Dragonereum

Where blockchain meets real gameplay

Version 0.0.4

9 January 2018

Abstract

Dragonereum is a criptocollectible game where users can own a dragon, trade their dragons, interbreed them and fight other dragons collecting rewards and achievements along the way. All of these is done on the blockchain in an open, trusted and decentralized manner. Our aim is to build first truly decentralized digital scarcity game where developers control as little as possible and the community has the ability to verify all the transactions and actions without a necessity to rely on any trusted third party.

This document discusses technological and economic implementation of the system describing such elements of it as: random number generation, decentralized breeding algorithm, fight implementation, game mechanics and a broader economic model which emphasize strength of a collectible game built on blockchain.

Additionally we want to give our users the right to choose the direction of project development.

Table of contents

Disclaimer

This whitepaper is provided for informational purposes only, and does not and will not create any legally binding obligation on the authors or on any third party, therefore, the representations herein should not be relied on. Dragonereum makes no representations or warranties (whether express or implied), and disclaims all liability arising from any information stated in the whitepaper.

No regulatory authority has examined or approved of any of the information set out in this whitepaper. Thus, no action has been or will be taken under the laws, regulatory requirements or rules of any jurisdiction. The publication, distribution or dissemination of this whitepaper does not imply that the applicable laws, regulatory requirements or rules have been complied with.

Introduction

Definitions

In this paper we assume that a reader possesses basic knowledge of blockchain, cryptocurrencies, smart contracts, dapps, etc so we go straight to our core features.

Collectible is any object regarded as being of value or interest to a collector [1]. Some of the most popular types of collectibles are: ancient coins, stamps, old books and maps, baseball cards [2].

Digital collectible is a sequence of bits stored on some medium which represent a game unit or artefact, digital custom art, etc. As any other peace of data it can be easily replicated, modified or removed from the user by the third party which issued or manage particular collectible asset.

Cryptocollectible is a unique non-fungible digital asset stored on a blockchain with some characteristics encoded in it. Use of blockchain allows users to track ownership of the collectible items and guarantee the authenticity or rarity of them without the necessity to rely on trusted third party. Similar to regular collectibles the price of cryptocollectibles is determined by the market agents based on perceived value of a particular item. During evaluation process those agents can take into consideration demand and supply, trends, perceived importance, scarcity of the item and previously paid price.

Gamified cryptocollectible is a cryptocollectible with the implemented capability to perform some actions by a user with collectible assets except for just storing and multiplying them on blockchain thus increasing user engagement and perceived value.

Current challenges

Current state of cryptocollectibles represent dim picture as there is limited scope of actions one can do with his or her items.

Unlimited supply reduces attractiveness of a particular cryptocollectibe and there is a small incentive to track other available items except for a small set of highly customized or custom-built assets.

Additionally, current solutions still heavily rely on centralized infrastructure diminishing advantages of blockchain usage.

The solution

We propose the system which is built on the basics of current technology but adds gamification to keep engagement levels high. Supply of cryptocollectibles is limited by proposed economic model. Our long term goal is to create fully decentralized game leveraging all of the unique advantages of the blockchain.

Technology limitations

As decentralized technologies are in their early period of development there are several limitations which we have to consider and which hopefully we will be able to overcome in the future as decentralized technology advances.

First of all there will be a relatively high price for all interactions with smart contracts (unless gas price will drop dramatically) and a possibility that many transactions will run out of gas. Transaction price could be minimized if there would be a solution to use off-chain protocols to perform computations. However at the moment there is no such solutions that will suit our needs so during launch we will perform all computations on-chain.

Additionally Ethereum network is currently limited to approximately 15 tx/sec. So if any particular application is heavily used this is automatically bring the whole system to a halt.

Secondly, as we rely on Metamask for our product the usage is only limited to desktop browsers. We hope that there will be solutions which will allow us to move to mobile but for now our platform can be used only with browsers supported by Metamask.

Also there might be a problem publishing an app on the App Store as our ecosystem require use of cryptocurrency payments instead of Apple’s In-app purchases.

Additionally we want to mention that proposed RNG is in reality PRNG and true randomness currently is not achievable on blockchain.

Dragonereum Overview

What is Dragonereum

Dragonereum takes few steps further current cryptocollectible offerings surpassing their limitations by providing higher engagement, decentralization and setting restrictions on the speed and possibility of breeding thus restraining unlimited growth of dragon population.

In addition to buying, selling and breeding their dragons users now can engage in fights, collect rewards and compete with other players for in-game achievements. And most of these features securely completed and stored on the blockchain removing the need for a trusted party.

While fulfilling our main objectives by creating complete infrastructure and ecosystem of Dragon Marketplace and Dragon fights we want to keep the system as simple as possible without unnecessary tokens.

Summing everything up, Dragonereum is a cryptocollectible PvP game where users collect and breed unique dragons engaging in fights along the way. Other users can watch fights and place bets on their favourite player if this feature is developed by the community.

Every Dragon is owned and controlled by a particular address on Ethereum network. Dragons are securely stored on the blockchain and can not be modified, replicated or destroyed by any third party (not even by us or Vitalik).

Each Dragon has its unique immutable set of genes which sets all features and traits of the dragon.

In this document we propose philosophical background, technical foundations and economic model of Dragonereum where dragon supply is limited by a proposed game mechanics.

Infrastructure and decentralization

Our aim is to create a game built on Ethereum network which will have as little ties to our servers as possible. It would be great to have 100% decentralized game however at the current state of technology it will be not feasible for players to play the game where every action require a payment so we kept only core features on-chain.

In the long run we hope to remove any dependency from us or any other third party with the advancement of technology.

At the moment the game is accessible via browser with Metamask plugin installed.

Data layer consists of two ERC721 [3] non-fungible tokens which will represent two states of a dragon (an egg and a dragon). Business logic layer consists of the breeding smart contract, fighting smart contract, marketplace smart contract.

Continuous integration

The update of a deployed smart contract on the Ethereum blockchain requires deployment of a completely new smart contract. This usually leads to either a fork of the dapp or to a disruption of user interactions with a dapp until everyone switches to a new version of the smart contract. Such updates for a blockchain game could lead to a disruption of game processes, lost cryptoassets and negative experience for users.

We are proposing the use of contract abstraction implemented by a mediating smart contract and cascading contract. We split our application into two layers: data layer and business logic layer. This approach allows us to update the dapp logic without necessity to migrate user data.

Obtaining randomness

There are two main approaches used for random number generation. The first relies on some physical phenomenon which is expected to be random. The second relies on some computational algorithm. The latter uses some initial value to generate a random number however this random number can be reproduced if the initial value (or source of it) and the algorithm is known to the attacker so this approach is not truly random and called pseudorandom.

For some industries (e.g. online gambling) obtaining randomness (RNG), or to be more precise pseudorandomness (PRNG) is a complex issue as random numbers which influences an outcome of the game should be equally unpredictable for all parties.

In legacy online apps there is a number of ways to obtain randomness independently or relying on third parties. However, users of such apps rarely have the possibility to validate the process of obtaining these random numbers.

On the other hand for decentralized apps (dapps) where an outcome relies on the random number this issue becomes even more important because of limited availability of methods, price associated with the use of current solutions and time required for RNG.

To build a proper decentralized game where an outcome is defined by random number following specifications of RNG must be met [10]:

1. RNG can generate a random number in short period of time (ideally in less than 1 second)
2. The RNG can be trusted and verified by being stored on a blockchain
3. RNG should be capable to serve large number of players simultaneously
4. The cost to obtain a random number should be reasonable (low gas/ transaction cost)

Let's take a look at different approaches for obtaining random numbers for Ethereum dapps available today.

Data from blockchain
It is possible to get a random (pseudo-random) number from the blockchain data directly i.e. based on block number, block hash, etc.

This approach is fast (random number is generated in a next mined block) and relatively cheap but there is a possibility that miners can influence an output of the block in order to affect generated number.

We will dig a bit deeper into this problem later.

Use of oracles
This approach is based on some trusted external source of randomness, e.g. www.random.org, which is then delivered to blockchain by other trusted external sources, e.g. http://www.oraclize.it [4].

This implementation is simple but the weak spot of it is that oracles also could be compromised [5] and it takes longer to get random number as two blockchain transactions are needed. In our tests it took around 30 seconds to get random value by this approach.

Commit-reveal
RNG by this method require two steps. During the first step participants send hashes of random values and deposit a pledge. During the second step, sent values are disclosed and the random number is generated based on this initial values [5]. The pledge is required to ensure faithfulness of participants. This method is used by RanDAO, Sleth, Maker-Darts [5].

The disadvantages of this method are that a fee is required by most implementations in order to involve participants for random number generation and that this method is a subject to DDOS attacks. The latter results in a loss of pledges by honest participants [5].

Future blocks
During this RNG a user sends a request into blockchain that the random number will be needed soon. This request is put into an request list by a RNG smart contract. In the next step a user requests a random number via another transaction after some blocks been mined by the network. Required action is taken using generated random number based on initial input and some hashing algorithm, e.g. sha256.

This approach is slower than other solutions (2 blocks will take about 30 seconds) and current limitations of Ethereum EVM allows us to look back only 256 blocks so a user could just skip generated number if it does not match his expectations.

Signidice algorithm
This approach of RNG is described in Gluk256 repository [5], [6]. Ethereum smart contract receives a newly generated public key from the owner, and await for a value from the user. After receipt of this value the owner hashes received value with a private key thus generating required random number, which might be confirmed by already published public key.

Regardless of the fact that there is a centralized application that confirm random numbers required for this solution it is still possible to use this method for our use cases. However we consider this approach being too complex for our initial requirements (more on this below).

Our solution
According to initial requirements RNG have to generate a random number as fast as possible (ideally in less than 1 second but game process is currently limited by Ethereum rate of block creation anyway). And the best solution in this regard is a use of data from a blockchain block.

We propose to use a hash of a block for getting random numbers. For more complexity and the possibility to serve multiple players at the same time, we mix sender’s address, unit count (dragons) or action count (fights). So if similar transactions are included in one block, they will get different random numbers even though hash of the block will be the same.

uint64(keccak256(keccak256(block.blockhash(block.number), _seed), msg.sender));

Where *seed* - dragon count or fight count
msg.sender - address of a sender

For complex genome generation we need more than 30 different random numbers. For that purpose we propose the use of bit-shift operations on initially generated random number.

To be precise we will use a remainder of the division as it is more readable and efficient

For example, if we receive initial random number of 142 and we need two additional random numbers following operations will serve our needs:

$$142 \% 10 = 2$$
$$142 \% 100 = 42$$

Where % is a remainder operator.

As a result we will get two independent random numbers: 2 and 42.

Finally, in the ideal decentralized world RNG will not be a subject to a miner attack. However in real world this factor can be neglected for our use cases as economically motivated attacker (in this case miner) will pursue higher payouts and thus mining another block will not be feasible as other miners will not wait and will submit a new block.

Also we will reconsider this assumption if an economic incentive for miner’s attack will increase over time/with a release of additional features in the game and/or new solutions for Ethereum EVM will emerge (e.g. 256 back lookup limit will be removed).

Basics of blockchain genetics

Major limitation of legacy games is a process of new character creation. Current solutions allow user to choose some characteristics of a character or supply a default character with preconfigured range of abilities. Additionally all current solutions are centralized by their design so the player does not control or have the ability to verify the process of game character creation, number of created characters or their set of skills.

As such the company behind the project at any moment can release newly created characters with better skill set or unique appearance thus destroying economic incentives of current players who invested time and money into their character (especially in cases where all created characters are collectibles).

We propose the system where any third party (even creators) has as little control of new character creation as possible after release of the system.

To build such a system all steps of the creation process should be done on blockchain. In order to do that two main principles must be met:

1. RNG have to be generated on blockchain or at least stored on blockchain
2. Breeding of characters also have to be done on blockchain via publicly accessible smart contract

RNG was already described in this document and here we’ll dig into on-chain breeding.

In the proposed product genome of dragons have very important role as it determines every aspect of dragon's character, appearance, fighting abilities and thus defines market value of a particular dragon as a cryptocollectible item. As such we tried to make genetics of dragons similar to genetics of real world animals nevertheless adding some additional features as we are talking about dragons, aren't we?

Two dragons upgraded to a certain level can produce an egg. This egg in some time will become a dragon which will share traits of both parents.

Dragon's genes define not only appearance of a dragon but also fighting abilities and other characteristics. We believe that a dragon with diamond claws should differ in fighting abilities from a dragon with stone claws so their genes also should differ and those features should also be inherited by following generations.

There are 3 main approaches for implementation of breeding algorithm:

- Centrally hosted breeding algorithm
- Off-chain breeding algorithm
- On-chain breeding algorithm

Let's look into details of all these approaches.

Centrally hosted breeding algorithm allows highly sophisticated breeding algorithm implementation and details of such algorithms can be kept in secret and therefore unpredictable for users (but this approach creates possibility for insider advantages). Additionally the possibilities for verification are limited for general users and community thus this system is not providing required level of trust.

Performing all necessary computations off-chain but storing associated data on-chain will be a great solution as it will reduce cost and will allow implementation of more complex algorithms. However current state-chain (off-chain) protocols does not provide sufficient solutions for implementations of state-chain breeding algorithm (more info on considered off-chain solutions below). We may reassess the process once suitable protocol will be implemented.

On-chain breeding algorithm currently limits possibilities for implementations as the complexity of the algorithm will lead to increase of gas usage by smart contract. However results of breeding could be easily verified by anyone as everything is performed and stored on blockchain.

Dragon breeding algorithm

According to initial requirements we propose the use of on-chain breeding for full decentralization. In order to keep costs as low as possible we reduced complexity of smart

contract to the level where users pay less but the predictability of breeding in not sacrificed at the same time.
We hope that with future releases of such protocols as Plasma Like by BANKEX [10] cost of breeding will be reduced and we will be able to switch to more advanced forms of breeding algorithms.

This approach allows us to create a truly trusted world of cryptocollectible dragons where everyone will be able to check what is going on during breeding process and even our team members will not be able to influence the results thus completely removing necessity of trust as it is the case with other solutions available today.

Off-chain solutions (state chains)

We think that it would be great to keep everything on-chain. However for the best gameplay experience, user engagement and ecosystem growth the barrier to entry should be as low as possible.

Current rise of gas price and associated delays with transaction processing on Ethereum network could be solved with the use of off-chain (state-chain) solutions.

There are following options of off-chain technology which were considered during our research:

- Raiden network
- Truebit
- ZoKrates

Raiden network is limited only to token transfers so it is not possible to use if for computational purposes.

Truebit is a scalable verification solution for blockchains which allows to do complex computations off-chain [7]. However those computations are carried out and verified by Truebit participants for a fee thus adding additional costs to the process making this solution less attractive for our use-cases.

ZoKrates - uses 0-knowledge approach [8]. It allows to implement complex algorithms on a C-like language, compile smart contract and deploy it to the Ethereum network. On the next step a user run the version of the code which is deployed on some server and then verifies the result by the contract and stores a result on the blockchain.

The draw back of this approach is a high gas usage which is higher or similar to our algorithms deployed directly to smartcontract. Also there is a scalability issue as only around 6 such computations could be stored in a single block. Additionally as it stated by developers: “this is a proof-of-concept implementation. It has not been tested for production” [8].

Game mechanics and sustainable game ecosystem

Our aim is to create a cryptocollectible game where those users who spend more time playing the game have the possibility to have higher valued dragons. We want to accomplish that by balancing incentives inside the game and creating in-game events which will keep user engagement levels high.

With our initial dapp a user will have the possibility to own and manage a dragon or a thunder of dragons. Every dragon comes with unique appearance and skills. So one will have to choose a race of the dragon to control, decide on fight strategy, which abilities to train with experience earned in fights and how to get better offsprings with even better skills.

The game starts with the purchase of a dragon egg or a dragon from another player or during initial dragon sale. In case of an egg a user know parents of the egg but not which dragon will appear from an egg as this is decided during egg hatch.

After receiving certain amount of experience and upgrading to a certain level a dragon will receive the possibility to breed. User will have to choose whether to get an upgrade for a dragons' skills or to get a new dragon.

Offspring will rarely outperform parents right from the hatchery. However initial skills would be generally higher than those of parent dragons. So with some training and a few updates the dragon eventually will outperform its parents.

As in a real world sometimes there might be a mutation which will make some traits of a new dragon much weaker or much stronger than those of parents. Mutations affect individual genes so any trait can get a boost or setback during breeding but no one can predict which gene will be affected and when.

As every next level up will require even more experience it will become more and more difficult to get dragons of future remote generations thus limiting total population of dragons and avoiding exponential unit number grows.

At the same time those gen0 dragons who will get additional skills instead of offsprings will become more valuable as they will have better fighting abilities even compared to future generations.

As such the system is designed in a way which encourage users to be active and allows them to choose the winning strategy of their own choice.

There are 11 dragon races in the game. Dragons of different races can interbreed creating unique offsprings.

Every dragon has the ability to fight. Fight winner gets experience points which could be converted into level ups and improvements of skills. Additionally winner receives the Gold.

The Gold is an in-game cryptocurrency which will be distributed to winners partially from the system itself and partially from owner of dragon which was defeated during a fight.

The Gold could be spent on in-game services (e.g. healing of a dragon after a fight) which will be offered by other players or a system. This will create a platform for social interactions between dragon owners.

Additionally players will have internal ranking system where they can earn in-game achievements and compete against other dragon owners.

How to get a Dragon?

During launch of the game public sale of gen0 dragons will be conducted.

Following variation of a dutch auction is proposed for the sale: every new gen0 dragon released for sale on the marketplace will be added after another gen0 dragon is bought.

The price is determined by following formula:

$$P = P_1 \times N_t\%$$

where P_1 - average price of last sales,

N_t - multiplicator which depends of number of sales during last hour.

Further details will be available soon.

Conclusion

This document describes a blockchain cryptocollectible PvP game built on Ethereum network which emphasizes advantages of the blockchain solutions for future blockchain gaming, gambling and collectible markets.

Technological and socioeconomic solutions for creation of a sustainable game ecosystem examined and discussed.

Please join our official Telegram channel to get news and updates:

https://t.me/dragonereum

References

1 "Collectable". TheFreeDictionary.com. Retrieved August 19, 2013.

2 https://en.wikipedia.org/wiki/Collectable

3 https://github.com/ethereum/eips/issues/721

4 https://github.com/oraclize/ethereum-examples/tree/master/solidity/random-datasource

5 https://github.com/DaoCasino/Whitepaper/blob/master/DAO.Casino%20WP.md

6 https://github.com/gluk256/misc/blob/master/rng4ethereum/signidice.md Accessed: 2016-08-26.

7 https://truebit.io

8 https://github.com/JacobEberhardt/ZoKrates

9 Tomas Draksas Dev. Update #6: Edgeless Dice RNG
https://medium.com/edgeless/dev-update-6-edgeless-dice-rng-daef755f26a0

10 https://bankex.com/ru/technologies/plasma-protocol