Cryptoshares for digital cooperatives

Author: Zvezdin Besarabov

National high school of mathematics and natrual sciences, 11-th grade, Sofia, Bulgaria

zvezdin@obecto.com

Under the guidance of Todor Kolev

Founder of Obecto Cooperative, Sofia, Bulgaria

tkolev@obecto.com

Abstract

The project implements a decentralized autonomous program (cryptocontract), which allows the establishment of digital cooperatives including people who work together on group projects and split incoming revenue proportionally to the labor invested by each participant. This model of organization is an alternative and an addition to the traditional capital company, created and owned by shareholders, which afterwards hire employees. With the model of digital cooperatives, the main focus is on the work done by each person. Profits from the cooperative are split proportionally based on the invested labor. This leads to a continuous change in the proportions between the shares of the participants in the cooperative, and makes the administration of the legal arrangements between them difficult. Because of these specifics, the practical implementation of such internal cooperative relationships can be realized only by using an information system in additional to the traditional (paper) contracts. In this project, we have implemented a cryptocontract in the programming language Solidity which is then executed in the Ethereum blockchain. The administration is done from a web application, which instead of a traditional back-end, creates a direct connection to the Ethereum blockchain network and uses it to process transactions and store data. Thus, nobody is able to manipulate the data outside of the cryptocontract terms and no data will ever be lost, since the processing and storage is split among the participating computers in the Ethereum network.

Contents

1. Introduction

Over the past 20 years there has been a boom of innovation and an increasing amount of new businesses start, especially in the technology sector. Access to information, knowledge and capital is more than ever before. A huge number of entrepreneurial programs, accelerators and Venture Capital funds have been created, which support the creation of new business with unprecedented scale for the human civilization so far.

However, no significant improvements in the viability of starting companies can be observed [1] (Figure 1).

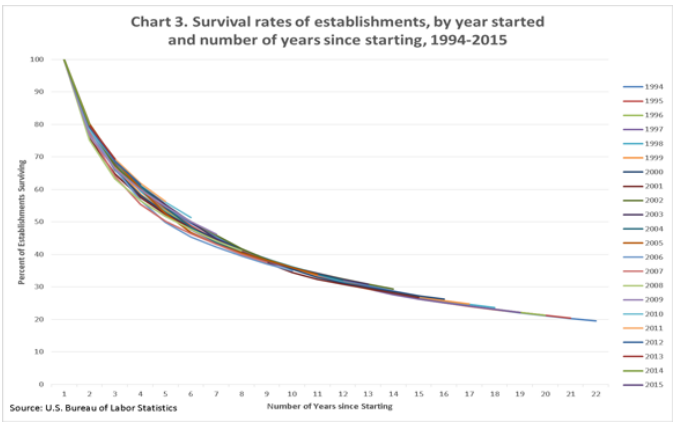

Figure 1: Percentage of the survived companies up to a certain age based on the year of foundation 1994-2015

One of the main reasons for the failure of starting businesses are the problems associated with the team. They have been ranked third by importance and identified as crucial for 23% of cases of failure of young companies [2].

To attract human resources, emerging businesses have two traditional instruments - attracting financial capital with which to hire employees or attract new partners who, free of charge, are willing to help with labor, knowledge and contacts to start the business, until the company becomes profitable.

Attracting capital by young companies is naturally hampered by the enormous risk associated with starting a new business. Because of that, venture capitalists are betting heavily on companies that can bring at least tenfold return on investment. These companies are often called "unicorns" because of their rarity. On the other hand, many good companies, which would generate triple or quadruple returns, remain unfunded and can not offer competitive conditions in the labor market. They have difficulties hiring workers, and those who get hired are often below the average professional level.

Attracting new partners, who can help with non-financial investment, solves the problem with the lack of initial capital. However, new risk arises from deterioration of the personal relationships between members, with the most common cause for discord between the partners in a startup being a dispute over the allocation of shares. This is especially pronounced when the founders invest labor instead of starting capital.

Commercial Law requires that any issuance of new shares in a company must be associated with a capital investment. It is theoretically possible to valuate the invested labor as capital investment, but such alignment is an unnatural approach to most legal systems, being applied with difficulty and with many additional risks. Because of these difficulties, the allocation of shares is forced to occur at the inception of the company, with changes being made rarely.

On the other hand, co-founders invest new labor daily, and thus, their expectations for the share they should have changes. Unless they invest their effort in the exact proportions in which they agreed Initially, imbalance between the inputted labor by each participant and his share will certainly occur. Thus, if the decisions on the allocation of new shares are blocked by one or more members, these imbalances could lead to the collapse of the company without any other problem in the business.

1.1. Proposed solution

In the current study, we propose an approach through which startups can attract investments in the form of labor, by forming digital cooperatives.

Such digital cooperatives are structures located within traditional capital based companies. Their function is to aggregate investment in the form of labor, which can be naturally measured as time. On the other hand, the company itself aggregates investment of capital, which can be measured in money. The two structures are connected by signing a Cooperation Agreement between the company and the participants in the cooperative (cooperators). The company utilizes the work of the cooperators and in turn directs part of its revenue back to the cooperative, where it is distributed based on the individual contribution of each cooperator (Figure 2).

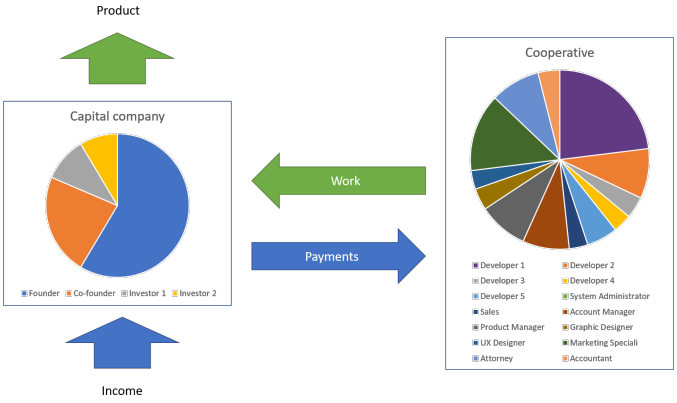

Figure 2: A digital cooperative and its connection with the capital company

This separation allows for different rules for repartitioning in each of the structures to be built. On one hand

the company founders, who have defined the business idea and gave the initial impetus, together with the investors, who invested capital in the company, can structure their relationships in a traditional way by taking stakes from the company itself. On the other hand, people investing time and effort working for the company can receive a proportionate share of the future benefits, without complications to the ownership structure of the company.

Despite the existing advantages, the organization of such a cooperative, is a difficult task that is hard to implement within the traditional legal system, which relies on signed paper documents. The complex and dynamic relationships between the cooperators, and their interactions with the capital-based company can be managed only by using an information system.

Unfortunately, the traditional information systems have a high degree of centralization by working on corporate servers. This makes the data therein not completely reliable, since there is a possibility that it can be manipulated by a hacker or a malicious employee who can access these centralized servers maliciously.

The advent of the blockchain technology offers a solution for this problem. The technology relies on consensus among many independent agents, rather than a centralized system. The initial application of the blockchain technology was to carry out monetary transactions and exchange cryptocurrency (Bitcoin [3]), but in the recent years the technology has developed rapidly and gets more and more widespread applications.

One of the important advances is the development of the Ethereum project [4]. It allows the realization of not only currency transactions in the blockchain, but also the implementation of cryptocontracts, representing autonomous software agents whose code is executed in a decentralized way among all computers in the Ethereum network. This decentralization and the cryptographic properties of the blockchain ensure that the cryptocontracts will be executed securely, based on their underlying rules, and that their data can not be manipulated beyond these rules.

The information system system we suggest, is based on an Ethereum cryptocontract which ensures a maximum transparency and data security, while guaranteeing that the underlying rules are strictly followed.

The project aims to enable young companies with insufficient capital to engage people who can invest in the success of the business with their time and labor. This would reduce the number of start-up businesses that fail due to a lack of capital and will allow more innovative products to reach the market. Even one percent increase in the success rate of start-ups can lead to huge amounts of value generated for the global economy.

1.2. Introduction to Ethereum

Figure 3: The value of one Ether based on the US dollar since the creation of Ethereum

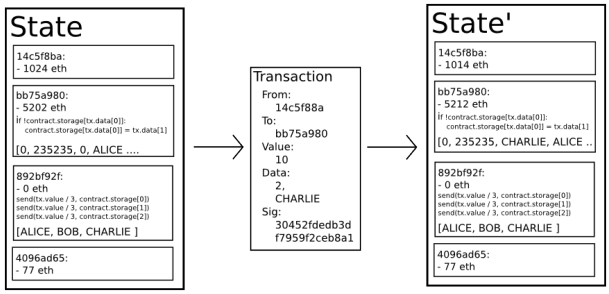

Figure 4: Ethereum's state transition system

Satoshi Nakamoto's introduction of Bitcoin in November 2008 has often been hailed as a radical development in money and currency as it's the first example of a digital asset which simultaneously has no backing or "intrinsic value" and no centralized issuer or controller. However, another arguably more important part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus. The most important aspect of such technology is the absence of an intermediary (centralized server, bank, company, etc.) between the originator and the recipient, as any changes to the data in this chain are made by consensus among all members of a decentralized network. Thus avoiding the need to trust third parties.

The blockchain platforms are already revolutionizing the technological development advancements, having a wide range of applications besides finance [5]. International laboratories for research in this direction are currently being established. Entrepreneurship worldwide focuses on the blockchain technology, developing systems for supply chain tracking, payments, privacy, consensus between members, evaluation of assets and other [6]. The blockchain provides the opportunity to restore online freedom and to transform the Internet entirely [7].

Ethereum is a cryptocurrency [8], where consensus in a decentralized network of nodes is achieved by asymmetric cryptography and blockchain technology [4]. At the base of each cryptocurrency stands the blockchain. The ledger (blockchain) of a cryptocurrency can be thought of as a state transition system, where there is a "state" consisting of the ownership status of all existing currency

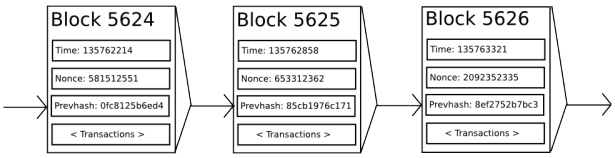

Figure 5: The basic structure of the blockchain

tokens and a "state transition function" that takes a state and a transaction, checks the input for validity, and outputs a new state which is the result (Figure 4). The new state is saved as a new entry in the blockchain. Thus each transaction on network is traceable verifiable by any entity, instead of trusting third parties for this purpose. In the case of Ethereum, the state consists of account objects, containing information about balance, and a code field, if the designated account is not owned by person but rather an autonomous application.

The structure of the blockchain can be easily implemented on a centralized server, which controls the state. But, to allow the creation and management of the decentralized system of a cryptocurrency, the state transition system has to be combined with a consensus system, ensuring the agreement between each member of the network on the order of transactions. The most common consensus process requires nodes in the network to continuously attempt to produce packages of transactions called "blocks" [9]. The compute complexity of the block creation is set so that on average every 15 seconds (in the case of Ethereum) a node in the network manages to create a valid candidate for the next block in the chain. When such is found, it is propagated throughout the network, with every node performing individual validation of the data. Upon successful validation, the network accepts the new block as the only version of "the truth". The creator of the block receives a certain amount of cryptocurrency tokens called "block reward". Each subsequent block with the same transactions is invalid and the network continues with the creation of the next block. Thus building consensus in a decentralized network is a competition between all nodes based on which will create a valid block first.

The verification of a new block checks the validity of the packaged transactions (whether they have been executed before, whether the sender's funds are sufficient, etc.), the result of executing the proof of work algorithm and other verifications on the detailed structure of the block. If all checks are completed successfully, the state transition function is called to transition to a new network state for each transaction in the block. Finally, the latest state is recorded in the block, which is appended to the blockchain. The blocks in the chain are connected, with each subsequent block pointing to the hash of the previous one (Figure 5).

Proof of work is an algorithm that aims to hamper the creation of new candidate blocks. Otherwise, every node will be able to attack the network, by creating its own attacked version of the blockchain and luring the others to migrate to it. In the case of Ethereum, proof of work verifies whether the hash from the hashing algorithm Ethash on the entire block, turned into a 256-bit number, is smaller than a certain dynamic variable. The only way to create a valid block is simply trial and error, repeatedly incrementing a counter in the block data and seeing if the new hash matches.

The entire system ensures security in a decentralized network. The only compromising case is if one node in the network holds 51% of the total hashing power. In oder to pull off a successful attack, the node must create a valid alternative version of the main blockchain (fork). However, any change in the state at any point in the chain will render all successive blocks from that moment on invalid for this alternative version of the blockchain. The node will have to recreate each subsequent block in the chain, for each respectively finding a valid solution to the proof of work algorithm. If the alternative version overtakes and outstrips the main blockchain in length, all nodes of the network will move to the alternative (and attacked) version and the changes in it will gain value. But in order for the attacker to overtake everyone else in the network, he will have to hold more than half of the total computing power, which at large enough networks is practically impossible. To solve this potential threat from centralized control, Ethereum will move to a different consensus algorithm - proof of stake [10], where instead of computing power, nodes invest in a particular proposal for a new block with digital tokens of the cryptocurrency. The next block is selected pseudo-randomly from the proposals.

1.3. Autonomous decentralized applications

The intent of Ethereum is to create an alternative protocol for building autonomous decentralized applications with particular emphasis on situations where rapid development time, security for small and rarely used applications, and the ability of different applications to very efficiently interact, are important. In the structure of the Ethereum state, except for account balances, code can be contained for execution in the form of transactions in the blockchain. Because the decentralized Ethereum network is bound on consensus, no one has control over the code execution or storage. A decentralized application can not be stopped, deleted or attacked in any way, unless the underlying rules set in the code allow it. With such decentralized applications we have a guarantee that the code will always execute without outside interference (except in the case of 51% attack). Because such an application resembles contractual relations, autonomous decentralized applications can serve as legal contracts, and are therefore referred to as "cryptocontracts" or simply contracts.

Unlike other cryptocurrencies, such as Bitcoin, the decentralized applications in Ethereum have access to their own memory to store their current state, can read data directly from the blockchain, can execute code of other

Figure 6: Tracking the current Ethereum network status in real time

decentralized applications in network and support a dedicated programming language called “Solidity”. The structure of such an application is similar to OOP programming, where each may comprise local and global variables, constructor, and private and public methods. The execution of the code is implemented in the Ethereum virtual machine.

Publishing such contracts is done by sending its compiled bytecode as a transaction without a recipient. The data is recorded in the next state of the network and the contract’s constructor is executed. The transaction returns an address, on which the published contract can be accessed by invoking its methods. Each method call is executed by a network transaction.

Because the contract’s state is part of the network state in the blockchain, all contract variables can be read and no outside access control is done for public method execution.

Ethereum builds a model against denial of service attacks. In order to prevent accidental or hostile infinite loops or other computational wastage in code, each transaction is required to set a limit to how many computational steps of code execution it can use. This limit should be lower than the global limit in the network. The fundamental unit of computation is ”gas”, equal to one computational step. Exceeding the limit stops the code execution and any changes to the state of the contract are reverted. The intent of the fee system is to require an attacker to pay proportionately for every resource that they consume, including computation, bandwidth and storage.

The current status of the network can be accessed easily in real time [11] (Figure 6).

2. Functionality of the proposed solution

The proposed solution is an electronic system that allows entrepreneurs to establish and operate digital cooperatives to attract investments in the form labor for the development of their activity. The system stores data for each cooperator, traces his labor investment and ensures fair and secure distribution of future revenue. Each activity and change in the cooperative is reflected in the system.

The cooperative is a structure of cooperators and a coordinator. The coordinator’s role is to coordinate the cooperative, as he is the person responsible for the actions of the cooperative.

Each cooperator is registered in the system. Cooperators are grouped into groups based on common activity and common source of revenue. The groups are called “Bonus pools”, or, in short “pools”. Each group receives a share of a certain financing source, which is managed by the company attached to the cooperative. It is possible for several groups to be funded from a single source, taking their share from it. All investments of the cooperators are carried out at group level. Revenue distribution executes after the transfer of funds to the group from its source of revenue, which are distributed between group members in proportion to their investment. The addition of a member in a group follows with his first investment in the activity of that group.

The investment of each cooperator is realized by accumulating money debt to him, proportional to the value of the labor done from his side, which is bought (decreased) with the transfer of income.

The balance of each cooperator in each group is measured in 3 separate digital currencies: tokens, slices and money. The tokens represent the current (unbought) debt to this member. Slices measure his overall investment (total debt accumulated over time) and money are equal to the funds transferred to him, which are monitored in the system only for transparency. Each cooperator holds a separate balance of tokens and slices for each group he invests in.

Reflecting the investment of the members in the system is carried out by the coordinator on a certain period of time (e.g. monthly) by issuing tokens to each cooperator in proportion to his investment in a particular group for that period.

The token issuing in each group is governed by the following parameters:

- **Total budget of the coordinator**: Total amount of tokens, which the coordinator is allowed to issue. When the budget runs out, the coordinator should propose a refill by initiating a voting in the group;
- **Monthly quota of the coordinator**: Total number of tokens that the coordinator has the right to issue on monthly basis. Payments outside the quota are a subject to a vote between the members of the group;
- **Monthly limit for transactions to a cooperator**: Threshold for issuing tokens to a member of the group. If a transaction is over this limit, a ballot is performed among the group members to accept the change;
- **Monthly micro-bonus quota per member**: Each member is entitled each month to issue tokens at his discretion to members of the group until his monthly allowance is used up. The transaction is not subject to a ballot.

The thresholds of the parameters can be changed by a vote among members of the group.

Full transparency is established in the cooperative, as all members are notified about any activity and change in the cooperative (section 3.3).

In case revenue is received, it is distributed to the group members in proportion to their investment. The process is explained in section 3.2.

2.1. Legal contracts

In order to create a cooperative, a physical contract for cooperation has to be signed from all cooperators. The contract contains the terms and conditions of the cooperative, a declaration of the financing sources, definition of the groups (bonus pools) and the roles of the cooperators.

The contract form [12] is "open source" and provided by the cooperative Obecto for free use. It can be found here , as well as on the home page of the system.

2.2. Democratic system for consensus

Cooperative consensus is achieved through democratic voting. Changes, affecting the cooperators or the cooperative as a whole, are voted for acceptance. Depending on the requested change, the voting for acceptance takes place at group level or at cooperative level. Voting at group level requires a $50\% + 1$ consensus among the group members. The weight of each member's vote is equal to his number of tokens in the group. The proposal is accepted and the change is implemented when the total weight of the votes accepting it is more than half of the total number of tokens in the group. In the event of voting on cooperative level, separate votes in each group each group are performed. The proposal is accepted only when all individual groups accept it.

Changes requiring a vote in a particular group to be accepted:

- Issuance of tokens to a group member, in case the group parameters require a vote for that transaction;
- Change of the group parameters.

Changes requiring a vote in the whole cooperative to be accepted:

- Change of the rules in the accepted by cooperators legal contracts (section 2.1);
- Change in the functionality of the information system.

The coordinator has the sole right to propose changes for acceptance, which ensures his agreement with any requested change. This is required, because the coordinator is practically the limited company behind the cooperative that brings legal and financial responsibility for the operations of the cooperative and defends the interests of all cooperators.

Votes must be submitted within one week from the start of the voting process. After this period, if the change is not accepted, the vote is considered unsuccessful.

2.3. Liquid democracy

Each cooperator has the right to delegate his voting rights in a certain group to another member of the group. Upon submission of his vote, if there are delegations to the member in question, he also votes on behalf of each person delegated to him.

If a cooperator has not submitted his vote in three consecutive votes, his voting rights are automatically delegated to the coordinator of the cooperative. Each cooperator has the ability to change or discontinue his delegation at any time .

The coordinator also has the right to delegate his powers to other members of the cooperative who can be referred to as administrators.

2.4. Measures against deadlock situations

Even if the system is attacked, the cooperative will not suffer direct losses. The aim of the system is to not issue payments directly, but to rather inform about them. Money with value do not pass trough the system, for now. The state of its data at any point of time is stored, thus there can be no loss of data if the system security is compromised. In such a situation, the coordinator is able to perform a "hard fork", and after the removal of the problem, the data can be migrated to a new version of the system.

3. Technical implementation

The project is divided into two main parts: a cryptocontract (also called "contract" and "bot") and a web application. The bot functions as a back-end, by managing the data of the system and autonomously executing its program code, without the web application having authority over it. The decentralized code implements all the functionality described in section 2. The web application is the front-end user interface which communicates directly with the contract in the blockchain of the cryptocurrency Ethereum.

3.1. The cryptocontract

At the heart of the project stands the cryptocontract. It is resembles the actual digital cooperative. It is written in the language Solidity. The main purpose for the use of such non-standard approach is to provide a reliable open-source system over which one has no power, and changes in its data can be made only after achieving consensus among its members. Everyone has the opportunity to create his own instance of the code of the contract in the network. The creator receives the necessary administrative rights for the initial setup. All the cooperative data is kept in the contract variables.

The data structure used for the storage of cooperative data in the code is constructed using techniques to reduce gas costs. The processes are described in section 4.2.

The entire interaction between the web application and the contract in the network is done by calling its methods.

All costs from the contract execution (gas) are covered by the member, who sent the particular transaction (method call).

3.2. Distribution of revenue

In the contract, gained tokens in a particular group are purchased in the presence of revenue. The value is distributed among each of the group members. The process is carried out in 3 phases:

1. Debt repayment. The current debt (tokens) to each group member is proportionately reduced, as based on the transferred amount and the proportion of the member tokens (which can be interpreted as shares), the value of the deducted tokens to each cooperator is calculated. However, the balance of the members grows proportionally to the deducted tokens;
2. If all tokens are redeemed and there are leftover money from the transfer, they are distributed in proportion to each member's slices. The distribution of the leftover bonus aims to reward general merit in the group;
3. The presence of residue from the previous step is caused by integer arithmetics. This type of numbers are used because of their accuracy and optimization. The rest of the amount is transferred to the newest member, because if it had to instead be distributed more democratically, the execution of the process alone would cost more than the negligible leftover value.

3.3. Transparency trough blockchain

Any change in the status of the contract by invoking a specific method is recorded as an event in the blockchain. Because of the functionality of the blockchain, no event will be deleted. Thus any change ever occurred the contract can be read and traced. This builds transparency in the cooperative because everyone is informed about the changes in the system at the moment of execution, and the whole history of events can be accessed at any time.

3.4. The web application

The web application builds the user interface by extracting data for the cooperative from the contract and calling its methods for the execution of a certain action. The implementation is based on Angular 2 Framework, and stylization is Google material design.

Trough the application the coordinator is able to create and publish his own instance of the decentralized contract. Then he and all members of the cooperative have the ability to connect to it. Hence the user has access to all the functionality described in section 2, comprising:

- Adding a cooperator in the cooperative;
- Creating a group (debt pool);
- Reflecting the investment of each member of a particular group;
- Sending revenue to a certain group.

Some form of consensus is required in order to perform most operations. This is described in more detail in section 2.2. In most cases a voting is started on the main page of the system.

Everyone has access to a global list of transactions (the events explained in section 3.3), as well as list of transactions from / to / about a specific person or project.

Data derived from the contract is structured in the different web pages of the application in order to visualize it in a friendly and understandable way (section 5).

3.5. The absence of back-end

So far the existence of a real back-end part is not mentioned. The purpose of the solution is exactly to eliminate such a centralized control system and the potential threats from the centralization. The data of such a system is located in one place, making it vulnerable to the will of the available authority above it. In order for the solution to provide the necessary security and guarantee, the decentralized cryptographic system in question that performs the function of a back-end is used. The cryptocontract manages all its data and there is no authority over this autonomous system.

Anyone willing to use the system installs the browser extension "Metamask". It carries out the direct connection to the Ethereum network. Without it, the standard method to implement such a connection is through a centralized Ethereum node server, which does not meet our needs. The extension allows the user to log in with his Ethereum account, the private key of which is stored and encrypted locally. The web application takes the address of the user's account from the extension, allowing it to show personalized content. Because the contract operates autonomously and in a decentralized way, it will perform instant and secure identification, omitting the need for a login form.

4. Obstacles overcome in the implementation

4.1. The creation of the cryptocontract

Solidity is a specific language for writing cryptocontracts. However, because of the limitaions of the Ethereum virtual machine (EVM), where they are executed, difficulties arise in the creation and execution of such contracts. Some of the overcome difficulties in the creation of the cryptocontract in the system are as follows:

1. **Gas limit:** The gas limit is a variable. The creator of a new block in Ethereum has the right to change the limit by a 1/1024th part of the current value. It

is in the interest of the members of the network to maintain a balance between the more funds they will receive from more complex blocks and their ability to process the data. Therefore, the gas limit, although variable, varies only at a certain interval in the network;

2. **Exceptions:** Only one exception in Solidity is available so far - the so-called "out of gas exception". This exception is thrown in the case of any error, and the effect on the contract is the same as if it reaches the gas limit;
3. **Contract size:** Publishing a contract in the network is a transaction. If the compiled bytecode of the contract is too large, the process will not be successful. Due to the low gas limit, the code size of the contract is very limited, which complicates the development of the needed functionality;
4. **Change of the published contracts** The published contracts can not be changed in terms of programming code. In order for the user to not be at risk of detecting a problem in an already published a contract, methods for a hard-fork of the system are provided (section 2.4);
5. **Storage location:** Variables declared as parameters of external functions, parameters of internal functions, in the body of functions and global variables in the source code are stored in a different type of memory, depending on the declaration. Assigning variables stored in different types of memory should be avoided, as this way the data is copied over to the target storage, instead of passing a reference to it, which is an expensive operation in terms of gas usage.

4.2. Optimization

The execution of a decentralized program differs radically from a centralized one. It is important that the implementation of our contract requires the least amount of computational steps possible. Lower execution costs are in our own interest. No operation in our contract should exceed the gas limit. Thus the following techniques and optimizations in the code have been taken:

1. **Complexity of the algorithms** Algorithms and operations without constant complexity are avoided. Loops, recursions and the like should be used as a last resort. No constant complexity means no guarantee that the code will not get past the gas limit of the transaction;
2. **Expensive operations** Writes in global variables have to be avoided if possible. Writing in this type of memory is the most expensive operation in terms of gas, as it changes the permanent state of the contract;
3. **Data Structures:** The structure of the variables in the local memory of the contract is constructed in such a way that access to certain elements is done by key-value pairs, which is constant complexity;
4. **Alignment of the declarations of variables:** The memory in Ethereum consists of series of 32-byte sectors. The storage of variables in the permanent memory is performed in the order of declarations. Variables are "packaged" in the sectors, so if after storing the previous variable the remaining space in the same sector is sufficient for the next variable, it is also stored into the sector without using a new one. For this prerequisite, the arrangement of variable declaration is an important step in the optimization of the system, thereby significantly reducing the gas usage.

4.3. Security

Security and transparency are among the main goals of the system. Being public and open source, caution has been taken for lightweight execution, protection against current attacks as well as different models for building a robust system. Some of the more notable security measures are as follows:

1. **Protection against "Re-Entrancy" attacks** This type of attacks is one of the most dangerous. It is often executed by recursively calling a certain method. The attack is carried out if contract B, whose method is invoked by calling a method of contract A, called the same (or other) method of contract A before the previous call was completed. Depending on the source code of a contract A, contract B can take advantage of various security flaws. In our case protection against this kind of recursive attacks is built;
2. **Model "Checks-Effects-Interactions":** This is a model for writing cryptocontract methods. In the first part, logical checks affecting the execution of the method body are executed. The second part makes changes to the variables that play a role in the first part. Finally, the actual action is performed - sending money to an account or calling another contract method. This model is often used along with other tactics against Re-Entrancy attacks;
3. **Access Control** Due to the publicity of the Ethereum network, an access system on different levels is integrated into the proposed solution. Each method of the contract has a certain level of access. When called, the contract compares the address of the caller with his level of access enshrined in its memory. In discrepancy, the execution of the method stops and the attempted unauthorized access to the system is recorded in the blockchain (section 3.3). Thus only the cooperators have the right to make changes to the status of the contract;
4. **Cooperator privacy** All contract variables are public. Therefore, the system does not retain sensitive or personal information about the cooperators. In its

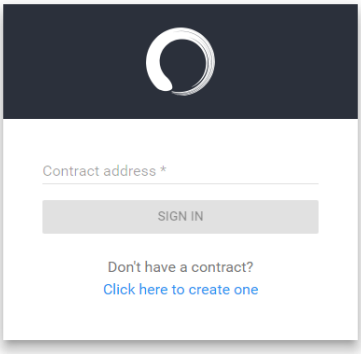

Figure 7: The form for system entry, requiring only the contract address as an input

memory, each member is anonymous and no relation to his real identity;

5. **Protection against stack depth call attacks:** Some operations in Solidity fail if the current depth of the stack reaches 1024. This is a limitation of the EVM. And because a certain stack depth can always be controlled and used as an attack against the contract, in our case there is an established protection against this type of attack;
6. **Protection against integer rollback:** Often overlooked, yet with the potential for tremendous damage [13], is the problem of integer rollback. All operations with such variables are secured either by logical checks or different structuring of the program logic. Thus avoiding unexpected results of operations and any targeted attacks of this type;
7. **"Failsafe" mode** The contract itself performs regular tests over its current condition. If a problem in the data is found that can not be automatically solved, the contract goes into a locked mode, waiting for the coordinator to address the problem;
8. **Protection against common web attacks:** Because the important decisions are executed by the contract, security in the web application is not important. However, measures have been taken against common web attacks (cross-site-scripting and the like).

5. Description of the functionality

Upon loading, the web application checks for the installed browser extension "MetaMask". If successful, the user enters the address of the contract to which he wants to connect (Figure 7). There can be numerous of different instances of the contract in the network, belonging to different cooperatives, without interference.

The main page of the system (Figure 8) displays personalized content for the user, such as his balance, in which groups he invests in and transactions affecting him. All data shown throughout the application is extracted from the memory of the decentralized autonomous program.

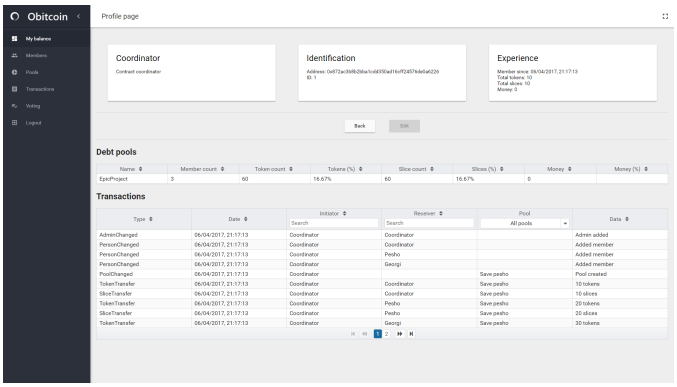

Figure 8: The home page of a member in the system

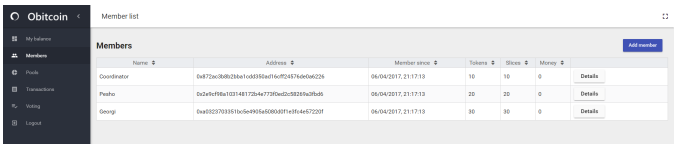

Figure 9: Global list of the cooperators and their overall balance. The user has the option to view a cooperator's profile in details, as well as add a new cooperator in the cooperative.

Figures 9 and 10 show the global lists of the cooperative members and bonus pools.

Figure 11 represents the global list of transactions. All changes to the cooperative state are displayed here. From this page the user has the option to reflect the investment of a cooperator in a group (Issue tokens), and to transfer revenue to a group (Buy tokens).

Figure 12 represents the page to reflect the investment of one or more cooperators in a group. A window is opened (Figure 13) when sending a transaction to the contract (including actions like adding a cooperator or creating a group), prompting for the user's permission to continue.

Upon a successful transaction, which can take up to a minute, the user is notified for the new changes to the cooperative (Figure 14).

After the execution of the last transaction, a voting was initiated for the acceptance of the change. Votes can be cast from the voting page (Figure 15).

If the voting reaches 50% + 1 consensus between the cooperators in the group, the change will take place (Figure 16).

In a similar manner, the user can transfer revenue, add or edit cooperators and create or edit bonus pools.

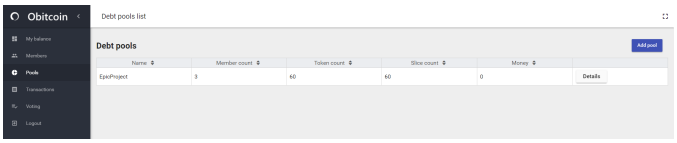

Figure 10: List of the bonus pools (groups) in the cooperative and their balance. The user can view a certain group in details, or create a new one.

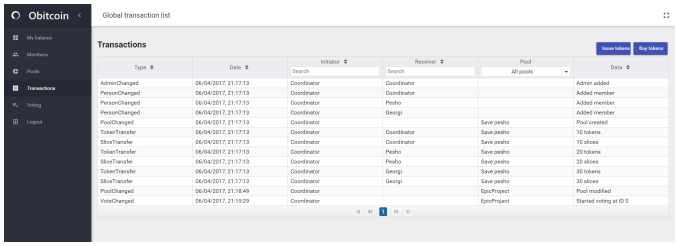

Figure 11: A global list of all cooperative actions / changes that ever occurred (section 3.3).

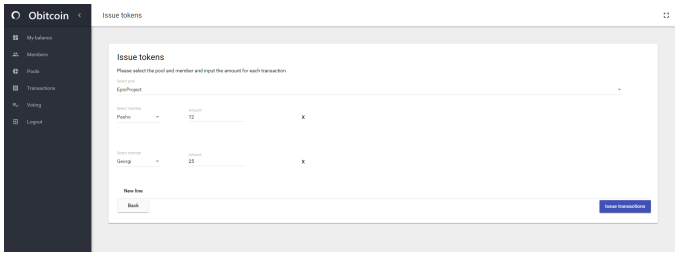

Figure 12: The form to reflect the investment done by one or more cooperators in a group.

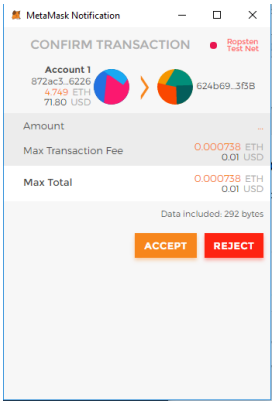

Figure 13: Pop-up window prompting for the user's permission to issue the transaction.

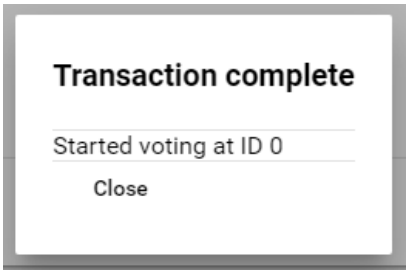

Figure 14: Notification for a successful transaction. In this case, a voting was initiated for the acceptance of the change.

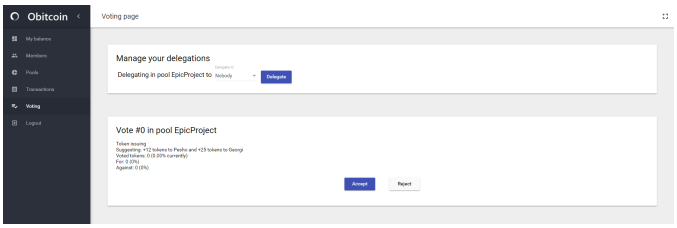

Figure 15: Page with all currently active votes. Hence, besides voting, the cooperators can delegate their voting rights in a particular group to another group member (section 2.3).

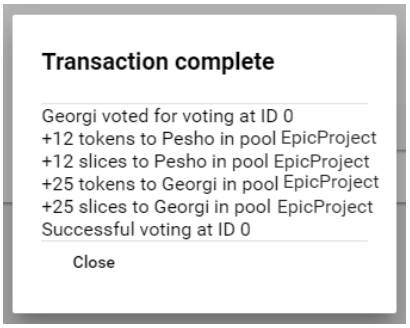

Figure 16: A notification for a successful voting, summarizing the changes in the cooperative.

Numerous rules and requirements affect the user's ability to execute a certain action, described in section 2.

6. Technologies

The following technologies are used in the technical implementation of the proposed solution:

- **Ethereum platform:** For the creation and execution of the project's core - the autonomous crypto-contract;
- **Web3:** JavaScript Dapp API to interface the Ethereum blockchain directly from the JavaScript front-end of the web application;
- **Solidity:** A programming language specific for the creation of cryptocontracts in Ethereum;
- **Metamask:** A browser extension, easing the communication between the client and the blockchain platform of Ethereum by directly injecting the Web3 API in the web application;
- **TestRPC:** Instrument for the creation of a local and private Ethereum blockchain network with only one node. It is mainly used to test contracts locally before releasing them in the main Ethereum network;
- **Truffle:** A development environment, testing framework and asset pipeline for Ethereum;
- **Angular 2:** A development platform for building web applications;

- **TypeScript, JavaScript, SASS, CSS and HTML:** Languages used in the web application's technical implementation;
- **Google Material Design with Bootstrap:** Modern and widespread stylization.

7. Practical application

The proposed solution was developed in close collaboration with the digital cooperative Obecto [14], which provides software development for start-up businesses [15]. The labor in the company is organized as a cooperative since the beginning of 2016 [16], with the original organization being carried out by signing multilateral protocols for the issued tokens and revenue distribution. In close cooperation with the founder of Obecto, Todor Kolev, we established together the digitization of the processes in the cooperative, implementing a solution based on the practical needs of such an organization.

For the development of the project we received support from attorney George Chisuse, who prepared the legal contracts, part of the presented solution. Mr. Chisuse also implemented this approach in his practice by consulting startups on how to develop cooperative structures.

The information system and legal contracts are being distributed freely and under an open source license. This will allow wider adaptation of the solution as well as its further development.

8. Conclusion

In this paper we offer a complete solution for the creation of digital cooperatives, which consists of an organizational mechanism, legal framework, cryptocontract on the Ethereum blockchain and an information system for the administration of the cooperative and the contract.

The provided tools make it possible to solve a problem, crucial for the failure of a huge number of startups - finding initial capital. An alternative approach is proposed for the financing of young companies through investment of labor, rather than capital. This is achieved by creating an internal cooperative structure within the company, which organizes the relationship between the company itself and the cooperators.

The presented solution solves another major problem identified as crucial to 23% [2] of the cases of startup failures - how to split the profits from the common activity. The system we have developed tracks the investment made by each cooperator, ensures fair split between the cooperators and builds consensus between the members of the system. This enables the complex relationships in cooperatives and the dynamic share allocation between them to be digitized, providing ease, convenience and security with the administration.

By providing new ways for people to perform and administer common business activities, the system has the potential to be revolutionary. Its rapid adaptation indicates the need for such a solution within our society.

Being aimed at startups, the proposed approach for building digital cooperatives will allow more innovative products to reach the market. Such acceleration in delivering innovation in turn would lead to an exponential increase of the value created within the world economy.

9. Acknowledgments

I would like to thank Todor Kolev, under whose guidance I developed the project, George Chisuse for the provided legal framework, Konstantin Delchev for the help on the development of my work during the summer research school and HSSIMI for the opportunity to develop my skills under the supervision of professionals.

10. References

[1] U. B. of Labor Statistics, Entrepreneurship and the u.s. economy, https://www.bls.gov/bdm/entrepreneurship/entrepreneurship.htm, 2016.
[2] C. Insights, The top 20 reasons startups fail, https://www.cbinsights.com/research-reports/The-20-Reasons-Startups-Fail.pdf, 2014.
[3] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, 2008.
[4] G. Wood, Ethereum: a secure decentralized generalised transaction ledger, yellowpaper.io, 2017.
[5] Futurism, The blockchain revolution has officially begun, https://futurism.com/the-blockchain-revolution-has-officially-begun/, 2017.
[6] Futurism, The birth of enterprise ethereum, https://futurism.com/this-is-the-birth-of-enterprise-ethereum/, 2017.
[7] Futurism, Ethereum can restore online freedom and transform the internet, https://futurism.com/videos/heres-ethereum-restore-online-freedom-transform-internet/, 2014.
[8] Ethereum, A next-generation smart contract and decentralized application platform, https://github.com/ethereum/wiki/wiki/White-Paper, 2017.
[9] V. Buterin, The question of mining, https://blog.ethereum.org/2014/03/20/the-question-of-mining/, 2014.
[10] V. Buterin, Understanding casper : Proof of stake consensus algorithms, https://blog.ethereum.org/2015/12/28/understanding-serenity-part-2-casper/, 2015.
[11] Ethereum, Network status, https://ethstats.net/, 2017.
[12] G. Chisuse, Cooperative common terms, https://docs.google.com/document/d/1lHQIQRQNa_2zp1j7d296UnXa0ZQSr_D6_2NTmTRFois/, 2017.
[13] Bitcoin, Value overflow incident, https://en.bitcoin.it/wiki/Value_overflow_incident, 2010.
[14] T. Kolev, A cooperative of developers working together and sharing common values, http://www.obecto.com/cooperative-developers-sharing-common-values/, 2017.
[15] T. Yavasheva, The big brother of startups, Economy 69 (2017).
[16] T. Yavasheva, Businesses without bosses, Economy 70 (2017).
[17] J. Rifkin, The zero marginal cost society: the internet of things, the collaborative commons, and the eclipse of capitalism, http://www.thezeromarginalcostsociety.com/, 2014.