. It is also understandable since the name as we use it today was first coined in the famous Satoshi Nakamoto's paper in 2008 [53].

This technology still presents many challenges and open questions for the research community concerning all aspects: security [32], blockchain is not immune to malicious users [21] and hackers and it has being exploited in different ways; regulations [68, 69, 38]; privacy [39]; the connection with the real-world [44], here oracles present a very important role which will be discussed in section 2.4.

2.3 Smart contracts

Smart contracts are programs deployed in a blockchain that can interact independently with the different participants of the blockchain (wallets, nodes, other smart contracts). They are self-executing [70], tamper resistant, distributed, immutable (once they are deployed they cannot be modified). The idea of smart contracts was originally proposed by Nick Szabo [71].

With the help of a Turing-complete virtual machine referred as Ethereum virtual machine (EVM), Ethereum is the first public blockchain platform to handle complex and customizable smart contracts. Every node in the Ethereum network runs an EVM implementation and executes the same instructions, making EVM the runtime environment for smart contracts [27]. Or as it is described in the official Ethereum documentation: "The EVM's physical instantiation can't be described in the same way that one might point to a cloud or an ocean wave, but it does exist as one single entity maintained by thousands of connected computers running an Ethereum client." [72]. Every Ethereum node runs on the EVM to maintain consensus across the blockchain.

The smart contracts can be written in different programming languages depending on the blockchain. For the EVM the most popular are Solidity and Viper. The contract code is compiled down to EVM bytecode and deployed on the blockchain for execution. Ethereum was the first blockchain to deploy smart contracts on a large scale, and it is now the most popular smart contract development platform. It may be used to create a variety of decentralized apps (DApps), such as digital rights management, crowdfunding, gambling, and so on.

Smart contracts have many advantages but they have to be carefully managed since small mistakes can cause big losses in hacks. During last year 2021, happened the biggest exploit of a smart contract so far where around \$611,000,000 in cryptocurrencies was stolen from Poly Network a cross-chain protocol [73]. Recently an Estonian company called Superfluid got hacked for a sum of 8.7M\$ by exploiting one unchecked parameter given in a function of the smart contracts [74].

2.3.1 EVM compatible blockchains

We have mentioned Ethereum as a blockchain but to be more precise, with that it was meant the Ethereum mainnet. Ethereum in itself is a protocol that is open source. That has allowed many other blockchain projects launch their own EVM with certain adjustments of their own, these are called EVM compatible blockchains, since the core concepts are the same and usually the same smart contract code can be deployed in any compatible chain without any change in the programming.

We can talk about the EVM as a supercomputer build up by thousands of other computers running interconnected as the nodes of a blockchain. Many EVM compatible are considered side-chains of Ethereum (sidechains described in subsection 2.2.3).

When we talk about EVM compatible blockchains we often refer to chains that use forks of

the EVM with small adaptations for their own projects. For instance Polygon already uses PoS as the consensus mechanism for the blockchain. They are often permissionless chains, but since the code is open source nobody can stop permissioned chains to be developed based on an EVM.

This compatibility allows different projects to target specific goals to solve with the help of blockchain and at the same time making the connection with other blockchains easy, allowing fast transfer of projects. They are often cheaper and faster alternatives.

Native token and its units

Ethereum has ether (ETH) as its native cryptocurrency. Ether is used to pay gas fees for the transactions recorded in the blockchain, it is also used in many applications as means of payment. It can be divided in smaller unit as it can have up to 18 decimals. In table 2 are presented the main units of ethers metric system [75].

Table 2. *Main units of ether*

| Unit | Wei Value | Wei |
|----------------------------|------------------|---------------------------|
| wei | 1 wei | 1 |
| Kwei (babbage) | 10^3 wei | 1,000 |
| Mwei (lovelace) | 10^6 wei | 1,000,000 |
| Gwei (shannon) | 10^9 wei | 1,000,000,000 |
| microether (szabo) | 10^{12} wei | 1,000,000,000,000 |
| milliether (finney) | 10^{15} wei | 1,000,000,000,000,000 |
| ether | 10^{18} wei | 1,000,000,000,000,000,000 |

All EVM compatible blockchains have their own native token to fulfill at least the purposes of paying transaction fees and they keep the same division of units as in Ethereum.

2.3.2 ERC-20 tokens

There are other kinds of cryptocurrencies that can be used in blockchains and in EVM compatible blockchains is even an standard that has become very popular for fungible tokens, the ERC-20 [76]. ERC stands for Ethereum Request for Comment and is the prefix for the standards used in the EVM. The Ethereum community reviews these documents called 'Ethereum Improvement Proposal' (EIP). They provide suggestions, and the developer who generated the document may alter it as a result. After working through the EIP process, the Ethereum community accepts some of these documents, finalizes them, and then developers implement them. This is how the document gets transformed into an ERC.

They have been used many times for raising funds [77] with a process called Initial Coin

Offering (ICO). Interested investors can purchase a new cryptocurrency token produced by a blockchain project through the ICO. The issued token may have some utility in relation to the project's product or service, or it may simply represent a stake in a company or project. ERC-20 tokens have been used in many blockchain projects as fungible tokens used to run different protocols by giving monetary incentives (tokens). This is the case of the EASYF used in the protocol proposed in this master's thesis. The code for the EASYF in the appendix one in listing 1.

2.4 Oracles

As previously mentioned one of the blockchain smart contracts main characteristics is the determinism of the programs, the execution of business logic code in a decentralized architecture where all executing nodes trust and agree on the execution outcomes [78].

One of the most significant problems of smart contract applications is the inability to access data outside of the blockchain, this is called the oracle problem. This is because the blockchain is a deterministic system, which means that every node in the network must be able to replay every transaction and smart contract code and get the same outcome.

There might be a case where a smart contracts needs to make a query from an external source like a web service API. While executing the contract at a given time a node might get value x from that query, and at the same time maybe with some milliseconds difference, another node processing the same block and smart contract call might get a different value $x' \neq x$ for whatever reason (data quickly changes, the query did not work...). This difference does not comply with the determinism of the blockchain. If, for example, the value of the EUR/USD asset is requested from the API, as this is highly volatile, these circumstances are quite likely to occur [79].

In simple terms, an oracle is a data supplier. An oracle answers inquiries about the real world through smart contracts. In most circumstances, a smart contract would be unable to know the information it needs to perform its function without the assistance of an oracle, since the only information that has all the time available is the one that resides in the blockchain.

Oracles give the Web3 ecosystem a method to connect to existing data sources, legacy systems, and advanced calculations. A whole new system of what are called decentralized oracle networks (DONs) have been developed to enable the implementation of hybrid smart contracts, in which on-chain code and off-chain infrastructure are coupled to provide complex decentralized applications (dApps) that react to real-world events and interact

with traditional systems [80].

2.4.1 Types of blockchain oracles

Beniiche in his study of blockchain oracles [81] proposes different types of oracles based on the following qualities:

- *Source*: what is the origin of the data. There are several options: it comes from software, hardware or directly from human interaction (in which case, it would still be intermediate by software).
- *Direction of information*: the data is coming into the blockchain (inbound) or is send from the blockchain to an external agent (outbound).
- *Trust*: is the origin of the data centralized or decentralized. Decentralization is regarded as more trustable since it can also avoid small errors that can pass unchecked.

A single oracle can be classified into several of these types. A centralized inbound software oracle, for example, is one that gets information from a company's website.

2.5 Decentralized autonomous organizations (DAO)

According to Singh and Kim "a Decentralized Autonomous Organization (DAO) is an organization that can run on its own predefined protocols without having any hierarchical management. It operates based on the predefined smart contract code such as EVM in Ethereum platform." [82].

There is not much research done yet about how people in DAOs deal with the socio-technical issues that arise from the conflict between pseudonymity while they still need to collaborate and trust one another [83].

In some DAOs, they issue tokens to offer their members governance rights, and blockchain users can trade in these governance tokens to become members of DAOs (with accompanying governance rights). As a result, by making distributed operational or strategic decisions, DAO members can contribute to the organization's goals. On-chain voting can be used to reach a decentralized consensus on these decisions [84].

There are several DAO's that work on a membership based on the ownership of NFT's [85]. There are several big projects based on this mechanism [86, 87, 88]. In these cases the token itself might represent ownership of a share of some treasury [85].

There have been several researches about how DAO can influence different organizations. For instance, in the university of Glasgow, an exploratory DAO was developed for storing students' grades untampered in the blockchain with a rewarding system which turned out to not be suitable according to their observations [89].

On the other hand there are certain applications regarding finance that have proven to be very effective and useful. One example of that is MakerDAO, which is regarded as one of the most advanced and well-known DeFi ecosystems, intends to use over collateralization to develop the Dai stablecoin, which may address many of the theoretical and practical issues that other stablecoins have faced [90].

DAO's are a very typical organization structure for DApps [84] and are in the core concepts of what is called Web3.

3. Novel blockchain protocol: Proof of Feedback

This master's thesis proposes a new blockchain protocol called the Proof of Feedback (PoF). The main idea of this protocol is to reward good feedback that is given from customers to different entities. That feedback needs to be evaluated in an open and transparent manner which is one of the concerns on the viability of this protocol.

The first idea and draft of this protocol was proposed by the Estonian company Easy Feedback Token OÜ in their whitepaper [1] for the ICO of the EASYF, an ERC-20 token deployed in the Polygon blockchain to be the rewarding token of the PoF. The idea in the whitepaper [1] is conceptually drafted and the technicalities on how to implement that protocol as well as what things are possible and which ones not, need to be investigated. The proposal is also open to be changed and adapted to the research results of this master's thesis.

The PoF is a protocol that needs to be regional so, it needs an implementation for each sovereign country. In the subsection 3.2.2 about the architecture of the protocol will be discussed what parts of the protocol are implemented locally. This will be further discussed in subsection 3.2.4.

The main purpose of this work is to design a prototype studying the different technical difficulties and obstacles that might arise during its implementation.

3.1 Existing related works

Looking for similar systems in research papers there has not been found any work related to rewarding feedback with the help of blockchain technology. Nonetheless there have been found systems implementing non-native consensus mechanism like systems that connect the blockchain with end users. Most research papers found with similar works are in the healthcare field [59, 91].

In the healthcare field there has been developed a system interacting with the blockchain with a consensus mechanism called Proof of Disease [59] that aims to solve different

challenges that have not been solve by previous electronic health records and health exchange information systems. It is similar to the this work in some of the requirements, since the Proof of Disease protocol needs to store data using a public ledger with different external parties (oracles) taking part in that process.

In the university of Glasgow was implemented a prototype system in the form of a DAO to grade students with the help of blockchain [89] and there are public rewards depending on those grades. But is hard to know up to what extend that system can be similar to the proposed protocol in this thesis.

3.2 Architecture of the protocol

3.2.1 Original idea

In figure 1 from the whitepaper [1] can be observed a minimal version of the main parts of the Proof of Feedback.

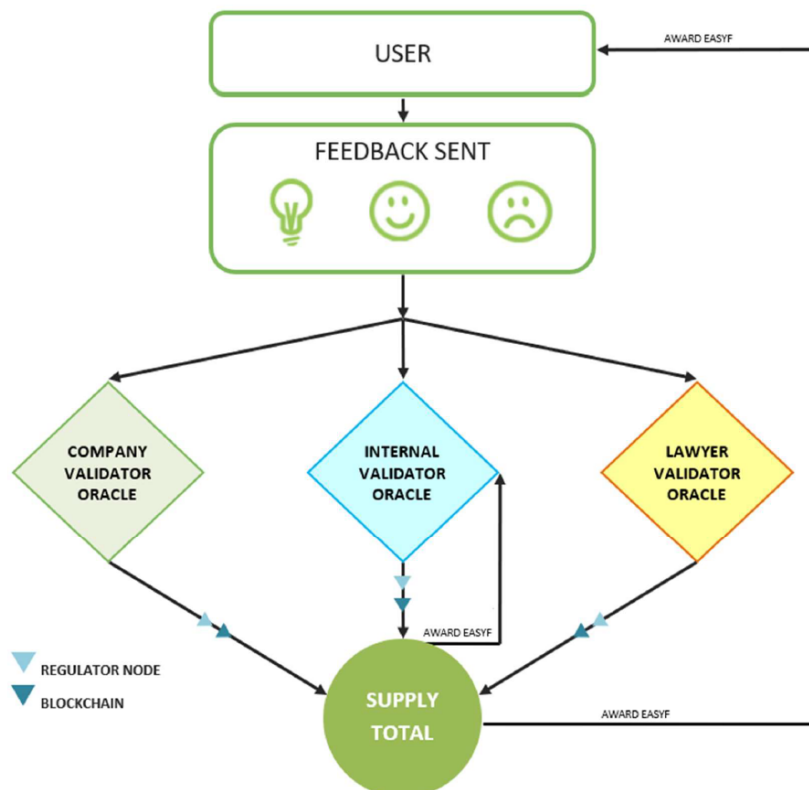


Figure 1. *PoF original concept from the Easy Feedback Token whitepaper [1]*

A user gives feedback through some user interface (UI) and then that feedback is passed to the correspondent validator (Company validator in the case of company registered in a paid service where they evaluate the feedback they have received; Internal validator for a

non-registered company; and the lawyer validator when the feedback is a complain needing legal intervention for solving the case). Those validators are composed of different juries that evaluate the feedback and pass that score to the regulator node that rewards the user and necessary participants of the internal validator.

In this original proposal there is no interaction with the blockchain until the last step. Making it a totally centralized process until the owner of the ERC-20 EASYF token minting contract interacts with the blockchain. This is done by a regulator node that would mint EASYF only in the last step. This work aims to study possible ways to implement such system with the help of blockchain technology. A more decentralized and transparent alternative would be to have all the validators as smart contracts in the blockchain.

In the next subsection 3.2.2 different protocol architecture possibilities will be described and analysed.

3.2.2 Possible architectures

There are mainly three layers with a possible fourth in the Proof of Feedback that will be later discussed in more detail but will be briefly introduced here:

1. **Feedback origin layer:** here is where starts the interaction from the end-user with the PoF by writing feedback to an specific company through the a DApp with a UI for providing feedback (in the original concept shown in figure 1 the first 2 blocks: user and feedback). That feedback is then passed to the next layer.
2. **Evaluation layer:** once the feedback from the user is received and registered it needs to be evaluated by a court of jurors that will decide if the feedback is worth a reward and they will give an evaluation to it.
3. **Rewarding layer:** in this last layer, primarily the customer received the reward in form of EASYF tokens and in some cases the jurors received a reward too.
4. **Minting layer:** In 2 of the proposed models this one is part of the rewarding layer. This level is composed by the contract of the protocol in charge of minting the ERC-20 tokens, either to directly reward a customer and jurors, or to provide liquidity to a contract.

In the aim of decentralizing this process as much as possible and making everything transparent as well as publicly traceable, the evaluation layer would be already in direct interaction with the blockchain by implementing the different validators in smart contracts.

Also giving companies the possibilities to directly mint EASYF tokens would lead to the

possibility of unbalance or totally biased rewarding, since it could be in their own interest to reward more their customers. This could motivate other companies might start doing the same up to a point where either the total supply is fastly minted making the protocol useless and the token losing all its value in the market. That is why it is proposed that the companies that want to reward using the PoF, can deploy their own company node and provide it with some EASYF token liquidity that they might acquire in the market.

PoF architecture 1

In the architecture shown in figure 2 the oracle functions that are specific for each country are concentrated in one regulator node which is unique for each country.

Since the minting of the EASYF can be done only by the owner of the contract and that is only one address, there would be the need to have an intermediary "minter" contract that would own the ERC-20 token contract. Then the different regulator nodes would have access to call this minter contract which, in turn, would mint the necessary tokens.

PoF ARCHITECTURE 1

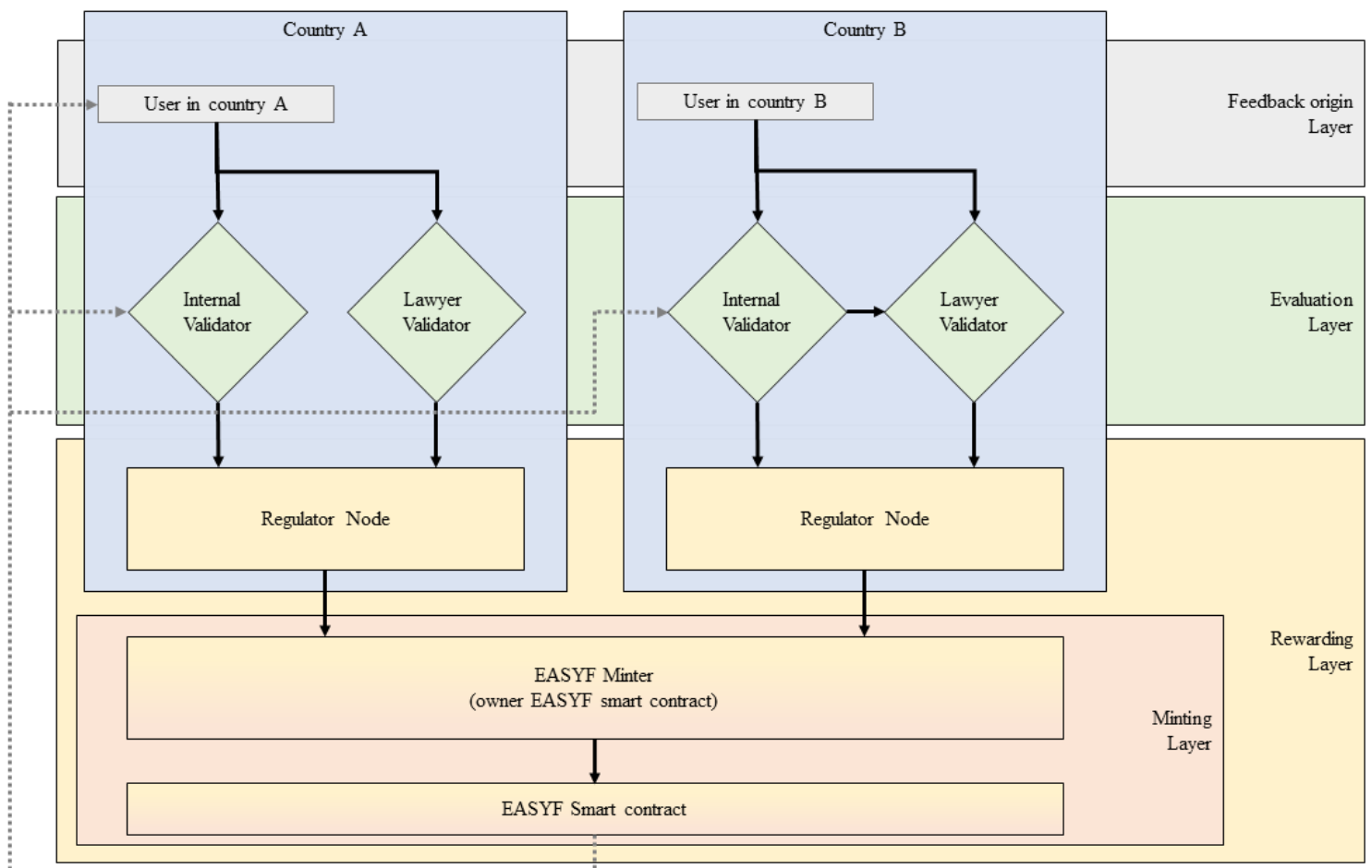


Figure 2. PoF architecture model 1.

PoF architecture 2

In this second proposal the regulator and minter node would be fused in one, decreasing the number of internal participants in the protocol and calls between each other. This has the downside that each internal and lawyer validators need to implement its own oracle calls to get external knowledge about what is the regional purchasing power. The oracle for the price of the EASYF could remain in the regulator node since it should be the same across all the global market.

As it can be seen in figure 3 the rewarding and minting layer are the same in this architecture.

PoF ARCHITECTURE 2

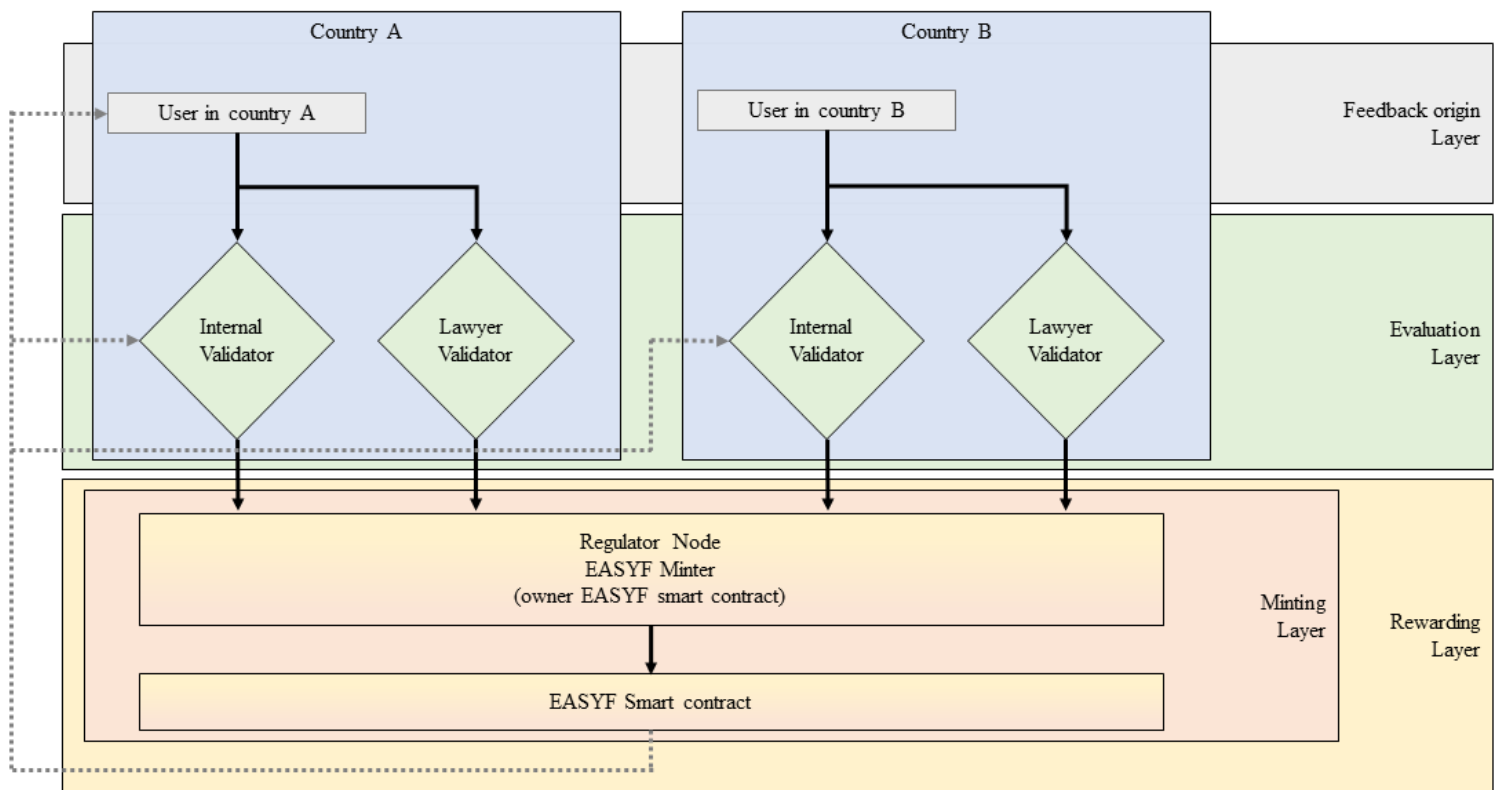


Figure 3. PoF architecture model 2.

PoF architecture 3

The third variant proposed in figure 4 is mostly like the first one in figure 2 but in this one the regulator nodes are provided some liquidity of EASYF and they can directly reward the customers and the jurors with their contract balance.

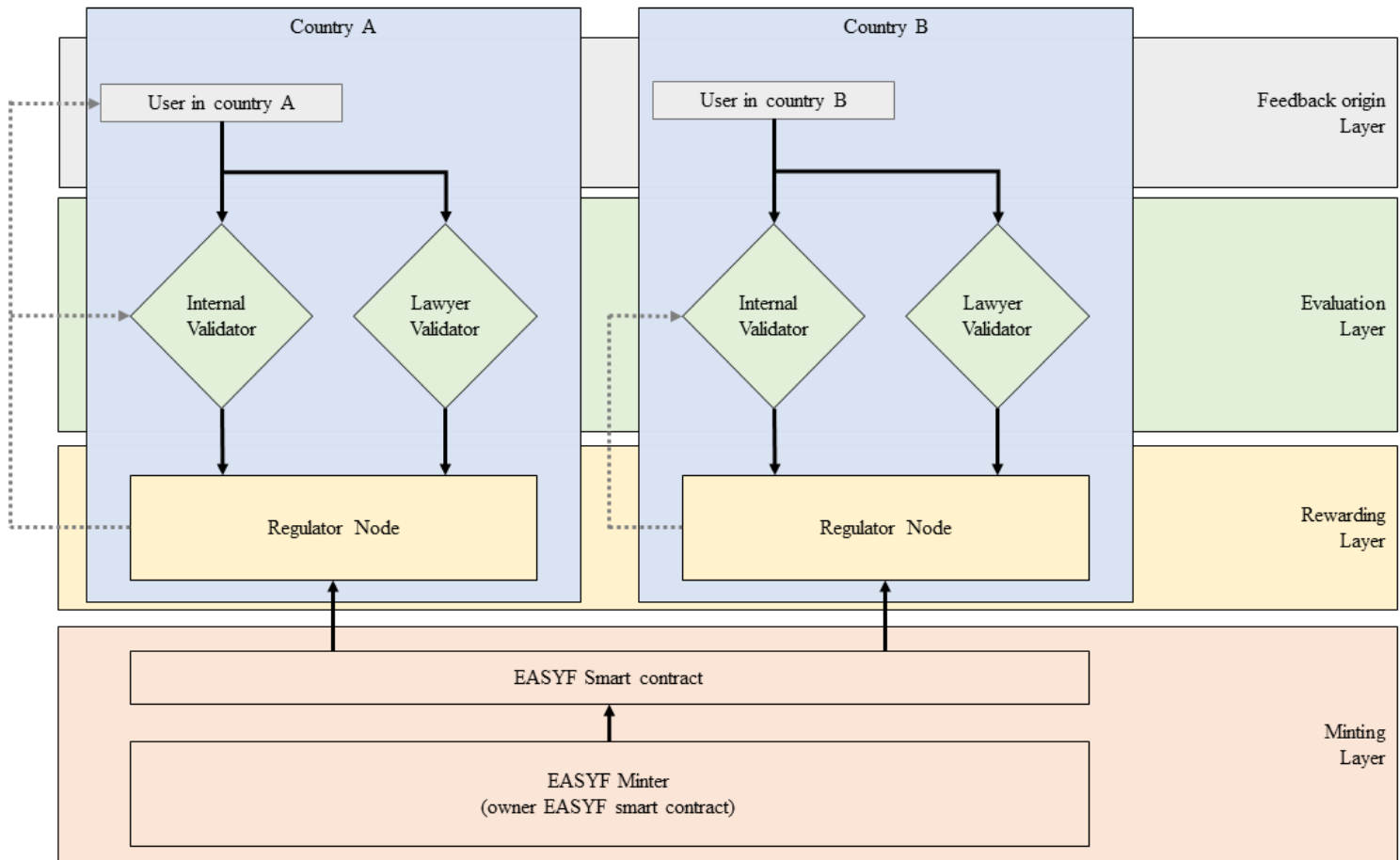


Figure 4. *PoF architecture model 3.*

In the diagram show in figure 4 the minter designates the person or entity owning the EASYF contract that would be the administrator at the beginning but with a possibility to be later passed to a regulating DAO that would decide on the supply of the token.

The main advantage of this architecture is that if the contracts of the validators, the regulator or the minting node were to be hacked, only the liquidity they have at the moment of the hack could be used and not take over all the minting leaving the value of the token to quickly drop due to the control over its supply in the PoF, the main system that gives it meaning. At the same time it has the drawback that at the beginning is still being owned by a single entity (that does not necessarily mean a single failure point, the ownership could be performed with a multi-signature wallet providing increased security) which is not a very decentralized governing mechanism. Even when handing over the ownership of the ERC-20 token contract to a DAO would not be much different security since the DAO is governed by a smart contract that, in turn, can be hacked. This leaves us with the main reason to use this architecture would be due to governing motives, since it could be controlled how much each regional implementation of the PoF can spend.

3.2.3 Feedback storage

The matter on how to store the feedback provided by the customer is very delicate, since it needs to be private but at the same time interact with a system that is public and transparent. How can data be stored in a public ledger but at the same time be private, meaning that only the interested parties can have access to it is the key question here.

We can analyze first who are those interested parties. First of all the customer providing the feedback, then the jurors that must evaluate it and third the company receiving that feedback. The rest of the society should not have access to the content of that feedback.

One possibility to store the feedback in a public but private way is by storing publicly the data encrypted with the public key of the interested parties (that public key would be the wallet addresses), and therefore only those parties could access that data. This data could be stored in the blockchain and stay there forever encrypted, but it has a cost, "Forever isn't free" [92]. Testing it in Polygon testnet the cost of storing 10 000 bytes of data is 297089520 GWei and if the cost of the token is 1.20€/MATIC that means that storing 10 000 bytes of data on-chain would cost 0.3565€. This is clearly not scalable. The contract used for testing this can be found in the appendix listing 6.

Another public option would be using the InterPlanetary File System (IPFS) [93]. According to the protocol documentation "IPFS is a distributed system for storing and accessing files, websites, applications, and data." [94]. It has become very popular in Web3 applications since it is an open and decentralized protocol that allows users to store information without it being inside a single entity infrastructure. This would be a more optimal solution. But still would require being encrypted with the public keys of the parties allowed to access that data (this requirement also applies to the on-chain storage solution).

The third option is keeping records of the feedback in a traditional and centralized manner in a database. This option is easier to handle, especially when taking on account data privacy laws. It is also true that feedback does not have to be stored forever. Its storage is essentially needed to be forwarded to the interested parties. This has been the option implemented in the prototype developed during this master's thesis.

3.2.4 Jurors

The jurors are external parties, persons interacting with the protocol through externally owned accounts (EOAs), as oracles in the sense that they provide real world data, that

is external to the blockchain. They are assigned feedback to evaluate according to a pre-established criteria.

There are some proposed requirements [1] that the jurors should fulfill to be deemed trustworthy and competent in the evaluation of feedback:

1. Having set up a company and/or to have been self-employed.
2. Have working experience in at least three sectors of activity. We might also have the possibility of having Jurors specialized in a single sector of activity, who will only have the option of evaluating feedback in that sector.
3. Have more than 15 years of work experience.

These requirements are a proposal not yet tested. The initial proposal for choosing the jurors is that the documentation will be sent to a notary for controlling the compliance with these requirements and is the entity administrating the Proof of Feedback that will elect the jurors. The number of jurors can grow overtime as long as they meet the requirements. The Court of Jurors is elected for every country where the PoF is implemented so that they can understand the local environment (language, market, legislation, etc).

This selection part is the one that can be delegated to a DAO and will be discussed in section 3.2.9

3.2.5 Regulator node

The rewarding of the customer and the jurors happen through this node. The validators provide the evaluation that the jurors have given and is the regulator that calculates what should be the right amount rewarded to the customer.

3.2.6 Validators

The role of the validators is to be the connection with the blockchain. Firstly, the user interacts with them in a direct or indirect way when registering feedback, and then are the jurors that provide their evaluations inserting them through the validators. There are different validators depending on two things: the relationship of the company with the administrator of the protocol; and the kind of feedback given.

Internal validator

The internal validator receives the feedback (customer address, feedback id) and needs to randomly select 3 jurors to give them the evaluation task. This is an extra challenge since the EVM is a deterministic computer and performing a random selection on-chain is an easily hackable task in the sense, that it could be predicted who are going to be the jurors, and therefore, the hacker could input information in a way that the agreed juror would get to evaluate the feedback previously set.

To avoid such previously fixed performance there would be needed an oracle to provide off-chain randomness that cannot be predicted. This kind of oracle already exists [95], this option though needs an extra call to an external contract and pay the fee for such call. Solving this problem will be discussed in the implementation section 3.3.2 solving this issue.

Lawyer validator

When the feedback is a complaint/claim that requires legal advice or remedies during the process, the lawyer validator will be the only one in charge of managing the rewards to users. In each jurisdiction, a renowned legal firm specializing in consumer protection will form the lawyer validator. Users that offer input that meets the following conditions would receive tokens from this legal validator oracle:

1. The feedback sent has a chance of being dealt with as a judicial or extrajudicial claim.
2. When a customer engages a law company to handle the claim, the claim is processed, and then after the judge has ruled the case, the lawyer validator would compensate the user with EASYF tokens.
3. Once the judge has handed down his sentence, the lawyer validator oracle would award EASYF tokens equal to 10% of the compensation received.

Company validator

In the original whitepaper [1] there was another kind of validator called company validator that was used for companies that are customers of Easy Feedback. This kind of validator led to have a company with minting capabilities and with no control by the rest of the network. This capability has been considered unsuitable for the system, removed and the possibility for a company to reward EASYF is been left in a way that they can have a company node but they have to provide it themselves with the liquidity by means of purchasing EASYF in the market.

3.2.7 Feedback evaluation

The final result of the evaluation will be the average of the 3 Jurors' scores [1].

For each of the three categories of the feedback that are evaluated: usefulness, originality and execution, the Court of Jurors will use the following grading scale:

| | | | | |
|----------|-----|------|--------|-----------|
| Very Low | Low | High | Medium | Very High |
| 0 | 1 | 2 | 3 | 4 |

Table 3. Table showing the corresponding evaluation grades

And, therefore, the maximum total sum of the 3 Jurors would be 36 points giving a final result of an average of 12 points. This average result needs to be translated to a value in EASYF tokens. That amount needs to depend on what is the current price of EASYF in the market and the Big Mac index [96] of the country where the company receiving the feedback is located. This will be further discussed in the subsection about the different oracles 3.2.8

All scores will be transformed into tokens by the equivalence of 12 points means the dollar equivalent of the US price of 2 Big Macs which according to the current Big Mac index for Estonia (2022) published by The Economist magazine \$4.43, would mean a reward of 8.86 dollars.

This value of around 8 dollars with the equivalence $1 \text{ EASYF} = 0.05 \text{ USD}$, would correspond initially to 160 EASYF, at the moment of listing in trading exchanges. The 160 tokens will vary depending on the value at which they are traded on the market.

3.2.8 Oracles

Once the feedback is in the blockchain there are still two important data needed from the real world, and therefore two external oracles are needed to provide them.

Token market price feeding

The ideal solution would be to have an external decentralized oracle hosted by a service like Chainlink [97] but since the token is still not traded in the market, such implementation is not feasible in the scope of this work. That is why it will be implemented using a smart contract written by the author of this thesis.

Big Mac index feeding

This index is well-known but there are still not public oracles implemented for it. The goal is to have such public oracle feeding information to the PoF, but such implementation has been left out the scope of this work and it has been implemented manually with a smart contract. And probably for this feed, since this index is updated once per year is not worth implementing a whole oracle system only for it.

3.2.9 Possibility of a ruling DAO

There are many aspects of this system that would still be managed and ruled by a single entity, the company implementing the system. This is not what is sought with the development of DApps in a blockchain (what is called Web3). This is a bottle neck for decentralization. Many of the functions executed by the initial administrator of the protocol could be given to a DAO.

Governing scope of the DAO

Here are discussed the different functions the DAO could perform:

- Approve new jurors for the different internal validators. That would require the development of an additional system to proposed new jurors and a mechanism for voters to revise their application and make a voting decision accordingly. This presents an extra difficulty due to the regional implementation of the internal validators.
- Revise and update the requirements for new jurors applications.
- Revise and update the evaluation criteria and methods.
- Audit the different systems of the Proof of Feedback (e.g.: draw system, the maximum rewards, the evaluation layer).
- Audit the amount of feedback each customer and each juror has received. This audits can be held in a decentralized way with the help of the Graph, a decentralized protocol for indexing data of the blockchain [98].
- Provide liquidity to the different regulator nodes by minting new EASYF-s. This would mean giving the DAO the ownership of the EASYF smart contract.

Note all these functions would need to be given at once, and they require further research.

Possible voting powers:

There are several voting powers that can be granted by:

- EASYF token holdings.
- Some certification that the participant has given feedback using the PoF (one voting power). This could be recorded by giving a feedback NFT (Non-Fungible Token) to each customer that has given feedback.
- Amount of times interacted with the PoF. This could also be implemented with the help of an NFT.
- Amount of times with successful feedback in the PoF.
- Delegated staking of EASYF.
- Using LP (Liquidity Provider) tokens that have been accumulated by staking.

3.3 Implementation prototype

For testing this protocol a reduced version has been implemented. Reduced in the sense that includes only one internal validator from the regional layer that covers one sector of the economic system; and one regional regulator node. In the prototype has been left out the lawyer validator since it involves more parties and legal matters out of the scope in this master's thesis.

The smart contracts implementation will be discussed in subsection 3.3.2. The interactions between the end-users and the jurors with the smart contracts will be explained in subsection 3.3.3. And the management of the feedback as well as users and jurors data will be detailed in subsection 3.3.4.

The whole system has a traditional client-server architecture with an added interaction with the blockchain for tokenising the process of rewarding feedback.

3.3.1 Blockchain Polygon

This prototype has been implemented in the EVM compatible blockchain Polygon. There are mainly two reasons to use this blockchain: the community behind the blockchain (we could say its popularity) and the cost per transaction. The native token of this blockchain is called MATIC.

Ethereum is a very popular blockchain in terms of market capital and daily users, but its raise in value (fiat price per each ether) and number of users forced it to have excessively high gas fees (over \$63 on average during November 2021) [99]. The effect of this has been scarcely studied, but it has been researched how it has a slight negative impact in the activity of DAO's [100]. Nevertheless, in this systems the cryptocurrency amounts that

are usually transfer should be relatively big compared to the gas fees. The design of the proposed system in this master’s thesis requires small fees for it to be feasible. This will be part of the validation process as discussed in section 4.1.1.

Nonetheless there are several alternative blockchains that are easily portable from one to another, due to the fact of being EVM compatible like Polygon, Binance Smart Chain (BSC), Avalanche, Fantom Opera... And Polygon comes often as a popular alternative that is cheaper and faster than Ethereum as per 2022. It might be claimed that BSC is more popular, even so is more centralized than Polygon.

3.3.2 PoF contracts

During the course of this thesis several smart contracts have been writing to test the different architectures. But using the proposed architecture shown in figure 4 there are mainly 3 different contracts: InternalValidator in appendix listing 2, RegionalRegulator shown in listing 3 and EASYFPriceFeed in listing 5. The last being the oracle code written for this thesis since the implementation of a decentralized oracle has become very expensive [101] and the requirements of the price feed oracle for this protocol are simple.

PoF feedback registration (Internal Validator Smart contract call)

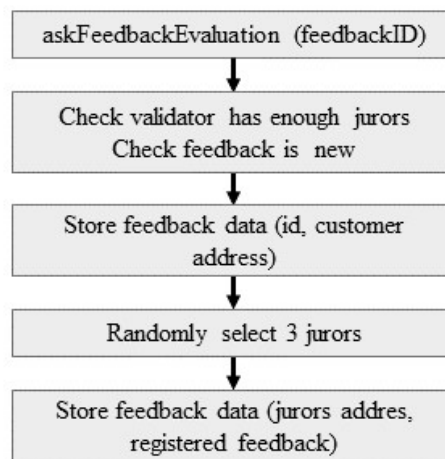


Figure 5. *PoF feedback registration smart contract call*

The process for registering feedback shown in figure 5 starts by any address calling the function `askFeedbackEvaluation` in which the user needs to provide the id of the feedback to be evaluated. The function starts by checking that the contract has enough jurors registered to start working and that the feedback id is new. Then the address of the feedback provider is stored in a structure of the state data in the internal validator contract. That structure is stored in a map where the index is the feedback id. Using the provided

feedback id the jurors are assigned randomly and this data is also stored in the structure. Then it needs to be evaluated by the jurors. All the feedback registration process happens in an internal validator smart contract.

PoF Feedback evaluation (Smart contracts calls)

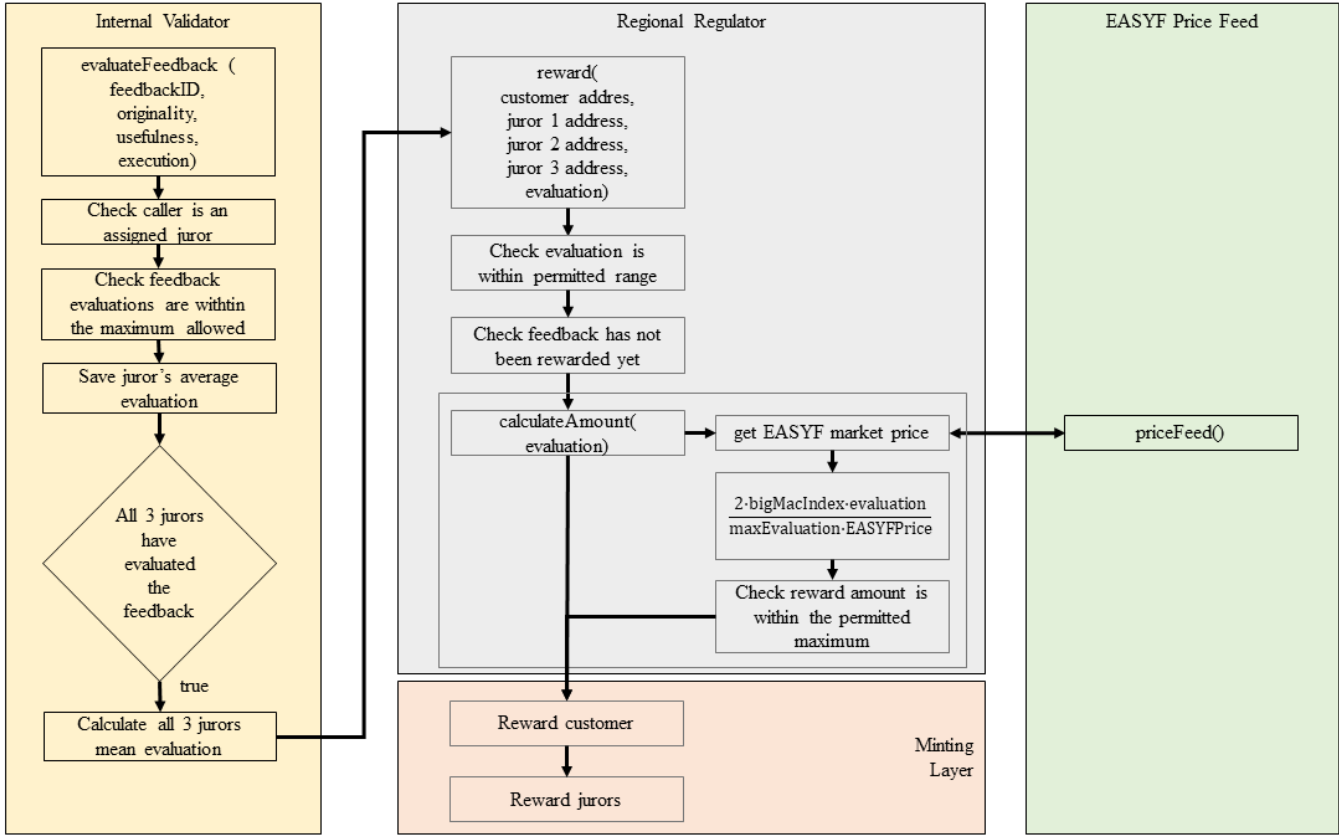


Figure 6. PoF feedback evaluation smart contract calls

Once the feedback is stored in the smart contract the evaluation process can start as shown in figure 6. Each juror needs to call the function `evaluateFeedback` providing the feedback id and their 3 scores evaluating that feedback. The function first checks that the jurors has been assigned that feedback and that the feedback evaluations are in the correct range. Then it stores the average evaluation of the juror in the structure of the feedback. Once the last juror inserts his/her evaluation the function calculates all 3 jurors average and calls the `reward` function of the regional regulator. It checks evaluation is within the permitted range and makes sure the feedback has not been rewarded yet. Then it calculates the amount to be rewarded using equation 3.1:

$$\frac{2 \cdot bigMacIndex \cdot evaluation}{maxEvaluation \cdot EASYFPrice} \quad (3.1)$$

The EASYF price is retrieved by calling the EASYF price feed oracle. If the amount to be rewarded does not exceed the maximum set in the smart contract then it proceeds to

reward the user and the jurors calling the ERC-20 smart contract functions.

Random jurors draw

To avoid an attack like the one to the Fomo3d game [102] where the seed for generating a random number was generated from data publicly available from the blockchain it needs to be given from the outside real world.

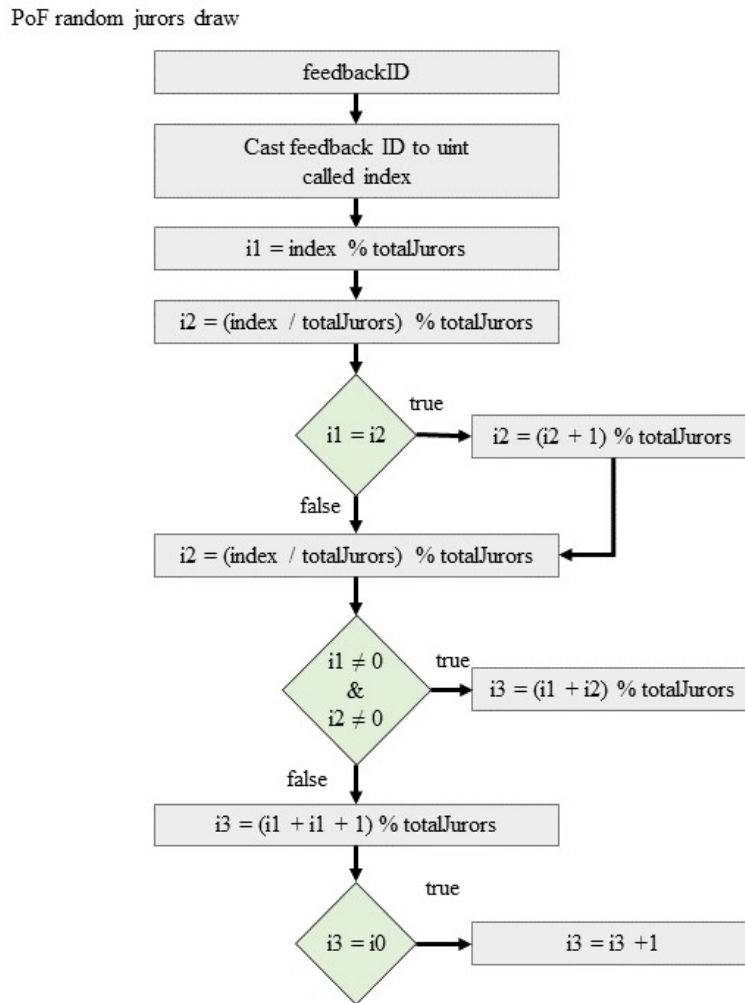


Figure 7. PoF random jurors draw algorithm

As previously mentioned in section 3.2.6, once a feedback is registered it needs to be randomly assigned to 3 different jurors. This poses the problem of generating randomness on-chain. But this may be solved by having a feedback id that is random (the id can be generated outside the blockchain and input as a parameter). Feedback id should be in the format of a valid 20-byte address [103]. For that it the feedback could be contained in a structure and ciphered using sha256 algorithm and take the right most 160-bits. This creates a practically collision free id that is almost impossible to predict unless the user inserting the feedback knows exactly the structure of the feedback content used to be hashed and add words getting a favorable hash. A simple fix would be to add a random number (nonce) to

that content or a timestamp in the back-end making it practically impossible to the user to predict the outcome.

Then in the smart contract can be used a modulo of the hash with the total number of jurors. The jurors' wallets are stored in an array, and the function `randomJurors` takes as a parameter a feedback id (160-bit address) that is first cast to an unsigned integer to draw the indexes as it is explained in figure 7 describing the algorithm.

The more jurors the less often collisions. With 7 jurors around 49% of the times there is some collision. With 17 there is already only 18% collisions. This has been empirically tested. It has also been proved that the jurors are uniformly assign for evaluation, meaning that there is no favourite index with this algorithm, all indexes are drawn with the same frequency.

3.3.3 PoF client

The UI for the user where the feedback can be provided has been focused on three types of feedback (mentioned in subsection 2.1.1): claims, complaints and suggestions. There is a simple landing page as shown in figure 8 where the user can start and decide whether to provide feedback or evaluate some by inserting the id of the feedback to be evaluated.

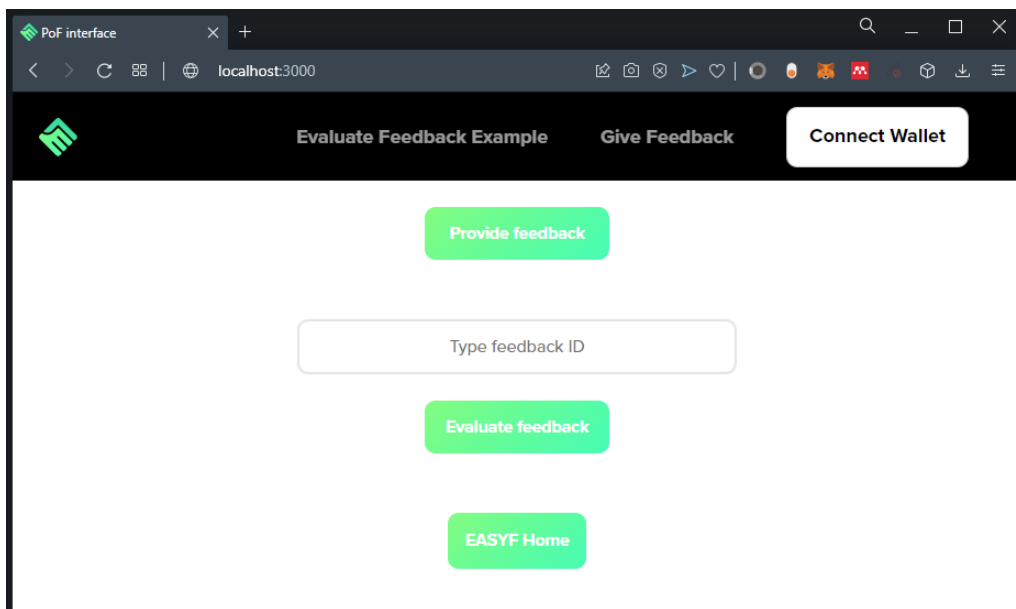
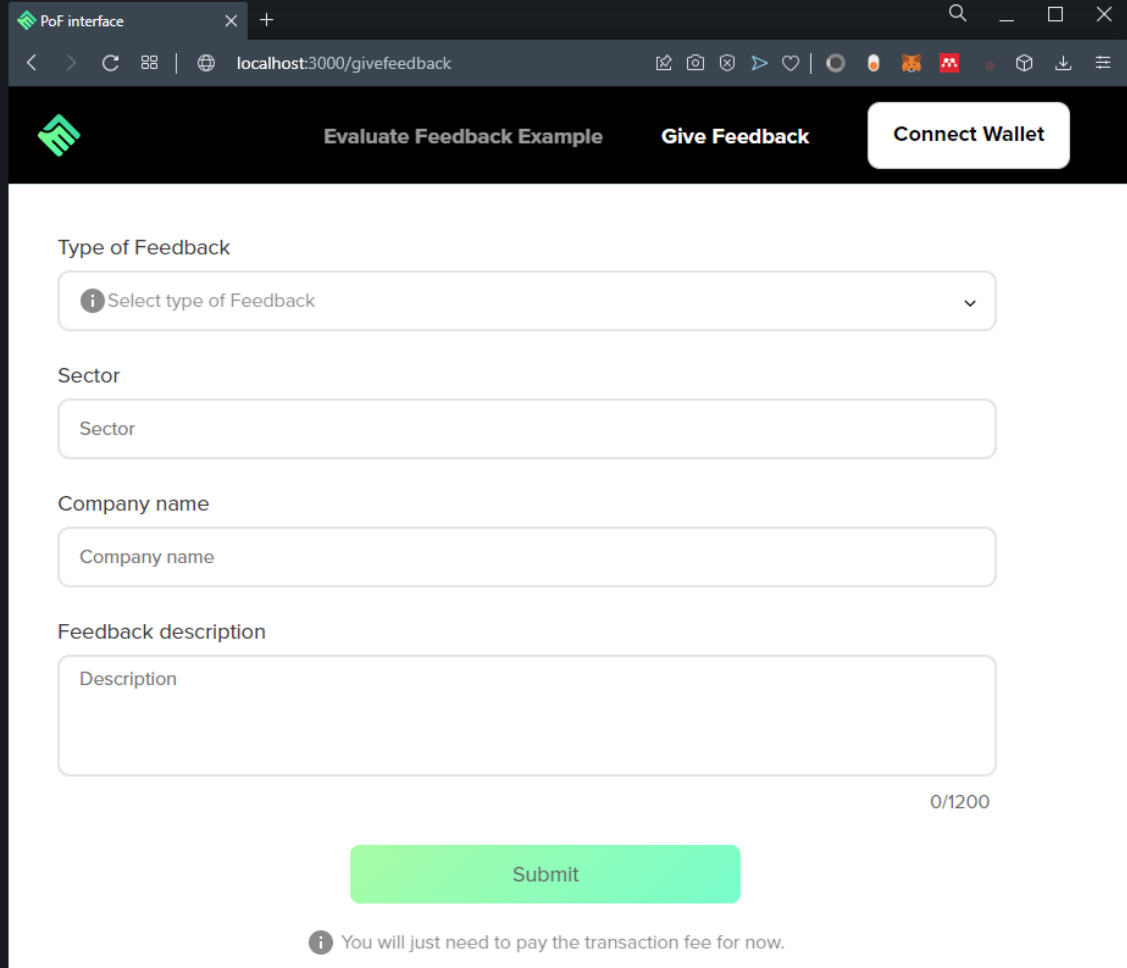


Figure 8. *PoF interface. Landing page*

Giving feedback

The first part of the interface is for the client to provide feedback to a company. In figure 9 can be seen the page that the client gets for providing feedback. Before providing feedback

the user needs to log in using a blockchain wallet (in the current version 2 of the most popular wallet service providers are supported: Metamask and Coinbase).



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/givefeedback'. The page title is 'PoF interface'. The header features a green logo on the left, and three navigation elements: 'Evaluate Feedback Example', 'Give Feedback', and a white button labeled 'Connect Wallet'. The main content area contains a form with the following fields:

- Type of Feedback:** A dropdown menu with the placeholder text 'Select type of Feedback' and a downward arrow.
- Sector:** A text input field with the placeholder text 'Sector'.
- Company name:** A text input field with the placeholder text 'Company name'.
- Feedback description:** A large text area with the placeholder text 'Description' and a character count '0/1200' at the bottom right.

Below the form is a prominent green 'Submit' button. Underneath the button, an information icon is followed by the text: 'You will just need to pay the transaction fee for now.'

Figure 9. *PoF interface. Customer view for giving feedback*

Apart from that, the user would need to fill some more data in order to give private and non-anonymous feedback. The required information are name, year of birth, gender and e-mail address. This data is needed to ensure several things:

- The user is at least 14 years old.
- Contact information for different purposes: notification of feedback being processed, notification when getting rewarded, contact for the company specially when solving a claim or complain.
- Basic information for future statistics and better training of the machine learning system that wants to be implemented in the future as a first filter for jurors.
- Whitelisting an address to be able to interact with the system. One aspect to be discussed is the necessity to perform a deeper KYC (Know Your Customer) process.

The process of providing feedback is simple. It starts by the user filling the form with

the necessary feedback information: type of feedback, sector in which the company is operating, the company that is receiving the feedback and the feedback itself.

Once the form is filled with correct data then it is send to the server where the feedback is stored in a private database. A flow chart describing the process can be observed in figure 10.

PoF Feedback registration data flow chart

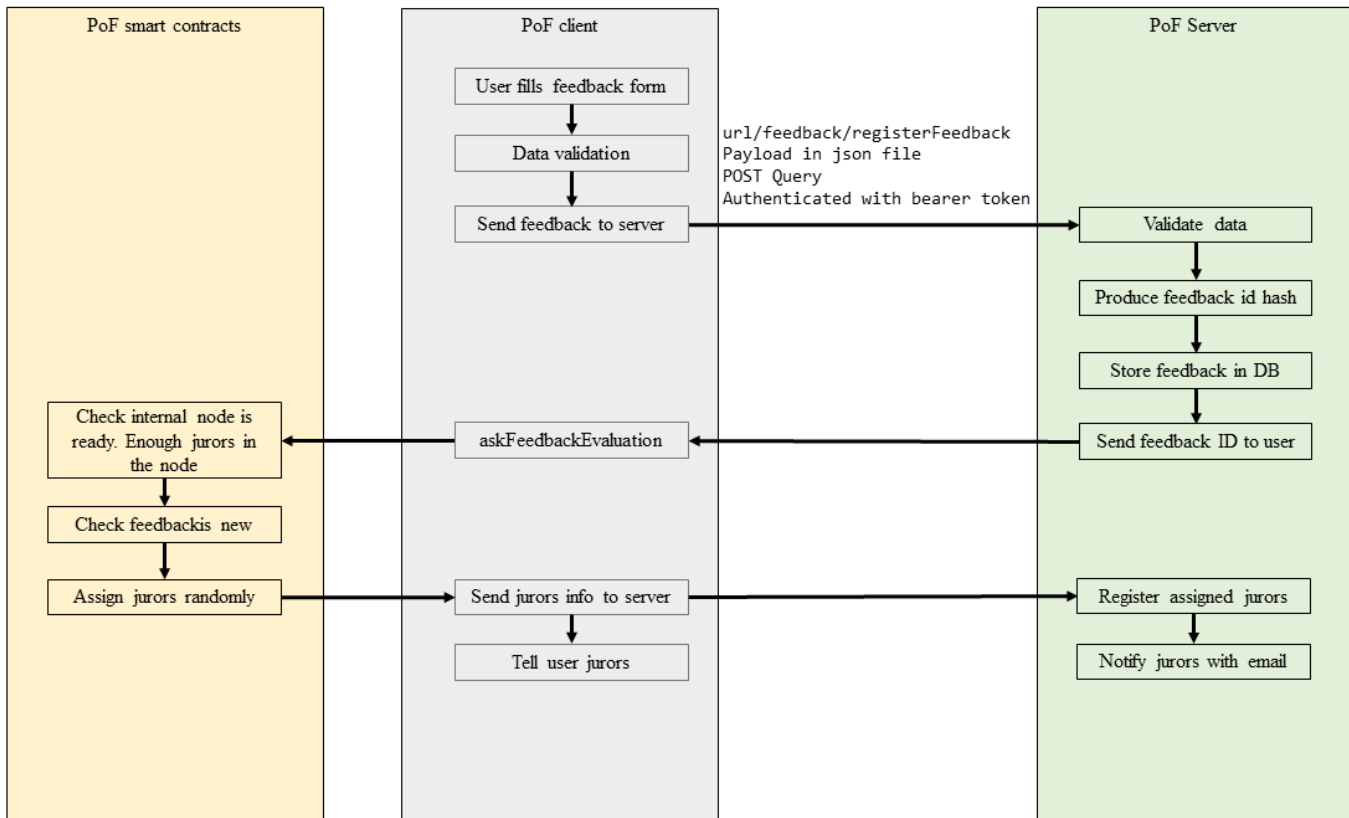


Figure 10. Flow chart for the feedback registration process

Evaluating feedback

Once a feedback has been registered in the blockchain and stored in the database the jurors can start evaluating it. Figure 11 shows a screenshot of the evaluation page the jurors are prompted and the process of evaluating.

The process for the jurors is very simple they just give their evaluation according to the criteria described in chapter 3.2.7. When the jurors submit their evaluation they interact with a the smart contract of the internal validator as described in the previous chapter 3.3.2. In the submission process that information is also sent to the server. This part has no influence in the rewarding process, but is necessary for the interaction with the PoF interface. Once all of them have evaluated the feedback, the smart contract automatically

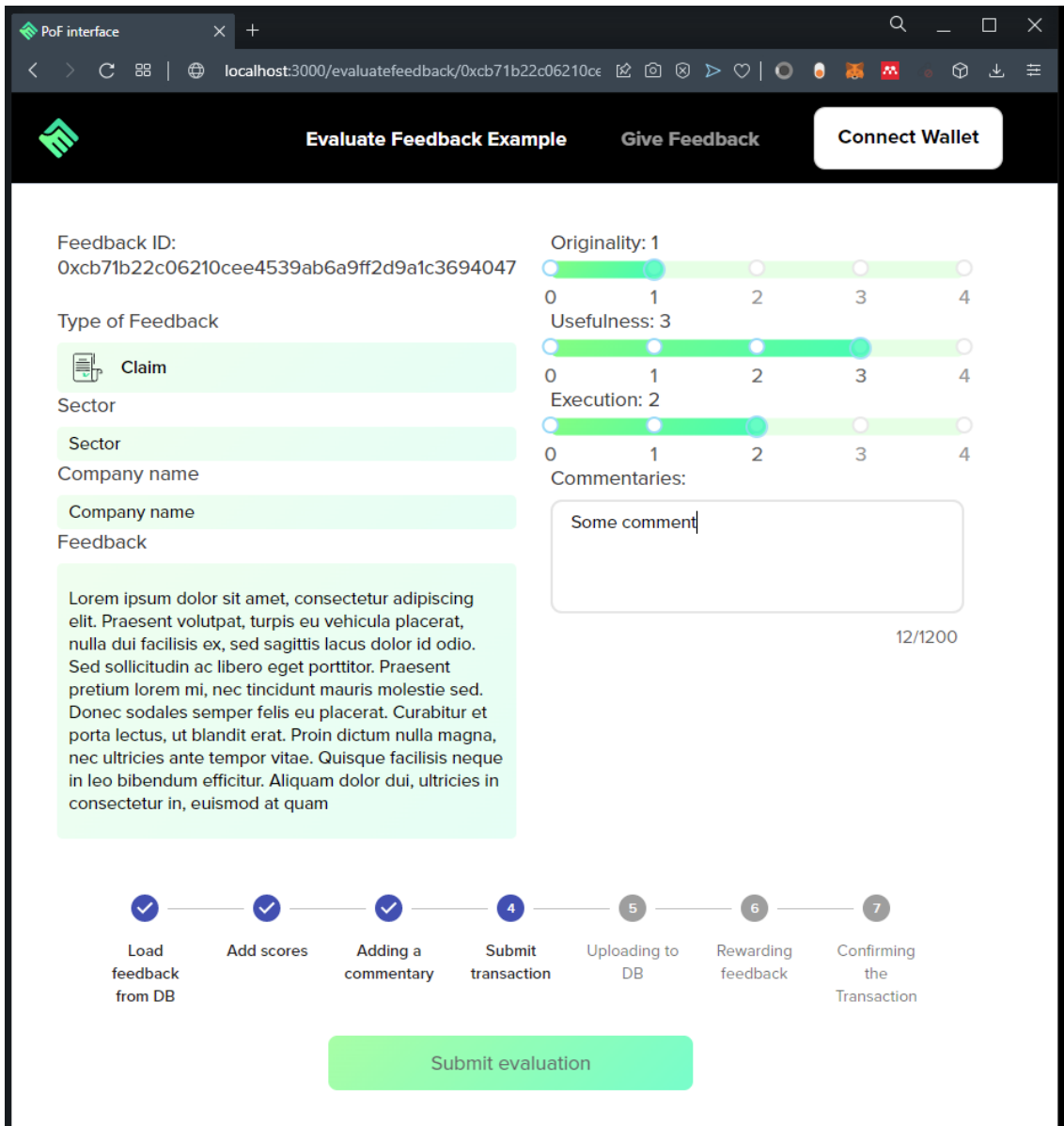


Figure 11. *PoF interface. Juror view while evaluating feedback*

finishes the rest of the process. In figure 12 is described the process interconnecting the client, server and smart contract.

That part happens in the blockchain and in the current implementation no information of the part is stored in the server, but it could be done by submitting the transaction address to the server and taking the information that is available in the logs thanks to the emitted events of the smart contract. It has the problem that if the user closes the application before sending the transaction address to the server, the feedback would still be rewarded but that information would stay unrecorded. This could be solved by implementing an indexing node with the help of the decentralized indexing protocol the Graph [98]. Which allows to index continuously information of the tracked events from certain smart contracts, which

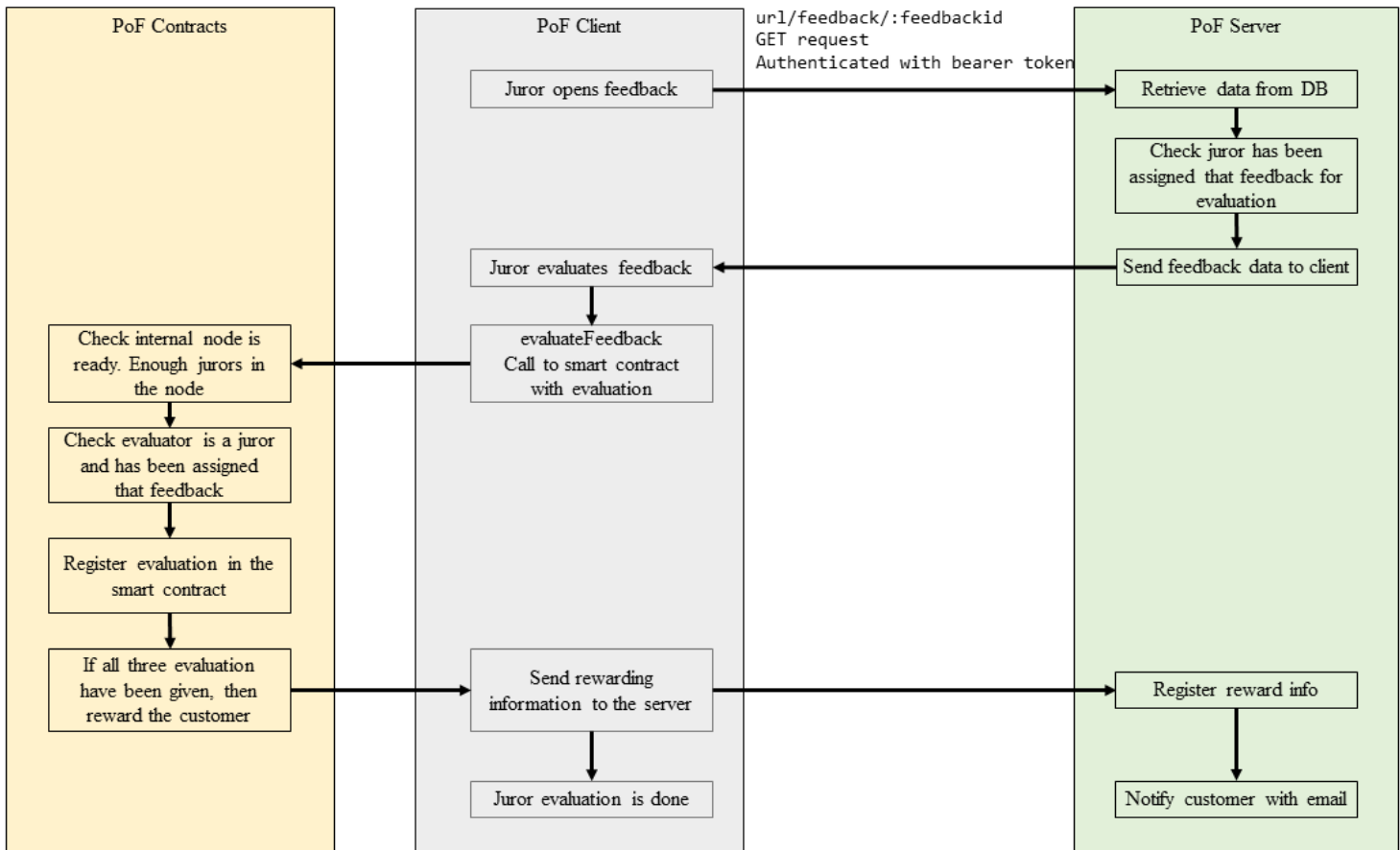


Figure 12. PoF interface. Feedback evaluation data flow chart

would allow the tracking of the rewarding part without the need of the client sending that information.

3.3.4 PoF server

The server part of the system manages all the database (DB) queries, creating the object models, registering new entries, updating them, retrieving them to send the information to the client and in general all traditional DB management actions.

Secure connection.

It ensures that users are connected with their wallet address and authenticated with a type of bearer token called JSON web token (JWT) [104] which is often used for assuring a secure private connection without need of all the credentials of the user going from client to server [105]. In this application the client asks for the authentication token which the server produces with an expiration time by using a fixed JWT string and signing it with the public wallet address of the user.

This secure connection allows to verify who is making the request, and send the data they are allowed to have access. For instance only jurors that have been assigned a feedback for evaluation can fetch that data. Once they request the data of a specific feedback by its id, it is checked from the server side that the juror has been assigned that feedback. This is important since any one can see the feedback ids that have been registered in the smart contract.

Main queries the server handles:

- Feedback related queries:
 - `/getfeedbackinfo` post request for retrieving feedback.
 - `/registerFeedback` post request for storing or updating feedback to the db. As discussed in 3.2.6 when the feedback is going to register the server provides an extra number to the content of the feedback in order to be hashed and returned to the client as feedback id.
- User-authentication related queries:
 - `/getToken` post request for getting authentication token. Needs user address
 - `/accountdetails` post request for registering/updating account (wallet address) information .
 - `/getaccountinfo` get request for getting account (wallet address) information.

4. Validation of the proposed system

There are mainly two aspects analyzed for the validation of the system: a general audit of the protocol and the interface in section 4.1 and to what degree is the whole system really decentralized in section 4.2. At the end of the validation chapter some possible improvements are presented in section 4.3.

4.1 Smart contracts audit

Like for any system involving ownership of some asset, in this case digital assets in the form of cryptocurrencies, there are standard mechanisms for auditing smart contracts and different tools to perform an automated audit [102, 106] as well as specialized companies like Certik [107]. The use of such tools has been left out the scope of this master's thesis.

The author of this master's thesis has summarized what seems to be general practice in the blockchain industry [108], since there is not much unified standard in open research for auditing smart contracts. Firstly the cost of deploying and operating the system is analysed in subsection 4.1.1, then the different possible vulnerabilities of the smart contracts are defined in subsection 4.1.2 and finally what can be the platform security flaws in subsection 4.1.3.

4.1.1 Gas costs analysis

The first analysis to validate the system is the cost of deploying and using such system. For that the different operations' costs have been measure using the developed prototype.

In table 4 are presented the gas cost of each function that have been measured while running the tests locally. Is important to take on account that the gas cost may vary a bit depending on the moment usage of the network. Also, when translating that cost to euros must be reminded that the conversion from the Polygon native cryptocurrency to euros might change overtime.

Also we can observe that the cost of deployment is very small, a maximum of 4 euro cents for the regional regulator. But then we need to process what is the cost each time a feedback

Table 4. Table the gas cost of the different smart contract functions

| Solc version: 0.8.12 | | Optimizer enabled | | Runs: 200 | Block limit: 30000000 gas | |
|----------------------|-----------------------|-------------------|---------|-----------|---------------------------|-----------|
| Methods | | 21 gwei/gas | | | 1.20 eur/matic | |
| Contract | Method | Min | Max | Avg | # calls | eur (avg) |
| EasyFeedBackToken | mint | 38651 | 55775 | 47213 | 32 | 0.00 |
| EASYFPriceFeed | setPriceFeed | 24799 | 46711 | 29736 | 100 | 0.00 |
| PoFInternalValidator | addJuror | 79701 | 113901 | 86032 | 54 | 0.00 |
| PoFInternalValidator | askFeedbackEvaluation | 148018 | 148419 | 148084 | 204 | 0.00 |
| PoFInternalValidator | evaluateFeedback | 61845 | 311161 | 132236 | 303 | 0.00 |
| PoFRegionalRegulator | addInternalNode | 74580 | 91680 | 90823 | 20 | 0.00 |
| PoFRegionalRegulator | setMaxEvaluation | 51203 | 59603 | 55403 | 2 | 0.00 |
| PoFRegionalRegulator | withdraw | - | - | 57276 | 1 | 0.00 |
| Deployments | | | | | % of limit | |
| EasyFeedBackToken | | - | - | 1044625 | 3.5 % | 0.03 |
| EASYFPriceFeed | | - | - | 268824 | 0.9 % | 0.01 |
| PoFInternalValidator | | 1189237 | 1189249 | 1189248 | 4 % | 0.03 |
| PoFRegionalRegulator | | 1625688 | 1625700 | 1625699 | 5.4 % | 0.04 |

is registered and evaluated. To have an idea of what that cost means the cryptocurrency units will be converted to the local fiat, euros. But the price of MATIC in relation to euros is volatile along time. Table 5 shows the variation in price during the last year (from April 2021 to April 2022). That data will be used to analyzed the possible operational variable cost.

Table 5. April 2021 - April 2022 historical MATIC price

| | MATIC/€ |
|--------------------|-------------|
| Minimum | 0,587305667 |
| Maximum | 2,543323738 |
| Average | 1,350823265 |
| Standard deviation | 0,371608122 |

The cost of askFeedbackEvaluation function is: $(148084/10^9) * 21 * 1,20 = 0,0037317168€$

The average cost of evaluateFeedback with multi evaluation function is: $(132236/10^9)* 21 * 1,20 = 0,0033323472€$

So the total cost per feedback is $0,0037317168 + 3 * 0,0033323472 = 0,0137287584€$

If the yearly prognosticated amount of feedbacks is 100 000 the cost per year would be 1 372,87584€.

From the previous cost tables we can make a prediction of the operational cost for the Proof of Feedback protocol as shown in table 6

But more interesting than the total cost is the relative cost of the value generated versus the cost of generating that value. The value generated is more stable since it has establish an amount of 2 BigMacs to reward feedback an another 2 to reward the jurors. That amount is updated once or twice per year, allowing to do a more precise analysis. Table 7 shows the

Table 6. *Future cost estimation in euros*

| Function | Gas cost | Ether/Gwei | Gas multiplier | MATIC/EUR | | | Cost | | |
|---------------------------------|----------|------------|----------------|-----------|------|------|--------|---------|---------|
| | | | | Min | Avg | Max | Min | Avg | Max |
| askFeedbackEvaluation | 148084 | 1000000000 | 21 | 0,58 | 1,35 | 2,54 | 0,0018 | 0,0042 | 0,0079 |
| evaluateFeedback | 132236 | 1000000000 | 21 | 0,58 | 1,35 | 2,54 | 0,0016 | 0,0038 | 0,0071 |
| Total cost per feedback | | | | | | | 0,0067 | 0,0155 | 0,0291 |
| Total cost per 100 000 feedback | | | | | | | 671,91 | 1545,40 | 2909,70 |

generated variable generated value (maximum, minimum and the average). The estimation has been done using the last BigMac index for Estonia 4,43 and the average conversion from USD to EUR during April 2021 to April 2022: 0,867916.

Table 7. *Generated value by feedback getting rewarded and jurors evaluating feedback*

| Generated value (€) | Customer | Jurors | Total |
|---------------------|----------|--------|-------|
| Min | 0,00 | 7,69 | 7,69 |
| Avg | 3,84 | 7,69 | 11,53 |
| Max | 7,69 | 7,69 | 15,38 |

And table 8 presents the variable relation of the generated value versus cost. It can be observed that even in the least favorable scenario when the generated value is the minimum (7,69€) and the cost is the maximum (0,0279€) the generated value is still more than 264 times bigger, which seems to justify the cost. In the best case scenario that ratio is even bigger with the generated value being more than 2288 times bigger than the cost.

Table 8. *Relation of generated value versus cost*

| Total | | | Cost | | |
|-----------|-----|-------|---------|--------|--------|
| | | | Min | Avg | Max |
| | | | 0,0067 | 0,0155 | 0,0291 |
| Generated | Min | 7,69 | 1144,46 | 497,59 | 264,28 |
| | Avg | 11,53 | 1716,69 | 746,38 | 396,42 |
| | Max | 15,38 | 2288,92 | 995,18 | 528,56 |

4.1.2 Contract vulnerabilities

There are mainly three types of vulnerabilities analysed:

1. *Reentrancy issues [109]*: it refers to issues that arise when a smart contract makes an external call to another smart contract function before the effects of the original smart contract call are resolved. Because the original contract's balance has not yet been changed, the external contract can then recursively call the original smart contract and interact with it in ways it should not be allowed to. There have been several serious and big attacks of this kind in terms of capital in DeFi projects (\$11.7M in

the case of Agave DAO and Hundred Finance [110] and \$2M in Revest Finance [111] and even bigger ones [112]).

2. *Integer overflows and underflows*: since the blockchain is a deterministic machine and smart contracts must be too, they work only with integers, and when a smart contract performs an arithmetic operation but for instance the result exceeds the storage limit of the smart contract (usually 18 decimal places) it leads to leaks or errors. As a result, inaccurate numbers may be estimated, or function calls being wasted.
3. *Front running opportunities*: Market purchases or sales can be predicted by poorly designed programming. As a result, others may be able to use and trade on the knowledge for their own gain. This affects mostly DeFi projects, but this prototype is not left out of this vulnerability.

Reentrancy issues

In the current prototype there are no calls to external untrusted contracts at the moment, so this kind of attack is more theoretic. But analyzing the points where there are calls to external contracts, there is one reentrancy vulnerability when the last juror has evaluated the feedback. There is one requirement to check if the feedback has been rewarded or not to avoid double rewarding but, that flag is set after the customers and the jurors have been rewarded (a call to the ERC-20 token contract), so that could be exploited but only by designated jurors. In theory all jurors are externally owned accounts (EOA), that means the callers of the functions are not contracts, and therefore they cannot have a `fallback` function that exploits that vulnerability. Figure 13 shows the point within calls where the reentrancy vulnerability can happen.

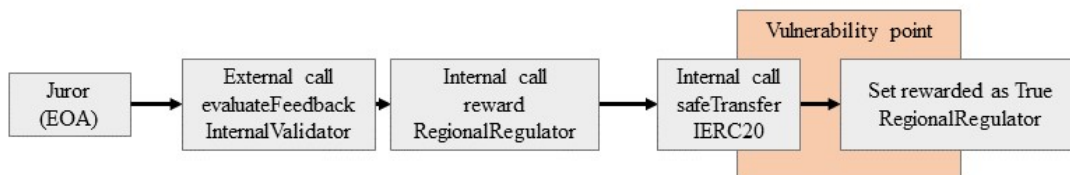


Figure 13. *PoF reentrancy vulnerability point*

To avoid any possibility of that exploit, for instance a juror is registered with an smart contract he/she controls, adding a `noReentrant` modifier that locks reentrancy calls would prevent that call. Though that kind of juror should not be allowed. The proposed modifier for avoiding that attack is shown in listing 4.1.

```

1 // attribute to be added in the contract
2 bool internal locked;
  
```

```

3
4     modifier noReentrant () {
5         require (!locked, "Locked: ongoing contract call");
6         locked = true;
7         _;
8         locked = false;
9     }

```

Listing 4.1. Lock modifier

There are also several modifiers: in the internal validator only jurors can call `evaluate` function and they are required to have that feedback assigned (this is checked within the function). There is also a modifier `onlyInternalValidator` that prevents anyone from calling the reward function in the regulator.

Integer overflows and underflows

The basic arithmetic operation security is handled by the standard `safeMath` library that most smart contract use in EVM blockchains.

In the proposed system there are several points where overflows or underflows might happen:

- *Evaluation number*: if the jurors were to enter an excessively high evaluation there could be a situation where the tokens rewarded would be unfairly high. Of course the front-end does not allow so, but the juror could still register the feedback directly using the block explorer of Polygon or through some node. To prevent that the smart contracts have a `maxEvaluation` parameter that cannot be exceeded.
- *Maximum evaluation*: if the maximum evaluation is not synchronized between internal validator and the regulator it might lead to an overflow when rewarding the feedback. To solve that problem there is a centralized (within PoF contracts) mechanism in which the maximum evaluation can only be updated using the regulator node and is that contract that updates the maximum evaluation in the internal nodes.
- *Regulator liquidity*: if the regulator does not have enough liquidity, it reverts the call preventing the situation where a feedback has been evaluated and cannot be rewarded due to lack of liquidity and a bug in the code so when the internal validator calls the regulator `reward` function and one of the things it requires is for the smart contract to have enough balance to proceed with the call.
- There is one for loop in the code for the internal validator that could be problematic if there would be millions of jurors registered in a smart contract, which should never be the case. This could only be exploit by hacking the owners wallet and registering

millions of jurors, which would be absurdly expensive for the hacker in terms of gas fees.

Front running opportunities

There are no sales in this project but there are a couple of points in the prototype that could become front running opportunities:

1. A customer and one or more evaluators agreeing on a certain feedback rewards. For that they would need to generate a favourable feedback id that would draw the evaluators that agreed on the process. With few jurors this would not be very hard to achieve since you could brute force a favourable hash by changing the feedback content with some number used as nonce. This vulnerability is stopped in the server by adding a secret nonce.

Of course, if the administrators of the system got to know about it, they can take the jurors out of the smart contract (with the function `removeJuror` and the user can be banned in the DApp (that functionality is added in the server code and there is the administrator role that can call that function). That does not stop any user from registering any feedback id in the validator, but that could not get rewarded since there is no register of that feedback. This is still a vulnerability, since any bot could flood the smart contract with unusable ids, potentially blocking real feedback being registered. That is why a possible improvement could be verifying users in a whitelist that the internal validators can check against.

There is another requirement for using any internal validator: there have to be at least 7 jurors registered in the contract to start operating. If there are not enough jurors, then the call is reverted.

2. The price feed for EASYF that would come from an external oracle, if it got hacked like it happened with DeusDao [113] it could be used to manipulate the amount of tokens rewarded and therefore drain funds from the contract. In the current prototype this is not a problem since that oracle is controlled by the administrators of the system. But even if that oracle got hacked or there are some problems with an external oracle the architecture of the system would not allow to drain more funds than the limited amount allocated to the regulator node. And at the same time there is an extra attribute in the contract `maxReward` against which the calculated amount to be rewarded is checked, and if it exceeds that maximum the call is reverted.

4.1.3 Platform security flaws

When a protocol or platform is ready for production and professional audits are performed, they look at the network that hosts the contracts, as well as the API that is used to interface with the DApp. Users could connect their wallets to fraudulent blockchain applications if a project is subject to a defacement attack where the UI hijacked. Also the system could be subject of DDoS (Distributed Denial of Service) attack and the interface would become useless.

The prototype developed during the course of this master thesis is no yet ready for production and does not have protection from DDoS or defacement attacks since it has been run locally and does not have an SSL certificate. But as specified in the chapter about the server 3.3.4 it does have an authenticated connection between the front-end and the back-end. This mechanism avoids external users accessing feedbacks data (only assigned evaluators).

4.2 Decentralization analysis

When analyzing how decentralized the proposed prototype is, the first task is finding out how many single failure points there are. Meaning, points that if they are exploited the whole system would not work. The first obvious answer would be the point where any of the smart contracts does not work, but that has been previously covered, and if it gets hacked most of the token funds still remain secured as long as the EASYF ERC-20 smart contract remains unhacked. Also the previously mentioned situation is more than one failure point. Each of the contracts has an owner that at the beginning would be the administrator of the system, but each of those contract owners (in the case they would become several different) would be a single failure point (excluding the EASYF token contract since the protocol needs tokens but not the contract). Here the interface part for the PoF is not considered a single failure point since, the system can still work without it.

But there is indeed a bottle-neck in the interface part, a point where one entity has control over that flow diminishing decentralization. If the server part stops working the system can be used, but the jurors have no way to know the contents of the feedback they need to evaluate.

Another point for analyzing decentralization is transparency of the protocol. The users' personal data is kept private for obvious reasons: complying with data protection laws as well as the fact that there is no need for that data to be public. The content of the feedbacks

is also kept private.

4.2.1 Preserving feedback requirements

As described at the beginning of this thesis the feedback must be private and non anonymous. In chapter 3.3.4 was discussed how the privacy of the feedback is ensured by the authentication mechanism which allows to securely store the data and provide it only to the jurors that need to evaluate it, as well as to the entity receiving the feedback.

The prototype presented in this work does not fulfill the non anonymity requirement, since the user can log in with a blockchain wallet and there is no further verification required as well as no credentials are asked to provide feedback. But it is easy to restrict the registering of feedback until the user has provided personal information the same way that when a juror that has not been assigned a feedback cannot retrieve information about it.

4.3 Possible improvements

One concern is the determinism of smart contracts, once it is deployed it cannot be changed. That is why for the prototype of this system there are mechanisms that prevent different exploits like funds getting locked (for that there is withdraw function that allows to return funds to the owner of the contracts); deprecated internal nodes with access to the regulator node (this is done by adding or removing linked validators to the regulator node); and so on. But not all possible exploits or vulnerabilities arise from the beginning as well as improvements to the contracts. That is why there exist mechanisms to upgrade contracts that have been developed in EVM compatible blockchains. Further research would be needed but it is possible to create smart contracts that are upgradable, secure and scalable [114].

Implementing a DAO is out of the scope of this work since it is a whole topic and implementation process on its own. But this would be one of the next improvements towards decentralization of the system.

The way the system is designed right now can be accessed only by people that has an EVM compatible wallet and that has certain knowledge on how to use Web3 applications. As per today, there are still many people that is not so familiar with these kind of systems, and therefore would be left out from this system. This is the main obstacle in the use of the proposed system since is meant to be for everyone.

There are at least two different possible solutions for people without a blockchain wallet:

1. Qredo [115]: is a layer 2 protocol for managing crypto assets. Where a dozens of wallets can be open and assigned to different customers so the customer without his/her own wallet can still get the rewards and claim them in the future once has opened a blockchain wallet. The main disadvantage of this solution is that in the ledger it can be seen as a many rewards going to one address, that is the bridge to the Qredo layer 2 solution, giving the impression that someone is unfairly benefiting from the system.
2. Having a smart contract with a list of clients, where the correspondent EASYF tokens are deposited. This is a bit more expensive solution but more transparent.

In both, solutions the management of the funds would still be centralized. These solutions have been thought and discovered in the course of this master's thesis and are a feasible future implementation.

In the current implementation there is lack of connection between the server side and the blockchain in the registering process. The authenticity of the data relies on the secure connection between client and server. It is true that the server provides the feedback ID to the client without which there is no way to go. But it is lacking the mechanism to check if the feedback has been indeed registered. The server does verify the jurors assignation and for that operation reads data from the blockchain.

5. Conclusions

Often is claimed that the main advantage in adopting blockchain technology is decentralization. While this claim is not far from reality, is not the only point as well as it does not mean that everything should be decentralized when adopting this technology. Blockchain helps in some other aspects:

- **Transparency:** by having a public record of transactions with a set of addresses.
- **Tokenization:** monetizing systems that were impossible or extremely cumbersome to do before.
- **Data verification:** Everybody can check what has happened in each transaction and know that it has not been tampered.

These are characteristics that give sense to the Proof of Feedback protocol implementation proposed in this master's thesis.

At the same time this master's thesis provides mainly the following contributions:

- An overview of the existing research literature about feedback and blockchain technologies, and a bit more deeply on smart contracts and EVM blockchains that implement them, as well as a summary on blockchain oracles.
- Several architecture possibilities for an EVM blockchain protocol meant to reward valuable feedback along with different proposals on how to store feedback with different levels of decentralization.
- A prototype built using smart contracts in an EVM blockchain has been shown that is technically possible to have a newly created and innovative protocol for rewarding valuable feedback with the help of blockchain technologies, that is the Proof of Feedback presented in this thesis.
- A DApp prototype (client and server type application) to function as an interface for interacting with the proposed protocol smart contracts.
- This master's thesis also proposes a new algorithm for drawing three numbers on-chain at random given an external seed in the form of a valid EVM address.
- An audit report analysing the prototype to validate the protocol and the initial user interface. The gas cost analysis in section 4.1.1 shows that the value generated

each time feedback is provided justifies by far the cost of operating this system. In addition, as analysed in the contract vulnerabilities section 4.1.2 and in the platform security flaws section 4.1.3 it seems that the protocol and the DApp are secure enough to be deemed feasible.

- An analysis about the degree of decentralization the system has. There are still some decentralization steps that could be taken as described in section 4.2, but the system is open and transparent enough to be implemented.
- An initial draft on how a DAO could be designed to rule the protocol: what functions in managing the protocol it could fulfill (3.2.9) and by what mechanisms could it be ruled (3.2.9).

There are still many things that could be improved in this master's thesis as pointed out in section 4.3. Additionally, scalability of the system remains to be tested out and a professional audit should still be performed from an external agent, since the analysis of the system done here has been performed by the same author of the master's thesis.

To conclude, this master thesis presents an innovative protocol to reward valuable feedback with the help of blockchain and has been proved feasible, secure and economically viable.

Bibliography

- [1] *Easy Feedback Token Whitepaper*. URL: https://easyfeedbacktoken.io/wp-content/uploads/2019/06/Easy_Feedback-Token_EFT_WhitePaper_en.pdf (visited on 02/18/2022).
- [2] Rachel Jug, Xiaoyin Sara Jiang, and Sarah M. Bean. “Giving and Receiving Effective Feedback: A Review Article and How-To Guide”. In: *Archives of Pathology & Laboratory Medicine* 143.2 (Feb. 2019), pp. 244–250. ISSN: 0003-9985. DOI: 10.5858/ARPA.2018-0058-RA.
- [3] Bob Thompson. “eService: Strategies for success in the customer age”. In: *RightNow Technologies* (2002). URL: https://mthink.com/legacy/www.crmproject.com/content/pdf/CRM3_wp_thompson.pdf.
- [4] Helena Holmström Olsson and Jan Bosch. “Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation”. In: *Lecture Notes in Business Information Processing*. Vol. 210. Springer, Cham, 2015, pp. 154–166. ISBN: 9783319195926. DOI: 10.1007/978-3-319-19593-3_13. URL: https://link.springer.com/chapter/10.1007/978-3-319-19593-3_13.
- [5] *UK Customer Satisfaction Index (UKCSI) - Institute of Customer Service - The state of customer satisfaction in the UK - July 2021*. Tech. rep. 2021. URL: <https://www.instituteofcustomerservice.com/research-insight/ukcsi/> (visited on 02/16/2022).
- [6] Helena Holmstrom Olsson and Jan Bosch. “From opinions to data-driven software R&D: A multi-case study on how to close the ‘open loop’ problem”. In: *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*. Institute of Electrical and Electronics Engineers Inc., Oct. 2014, pp. 9–16. ISBN: 9781479957941. DOI: 10.1109/SEAA.2014.75.
- [7] Bob Thompson. “The loyalty connection: Secrets to customer retention and increased profits”. In: *RightNow Technologies & CRMguru* 18 (2005).
- [8] Kurt Schneider. “Focusing spontaneous feedback to support system evolution”. In: *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference, RE 2011*. 2011, pp. 165–174. ISBN: 9781457709234. DOI: 10.1109/RE.2011.6051645.

- [9] Petrișor Blidaru Onorel. “The Importance of Feedback in Organizational Communication.” In: *Social-Economic Debates* 8.1 (2019), pp. 30–38. ISSN: 2248-3837.
- [10] Paul Resnick and Richard Zeckhauser. “Trust among strangers in internet transactions: Empirical analysis of eBay’s reputation system”. In: *Advances in Applied Microeconomics* 11 (Oct. 2002), pp. 127–157. ISSN: 02780984. DOI: 10.1016/S0278-0984(02)11030-3/FULL/PDF. URL: [https://www.emerald.com/insight/content/doi/10.1016/S0278-0984\(02\)11030-3/full/html](https://www.emerald.com/insight/content/doi/10.1016/S0278-0984(02)11030-3/full/html).
- [11] Judith A. Chevalier and Dina Mayzlin. “The Effect of Word of Mouth on Sales: Online Book Reviews.” in: *Journal of marketing research* 43.3 (Oct. 2006), pp. 345–354. ISSN: 00222437. DOI: 10.1509/JMKR.43.3.345. URL: <https://journals.sagepub.com/doi/full/10.1509/jmkr.43.3.345>.
- [12] Phillip Dawson et al. “What makes for effective feedback: staff and student perspectives”. In: *Assessment & Evaluation in Higher Education* 44.1 (Jan. 2018), pp. 25–36. ISSN: 1469297X. DOI: 10.1080/02602938.2018.1467877. URL: <https://www.tandfonline.com/doi/abs/10.1080/02602938.2018.1467877>.
- [13] Berry O’Donovan, Chris Rust, and Margaret Price. “A scholarly approach to solving the feedback dilemma in practice”. In: *Assessment & Evaluation in Higher Education* 41.6 (Aug. 2016), pp. 938–949. ISSN: 1469297X. DOI: 10.1080/02602938.2015.1052774. URL: <https://www.tandfonline.com/doi/abs/10.1080/02602938.2015.1052774>.
- [14] John Gerdes, Betsy Bender Stringam, and Robert G. Brookshire. “An integrative approach to assess qualitative and quantitative consumer feedback”. In: *Electronic Commerce Research* 2008 8:4 8.4 (Oct. 2008), pp. 217–234. ISSN: 1572-9362. DOI: 10.1007/s10660-008-9022-0. URL: <https://link.springer.com/article/10.1007/s10660-008-9022-0>.
- [15] Frederick F Reichheld. “The One Number You Need to Grow”. In: *Harvard business review* 81.12 (2003), pp. 46–55. URL: <https://hbr.org/2003/12/the-one-number-you-need-to-grow>.
- [16] Thomas A. Burnham and Jeffrey A. Wong. “Factors influencing successful net promoter score adoption by a nonprofit organization: a case study of the Boy Scouts of America”. In: *International Review on Public and Nonprofit Marketing* 15.4 (Dec. 2018), pp. 475–495. ISSN: 18651992. DOI: 10.1007/s12208-018-0210-X/TABLES/6. URL: <https://link.springer.com/article/10.1007/s12208-018-0210-x>.

- [17] Nicholas I. Fisher and Raymond E. Kordupleski. “Good and bad market research: A critical review of Net Promoter Score”. In: *Applied Stochastic Models in Business and Industry* 35.1 (Nov. 2018), pp. 138–151. ISSN: 1526-4025. DOI: 10.1002/ASMB.2417. URL: <https://onlinelibrary.wiley.com/doi/full/10.1002/asmb.2417><https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.2417><https://onlinelibrary.wiley.com/doi/10.1002/asmb.2417>.
- [18] Sven Baehre et al. “The use of Net Promoter Score (NPS) to predict sales growth: insights from an empirical investigation”. In: *Journal of the Academy of Marketing Science* 50.1 (July 2021), pp. 67–84. ISSN: 15527824. DOI: 10.1007/s11747-021-00790-2/TABLES/8. URL: <https://link.springer.com/article/10.1007/s11747-021-00790-2>.
- [19] Marilyn Strathern. “‘Improving ratings’: audit in the British University system”. In: *European Review* 5.3 (1997), pp. 305–321. DOI: 10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4.
- [20] Sinclair Davidson, Primavera De Filippi, and Jason Potts. “Blockchains and the economic institutions of capitalism”. In: *Journal of Institutional Economics* 14.4 (Aug. 2018), pp. 639–658. ISSN: 17441382. DOI: 10.1017/S1744137417000200.
- [21] Dylan Yaga et al. “Blockchain Technology Overview”. In: *arXiv preprint arXiv:1906.11078* (June 2019). DOI: 10.6028/NIST.IR.8202. arXiv: 1906.11078v1. URL: <http://arxiv.org/abs/1906.11078><http://dx.doi.org/10.6028/NIST.IR.8202>.
- [22] Liuwen Yu et al. “Enhancing Trust in Trust Services: Towards an Intelligent Human-input-based Blockchain Oracle (IHiBO)”. In: *Proceedings of the 55th Annual Hawaii International Conference on System Sciences, 2022*. URL: <https://orbilu.uni.lu/handle/10993/50022>.
- [23] Samuel Yousefi and Babak Mohamadpour Tosarkani. “An analytical approach for evaluating the impact of blockchain technology on sustainable supply chain performance”. In: *International Journal of Production Economics* 246 (Apr. 2022), p. 108429. ISSN: 0925-5273. DOI: 10.1016/J.IJPE.2022.108429.
- [24] Huizhen Liu et al. “Research on Logistics Information Management System Based on Blockchain Perspective”. In: *Academic Journal of Business & Management* 4 (2022), pp. 26–29. DOI: 10.25236/AJBM.2022.040105.
- [25] Sanjeev Verma and Ashutosh Sheel. “Blockchain for government organizations: past, present and future”. In: *Journal of Global Operations and Strategic Sourcing* (2022). ISSN: 23985364. DOI: 10.1108/JGOSS-08-2021-0063/FULL/PDF.

- [26] Primavera De Filippi, Morshed Mannan, and Wessel Reijers. “Blockchain as a confidence machine: The problem of trust & challenges of governance”. In: *Technology in Society* 62 (Aug. 2020). ISSN: 0160791X. DOI: 10.1016/J.TECHSOC.2020.101284/BLOCKCHAIN_AS_A_CONFIDENCE_MACHINE_THE_PROBLEM_OF_TRUST_CHALLENGES_OF_GOVERNANCE.PDF.
- [27] Shuai Wang et al. “Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.11 (Nov. 2019), pp. 2266–2277. ISSN: 21682232. DOI: 10.1109/TSMC.2019.2895123.
- [28] Arijit Sengupta and Hemang Subramanian. “User Control of Personal mHealth Data Using a Mobile Blockchain App: Design Science Perspective”. In: *JMIR mHealth and uHealth* 10.1 (Jan. 2022), e32104. ISSN: 22915222. DOI: 10.2196/32104. URL: <https://mhealth.jmir.org/2022/1/e32104>.
- [29] M. M. Kamruzzaman et al. “Blockchain and Fog Computing in IoT-Driven Healthcare Services for Smart Cities”. In: *Journal of Healthcare Engineering* 2022 (Jan. 2022), pp. 1–13. ISSN: 2040-2295. DOI: 10.1155/2022/9957888.
- [30] Guangquan Xu et al. “SG-PBFT: a Secure and Highly Efficient Distributed Blockchain PBFT Consensus Algorithm for Intelligent Internet of Vehicles”. In: *Journal of Parallel and Distributed Computing* (Feb. 2022). ISSN: 0743-7315. DOI: 10.1016/J.JPDC.2022.01.029. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0743731522000363>.
- [31] Comp Sci et al. “Performance analysis of lightweight Internet of things devices on blockchain networks”. In: *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES* 30.2 (Feb. 2022), pp. 328–343. ISSN: 1300-0632. DOI: 10.3906/elk-2103-110. URL: <https://github.com/ConsenSys/quorum>.
- [32] Remzi Gürfidan and Mevlüt Ersoy. “A new approach with blockchain based for safe communication in IoT ecosystem”. In: *Journal of Data, Information and Management* 2022 (Feb. 2022), pp. 1–8. ISSN: 2524-6364. DOI: 10.1007/s42488-021-00063-1. URL: <https://link.springer.com/article/10.1007/s42488-021-00063-1>.
- [33] Ahmed Alkhateeb et al. “Hybrid Blockchain Platforms for the Internet of Things (IoT): A Systematic Literature Review”. In: *Sensors* 22.4 (2022), p. 1304.
- [34] Lei Xu et al. “New Gold Mine: Harvesting IoT Data Through DeFi in a Secure Manner”. In: *Blockchain – ICBC 2021*. Springer, Cham, Dec. 2021, pp. 43–58. DOI: 10.1007/978-3-030-96527-3_4. URL: https://link.springer.com/chapter/10.1007/978-3-030-96527-3_4.

- [35] Hajar Moudoud, Soumaya Cherkaoui, and Lyes Khoukhi. “An IoT Blockchain Architecture Using Oracles and Smart Contracts: The Use-Case of a Food Supply Chain”. In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2019-Septe* (Sept. 2019). DOI: 10.1109/PIMRC.2019.8904404.
- [36] Guojun Ji et al. “Timing of blockchain adoption in a supply chain with competing manufacturers”. In: *International Journal of Production Economics* 247 (May 2022), p. 108430. ISSN: 0925-5273. DOI: 10.1016/J.IJPE.2022.108430.
- [37] Soumyadeb Chowdhury et al. “Blockchain technology adoption for managing risks in operations and supply chain management: evidence from the UK”. In: *Annals of Operations Research* (Jan. 2022), pp. 1–36. ISSN: 0254-5330. DOI: 10.1007/S10479-021-04487-1/TABLES/11. URL: <https://link.springer.com/article/10.1007/s10479-021-04487-1>.
- [38] Himanshu Falwadiya and Sanjay Dhingra. “Blockchain technology adoption in government organizations: a systematic literature review”. In: *Journal of Global Operations and Strategic Sourcing* ahead-of-p.ahead-of-print (Feb. 2022). ISSN: 2398-5364. DOI: 10.1108/JGOSS-09-2021-0079. URL: <https://www.emerald.com/insight/content/doi/10.1108/JGOSS-09-2021-0079/full/html>.
- [39] Comp Sci, Murat Osmanoğlu, and Ali Aydın Selçuk. “Privacy in blockchain systems”. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 30 (2022), pp. 344–260. DOI: 10.3906/elk-2105-183.
- [40] Dejan Vujičić, Dijana Jagodić, and Siniša Randić. “Blockchain technology, bitcoin, and Ethereum: A brief overview”. In: *2018 17th International Symposium on INFOTEH-JAHORINA, INFOTEH*. Vol. 2018-Janua. East Sarajevo, Bosnia and Herzegovina: Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 1–6. ISBN: 9781538649077. DOI: 10.1109/INFOTEH.2018.8345547.
- [41] *Bitcoin - Open source P2P money*. URL: <https://bitcoin.org/en/> (visited on 02/16/2022).
- [42] *Home | ethereum.org*. URL: <https://ethereum.org/en/> (visited on 02/16/2022).
- [43] *Home | IOTA*. URL: <https://www.iota.org/> (visited on 02/16/2022).
- [44] Abhimanyu Rawat, Vanesa Daza, and Matteo Signorini. “Offline Scaling of IoT Devices in IOTA Blockchain”. In: *Sensors* 22.4 (2022), p. 1411.
- [45] Shubhani Aggarwal and Neeraj Kumar. “Cryptographic consensus mechanisms”. In: *Advances in Computers* 121 (Jan. 2021), pp. 211–226. ISSN: 0065-2458. DOI: 10.1016/BS.ADCOM.2020.08.011.

- [46] Yang Xiao et al. “A Survey of Distributed Consensus Protocols for Blockchain Networks”. In: *IEEE Communications Surveys and Tutorials* 22 (2 Apr. 2020), pp. 1432–1465. ISSN: 1553877X. DOI: 10.1109/COMST.2020.2969706.
- [47] *Proof-of-stake (PoS) | ethereum.org*. 2022. URL: <https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/#top> (visited on 02/16/2022).
- [48] Shijie Zhang and Jong Hyouk Lee. “Analysis of the main consensus protocols of blockchain”. In: *ICT Express* 6 (2 June 2020), pp. 93–97. ISSN: 2405-9595. DOI: 10.1016/J.ICTE.2019.08.001.
- [49] Bin Cao et al. “Performance analysis and comparison of PoW, PoS and DAG based blockchains”. In: *Digital Communications and Networks* 6.4 (Nov. 2020), pp. 480–485. ISSN: 2352-8648. DOI: 10.1016/J.DCAN.2019.12.001.
- [50] Ling Ren and Srinivas Devadas. “Proof of Space from Stacked Expanders”. In: *Theory of Cryptography*. Ed. by Martin Hirt and Adam Smith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 262–285. ISBN: 978-3-662-53641-4.
- [51] Lin Chen et al. “On security analysis of proof-of-elapsed-time (poet)”. In: *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.
- [52] Mohammad Wazid et al. “A Tutorial and Future Research for Building a Blockchain-Based Secure Communication Scheme for Internet of Intelligent Things”. In: *IEEE Access* 8 (2020), pp. 88700–88716. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2992467.
- [53] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized Business Review* (2008). DOI: 10.1007/s10838-008-9062-0. arXiv: 43543534534v343453. URL: <https://www.debr.io/article/21260.pdf>.
- [54] *An Introduction to IOTA | IOTA Wiki*. 2022. URL: <https://wiki.iota.org/learn/about-iota/an-introduction-to-iota#consensus-in-a-blockchain> (visited on 02/16/2022).
- [55] *What Is Polygon | Polygon Technology | Documentation*. 2022. URL: <https://docs.polygon.technology/docs/validate/polygon-basics/what-is-polygon/> (visited on 03/14/2022).
- [56] *What Is BscScan and How to Use It? | Binance Academy*. 2021. URL: <https://academy.binance.com/en/articles/what-is-bscscan-and-how-to-use-it> (visited on 02/16/2022).

- [57] *Avalanche Blockchain Consensus | Avalanche Docs*. URL: <https://docs.avax.network/learn/platform-overview/avalanche-consensus/> (visited on 02/16/2022).
- [58] Anatoly Yakovenko. “Solana: A new architecture for a high performance blockchain v0. 8.13”. In: *Whitepaper* (2018).
- [59] Asoke K Talukder et al. “Proof of Disease: A Blockchain Consensus Protocol for Accurate Medical Decisions and Reducing the Disease Burden”. In: *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBD-Com/IOP/SCI)*. 2018, pp. 257–262. DOI: 10.1109/SmartWorld.2018.00079.
- [60] Ali Arjomandi-Nezhad et al. “Proof of humanity: A tax-aware society-centric consensus algorithm for Blockchains”. In: *Peer-to-Peer Networking and Applications* 14.6 (Nov. 2021), pp. 3634–3646. ISSN: 19366450. DOI: 10.1007/s12083-021-01204-4/TABLES/1. URL: <https://link.springer.com/article/10.1007/s12083-021-01204-4>.
- [61] *Proof Of Humanity*. URL: <https://www.proofofhumanity.id/> (visited on 02/09/2022).
- [62] Yang Ming et al. “Blockchain-Enabled Efficient Dynamic Cross-Domain Deduplication in Edge Computing”. In: *IEEE Internet of Things Journal* (2022), pp. 1–1. ISSN: 2327-4662. DOI: 10.1109/JIOT.2022.3150042. URL: <https://ieeexplore.ieee.org/document/9708087/>.
- [63] Vida J. Morkunas, Jeannette Paschen, and Edward Boon. “How blockchain technologies impact your business model”. In: *Business Horizons* 62 (3 May 2019), pp. 295–306. ISSN: 0007-6813. DOI: 10.1016/J.BUSHOR.2019.01.009.
- [64] Amritraj Singh et al. “Sidechain technologies in blockchain networks: An examination and state-of-the-art review”. In: *Journal of Network and Computer Applications* 149 (Jan. 2020), p. 102471. ISSN: 1084-8045. DOI: 10.1016/J.JNCA.2019.102471.
- [65] Shaofeng Lin et al. “Research on Cross-chain Technology of Blockchain”. In: *Proceedings - 2021 6th International Conference on Smart Grid and Electrical Automation, ICSGEA 2021* (May 2021), pp. 405–408. DOI: 10.1109/ICSGEA53208.2021.00098.

- [66] Harsh Sheth, Harsh Sheth, and Janvi Dattani. “Overview of Blockchain Technology”. In: *Asian Journal For Convergence In Technology (AJCT) ISSN -2350-1146* 0 (0 Apr. 2019). URL: <https://asianssr.org/index.php/ajct/article/view/728>.
- [67] Jesse Yli-Huumo et al. “Where Is Current Research on Blockchain Technology?—A Systematic Review”. In: *PLOS ONE* 11.10 (Oct. 2016), e0163477. ISSN: 1932-6203. DOI: 10.1371/JOURNAL.PONE.0163477. URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0163477>.
- [68] Zixuan Wang. “Analysis of risks and regulatory issues in the development of blockchain finance”. In: *Academic Journal of Business & Management* 4 (2022), pp. 61–66. DOI: 10.25236/AJBM.2022.040111.
- [69] Diego Cagigas et al. “Explaining public officials’ opinions on blockchain adoption: a vignette experiment”. In: *Policy and Society* (Feb. 2022). ISSN: 1449-4035. DOI: 10.1093/POLSOC/PUAB022. URL: <https://academic.oup.com/policyandsociety/advance-article/doi/10.1093/polsoc/puab022/6524356>.
- [70] Bhabendu Kumar Mohanta, Soumyashree S. Panda, and Debasish Jena. “An Overview of Smart Contract and Use Cases in Blockchain Technology”. In: *2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018* (Oct. 2018). DOI: 10.1109/ICCCNT.2018.8494045.
- [71] Nick Szabo. “Formalizing and Securing Relationships on Public Networks”. In: *First Monday* 2.9 (Sept. 1997). ISSN: 1396-0466. DOI: 10.5210/FM.V2I9.548. URL: <https://firstmonday.org/ojs/index.php/fm/article/view/548/469%20https://firstmonday.org/ojs/index.php/fm/article/view/548>.
- [72] *Ethereum Virtual Machine (EVM)* | [ethereum.org](https://ethereum.org/en/developers/docs/evm/). URL: <https://ethereum.org/en/developers/docs/evm/> (visited on 02/18/2022).
- [73] *Rekt - Poly Network - REKT*. 2021. URL: <https://rekt.news/polynetwork-rekt/> (visited on 02/17/2022).
- [74] *Rekt - Superfluid - REKT*. 2022. URL: <https://rekt.news/superfluid-rekt/> (visited on 02/17/2022).
- [75] *Ether — Ethereum Homestead 0.1 documentation*. URL: <https://ethdocs.org/en/latest/ether.html> (visited on 05/03/2022).

- [76] Reza Rahimian and Jeremy Clark. “TokenHook: Secure ERC-20 smart contract”. In: (July 2021). DOI: 10.48550/arxiv.2107.02997. arXiv: 2107.02997. URL: <https://arxiv.org/abs/2107.02997v1>.
- [77] Paul Cuffe. “The role of the ERC-20 token standard in a financial revolution: the case of initial coin offerings”. In: *IEC-IEEE-KATS Academic Challenge, Busan, Korea, 22-23 October 2018*. IEC-IEEE-KATS. 2018.
- [78] Hamda Al-Breiki et al. “Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges”. In: *IEEE Access* 8 (2020), pp. 85675–85685. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2992698.
- [79] VITO FERRULLI. “On demand decentralized oracles for blockchain: a new Chainlink based architecture”. In: (2022). URL: <https://etd.adm.unipi.it/t/etd-02062022-124127/>.
- [80] *What Is an Oracle in Blockchain? » Explained | Chainlink*. URL: <https://chain.link/education/blockchain-oracles> (visited on 05/01/2022).
- [81] Abdeljalil Beniiche. “A Study of Blockchain Oracles”. In: (Mar. 2020). URL: <https://arxiv.org/abs/2004.07140v2>.
- [82] Madhusudan Singh and Shiho Kim. “Blockchain technology for decentralized autonomous organizations”. In: *Advances in Computers* 115 (Jan. 2019), pp. 115–140. ISSN: 0065-2458. DOI: 10.1016/BS.ADCOM.2019.06.001.
- [83] Michael Fröhlich et al. “Blockchain and Cryptocurrency in Human Computer Interaction: A Systematic Literature Review and Research Agenda”. In: *arXiv preprint arXiv:2204.10857* (2022).
- [84] Xi Zhao et al. “Task management in decentralized autonomous organization”. In: *Journal of Operations Management* (Apr. 2022). ISSN: 1873-1317. DOI: 10.1002/JOOM.1179. URL: <https://onlinelibrary.wiley.com/doi/full/10.1002/joom.1179> <https://onlinelibrary.wiley.com/doi/abs/10.1002/joom.1179> <https://onlinelibrary.wiley.com/doi/10.1002/joom.1179>.
- [85] Sarah Allen et al. “NFTs for Art and Collectables: Primer and Outlook”. In: (2022).
- [86] *Doodles*. URL: <https://doodles.app/> (visited on 05/03/2022).
- [87] *Song a Day World*. URL: <https://songaday.world/songadao/> (visited on 05/03/2022).
- [88] *MODA DAO - Home, Whitepaper*. 2021. URL: <https://www.modadao.io/%20https://gateway.pinata.cloud/ipfs/QmX9tuaHepxwaTwTU5TnWDksWNSSm> (visited on 05/03/2022).

- [89] John Rooksby and Kristiyan Dimitrov. “Trustless Education? A Blockchain System for University Grades”. In: 2017.
- [90] Martin Brennecke et al. “The De-Central Bank in Decentralized Finance: A Case Study of MakerDAO”. In: *Proceedings of the 55th Annual Hawaii International Conference on System Sciences*. 2022.
- [91] Efthymios Chondrogiannis et al. “Using blockchain and semantic web technologies for the implementation of smart contracts between individuals and health insurance organizations”. In: *Blockchain: Research and Applications 3.2* (June 2022), p. 100049. ISSN: 2096-7209. DOI: 10.1016/J.BCRA.2021.100049. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2096720921000440>.
- [92] *Forever Isn't Free: The Cost of Storage on a Blockchain Database* | by Jamila Omaar | *IPDB Blog* | *Medium*. 2017. URL: <https://medium.com/ipdb-blog/forever-isnt-free-the-cost-of-storage-on-a-blockchain-database-59003f63e01> (visited on 01/31/2022).
- [93] *IPFS Powers the Distributed Web*. URL: <https://ipfs.io/> (visited on 02/01/2021).
- [94] *What is IPFS?* | *IPFS Docs*. URL: <https://docs.ipfs.io/concepts/what-is-ipfs/> (visited on 02/18/2022).
- [95] *Introduction to Chainlink VRF* | *Chainlink Documentation*. URL: <https://docs.chain.link/docs/chainlink-vrf/> (visited on 03/14/2022).
- [96] P Woodall. “The Big Mac Index”. In: *The Economist* (1986). URL: <https://www.economist.com/big-mac-index>.
- [97] Lorenz Breidenbach et al. *Chainlink 2.0: Next steps in the evolution of decentralized oracle networks*. 2021. (Visited on 05/03/2022).
- [98] *Introduction - The Graph Docs*. URL: <https://thegraph.com/docs/en/about/introduction/> (visited on 03/18/2022).
- [99] *Ethereum Gas Fees Are Red Hot, Infuriating Users and Boosting Rivals*. URL: <https://markets.businessinsider.com/news/currencies/ethereum-transaction-gas-fees-high-solana-avalanche-cardano-crypto-blockchain-2021-12> (visited on 05/01/2022).
- [100] Youssef Faqir-Rhazoui et al. “Effect of the Gas Price Surges on User Activity in the DAOs of the Ethereum Blockchain”. In: *Conference on Human Factors in Computing Systems - Proceedings* (May 2021). DOI: 10.1145/3411763.3451755.

- [101] Mudabbir Kaleem and Weidong Shi. “Demystifying Pythia: A Survey of ChainLink Oracles Usage on Ethereum”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12676 LNCS (2021), pp. 115–123. ISSN: 16113349. DOI: 10.1007/978-3-662-63958-0_10/FIGURES/5. URL: https://link.springer.com/chapter/10.1007/978-3-662-63958-0_10.
- [102] Daojing He et al. “Smart Contract Vulnerability Analysis and Security Audit”. In: *IEEE Network* 34 (5 Sept. 2020), pp. 276–282. ISSN: 1558156X. DOI: 10.1109/MNET.001.1900656.
- [103] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper 151.2014* (2014), pp. 1–32.
- [104] M. Jones, J. Bradley, and N. Sakimura. “JSON Web Token (JWT)”. In: (May 2015). DOI: 10.17487/RFC7519. URL: <https://www.rfc-editor.org/info/rfc7519>.
- [105] Prajakta Solapurkar. “Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario”. In: *Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016* (2016), pp. 99–104. DOI: 10.1109/IC3I.2016.7917942.
- [106] Petar Tsankov et al. “Securify: Practical Security Analysis of Smart Contracts”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* 18 (2018). DOI: 10.1145/3243734.
- [107] *CertiK Blockchain Security Leaderboard*. URL: <https://www.certik.com/> (visited on 05/01/2022).
- [108] *What Is a Smart Contract Security Audit? | Binance Academy*. URL: <https://academy.binance.com/en/articles/what-is-a-smart-contract-security-audit> (visited on 05/01/2022).
- [109] Noama Fatima Samreen and Manar H. Alalfi. “Reentrancy Vulnerability Identification in Ethereum Smart Contracts”. In: *IWBOSE 2020 - Proceedings of the 2020 IEEE 3rd International Workshop on Blockchain Oriented Software Engineering* (Feb. 2020), pp. 22–29. DOI: 10.1109/IWBOSE50093.2020.9050260.
- [110] *Rekt - Agave DAO, Hundred Finance - REKT*. URL: <https://rekt.news/agave-hundred-rekt/> (visited on 05/01/2022).
- [111] *Rekt - Revest Finance - REKT*. URL: <https://rekt.news/revest-finance-rekt/> (visited on 05/01/2022).
- [112] *Hack Solidity: Reentrancy Attack | HackerNoon*. URL: <https://hackernoon.com/hack-solidity-reentrancy-attack> (visited on 05/01/2022).

- [113] *Rekt - Deus DAO - REKT*. URL: <https://rekt.news/deus-dao-rekt/> (visited on 05/01/2022).
- [114] Van Cuong Bui et al. “Evaluating Upgradable Smart Contract”. In: *Proceedings - 2021 IEEE International Conference on Blockchain, Blockchain 2021* (2021), pp. 252–256. DOI: 10.1109/BLOCKCHAIN53845.2021.00041.
- [115] Kealan Mccusker and Brian Spector. “Qredo Network - Yellow Paper”. In: (2020). URL: <https://milagro.apache.org..>

Appendices

Appendix 1 - "Proof of Feedback" contracts

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.6.0 <0.8.0;
4
5 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
6 import "@openzeppelin/contracts/token/ERC20/ERC20Capped.sol";
7 import "@openzeppelin/contracts/access/Ownable.sol";
8
9 contract EasyFeedBackToken is ERC20Capped, Ownable {
10     constructor ()
11     ERC20("EasyFeedback", "EASYF")
12     ERC20Capped(179141000000 * 1 ether)
13     {
14         // Mint 1% of total supply
15         mint(msg.sender, (1791410000 * 1 ether));
16     }
17
18     event Burned(address indexed burner, uint256 burnAmount);
19
20     event Minted(
21         address indexed minter,
22         address indexed receiver,
23         uint256 mintAmount
24     );
25
26     function mint(address _to, uint256 _amount) public onlyOwner {
27         require(_amount > 0, "ERC20: Cannot mint 0 tokens");
28         _mint(_to, _amount);
29         emit Minted(owner(), _to, _amount);
30     }
31
32     function burn(uint256 _amount) public {
33         require(_amount > 0, "ERC20: Cannot burn 0 tokens");
34         _burn(msg.sender, _amount);
35         emit Burned(msg.sender, _amount);
36     }
37 }
```

Listing 1. ERC-20 token EASYF smart contract

```

1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;
4
5 import "@openzeppelin/contracts/access/Ownable.sol";
6
7 /**
8  @title PoF Regional regulator interface
9  */
10 interface IPoFRegionalRegulator {
11     function reward(
12         address payable,
13         address payable,
14         address payable,
15         address payable,
16         address,
17         uint256
18     ) external;
19 }
20
21 /**
22  @title PoF Internal Validator
23  @author Javier Ortin
24  @notice Proof of Feedback validator registering the users' feedback
25         then
26         randomly drawing the jurors assigned for evaluating the
27         feedback
28         and sending the final decision to the regulator node
29  */
30
31 contract PoFInternalValidator is Ownable {
32     ///Events of the contract.
33
34     event AddedJuror(address jurorAddress);
35     event RemovedJuror(address jurorAddress);
36     event UpdatedMaxEvaluation(uint256 newMaxEvaluation);
37     event FeedbackRegistered(address feedbackID);
38     event FeedbackEvaluated(
39         address feedbackID,
40         address juror,
41         uint256 originality,
42         uint256 usefulness,
43         uint256 execution,
44         uint256 alreadyEvaluated
45     );
46
47     /// State variables

```

```

46     /// @dev Map used to whitelist jurors
47     mapping(address => bool) public isJuror;
48
49     /// @dev Array containing the addresses of the jurors
50     address[] private jurors;
51
52     /// @dev total amount of jurors in the internal validator
53     uint256 totalJurors = 0;
54
55     /**
56     @notice Structure for given feedback
57     @dev customer is a payable address
58     @dev evaluated how many jurors have already evaluated
59     */
60     struct Feedback {
61         address customer;
62         address juror1;
63         address juror2;
64         address juror3;
65         mapping(address => uint256) evaluations;
66         bool registered;
67         uint256 evaluated;
68         uint256 finalEvaluation;
69     }
70
71     /// @dev map for storing feedbacks' information
72     mapping(address => Feedback) public feedbacks;
73
74     /**
75     @dev minimum amount of jurors that the internal validator must have
76         to have a fair random selection of jurors and start operating
77     */
78     uint256 public minJurors = 7;
79
80     /// @notice PoF Regional Regulator to which this validator connects
81     address public pofRegulator;
82
83     /// @notice Maximum points that a feedback can get
84     uint256 public maxEvaluation;
85
86     /// @notice Code of the country (ISO 3166) where this validator
87         operates
88     /// @dev Code should follow the ISO 3166 standard
89     string public countryCode;
90
91     constructor (
92         address _pofRegulator,

```

```

92     uint256 _maxEvaluation,
93     string memory _countryCode
94 ) {
95     pofRegulator = _pofRegulator;
96     maxEvaluation = _maxEvaluation;
97     countryCode = _countryCode;
98 }
99
100 /// Modifiers
101
102 /**
103  * @dev Reverts if called by any account that is not the PoF
104     regulator.
105  */
106 modifier onlyRegulator() {
107     require(
108         pofRegulator == _msgSender(),
109         "Regulator: caller is not PoF Regulator"
110     );
111     _;
112 }
113
114 /**
115  * @dev Throws if called by any account that is not a juror.
116  */
117 modifier onlyJuror() {
118     require(
119         isJuror[_msgSender()],
120         "Juror: caller is not a PoF Internal validator juror"
121     );
122     _;
123 }
124
125 /// Functions
126
127 /**
128  * @param juror new address to be added as a juror
129  * @dev adds a new juror address to the map and array of jurors
130  */
131 function addJuror(address juror) public onlyOwner {
132     require(juror != address(0x0), "Zero address cannot be an juror
133         ");
134     require(!isJuror[juror], "Address is already a juror");
135     jurors.push(juror);
136     isJuror[juror] = true;
137     totalJurors++;
138     emit AddedJuror(juror);
139 }

```

```

137
138  /**
139  @param juror address to be removed
140  @dev removes a juror address from the map and array of jurors
141  */
142  function removeJuror(address juror) public onlyOwner {
143      require(isJuror[juror], "Address must be a juror");
144      for (uint256 i = 0; i < totalJurors; i++) {
145          if (jurors[i] == juror) {
146              jurors[i] = jurors[totalJurors - 1];
147              jurors.pop();
148              isJuror[juror] = false;
149              totalJurors--;
150              emit RemovedJuror(juror);
151          }
152      }
153  }
154
155  /**
156  @notice Set new maximum evaluation that a feedback can have
157  @param _maxEvaluation new maximum evaluation points
158  */
159  function setMaxEvaluation(uint256 _maxEvaluation) public
160      onlyRegulator {
161      maxEvaluation = _maxEvaluation;
162      emit UpdatedMaxEvaluation(_maxEvaluation);
163  }
164
165  /**
166  @notice Function to register the feedback in the blockchain
167  @param feedbackID id of the given feedback in the form of a valid
168      address.
169  */
170  function askFeedbackEvaluation(address feedbackID) public {
171      require(
172          totalJurors >= minJurors,
173          "There are not enough jurors to use the PoF"
174      );
175      require(!feedbacks[feedbackID].registered, "Feedback must be
176          new.");
177      feedbacks[feedbackID].customer = _msgSender();
178      address juror1;
179      address juror2;
180      address juror3;
181      (juror1, juror2, juror3) = randomJurors(feedbackID);
182      feedbacks[feedbackID].juror1 = juror1;
183      feedbacks[feedbackID].juror2 = juror2;

```

```

181     feedbacks[feedbackID].juror3 = juror3;
182     feedbacks[feedbackID].registered = true;
183     emit FeedbackRegistered(feedbackID);
184 }
185
186 /**
187  @notice Function that only jurors can call to evaluate the feedback
188         they have been assigned
189  @param feedbackID id of the given feedback in the form of a valid
190         address.
191  @param originality evaluation points for this criteria
192  @param usefulness evaluation points for this criteria
193  @param execution evaluation points for this criteria
194  */
195 function evaluateFeedback(
196     address feedbackID,
197     uint256 originality,
198     uint256 usefulness,
199     uint256 execution
200 ) public onlyJuror {
201     require(
202         _msgSender() == feedbacks[feedbackID].juror1 ||
203         _msgSender() == feedbacks[feedbackID].juror2 ||
204         _msgSender() == feedbacks[feedbackID].juror3,
205         "The juror must have the feedback assigned for evaluation"
206     );
207     require(
208         originality <= maxEvaluation &&
209         usefulness <= maxEvaluation &&
210         execution <= maxEvaluation,
211         "Evaluation cannot be more than the maximum"
212     );
213     feedbacks[feedbackID].evaluations[_msgSender()] =
214         (originality + usefulness + execution) /
215         3;
216     feedbacks[feedbackID].evaluated++;
217     emit FeedbackEvaluated(
218         feedbackID,
219         _msgSender(),
220         originality,
221         usefulness,
222         execution,
223         feedbacks[feedbackID].evaluated
224     );
225     if (feedbacks[feedbackID].evaluated == 3) {
226         finalEvaluation(feedbackID);
227     }

```



```

226     }
227
228     /**
229     @notice Calculation of the final evaluation. Arithmetic mean.
230     @param feedbackID id of the given feedback in the form of a valid
           address.
231
232     */
233     function finalEvaluation(address feedbackID) private {
234         uint256 evaluation = (
235             feedbacks[feedbackID].evaluations[feedbacks[feedbackID].
                juror1] +
236             feedbacks[feedbackID].evaluations[feedbacks[feedbackID].
                juror2] +
237             feedbacks[feedbackID].evaluations[feedbacks[feedbackID].
                juror3]) /
238             3;
239         feedbacks[feedbackID].finalEvaluation = evaluation;
240         IPoFRegionalRegulator(pofRegulator).reward(
241             payable(feedbacks[feedbackID].customer),
242             payable(feedbacks[feedbackID].juror1),
243             payable(feedbacks[feedbackID].juror2),
244             payable(feedbacks[feedbackID].juror3),
245             feedbackID,
246             evaluation
247         );
248     }
249
250     /**
251     @notice Algorithm for randomly selecting the jurors.
252     @param feedbackID id of the given feedback used as seed.
253     @return address 3 addresses of the jurors selected.
254     @dev this function can only be called internally
255     */
256     function randomJurors(address feedbackID)
257         private
258         view
259         returns (
260             address,
261             address,
262             address
263         )
264     {
265         uint256 index = uint256(uint160(address(feedbackID)));
266         uint256 i1 = index % totalJurors;
267         uint256 i2 = (index / totalJurors) % totalJurors;
268         uint256 i3;

```

```

269     if (i2 == i1) i2 = (i2 + 1) % totalJurors;
270     if (i1 != 0 && i2 != 0) i3 = (i1 + i2) % totalJurors;
271     else {
272         i3 = (i1 + i2 + 1) % totalJurors;
273         if (i3 == 0) ++i3;
274     }
275     return (jurors[i1], jurors[i2], jurors[i3]);
276 }
277 }

```

Listing 2. Internal validator smart contract

```

1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;
4
5 import "@openzeppelin/contracts/access/Ownable.sol";
6 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
7 import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
8
9 /**
10 @title EASYF price feed oracle interface
11 */
12 interface IEASYFPriceFeed {
13     function priceFeed() external returns (uint256);
14 }
15
16 /**
17 @title EASYF price feed oracle interface
18 */
19 interface IPoFInternalValidator {
20     function setMaxEvaluation(uint256) external;
21 }
22
23 /**
24 @title PoF Regional regulator smart contract
25 @author Javier Ortin
26 @notice PoF smart contract in charge of sending the rewards to the end
    user
27 @dev To this contract connect several PoF internal validators and it
    has only one
28     call to an external oracle to provide the market price of EASYF
29 */
30 contract PoFRegionalRegulator is Ownable {
31     using SafeERC20 for IERC20;
32
33     ///Events of the contract.
34

```

```

35     /// @dev Amount in EASYF.
36     event Rewarded(
37         address beneficiary,
38         address feedbackID,
39         uint256 amount,
40         uint256 evaluation
41     );
42     event UpdatedMaxReward(uint256 newMaxReward);
43     event UpdatedJurorsReward(uint256 newMaxReward);
44     event UpdatedBigMacIndex(uint256 newBigMacIndex);
45     event UpdatedMaxEvaluation(uint256 newMaxEvaluation);
46     event AddedInternalValidator(address newInternalValidator);
47     event RemovedInternalValidator(address newInternalValidator);
48
49     /// State variables
50
51     /// @dev Map used to store which feedbacks have been already
52     ///     rewarded to
53     ///     avoid rewarding the same feedback twice
54     mapping(address => bool) public rewarded;
55
56     /// @dev Map used to whitelist internalValidators
57     mapping(address => bool) public isInternalValidator;
58     address[] public internalValidators;
59
60     /// @notice ERC-20 token: EASYF used as a reward
61     address public easyf;
62
63     /// @notice EASYF price feed oracle
64     address public easyfPriceFeed;
65
66     /**
67     @notice Maximum amount rewarded.
68         Safety feature. In case some oracle is hacked
69         there is still a maximum amount that cannot be surpassed
70     @dev     Units are EASYF (ERC-20 ether unit)
71     */
72     uint256 public maxReward;
73
74     /// @notice reward for jurors for each evaluation
75     /// @dev     Units are EASYF (ERC-20 ether unit)
76     uint256 public jurorsReward;
77
78     /// @notice BigMac index of the region where this regulator is
79     ///     implemented
80     /// @dev it should be multiplied by 100 to work with integers
81     uint256 public bigMacIndex;

```

```

80
81     /// @notice Maximum points that a feedback can get
82     /// @dev In the future might be decided to take some different
83     scala
84     /// but in general it should not be changed
85     uint256 public maxEvaluation;
86
87     /// @notice Code of the country (ISO 3166) this node regulates
88     /// @dev Code should follow the ISO 3166 standard
89     string public countryCode;
90
91     bool internal locked;
92
93     constructor (
94         address _easyf,
95         address _easyfPriceFeed,
96         uint256 _maxReward,
97         uint256 _jurorsReward,
98         uint256 _bigMacIndex,
99         uint256 _maxEvaluation,
100        string memory _countryCode
101    ) {
102        easyf = _easyf;
103        maxReward = _maxReward;
104        jurorsReward = _jurorsReward;
105        bigMacIndex = _bigMacIndex;
106        easyfPriceFeed = _easyfPriceFeed;
107        maxEvaluation = _maxEvaluation;
108        countryCode = _countryCode;
109    }
110
111    /// Modifiers
112
113    /**
114     * @dev Throws if called by any account that is not a PoF Internal
115     * validator.
116     */
117    modifier onlyInternalValidator() {
118        require (
119            isInternalValidator[msg.sender],
120            "InternalValidator: caller is not a PoF Internal validator"
121        );
122        _;
123    }
124
125    /**
126     * @dev Throws if trying a reentrancy attack.

```

```

125     */
126     modifier noReentrant () {
127         require (!locked, "Locked: ongoing contract call");
128         locked = true;
129         _;
130         locked = false;
131     }
132
133
134     /// Functions
135
136     /**
137     @notice Set new maximum amount that can be rewarded
138     @param _maxReward new maximum reward amount of EASYF tokens
139     */
140     function setMaxReward(uint256 _maxReward) public onlyOwner {
141         require (_maxReward > 0, "Maximum reward cannot be zero");
142         maxReward = _maxReward;
143         emit UpdatedMaxReward(maxReward);
144     }
145
146     /**
147     @notice Set new bigMac Index. Usually updated once or twice per
148         year.
149     @param _bigMacIndex bigMac index of the country
150     */
151     function setBigMacIndex(uint256 _bigMacIndex) public onlyOwner {
152         require (_bigMacIndex > 0, "BigMac index cannot be zero");
153         bigMacIndex = _bigMacIndex;
154         emit UpdatedBigMacIndex(bigMacIndex);
155     }
156
157     /**
158     @notice Set new maximum evaluation that a feedback can have
159     @param _maxEvaluation new maximum evaluation points
160     */
161     function setMaxEvaluation(uint256 _maxEvaluation) public onlyOwner
162     {
163         require (_maxEvaluation > 0, "Maximum evaluation cannot be zero"
164             );
165         maxEvaluation = _maxEvaluation;
166         for (uint256 i = 0; i < internalValidators.length; i++) {
167             IPoFInternalValidator(internalValidators[i]).
168                 setMaxEvaluation(
169                     _maxEvaluation
170                 );
171         }

```

```

168     emit UpdatedMaxEvaluation(_maxEvaluation);
169 }
170
171 /**
172  @param internalValidatorAddress new address to be added as an
        internal validator
173  @dev adds a new internal validator address to the map
        internalValidators
174  */
175  function addInternalValidator(address internalValidatorAddress)
176      public
177      onlyOwner
178  {
179      require(
180          internalValidatorAddress != address(0x0),
181          "Zero address cannot be an internal validator"
182      );
183      require(
184          !isInternalValidator[internalValidatorAddress],
185          "Address is already an internal validator"
186      );
187      internalValidators.push(internalValidatorAddress);
188      isInternalValidator[internalValidatorAddress] = true;
189      emit AddedInternalValidator(internalValidatorAddress);
190  }
191
192 /**
193  @param internalValidatorAddress internal validator address to be
        removed from the map
194  @dev removes a new internal validator address to the map
        internalValidators
195  */
196  function removeInternalValidator(address internalValidatorAddress)
197      public
198      onlyOwner
199  {
200      require(
201          isInternalValidator[internalValidatorAddress],
202          "Address is not an internal validator"
203      );
204      for (uint256 i = 0; i < internalValidators.length; i++) {
205          if (internalValidators[i] == internalValidatorAddress) {
206              internalValidators[i] = internalValidators[
207                  internalValidators.length - 1
208              ];
209              internalValidators.pop();
210              isInternalValidator[internalValidatorAddress] = false;

```

```

211         emit RemovedInternalValidator(internalValidatorAdress);
212     }
213 }
214 }
215
216 /**
217 @notice Calculates the amount of token an evaluation is worth
218 @param evaluation amount of points a feedback has been given
219 @dev only to be called by reward function.
220     No need to divide bigMacIndex and easyfPrice since both are
221     multiplied by 100 and
222     one is diving the other
223 */
224 function calculateAmount(uint256 evaluation) private returns (
225     uint256) {
226     uint256 calculatedReward = (2 * bigMacIndex * evaluation * 1
227         ether) /
228         (maxEvaluation * IEASYFPriceFeed(easyfPriceFeed).priceFeed
229             ());
230     require(
231         (maxReward * 1 ether) >= calculatedReward,
232         "Reward amount cannot be bigger than maximum reward"
233     );
234     return calculatedReward;
235 }
236
237 /**
238 @notice Calculates and sends the reward to the customer
239 @param customer address of the customer that sent the feedback
240 @param juror1 address of the juror1 to be rewarded
241 @param juror2 address of the juror2 to be rewarded
242 @param juror3 address of the juror3 to be rewarded
243 @param feedbackID id of the given feedback. Use to avoid double-
244     rewarding
245 @param evaluation points the evaluation has received.
246 */
247 function reward(
248     address payable customer,
249     address payable juror1,
250     address payable juror2,
251     address payable juror3,
252     address feedbackID,
253     uint256 evaluation
254 ) public onlyInternalValidator noReentrant{
255     require(evaluation > 0, "Evaluation cannot be zero");
256     require(
257         evaluation <= maxEvaluation,

```

```

253         "Evaluation points cannot be bigger than the maximum points
254         ";
255     require(!rewarded[feedbackID], "Feedback has been already
256         rewarded");
257
258     /// Calculate how many tokens an evaluation means
259     uint256 amount = calculateAmount(evaluation);
260     require(
261         IERC20(easyf).balanceOf(address(this)) >= amount,
262         "Not enough liquidity in the regional regulator"
263     );
264
265     /// Reward the customer
266     IERC20(easyf).approve(address(this), amount);
267     IERC20(easyf).safeTransferFrom(address(this), customer, amount)
268         ;
269
270     /// Reward the jurors
271     IERC20(easyf).approve(address(this), jurorsReward);
272     IERC20(easyf).safeTransferFrom(address(this), juror1,
273         jurorsReward);
274     IERC20(easyf).approve(address(this), jurorsReward);
275     IERC20(easyf).safeTransferFrom(address(this), juror2,
276         jurorsReward);
277     IERC20(easyf).approve(address(this), jurorsReward);
278     IERC20(easyf).safeTransferFrom(address(this), juror3,
279         jurorsReward);
280
281     /// Set the feedbackID as already rewarded
282     rewarded[feedbackID] = true;
283
284     emit Rewarded(customer, feedbackID, amount, evaluation);
285 }
286
287 /**
288 @notice Sends all liquidity to the owner who later can burn it or
289 relocated
290 */
291 function withdraw() public onlyOwner {
292     uint256 amount = IERC20(easyf).balanceOf(address(this));
293     /// Send funds to the owner of the contract
294     IERC20(easyf).approve(address(this), amount);
295     IERC20(easyf).safeTransferFrom(address(this), owner(), amount);
296 }

```

Listing 3. Regional Regulator smart contract


```

1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;
4
5 import "@openzeppelin/contracts/access/Ownable.sol";
6 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
7 import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
8
9 /**
10 @title Company node
11 @author Javier Ortin
12 @notice Proof of Feedback company node in charge of sending the rewards
13         to the end user
14 @dev Liquidity should be provided by the company
15 */
16 contract PoFCompany is Ownable {
17     using SafeERC20 for IERC20;
18
19     ///Events of the contract.
20
21     /// @dev Amount in EASYF.
22     event Rewarded(
23         address beneficiary,
24         address feedbackID,
25         uint256 amount,
26         uint256 evaluation
27     );
28     event UpdatedMaxReward(uint256 newMaxReward);
29     event UpdatedMaxEvaluation(uint256 newMaxEvaluation);
30
31     /// State variables
32
33     /// @dev Map used to store which feedbacks have been already
34         rewarded to
35     ///         avoid rewarding the same feedback twice
36     mapping(address => bool) public rewarded;
37
38     /// @notice ERC-20 token contract: EASYF used as a reward
39     address public easyf;
40
41     /// @notice Maximum amount rewarded
42     /// @dev in eth -> EASYF token
43     uint256 public maxReward;
44
45     /// @notice Maximum points that a feedback can get
46     uint256 public maxEvaluation;

```

```

46     string public company_name;
47
48     /// @notice Contract constructor
49     constructor(
50         address _easyf,
51         uint256 _maxReward,
52         uint256 _maxEvaluation,
53         string memory _company_name
54     ) {
55         easyf = _easyf;
56         maxReward = _maxReward;
57         maxEvaluation = _maxEvaluation;
58         company_name = _company_name;
59     }
60
61     /// Functions
62
63     /**
64     Set new maximum amount that can be rewarded
65     @param _maxReward new maximum reward amount of EASYF tokens
66     */
67     function setMaxReward(uint256 _maxReward) public onlyOwner {
68         require(_maxReward > 0, "Maximum reward cannot be zero");
69         maxReward = _maxReward;
70         emit UpdatedMaxReward(maxReward);
71     }
72
73     /**
74     Set new maximum evaluation that a feedback can have
75     @param _maxEvaluation new maximum evaluation points
76     */
77     function setMaxEvaluation(uint256 _maxEvaluation) public onlyOwner
78     {
79         require(_maxEvaluation > 0, "Maximum evaluation cannot be zero"
80             );
81         maxEvaluation = _maxEvaluation;
82         emit UpdatedMaxEvaluation(_maxEvaluation);
83     }
84
85     /**
86     @notice Calculates the amount of token an evaluation is worth
87     @param evaluation amount of points a feedback has been given
88     @dev since this is own by a company they don't need to follow the
89         oracle standard of the PoF. They still would theoretically
90         follow it
91     */
92     function calculateAmount(uint256 evaluation)

```

```

90     private
91     view
92     returns (uint256)
93     {
94         return (maxReward * evaluation * 1 ether) / maxEvaluation;
95     }
96
97     /**
98     @notice Calculates and sends the reward to the customer
99     @param customer address of the customer that sent the feedback
100    @param feedbackID id of the given feedback. Use to avoid double-
101           rewarding
102    @param evaluation points the evaluation has received.
103    */
104    function reward(
105        address payable customer,
106        address feedbackID,
107        uint256 evaluation
108    ) public onlyOwner {
109        require(evaluation > 0, "Evaluation cannot be zero");
110        require(
111            evaluation <= maxEvaluation,
112            "Evaluation points cannot be bigger than the maximum points
113             "
114        );
115        require(!rewarded[feedbackID], "Feedback has been already
116             rewarded");
117
118        /// Calculate how many tokens an evaluation is worth
119        uint256 amount = calculateAmount(evaluation);
120        require(
121            IERC20(easyf).balanceOf(address(this)) >= amount,
122            "Not enough liquidity in the company node"
123        );
124
125        /// Reward the customer
126        IERC20(easyf).approve(address(this), amount);
127        IERC20(easyf).safeTransferFrom(address(this), customer, amount)
128            ;
129
130        /// Set the feedbackID as already rewarded
131        rewarded[feedbackID] = true;
132
133        emit Rewarded(customer, feedbackID, amount, evaluation);
134    }
135 }

```

Listing 4. Company validator smart contract

```

1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;
4
5 import "@openzeppelin/contracts/access/Ownable.sol";
6
7 /**
8  @title Oracle for the EASYF token price
9  @author Javier Ortin
10 @notice price feed for other contracts to get the USD trading price of
    EASYF token
11 @dev The owner can update the current trading price (USD) of the EASYF
    token.
12 Interface to add in the contract:
13 interface IEASYFPriceFeed {
14     function priceFeed() external returns (uint256);
15 }
16 */
17 contract EASYFPriceFeed is Ownable {
18     /// Events
19     event UpdatedPriceFeed(uint256 price);
20
21     /// State variables
22     /// @notice state variable to call for getting EASYF price
23     uint256 public priceFeed;
24
25
26     constructor(
27         uint256 _priceFeed
28     ) {
29         priceFeed = _priceFeed;
30     }
31
32     /**
33     Sets the current EASYF token price.
34     @param price new price in USD
35     */
36     function setPriceFeed(uint256 price) public onlyOwner {
37         priceFeed = price;
38         emit UpdatedPriceFeed(priceFeed);
39     }
40 }

```

Listing 5. EASYF price feed oracle smart contract

```

1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;

```

```

4
5 import "@openzeppelin/contracts/access/Ownable.sol";
6
7 /**
8  @title Contract for testing the cost of storing feedback directly in
      the blockchain
9  @author Javier Ortin
10 @notice simple contract that stores bytes of any kind of data assigned
      to an id.
11 @dev Not intended to be used in production
12 */
13 contract FeedbackStorage is Ownable {
14     event FeedbackRegistered(address feedbackID, address customer);
15
16     /**
17     @notice Structure for given feedback
18     */
19     struct Feedback {
20         address customer;
21         bytes feedbackData;
22         bool registered;
23     }
24
25     /// @notice map for given feedbacks id -> Feedback struct
26     mapping(address => Feedback) public feedbacks;
27
28     /**
29     Insertion of the feedback in to the blockchain
30     @param feedbackID id of the given feedback in the form of a valid
      address.
31     @param feedbackData bytes contain the encrypted feedback data
32     */
33     function registerFeedback(address feedbackID, bytes calldata
      feedbackData)
34         public
35     {
36         require(!feedbacks[feedbackID].registered, "Feedback must be
      new.");
37         feedbacks[feedbackID].customer = _msgSender();
38         feedbacks[feedbackID].feedbackData = feedbackData;
39         feedbacks[feedbackID].registered = true;
40         emit FeedbackRegistered(feedbackID, feedbacks[feedbackID].
      customer);
41     }
42 }

```

Listing 6. Feedback storing smart contract