D-Drops: A secure decentralized value creation and distribution protocol with an effort-based reward system.

Amsterdam version 0.0.1 - 31-12-2021

D-DROPS Team

Abstract: Decentralization has caused the value generated with new technology to no longer end up in the pockets of a select few at the top of the chain. Instead, it is proportionally distributed among the individual providers who enable the technology. As such, there are no longer billions of dollars locked up in the billionaires' bank account, as a figure of speech, rather it is free and distributed amongst a larger portion of the people.

D-Drops utilizes this new and unique phenomenon to enrich the world with digital valuable items and create fun and exciting experiences for the people. In this paper, we discuss its design and implementation as well as the issues it faces and how they are tackled. Furthermore, we will discuss the opportunities it provides as well as its future implementations as envisioned by us.

(1) Introduction

Nowadays few people don't own a smartphone. We rely on them for almost everything, from keeping in touch with our friends and family to entertainment and staying up to date with the fast-moving modern civilization. The smartphone has almost become part of our body and we can no longer imagine a life without it. It is bringing the digital world closer and closer to our lives and almost embedding the digital world into reality. It will not take too long until what is real and what is virtual become almost indistinguishable from each other.

This development has not gone past the D-Drops team. We realize that the boring everyday reality can be made much more fun and exciting by new emerging technology such as AR, VR, and blockchain. The financial independence provided by the blockchain, the submerging of reality with the virtual world provided by AR, all of it introduce great opportunities and possibilities for the future. The D-Drops concept aims to take full advantage of the emerging markets and help build the future of work and entertainment.

D-Drops provides its users with an app to locate and hunt for treasures with real value. With its AR layer, we aim for the future where virtual reality and the physical world converge and submerge. In the following, a technical description of D-Drops smart contract architecture is given. At the end of this paper, we briefly discuss the issues and application of the D-Drops concept.

(2) A system of states

The D-Drops paradigm can be viewed as a system of four finite-state transition machines. We define the collection of these states transition machine as the world state and refer to the underlying four states transition machines as the substates; formally we define the world state as:

$$\boldsymbol{\delta_W} \equiv (\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C) \qquad [1]$$

Where substate δ_{TC} is an array of the id number of the active treasure chest which are available to be claimed.
Substate is a mapping between the claimant's 160-bit address and a treasure chest id number. This substate maps out all currently claimed but not yet approved treasure chests. Each claimant can only claim one chest at a time. Claiming another treasure chest while one's current claim is being processed effectively overwrites the claim. Substate consists out of a mapping between validators' 160-bit address and an array of treasure chest id numbers which are randomly assigned to this validator by the smart contract. This substate maps out the to-be-validated claim requests per validator. And lastly, substate is a list of treasure chest id numbers that have been claimed and approved. This substate represents all the approved treasure chest claims.

A transition from one world state, σ_W^i, to another, σ_W^{i+1}, comes about exclusively through a transition of any of its substates. Formally:

$$\boldsymbol{\delta_W^{i+1}} \equiv \boldsymbol{\nabla}(\boldsymbol{\delta_W^i}, \delta_{TC}^{i+1}, \delta_{Cl}^{i+1}, \delta_V^{i+1}, \delta_C^{i+1}) \qquad [2]$$

Where ∇ is the world state transition function. The validity of a transition is of crucial importance since an invalid state transition implies bad and or critically flawed code and or logic exploitable by malicious actors. An invalid world state transition is one in which the substates maintain their i-th state while the world state transitions into its i-th plus one state. The opposite where the world state transitions in a new state without a transition in any of its substates is equally invalid. A valid world state transition comes about through a transition in the state of one or more of the substates.

Note: *In the subsequent section when it is referred to a transaction it should be noted that not every transaction on the Ethereum network is meant. Instead, the transactions talked about in this paper are those which cause directly or indirectly a state transition in one of the substates of the world state. The world state itself is entirely dependent on its substates and thus it is only implicitly dependent on these transactions.*

2.2 Dependency of states

A transition in any of the substates is permissible exclusively through a transaction furthermore whilst a transition in the substate δ_{TC} is permitted independently of the other substates a transition in states ($\delta_{Cl}, \delta_V, \delta_C$) are always accompanied by a transition in at least one of the other substates. Thus, a transition in one of these states is only valid if and only if a transition in at least one of the other substates occurs within the same transaction. Since a transition in any of the substates results in a transition in the world state a substate transition is always accompanied by a transition in the world state. The exact dependencies between the different substates are derived below.

A transition in substate δ_{TC} can occur in the event of a new treasure chest drop. Since this treasure chest is not yet claimed and not up for validation the treasure drop event does not trigger a transition in the substates ($\delta_{Cl}, \delta_V, \delta_C$). Thus, a valid state transition of substate δ_{TC} can formally be defined as follows.

$$\delta_{TC}^{i+1} = L_{TC}(\delta_{TC}^{i}, T),$$

where [3]

$$L_{TC} \equiv \begin{matrix} l_{TC}(\delta_{TC}^{i}, T) \\ \nabla(\boldsymbol{\delta_W^i}, \delta_{TC}^{i+1}) \end{matrix}\Bigg\}$$

Where l_{TC} is the state transition function and L_{TC} is the valid state transition function. Note since there are valid and invalid state transitions, we refer to the state transition function with a lowercase letter “l” and to the valid state transition function with the uppercase letter “L”.

When a treasure chest is found the finder can claim this treasure chest by sending a claim request. This claim request triggers a state transition in substate δ_{Cl} however in order to block others from claiming the same treasure chest the substate δ_{TC} must be altered in such a way to prevent others from sending a claim request thus a transition in substate δ_{TC} is required. At the same time, the claim request needs to be validated and thus the claim must be sent to the validators thus requiring a transition in the substate δ_V. We can conclude that for a transition in substate δ_{Cl} to be valid it must go hand in hand with a transition in substates (δ_{TC}, δ_V); formally stated.

$$\delta_{Cl}^{i+1} = L_{Cl}(\delta_{Cl}^{i}, T),$$

where [4]

$$L_{Cl} \equiv \begin{matrix} l_{Cl}(\delta_{ChCl}^{i}, T) \\ l_{TC}(\delta_{TC}^{i}, T) \\ l_V(\delta_V^{i}, T) \\ \nabla(\boldsymbol{\delta_W^i}, \delta_{TC}^{i+1}, \delta_{Cl}^{i+1}, \delta_V^{i+1}) \end{matrix}\Bigg\}$$

Once a treasure claim is validated the claim needs to be removed and the treasure chest object needs to be updated with the address of the claimant and the addresses of the validators and added to the substate δ_C. Thus, a valid transition in substate δ_V requires a transition in substates (δ_{Cl}, δ_C), formally stated.

$$\delta_V^{i+1} = L_V(\delta_V^{i}, T),$$

where

$$L_V \equiv \begin{matrix} l_V(\delta_V^{i}, T) \\ l_{Cl}(\delta_{Cl}^{i}, T) \\ l_C(\delta_C^{i}, T) \\ \nabla(\boldsymbol{\delta_W^i}, \delta_C^{i+1}, \delta_{Cl}^{i+1}, \delta_V^{i+1}) \end{matrix}\Bigg\}$$

[5]

A transition in the substate δ_C can only be triggered through one of the other substate transitions. Furthermore, a transition in substate δ_C does not trigger any other state transition with exception of the world state and thus we can formally define the valid state transition function of substate δ_C as follows.

$$L_C \equiv \begin{matrix} l_C(\delta_C^{i}, T) \\ \nabla(\boldsymbol{\delta_W^i}, \delta_C^{i+1}) \end{matrix}\Bigg\} \quad [6]$$

The transactions required for a state change in any of the substates is non other than an Ethereum transaction. As such the security of

the D-Drops state machine at a low level is completely coupled to that of the Ethereum network. On the higher level, the security of the D-Drops state machine is completely and unambiguously dependent on the code, data architecture, and logic of the underlying smart contracts. It is therefore of crucial importance that all the smart contract logic and data structure is designed carefully and tested exhaustively.

2.3 System validation

In the previous section, several valid substate transition functions have been defined. It has been shown that albeit some substate transitions are permitted independently of each other most substate transitions require another substate transition to occur immediately after itself. Knowing the dependency of each substate together with its valid state transition function is very important for the system validation process as will be discussed in this section.

First, we introduce some new concepts to help us describe the system validation process more precisely. We define the state number of a state as the total amount of state transitions it has undergone prior to the current state; formally:

$$Num(\delta_x^i) \equiv \begin{cases} & n, i \in \aleph : n < i \\ i: & where\ \delta_x^i \in (\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C, \boldsymbol{\delta_W}) \\ & and\ n\ the\ number\ of\ state\ transitions \end{cases} \quad [7]$$

Below the amount of state transition of the state to the right with respect to one state transition of the state to the left is given. It should be noted that in order to prevent infinite loops we have chosen to not count the state dependency if it was linked to a true or false conditional statement. To give an example, we do not count the state dependency $\boldsymbol{\delta_V} - \delta_{Cl}$ because whether this dependency is true or not depends on whether the claim has been approved by the validators or not. Thus, giving a state dependency of 1-0.

$$\delta_{TC} - \boldsymbol{\delta_W} : 1 - 1$$

$$\boldsymbol{\delta_{TC}} - \delta_x : 1 - 0, \quad where\ \delta_x \in (\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C)$$

$$\delta_{Cl} - \delta_{TC} : 1 - 1$$

$$\delta_{Cl} - \delta_V : 1 - 1$$

$$\delta_{Cl} - \delta_C : 1 - 0$$

$$\delta_{Cl} - \boldsymbol{\delta_W} : 1 - 1$$

$$\delta_V - \delta_{TC} : 1 - 0$$

$$\delta_V - \delta_{Cl} : 1 - 0$$

$$\delta_V - \delta_C : 1 - 1$$

$$\delta_V - \boldsymbol{\delta_W} : 1 - 1$$

$$\delta_C - \delta_{TC} : 1 - 0$$

$$\delta_C - \delta_{Cl} : 1 - 0$$

$$\delta_C - \delta_V : 1 - 0$$

$$\delta_C - \boldsymbol{\delta_W} : 1 - 1 \quad [8]$$

Given the scheme above and the dependencies between the transition functions the following can be stated. For every state transition in δ_{TC} there is one state transition in the world state $\boldsymbol{\delta_W}$. For every state transition in substate δ_{Cl} there are three state transitions in the world state. For every state transition in substate δ_V there are two state transitions in the world state. Lastly, for every state transition in substate δ_C there is one state transition in the world state. Formally stated:

$$\boldsymbol{i_{\delta_W}} = i_{\delta_{TC}} + 3i_{\delta_{Cl}} + 2i_{\delta_V} + i_{\delta_C} \quad [9]$$

Note that since we choose to neglect looping dependencies this formula only represents a lower bound for the total amount of state transitions of the given state with respect to the other substates.

In a similar fashion, the following lower bound formulas for the total amount of state transitions of a given state with respect to the other substates can be derived.

$$i_{\delta_{TC}} = i_{\delta_{Cl}}$$

$$i_{\delta_{Cl}} = i_{\delta_{TC}} + i_{\delta_V}$$

$$i_{\delta_V} = i_{\delta_C}$$

$$i_{\delta_C} = 0 \quad [10]$$

Similarly, an upper bound for each state number can be defined. We define the total state change of all the substates summed together as

$$Sum(\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C) \equiv i_{\delta_{TC}} + i_{\delta_{Cl}} + i_{\delta_V} + i_{\delta_C} \quad [11]$$

where i_{δ_C} is the state number of the substate δ_C and so forth. The state number is the total amount of state transitions of a given state. As each state number is always updated when a state transition occurs. We can state with great certainty that the total state transitions

of a given state is captured by the state number. Furthermore, since every substate transition triggers at least one world state transition it can be stated that the total sum of all state transitions has to be greater than or equal to the total amount of world state transitions; formally

$$Sum(\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C) \geq \boldsymbol{i_{\delta_W}} \quad [12]$$

Similarly, we have for the upper bound of each substate the following relationships.

$$\boldsymbol{i_{\delta_W}} \geq i_{\delta_{TC}}$$

$$\boldsymbol{i_{\delta_W}} \geq i_{\delta_{Cl}}$$

$$\boldsymbol{i_{\delta_W}} \geq i_V$$

$$\boldsymbol{i_{\delta_W}} \geq i_{\delta_C} \quad [13]$$

A check for these identities does not exclude any internal inconsistencies it does provide the means to limit and quickly identify internal inconsistencies.

(3) The treasure chest object

A treasure chest object contains all the data about the treasure chest location within the AR layer as well as its content and id number. A treasure chest of type x, m_x, comprises the following fields.

Type: A string describing the type of the treasure chest; formally m_T.

ID number: A scalar value equal to the number of chests dropped by the smart contract counting from the genesis state up to and including the current treasure chest; formally m_{id}.

Center: A scalar pair that represent the longitude and latitude coordinates of the center of the uncertainty circle; formally m_c.

Radius: The radius of the location uncertainty circle; formally m_r.

LocationHash: The keccak 256-bit hash of the randomly generated longitude and latitude coordinates of the treasure chest's exact location; formally m_L.

Value: The total value in sacha (the smallest denominator of the DOP token 1DOP = 10^9 sacha) of the treasure chest content; formally m_v.

Claimant: The 160-bit address of the address currently claiming ownership of the treasure chest; formally m_{Cl}.

Validators: An array of length five containing the 160-bit addresses of the validators of the treasure chest claim; formally m_V.

Data: An arbitrary byte array containing data relevant to this treasure chest; formally m_D.

Link: The link to the uploaded media which functions as proof of finding

The treasure chest struct is the fundamental building block of the state system of the D-Drops universe. It is comparable to the account system in the Ethereum paradigm. The treasure chest struct contains all the relevant information needed for the state system to function properly.

All treasure chest objects inhabit one and the same mapping between the treasure chest id number and the treasure chest objects, distinguishing themselves only by their state. We define several states for the treasure chest object. The ACTIVE(m), EMPTY(m), CLAIMED(m) and the COMPLETED(m) state.

A treasure chest object in the ACTIVE state has non-empty locationHash, center, radius, and value fields while having empty claimant and validators fields; formally

$$ACTIVE(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v) \notin \emptyset \cap (m_{Cl}, m_V) \in \emptyset \quad [14]$$

Where M is the collection of all treasure chest objects.

A treasure chest object in the EMPTY state has none of its fields defined; formally

$$EMPTY(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}, m_V) \in \emptyset \quad [15]$$

A treasure chest object in the CLAIMED state has all but its validators field defined; formally

$$CLAIMED(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}) \notin \emptyset \cap m_V \in \emptyset \quad [16]$$

A treasure chest object in the COMPLETED state has all its fields defined; formally

$$COMPLETED(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}, m_V) \notin \emptyset \quad [17]$$

Note that in the discussion above no mention has been made of the Data field. This is because the Data field has no effect on the state of a treasure chest object and exists solely to contain data relevant to the treasure chest object.

(4) Three processes

Now that we have worked out the formal description of the smart contract architecture, we can talk about the actual processes that this architecture is designed to support.

4.1 Treasure drop process

The first and perhaps most important thing that the smart contracts need to be able to do is to drop treasure chests across the world. Furthermore, the location of these treasure chest needs to reflect the location of the holders and we also require them to not drop somewhere in the Atlantic. This is very easily solved by implementing a world domain. The world domain is designed to be a circle of a certain size that will switch its center every time a treasure chest is dropped and drop the treasure within its boundaries. Each continent has its own domain specifications i.e., circle size and circle center coordinates so in total there are (not counting Antarctica for obvious reasons) 6 domain specifications each for every continent. In order to keep the drops random as well as have them reflect the place of residence of the community members the following scheme has been implemented:

A list will keep track of the location of the holders on a continental level. Holders will be able to update their own location in this list. In order to prevent abuse of the system, there will be a minimum holding requirement in order to be able to put one's continental location on this list. When a Treasure chest is about to drop a random number will be generated and used as an index for the list with holders' locations. This will keep the drops random while at the same time prioritizing the continents with the most holders.

4.2 Claiming process

When a person finds a treasure, they are required to send a claim request to the smart contract. With this request one must provide the following data:

- Longitude and latitude of the treasure
- Treasure id number
- A link to the picture or video which will be used as proof of finding

As stated above when sending a claim request, one must also provide evidence in the form of a picture or video to show they have actually traveled to the location of the treasure chest. Once a claim request is sent the user is free to continue their search for the next treasure.

4.3 Validation process

Community members with a certain minimum amount of DOP tokens will be able to function as validators. The validators in the D-Drops concept can be compared to the miners in Ethereum and BTC. The validators task is to approve or reject a claim request based on the provided evidence i.e. picture or video provided by the claimant. Google and google maps will be the most significant tool of the validators. As their task will be to figure out if the picture or video is taken in the vicinity of the treasure location. Note that we do not require the picture to be exactly at the place of the treasure chest location as it is not always possible to provide a good proof of travel think for instance of a forest filled with threes or a desert. Below you can find the exact requirements a proof of travel has to meet.

- The picture or video needs to be taken at least *10m in the vicinity of the treasure.
- In the picture or video the user needs to show a paper in his/her hand with their wallet address on it. Note: It is not required to be on the picture yourself.
- The picture or video needs to be as clear as possible and not blurry

When a claim request is approved, the claim fee of 10% will be divided according to the following scheme:
5% validators reward
3% burn
2% treasure wallet

(5) Issues

5.1 Spoofing prevention

D-Drops treasure hunt relies on GPS location to check whether a person has found and in fact reached the location of the treasure. As such one can immediately raise the argument that manipulating GPS location of a smartphone is easily achieved and thus nothing is preventing people from cheating and quickly collecting all the treasures using cleverly programmed AI.

We see this issue and have implemented some solutions already in the design of the concept. And other solutions will be added to complement the existing solutions, in the future. The solution relies on the validation of treasure chest claims by validators. How that works is described below.

First, we will encode the exact location of the treasure chest object with the keeccak-256 hash function and will not store the exact location coordinates themselves. This will require the treasure hunter to unlock the hash by means of trial and error. Hence walking around within the uncertainty circle until the hash of the treasure hunter's location equals the hash stored in the treasure chest objects LocationHash field; formally m_L. Requiring the user to send the latitude and longitude to the smart contract and have the smart contract compute and compare the hashes prevents the obvious exploit of feeding the smart contract its own data and bypassing the whole search part of the treasure hunt. Although this ensures the "has to search to find" requirement the design is still susceptible to spoofing i.e., manipulating GPS location of your smartphone to appear somewhere different than the actual location of the smartphone. In order to tackle this problem, we have introduced the validation system to the D-Drops concept. The validation system works in a similar fashion as the proof of stake mechanism in the Cardano ecosystem. Five random validators are picked out of the list of validators who must validate the treasure chest claim and approve it if thought to be genuine. The way this is done is by means of a photo or video that will be uploaded directly on chain via a base64 image to string function. The validators are required to review the submitted proof and validate it accordingly. The validators can be thought of as miners who validate or approve a treasure chest claim in return for a reward. The exact details of the validation system, as well as future upgrades to the system, will be published and discussed in the upcoming new version of the whitepaper.

5.2 Location imprecision

The GPS location accuracy of our smartphones depends on our environment and location w.r.t. the satellites. As such sometimes the GPS can have an inaccuracy of tens of meters. To address this issue, we have designed the Dapp in such a way that it will switch to radar mode when the treasure hunter moves inside the uncertainty circle. This way it does not matter where the treasure hunter's actual location is w.r.t. the real world rather what matters is how many steps one needs to take and in which direction to get to the treasure. You see if the GPS locator shows you to be x number of steps away from the location of the treasure all you have to do is walk that number of steps to the direction of the treasure. As such it users location w.r.t. the real world doesn't matter all that matters is users position w.r.t the treasure chest. Of course, one does not know the exact location of the treasure, this example is used just to explain the solution for the GPS inaccuracy problem. In short, it does not matter what your actual location is and if the GPS shows you to be 10 meters to the left or right w.r.t. the real world rather what matters is how far and in what direction one needs to walk to get to the treasure from the location where the GPS locator thinks that you are located.

(6) Use cases for the D-Drops concept and Dapp.

At the first glance, one might be tempted to limit the use of the D-Drops concept and Dapp to only treasure hunting which of course is as exciting as it is but there is more to it than just that.

As we will make it possible for users to drop their own treasures and create their own quests it will be possible for users of all kinds i.e. the normal Joe, multimillion dollar companies, celebrities and other prominent figures to create a small treasure hunt for their families, promote their new song, art or movie. Organize giveaways in and around their malls etc. Thus, the concept of D-Drops is not only limited to hunting for treasures which are created by the contract it also allows all possible and complex variations of the concept limited only to one's imagination and creativity.

Conclusion

The D-Drops concept sees the future and aims to take full advantage of the emerging markets of AR and blockchain technology. By providing a rich layer filled with real value treasures, exciting fun experiences with endless possibilities D-Drops plans to take the world by storm. Its carefully designed system and robust and fair validation protocols are designed to ensure security by means of the proof of stake mechanism. Happy Hunting!

List of symbols

$\boldsymbol{\delta_W}$	World State
δ_{TC}	Treasure chest list substate
δ_{Cl}	Claimed treasure chest list substate
δ_V	To be validated treasure chest list substate
δ_C	Completed treasure chests list substate
$\boldsymbol{\nabla}$	World state transition function
L_{TC}	Treasure chest list substate valid transition function
T	Transaction
L_{Cl}	Valid state transition function of δ_{Cl}
L_V	Valid state transition function of δ_V
L_C	Valid state transition function of δ_C
$Num(\delta_x^i)$	State number of the given state
$\boldsymbol{i_{\delta_W}}$	State number of the world state
$i_{\delta_{TC}}$	State number of δ_{TC}
$i_{\delta_{Cl}}$	State number of δ_{Cl}
i_{δ_V}	State number of δ_V
i_{δ_C}	State number of δ_C
m	Treasure chest
M	Treasure chest collection
l_{TC}	State transition function of δ_{TC}
l_{Cl}	State transition function of δ_{Cl}
l_V	State transition function of δ_V
l_C	State transition function of δ_C
$Sum(\delta_{TC}, \delta_{Cl}, \delta_V, \delta_C) \equiv i_{\delta_{TC}} + i_{\delta_{Cl}} + i_{\delta_V} + i_{\delta_C}$	The sum of the state numbers of all states

$$ACTIVE(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v) \notin \emptyset \cap (m_{Cl}, m_V) \in \emptyset$$

$$EMPTY(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}, m_V) \in \emptyset$$

$$CLAIMED(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}) \notin \emptyset$$

$$m_V \in \emptyset$$

$$COMPLETED(m) \equiv \forall m \in M: (m_c, m_r, m_L, m_v, m_{Cl}, m_V) \notin \emptyset$$