Decentralized Peer to Peer Game Platform, Truly Random Number Service from DPOS, and Cross-chain transfer using System Escrow

bitsharestalk.org, bitsuperlab
hackfisher@gmail.com
2014-08-04

1.0 Introduction

BitShares Play is an experiment to demonstrate and validate how a decentralized, autonomous, and provable fair game platform system is working. Systems like this are also known as Distributed Autonomous Corporations (DAC).

Games based on randomness have come to rely almost exclusively on trusted third parties to provide random feeds. While this system works well enough, it still suffers not only from the centralized trust based model, but also from the possibility of cheating by players or the trusted entity of the system. Even though some crypto based games on the Internet have provable random feeds which can be verified by public, hidden players still can cheat by submitting selective favorable transactions because they know the random secret in advance. The cost of mediation and trusted entity increases system transaction costs, possibility and return of being lucky. The players cannot make sure it's a fair game without trusting the third party. This reduces the fun, ease of motion, and charitable activities based on these games.

BitShares Play is to resolve these problems. To understand this, it can be separated to several parts:

- The crypto technology and the blockchain concept from Satoshi's "Bitcoin: A Peer-to-Peer Electronic Cash System" makes the system decentralized and trustless.
- A consensus algorithm like DPOS to update and maintain the network and public ledger while maintaining the security and stability of the system.
- A truly random number generator (RNG) which is based on the concept of provable distributed secret feeds. There is an algorithm from DPOS, which is used to shuffle the delegates ordering randomly in each round.
- A platform framework with different layers: the core layer, which performs the blockchain and ledger tasks, and the rule layer, which is designed to allow others to develop play rules, and thus games, on it and keep the tokens economically self-sustained and safe at the same time.
    - The later one is achieved by different rules with their own tokens (chips), by pegging their inner exchange price with the system token according to their collateral PLS supply and their current chip supply. At the same time, the collateral can be added or covered by the market users according to current price.
    - The purpose of the combinatorial number system (CNS) is to normalize the diversity of different rule inputs and outputs, and make it easy to design the play rules.
- A concept of cross chain trading relies on system escrow and user issued assets on decentralized exchanges like BitShares X. This requires the user issued assets to support system escrow too, and will make the exchange a dependency of this system. This DAC will support a similar kind of system escrow for other game DACs (with provable supply feed) to plug into.

2.0 System Tokens

Play shares (PLS) are the tokens of the system. PLS provide an opportunity to expand on our model for token distribution, PLS may explore models where a game rewards winners the tokens of the game system itself. People can own PLS for the purpose of exchanging them for chips to play games, acting as employees/delegates, or owning the DAC (shareholder). There are a lot of demand types:

1. People playing the game need to pay or bet for tickets in the process of playing.
2. Owning the DAC means that the owners of PLS will be paid with dividends from the revenue.
3. Acting as employees (delegates) of the DAC means that if delegates want to get advantage in DPOS process, they would like to own PLS to vote for their own blocks.
4. To own shares in future DACs of the franchise. This is the Daddy-DAC of a proto-family.
5. Donate to charity feeds, people can choose to donate their chip rewards to charity feeds.

People can buy PLS in exchange for "chips" in the inner system, which can be used to play the games. All spent chips are rewards for players except for a small percent being the "house edge". Part of this will be use to either pay the delegates, donate to charity, or pay the a dividend to the shareholders by destroying them. In the system, there is no single central entity that benefits from the house edge.

Investors can buy/sell PLS in the open market without playing the game. Players can buy/sell PLS in the open market without thinking of it as investing. Both drive demand for PLS. This makes holding PLS and winning PLS a good thing for both types of buyers.

The other kind of tokens in this system are called "chips", there will be different chips in this system. Normally, one chip asset for each game. Chips are created or destroyed by adding or covering the PLS collateral according to the current price determined by the market , to be more specific, the price is determined by the ratio of current PLS collateral and total supply of chips. This means there will be a flexible supply for the chips, according to the requirement of the game chips caused by the elevated popularity or profitability of the game. But at the same time, no chip is created from nothing because every chip is backed by PLS people that need to pay when they buy chips. The shares will be return after selling/covering chips. The initial collateral and supply is setted by game creators, so is the price, once the price is setted, the creator cannot affect it anymore, it can only be affected by the total PLS supply, the collateral people add to the chips, or the game rules.

2.1 Introduction to inner exchange model between PLS and chips:

It is different from BitShares market issued asset like BitUSD and user issued asset. There is an exchange model between PLS and chips, and no market, which is part the DAC consensus:

Every chip asset is created with some PLS collateral recorded by the system, and a total supply. After creation , the PLS amount will be frozen as collateral in the system's balance.

The game itself is free to adjust its supply according to its own business model, but there will be a system convert price between the chip and PLS, according to the PLS collateral and chip supply:

1 chip = (PLS collateral amount / chip supply amount) * 1 PLS

This means anyone can buy/sell (in other word create/destroy) chips according to the price of current block, the amount of PLS used to buy chip will be added to that collateral of the chips, and the new chips will

increase the supply. In the next block, there will be a new price, according to the updated PLS collateral supply and chip supply.

This way, the most popular games will have more collateral, the most profitable game will have a better chip prize with an increase related to PLS and thus other equities. The games with unrestrained dilution similar to Tencent's Q Coin is free and OK, but will have the drawback of a potential drop down in price.

One more thing, every game that wants to use chips as their game tokens requires:

1. A provable total supply of the game tokens.
2. A way for a 1: 1 transfer between chips in BitShares Play and their system, using cross-chain trading or by supporting system escrow. The easiest alternative would be to develop a game inside the BitShares Play platform, or using the Play Database and system as balance record of the game.

3.0 Consensus Algorithm using DPOS

The advantages of DPOS are that it allows for faster block confirmation times and also allows for scalability to the level of VISA's 10,000 payment transactions per second. the system is still decentralized in that there is neither a single point of failure and nor is there a single point of control. Delegates have the rather simple job to sign new blocks and can be fired on command if they do not perform their duties. Consensus on a whole is reached over all stakeholders, whereas in traditional Proof-Of-Work schemes only hashing shares contribute to the network consensus.

The Delegates are the entities which make all these magic happen. They are involved in the decentralize system as a key part that people can vote for to represent the current consensus, and change consensus of the system, which could help the system to upgrade and reform itself.

In the DPOS peer to peer (P2P) game system, delegates are playing an even more important roles, because they not only collect transactions, produce and sign blocks on their scheduled time slots, but also are the source of the provable distributed random secret services. Please refer section 4.2.

4.0 Truly Random Number Generation (RNG) in Decentralized System

4.1 General Thinking of RNG

- **Trusted Third Party Feeds**

The random number generator most often used in games is a feed from an existing lottery game, e.g. New York Lottery Quick Draw numbers. This is not viable because the feed can be cheated, and can not even be proven that the result is not selected in advance, which means some guy can decide the game result. The danger is that players have to trust a single point that could cheat or fail.

- **Benevolent Entity with A Provable Secret Key**

The idea is that the RNG should be a provable and deterministic process. The P2P nodes, or the players, could verify the fairness of the RNG after the ticket purchase and jackpots round.

A provable approach is achieved by publishing a one-way hash of the random selected secret ahead of time, such that participants can verify that hash one block later, when the secret is revealed.

This is easy to achieve by simply delegating the work to one central instance but unsafe benevolent entity. But there is a flaw here, benevolent entities with a secret key (e.g. classic satoshi dice) have the possibilities to cheating by submitting selective transactions. Benevolent entities have some advantage to others in this case, the secret is as random to the entity as other players so it can make use of that knowledge.

- **Future Event**

Another approach is to use future events as the random result, the randomness of future events could be determined and revealed at the same time. But the future event should be carefully selected, because there are cases that some entities could influence the result of future event. This can be resolved by selecting future event that can be hardly influenced/predicted, or by reducing the influence to future event from entities(e.g. increasing number of factors).

There are future events which are hard/impossible to predict("calculated out" in other words), so they are not determined thus random to the observers until they happen or appear. They will be determined and happen at the same time, and are revealed right away (no need for calculations), but not before that point.

- **Using of Randomness from A Blockchain**

We can add difficulty for a player to influence the RNG process by introducing proof of work (POW), this will make the player's factor more independent, prevent collusion or make it economically unprofitable to cheat . However, with significant hashing power one can more likely profit by submitting the block than holding out hoping to win the lottery.

For example, some data could be hashed and used as the feed from the Bitcoin blockchain and hybrid the data before generating a random number. As Bitcoin mining involves randomness, it's more secure for a random number generation.

Assuming there is a miner who is going to re-select another result rather than distribute it, he is losing his competition advantage to other miners. The time it takes before he can observe the result cannot be used to his advantage. This is the meaning of economically impossible, because POW reduces the miner's influence to cost of time.. Further, trying to find a collision is difficult with a probability space of > 476127 (to get third level prize of double color lottery), given that the miner has a maximum of 10x the time before other miners have the block for distribution, the possibility is still very low < 1/47612.

So a mining based approach creates a decentralized random factor that is probably good enough to start a DAC. But it's true that, even with the help of POW, miners still have a chance to attack. Miners, or more likely mining pool admins, have the chance of cheating by selectively discarding unfavourable blocks. The randomness would be better generated out of control of any entity.

- **Provable Distributed Randomness**

Without mining we could see something along the lines of each member of the Board of Directors (BOD) generating a secret random number in advance, revealing the hash of the network. Then after the

designated drawing block all members of the BOD reveal their secret key. The secrets are all hashed together along with the hash of one block header of the drawing block.

With this particular structure the BOD would be committing to their secret numbers long before the hash of the drawing block could be known. The only way to rig the drawing is for the BOD members to collude. If even one member is honest and keeps their information secret then the others are SOL.The more board members you have the harder it is for them to collude.

The entire process can be boiled down to the following process without a BOD.

1. Anyone who wishes to contribute to the Random Number Generation process publishes the hash of their secret HASH(S).
2. After all HASH(S) has been published all participants have an opportunity to publish S
3. After all participants published their S, HASH(S[0...N]) is calculated as the chosen random number.

Anyone concerned about the randomness of the result can participate in the process by publishing two transactions. Everyone else can simply choose to trust that the others are not colluding. If there is even one honest individual in the batch then the outcome is random . If all of the BOD contribute to the process then it can be assumed that there is a high probability that at least one of them is honest.

In this way everyone who wants it to be provably fair 'for certain' can know for sure that it was fair if they pay the minimum transaction fee. Everyone else can simply trust that it is fair and take the risk that everyone else is colluding against them. You could go as far as to have every ticket purchase include its own secret. Once the purchase window is over, everyone reveals their secrets. The hash of all secrets becomes the winning number. Because no valid transaction should ever be excluded from the block-chain for more than 1 or 2 rounds of the BOD we can safely assume that no one could know the random number generated in the end. But in the case, there are too many tickets involved in the RNG process, the time cost to collect all the secrets could be very high, it can not be guaranteed that all secrets can be collected before deadline.

4.2 Provable Distributed RGN Algorithm from DPOS:

It is a good balance to choose the RNG BOD members to be the 101 delegates of DPOS, this also allocates the cost of making sure it is provably secure to those who care about it the most. This means we would probably stick with letting the BOD do the drawing because they have a 99% uptime guarantee during RNG and are in general trusted. As long as one of them is honest then the resulting number is truly random.

With DPOS we have 100 nodes that should have near 99.9% uptime which means we can reliably produce a secure random number with assuming that just 1 out 100 is honest.

Code:

```
struct Block
{
  hash  secret;  // HASH( S[n] ) where n is the index in the array of secrets generated by this delegate
  hash  revealed_secret; // S[n-1]
};
```

For each block add a header field containing HASH(S[n]) where S[n] is a secret to be revealed next time this delegate produces a block. Also include in the header S[n-1].

We now have a stream of secrets being revealed once per block (15 to 30 seconds)... from this stream of secrets we can generate the random number R for the block as:

Code:

```
if( first_block_produced_by_delegate ) then Block[HEAD].revealed_secret = 0
ASSERT( HASH( Block[HEAD].revealed_secret) ==  GetLastBlockProducedByDelegate(Block[HEAD].delegate_id).secret )

R = HASH( Block[HEAD].revealed_secret )
for( uint32_t i = 1; i < 100; ++i )
{
    R = HASH( Block[HEAD-i].revealed_secret + R) // where + is concat
}
```

R = random number generated this block.

Every R is calculated from secrets introduced by all 100 delegates that were revealed after they committed to them. If at least one of the 100 delegates is honest then the resulting R is truly random.

Actually, "Block[HEAD].revealed_secret" the S[n-1] generated by HEAD's delegate in last round (each round 100 delegates' blocks). If we require the least security level of "at least one of the 100 delegates is honest then the resulting R is truly random", jackpots should be drawn out using 100th block's R when there are 100 blocks following from the block where ticket purchase transactions are accepted.

“Distributed” means that the random number of one block is generated by the previous 101 delegates’ revealed secret, so as long as at least one of them are honest, the resulting random number is truly random. “Provable” means that they need to publish the hash of their secrets to the blockchain in the next round. previous block which is at least ahead of a cycle (every delegate have at most one chance in one round, 101 blocks), so they can not cheating the resulting number by analyzing the player’s transactions or other delegates’ published secrets, and revealing selective favourable secret. The revealed secret’s hash must be the hash they published last time.

4.3 Why shuffle matters to DPOS?

Enough distributed factor sources are still not enough for us to get that randomness, we need to depend on the randomness process of collecting/communication to reveal them later. A shuffle makes all the 101 active delegates be involved in the process of collecting randomness, otherwise there could potentially be an attack which could affect the randomness collecting process, e.g.[10]. The shuffle makes that each delegate can only have one publish chance in each round, so they can not affect the random collecting process by introducing a new secret they know combined with predicting the delegates’ order, because the orders now is determined by the last randomness of the round, and each delegate only has one chance to publish its secret. All secret published by delegates are used to shuffle the delegates order of the next round, which means that the delegate cannot collude to control the orders if at least one of them are honest.

4.4 The way to resolve “Last Delegate being Evil Problem”[11]

In DPOS RNG algorithm, an evil delegate can choose to throw away an unfavorable random result by intentionally missing the block on his slot turn. This is the only thing they can do, but could potentially be a problem, when the ticket draw interval is less than a round (101 blocks), because an evil delegate can

predict which block he will produce in that draw interval and choose to buy the ticket which will draw in that block.

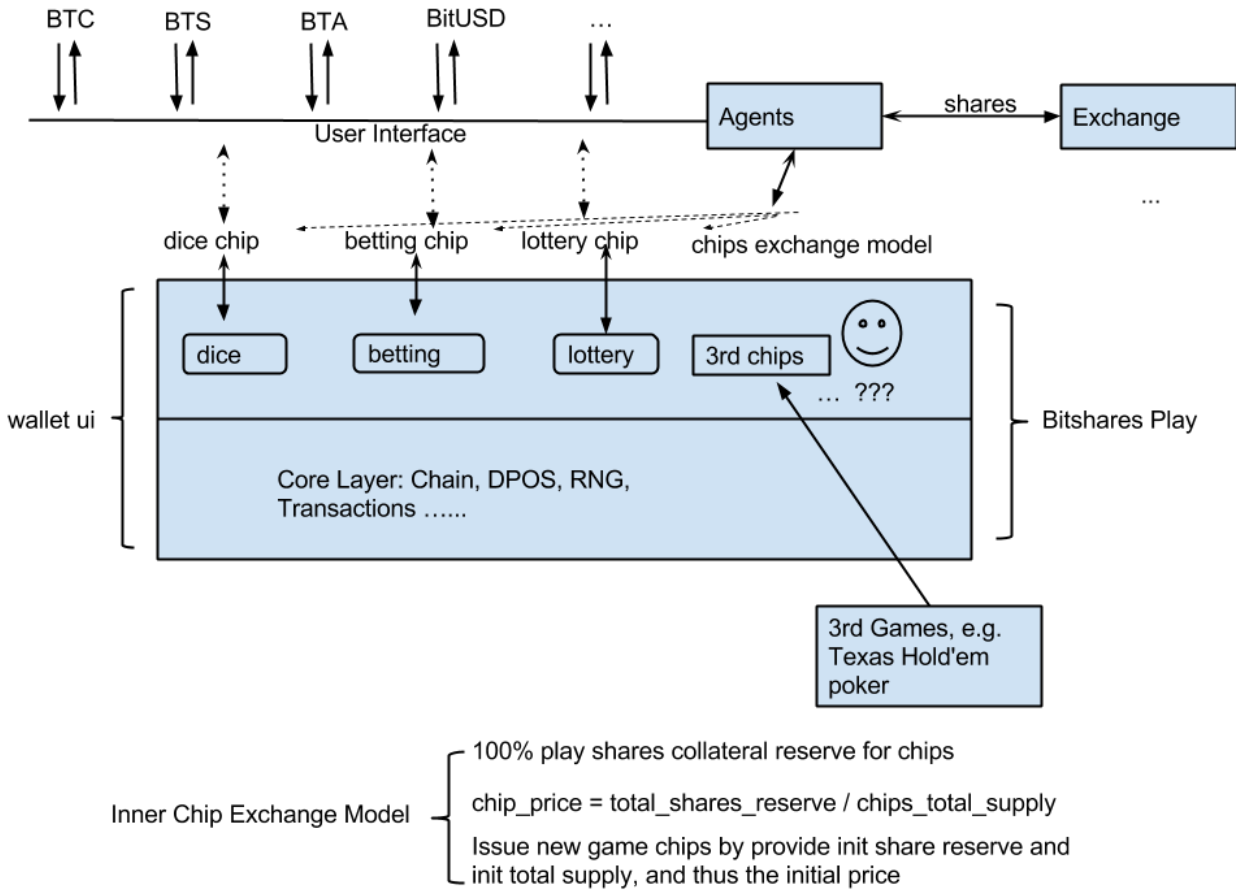
If the ticket draw interval is larger than 101 blocks, which means there will be at least one shuffle in that period, then the evil delegate can not predict which block he will produce. Then his only strategy is to guess or put tickets in each of the 101 blocks. When guessing, his chance is 1/101 and the expected return he can get back by attacking is the price of that ticket when he lose, because the delegate after him will continue to replace him and draw randomly. If he chose to put one ticket in each blocks, his attack cost is (101 * block_ticket_sale), but what is the expected return, still the ticket sale he put in one block, such that the total attack would be performed at a loss.

For some games 101 blocks draw interval is too long for their requirement, and a quicker drawing is needed. The solution is as follows, the ticket result will be drawn by 2 delegates:

The first delegate's random number is only in charge of producing a number X between 1 and 3 that determines that the Xth block after him will draw the random number for the ticket. The 2th delegate could be the evil guy who is trying to attack, but he can not predict who will produce the right drawing block before 4 blocks, his attack cost is (3 * block_ticket_sale), but his expected return is still only 1 block_ticket_sale. The only thing game rule need is to set the draw interval 1 block before the first delegate.

Note: block_ticket_sale means all the ticket sale the evil delegate buy in one block.

8.0 The Path to A Game Platform and Ecosystem



8.1 Rule Layer and Core Layer

BitShares Play is designed in two abstract layers, the rule layer and the core layer, which will make it very easy to create an alt-Play, for BitShares Play family. Users can customize the BitShares Play DAC, by customizing the rule configuration and the rule layer API. On the blockchain, there are only ranked lucky numbers and winning number, no rule configuration and jackpots drawing information. So all transactions much the core layer and part of rule could be accepted.

The communication and cooperation between these two layers are using the combinatorial number system (CNS) to map them. CNS will be described below.

The Play family DAC creators can select whatever kind of rules they like, color counts, ball count, k to select, prize definition etc. The outer rule layer will also provide a simple API for inheritors to customize prize rules. Here is an example to demonstrate what a rule will look like.

8.2 Large Prize Payout

To prevent winners of large jackpots from dumping their huge shares on the market, the payout should be delayed and spread over many blocks. This mechanism should be part of transaction validation, using similar feature like "nLockTime" from the Bitcoin protocol to lock/freeze the payment for several blocks. The

BitShares toolkit's transaction have a similar field called "valid_until", which should be the same with “nLockTime”. That is, if an output is a "reward" output, they will be split to up to N parts, each with 1 to N lock time and can only be spent after 1 to N blocks.

8.3 Common Requirements for Game Rules

- RNG (Random Number Generation). A random number needs to be generated by the BitShares Play DAC, which is used to calculate the winning number.
- Play/Game rule definition. There are a lot of game rules, lottery, dice, etc, but their models are pretty similar. Actually they have a lot in common, so that they can be combined to an abstract model/layer to define the rules.
- We need a mapping method to link the lucky number and winning number to continuous nature numbers, so we can simplify the problem space to a RNG for natural numbers. The lucky numbers are selected by users according to a rule model.
- Well designed rule models with a good economic balance are necessary, which could keep the DAC being self-sustained and continuous. There should be no jackpots giving out by failure because of rewards flaws.
- Large scale of prize should not corrupt the market of BitShares Play, e.g. prize owners might dump their jackpot to the market.

8.4 Combinatorial Number System(CNS) for Mapping Combinations to Natural Numbers

CNS is used to map the rule layer model to the core layer’s RNG process This is especially useful in mapping the lottery combination inputs to natural numbers, which can help custimize different rules with only some simple script language or configuration file.

In mathematics, and in particular in combinatorics, the **combinatorial number system** of degree k (for some positive integer k), also referred to as **combinadics**, is a correspondence between natural numbers (taken to include 0) N and k-combinations, represented as strictly decreasing sequences $c_k > ... > c_2 > c_1 \geq 0$. Since the latter are strings of numbers, one can view this as a kind of numerical system for representing N, although the main utility is representing a k-combination by N rather than the other way around. Distinct numbers correspond to distinct k-combinations, and produce them in a lexicographical order; moreover the numbers less than correspond to all k-combinations of { 0, 1, ..., n − 1}. The correspondence does not depend on the size n of the set that the k-combinations are taken from, so it can be interpreted as a map from **N** to the k-combinations taken from **N**; in this view the correspondence is a bijection.
The number N corresponding to $(c_k,...,c_2,c_1)$ is given by

$$N = \binom{C_k}{k} + \cdots + \binom{C_2}{2} + \binom{C_1}{1}$$

The fact that a unique sequence so corresponds to any number N was observed by D.H. Lehmer.[1] Indeed a greedy algorithm finds the k-combination corresponding to N: take c_k maximal with , then take c_{k-1} maximal with , and so forth. Finding the number N, using the formula above, from the k-combination $(c_k,...,c_2,c_1)$ is also known as "ranking", and the opposite operation (given by the greedy algorithm) as "unranking"; these operations are known by those names in most Computer algebra systems, and in computational mathematics.

Ranking/Unranking algorithm for multi-color balls lottery:

1. Given that there are c kinds of colored balls, each kind of balls has B_i balls, numbered from 0, 1, … B_i, users have to choose K_i balls from each kind ball as the combination of this color ball's combinations.
2. The final user selected balls are actually ball combination groups, each color has a related combination group, each group i is a combination with K_i balls select, such as $(C_1, C_2, C_3, \dots, C_{K_i})_{G_i}$.
3. First, mapping the Gi group combination to nature number using ranking algorithm of CNS, The number RG_i is the corresponding ranking number calculated according to $(C_1, C_2, C_3, \dots, C_{K_i})_{G_i}$, which are ordered incrementally.

$$RG_i = \binom{C_{K_i}}{K_i} + \cdots + \binom{C_2}{2} + \binom{C_1}{1}$$

4. For each group, there is a probability space of

$$SG_i = \binom{B_i}{K_i}$$

5. Then the final ranking number R is calculated as following

$$R = \sum_{i=0}^{c-1} (RG_i \times \prod_{x=0}^{i-1} SG_x)$$

6. The unranking is the reverse of the 3 to 5 steps.

In the ticket purchase block, the user selected combination group is converted to its ranking number as the lucky number which will be broadcasted in the claim_ticket transaction, further to be stored in blockchain.

After the winning number is out, and before the jackpots matching process, the winning number will be converted to combination groups using the unranking algorithm, the winning combination groups are used for detail jack pots much in the rule layer. By the same token, because nodes can only know the ranking of lucky combination groups, and can not generate the jackpot just according to ranking winning and lucky number, the lucky number also needs to be unranked before match.

The winning number is random number created by RNG, it has a u64_t type, but before the unranking, we should notice that the is a maximum probability space to the possible combination groups, which should be

$$TRG = \prod_{i=0}^{c-1} SG_i$$

The winning number need to do $(mod\ TRG)$ before unranking.

9.0 Cross-chain transfer based on Decentralized User Issued Asset Escrow

Systems like BitShares X have user issued assets which can represent tokens of some digital equity. If systems like this support escrow mechanisms, which mean destroy some Bit Asset (say PLS assets) in the exchange, and create same amount of tokens(PLS) in the BitShares Play system, and vice versa. This

is achieved by consensus communication between two systems, for example, if BitShares Play detects that some amount of PLS assets are sent to a escrow address, then, PLS with same amount will be created in the BitShares Play system, and vice versa. An escrow address is some special address of the system that no one knows the private key of.

This sound similar to Bitcoin's concept of side chains, but the difference is that with bitcoin side chain, there is only one token in that system, so any kind of escrow will have the risk to dilute that token or double spending. This is why we would need merged mining. But here, there are special user issued asset for its chain, representing exactly the same token mirror of it own chain.

9.0 References

[1] https://bitcoin.org/bitcoin.pdf
[2] http://en.wikipedia.org/wiki/Combinatorial_number_system
[3] http://www.bitshares.org/security/delegated-proof-of-stake.php
[4] http://chancecoin.com/technical
[5] https://classic.satoshidice.com
[6] http://letstalkbitcoin.com/bitcoin-and-the-three-laws-of-robotics/
[7] http://trade.500.com/dlt/
[8] http://blog.bifubao.com/en/2014/03/16/proof-of-reserves/
[9] https://bitsharestalk.org/index.php?topic=4164.0
[10] https://bitsharestalk.org/index.php?topic=4009.msg59991#msg59991
[11] https://bitsharestalk.org/index.php?topic=6764.0