# Lucky Games on Blockchain

Quyet Thang Nguyen
Ecosystem Developer
Fredo Tech Company
Ha noi City Vietnam

August 30, 2018

## 1 Introduction

Randomize in Blockchain - a Deterministic System?

**Main focus** the theme will be presented about the randomize on the Blockchain, where all information is public. In this article, we don't focus on how to create a random number, but on Lucky games on Blockchain, how to choose a winner from a lot of players so that the selection is fair for everyone and everyone can check that fairly.

## 2 Problems describe

In blockchain there are two main problems with randomizing. The first problem is that the miner (block creator of a round) always knows the result sooner than anyone else. Therefore, he always has a big advantage when he participates in the game in which the result is used. This makes the game no longer fair for everyone else.
The second problem is the collaborating party, which is actually an extension of the first problem. In this article, an analysis on this issue, how badly it affects a lucky game.

# 3 Solution approaches

A randomized result is fair and usable only if it is not previously known or unchangeable even someone knows it beforehand. Since Blockchain is a deterministic system, in my opinion, it is impossible to go in the first direction. All information in Blockchain is known to all. that is, before the miner the random on blockchain public, anyone else can also calculate that. Assuming only the miner would know the result before it is written in the block. Then the miner would have some information that only he knows. Then the blockchain would not decentralize anymore.

## 3.1 Existing Chaindata as a seed

The usable blockchain data would be the timestamp, block difficulty or such a blockhash, anything that exiting on chain (in created blocks). The most commonly used value is the blockhash. There are two cases for this solution class. The first would be the data from previous blocks. The second is the data from current block. In the first case, everyone can calculate the result and its not secret anymore. The next looks better, since only the miner knows the result before the block is sealed. The problem with centrailize and decentralize is affected in this case, since the miner can manipulate them or use it to make the result not fair anymore.
The optimal environment for this solution class would be the proof of work like BTC, where the reward for blockhash inventors is relatively large. Every miner in this system tries to find the block hash as fast as possible. But in the case that the reward for the lucky game is much larger than the block reward, the data from the current block is no longer reliable. In addition, the block time of BTC is long (about 15min at the moment), the players have to wait relatively long for a game round.

## 3.2 Seed distribution of each participant

The basic idea of this solution class is that each participant can influence the random outcome. In this part I will analyze the ranDAO algorithms. "RanDAO was first designed to be a better deterministic source of randomness on the blockchain because "miners cannot be trusted". RanDAO is a 2-phase commit-and-reveal scheme, where participants first commit a hashed version of their seed and then reveal the original seed at a later time. All seeds

committed and revealed by the participants will be used to generate the final random number. The integrity of this scheme relies on the assumption that not all parties of the commit-and-reveal process are from the same cartel and colluding with one another."
The idea has a drawback, you can not guarantee that all participants verify the hash sender. Thus, there is the case that some players with intention not verified. RanDAO has a solution for this case, stating that the participant loses his bet sum if he does not confirm. The question is, is the measure fair for all participants. We'll look at an example with three players and two of them is a group. If it's fair, the lonely player has a third chance to win and the group has two third chance to win. But that is not the case in fact. When the lonely Player verified, the group can start calculate the result and they have total three selections : only one of them verify (2 cases) or both of them verify. To win the game, the lonely player has to win in every selection of the other group. The chance of this player is $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} = \frac{1}{12}$ . that mean the solution of ranDAO can not prevent the collaborating party attack.

# 4 Binary Lucky Game

## 4.1 Anti collaborating party

**Lemma :** *in the case with two players the solution ranDAO works absolutely fair.*

**Proof :** in this case we do not have to look at the collaboring party because there are not enough players for the case. The problem would be that the second comfirmed player always knows or can calculate the result before. After a player has sent his secret number, the other player already knows the result. Nevertheless, he can not change anything. In case he wins with his secret number, he will send it off. Else, if he already knows that he lose with his secret number. Whether he sends it off or not, does not matter anymore. The chance always stay at 50:50 for both players.

### 4.2 Binary tree

To keep it fair, the binary lucky game player shares in pairs 1 vs 1. In the event that the number of players is an exponent of 2, we have a total of $log_2 n$ rounds to find the winner. The matches look like a complete binary tree.

**Complete binary tree :** *A complete (perfect) binary tree is a binary tree in which all interior nodes have two children and all leaves have the same depth or same level.*

At the beginning, the positions of players are set on pages of the binary tree. Randao is applied in every couple of every round, until the root is reached. Whoever arrives at the root will be the winner of the game. Thus, the chance of winning for each player is $\frac{1}{2}^{log_2 n} = 2^{-log_2 n} = \frac{1}{n}$. This is fair for all and immune collaborating party attack.

Demo on Github : https://github.com/bestboyvn87/bettingDapp

### 4.3 Minus points

It clearly to move into the next round only when all players are confirmed or the time has expired for confirming. In worst case for a game with n players, ca blocktime times $log_2 n$ is needed to find the winner of a game. In addition, for $log_2 n$ time confirmations cost n times transaction fee. For small games it is a big problem with economics. In the next part an extension of this algorithm will be presented, to solve the above problems.

## 5 Random with different weight

### 5.1 Problem presentation

Given are participants who play a game together. Player i ( $1 \leq$ i $\leq$ n) invested $w_i$ to play. how can you choose a winner from the players, so that the chance of winning is fair for all.
Set $W = \sum_{i=1}^{n} w_i$ To be fair for every players, the winning chance of player i with the invested amount $w_i$ should be $\frac{w_i}{W}$

### 5.2 Solution on Blockchain

# References


[1] Designing Random Number Generators *: https://medium.com/@kkenji1024/designing-random-number-generators-41e653782a7f*

[2] Proof-of-Randomness: UltraYOLO's Random Number Generation (RNG) on the Blockchain *: https://medium.com/ultrayolo/ultrayolos-random-number-generation-rng-on-the-blockchain-proof-of-randomness-c8e35d538d5a*

[3] Predicting Random Numbers in Ethereum Smart Contracts *: https://blog.positive.com/predicting-random-numbers-in-ethereum-smart-contracts-e5358c6b8620*