Decentralized Peer to Peer Game Assets Platform, Integration with Third Party Games using Smart Contract

hackfisher@gmail.com

2014-08-04

1.0 Introduction

BitShares Play(PLAY) is an experiment to demonstrate and validate how a decentralized, autonomous game assets platform might be implemented. The platform features multiple provable games of chance, as well as the ability to integrate various third party games and their asset systems. Shares of PLAY(PLS) combined with all of the assets of both built-in and third party games form a free marketplace and exchange. Systems like this are also known as Decentralized Autonomous Companies (DAC).

The basic idea of a game asset platform is that it contains an inner exchange model. Each game asset in PLAY is part of a contract with the system defining the assets supply and the PLS collateral ratio required to issue its shares. Game assets can be issued by providing PLS shares as collateral, and can be destroyed to recover PLS. The price of asset shares is determined by the assets defining contract using the current total supply and current collateral. So the total supply mechanism of game assets is also part of the contract with the system. For built-in games of chance the total supply changing rules is part of the DAC consensus.

Chance games have come to rely almost exclusively on trusted third parties to provide random feeds. While this system works well enough, it is based on a centralized trust model, and suffers from the possibility of cheating by players. Even though some crypto based games on the Internet have provably random feeds which can be verified by the public, hidden players can cheat by submitting selective favorable transactions because they know the random secret in advance. Thus, players are forced to trust that a game operator is scrupulous. This reduces demand for these games and prevents those who simply do not trust the operator from playing.

PLAY game assets (CHIP) are not limited to built-in games. They can be used as economic system by third party games and even centralized games by providing a contract which matches the requirements of the PLAY games assets inner exchange. This contract could be a consensus inside of or between DACs, or a smart contract between a DAC and centralized game. This can be achieved using smart contracts technology like smart oracles or Orisi.

2.0 System Tokens

PLAY shares (PLS) are the tokens of the system. PLS are designed first to be the shares of the PLAY System, and thus the unit in which dividends are paid to PLS holders and delegates of PLAY are paid for their contributions to the system. Secondly, PLS are the token with which users purchase game assets to play games. PLS provides an opportunity to expand the model of token distribution, and may explore models in which a game rewards winners with the tokens of the game system itself.

There are other tokens in the system such as games assets, including user-issued assets or PLS backed assets, etc. The PLS backed assets(CHIPS) is special in this system, 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 to the total supply of CHIPS. This means there will be a flexible supply for the chips according to the user demand for the game. As a game becomes increasingly profitable and popular more PLS will be directed as collateral for that game increasing its number of CHIPS. At the same time, no chip is created from nothing because every chip is backed by PLS. These PLS will be returned when destroying chips. The initial collateral and supply is set by the game's creators, as is the price. Once the price is set, the creator cannot alter it anymore, it is calculated by the total PLS supply, the collateral people add to the chips, and the game rules. There will be different CHIP in this system, normally one CHIP asset for each game.

People can buy PLS to exchange for CHIPS in the inner system, which can be used to play the games. All spent CHIPS are returned as rewards for players except for a small percent as the "house edge". House profits are managed by the delegates, who may take them as pay or donate them to charity. A portion may also be destroyed, resulting in dividends for the PLS shareholders. In this system, functioning with honest delegates, 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 desirable for both types of buyers.

PLAY CHIPS assets provides a good economic model to protect the user's game assets from being diluted. Because the platform must find way to guarantee that the new created chips must be created by providing play shares as collaterals, or according the mechanism as a game contract hardcoded in PLAY. Games in PLAY should not be given the ability to issue chips as they want (as benevolent entity).

2.1 Introduction to default inner exchange model between PLS and chips:

CHIPS are different from BitShares X market issued assets like BitUSD and user issued assets. There is an exchange model between PLS and chips, and no market, which is part the BitShares Play 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.

Game providers can code their business model into the game as they want, but the rules related to total supply and the PLS collaterals must be transparent in PLAY. There will be a system conversion 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 words create/destroy) chips according to the price of the 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 each block, the price will be recalculated, according to the updated PLS collateral supply and chip supply.

In this way, the more popular games will have more collateral, and the more profitable games will have a better chip price relative to PLS and thus other equities. Games with dilution are also possible, but being different with unstrained dilution similar to Tencent's Q coin, the contract between game and PLAY, total supply and dilution rules must be hardcoded in the contract rule.

In order to use chips as their game tokens, each game contract requires:

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

3.0 Consensus Algorithm using DPOS

Crypto technology and the blockchain concept from Satoshi's "Bitcoin: A Peer-to-Peer Electronic Cash System" enables a decentralized and trustless ledger, allowing accounts to exist. Consensus algorithms like proof of work (POW) and DPOS are required to update and maintain the network and public ledger while maintaining the security and stability of the system.

The advantage of DPOS is that it allows for faster block confirmation and also allows for scalability to the level of VISA's 10,000 payment transactions per second. Meanwhile the system is still decentralized in that there is neither a single point of failure nor a single point of control. Delegates have the rather simple job of signing new blocks and can be fired on command if they do not perform their duties. Consensus on a whole is reached by stakeholders, whereas in traditional Proof-Of-Work schemes only hashing participants contribute to the network consensus, leaving shareholders who don't mine unrepresented.

The Delegates are the entities which make all the magic happen. They are involved in the decentralized 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 role, because they not only collect transactions, produce and sign blocks on their scheduled time slots, but also provide provably fair, distributed random secrets for use in games. Please refer to section 4.2 for details.

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; it cannot even be proven that the result is not selected in advance, which means someone with inside access could fix the outcome. The danger is that players have to trust a single entity that could cheat or fail.

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

Ideally the RNG should be a provably random and unpredictable beforehand, but also deterministic and easily reproducible for verification after the fact. The P2P nodes, or the players, should be able to verify the fairness of the RNG after the ticket purchase and jackpots round.

A provable approach is achieved by publishing an 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 a central entity, but there is a flaw here: Any entity with a secret key (e.g. classic satoshi dice) has the possibility to cheat by submitting selective transactions. Benevolent entities thus have an advantage, the secret is not as random to them as to other players so they can make use of that knowledge. Requiring trust that an entity will be continually benevolent is a serious weakness.

- **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 in which some entities could influence the result of future event. This can be resolved by selecting a future event that is difficult toy influence/predict, or by reducing the influence the entities have over the future event (e.g. increasing number of factors).

There are future events such as some from radioactive sources which are hard or impossible to predict or calculate out. Because they cannot be determined by observers in advance, they are practically 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 cannot be anticipated.

- **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 hope to win the lottery.

For example, some data could be hashed as salt, aggregated with a Bitcoin block hash, and that aggregate hashed to generate a random number. As Bitcoin mining involves randomness, it's more secure for random number generation.

Assuming there is a miner who is going to re-select another result rather than distribute it, he is losing his competitive 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**

POW can create a competitive environment in which each participant (miner) cannot manipulate the random result or at least economically unprofitable. Without POW we could see an approach of distributing the factors to as many entities as possible, such that each entity can not cheat. To be simple, imagine these

entities being a 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 unable to predict the result. The more board members you have the harder it is for them to collude.

4.2 Provable Distributed RGN Algorithm from DPOS:
The approach of provable distributed secret feeds can lead to a truly random number generator (RNG) algorithm. DPOS is using such a RNG algorithm, there it is used to shuffle the delegates ordering randomly in each round.

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.

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.

“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. The revealed secret’s hash must be the hash they published last time. They can not cheat by revealing selective favourable secret because the hash of the secret must be included in previous produced block.

So in DPOS, we can have the pseudo-code with fixed number of delegates as following:

Code:
```
struct Block
{
   hash  HASH( S[n] ) // where n is the index of secrets generated by this delegate
   hash  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
}
```

Where the random number generated by this block is denoted by *R*.

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`" is 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.

In this way everyone else can simply trust that it is fair and take the risk that everyone else is colluding against them. In the chance game case, 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 that case there are too many tickets involved in the RNG process, the time cost to collect all the secrets could be very high, and it can not be guaranteed that all secrets couldbe collected before the deadline.

4.3 Shuffle In DPOS

A shuffle in DPOS occurs once all 101 delegates have signed a block, and the order of the delegates is shuffled randomly according to the random seed. This is a way to keep the decentralized consensus from being attacked by malicious delegates. Each delegate only has the choice to publish the block or not on his turn, aka. reveal the secret or not. If there is no shuffle, malicious delegates can influence the random seed by not revealing the secret, and choose the favorable delegates order as he want.

We can summarize this as not enough randomness in the process of collecting distributed random factors, so a shuffle is added, introducing the randomness into collecting process. 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 sure that each delegate can only publish one chance in each round(assuming that there is no missing block), 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.

We can define *BLOCK_TICKET_SALE* as all the ticket sale one malicious delegate will buy in one 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 *X*th 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.

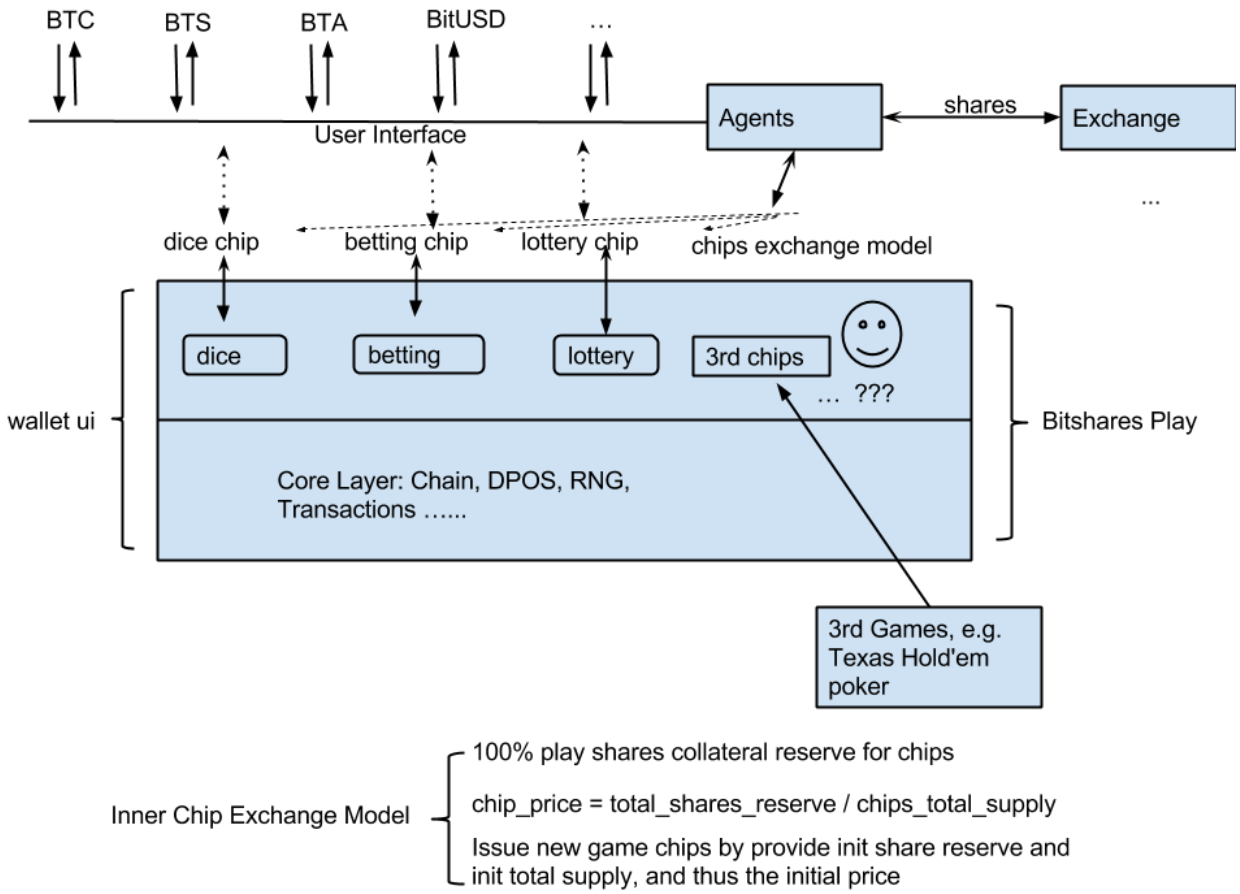
5.0 The Path to A Game Platform and Ecosystem

5.1 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 decouple the game integration and BitShares Play asset model. In the rule model, people can develop built in games inside the BitShares Play DAC, or integrate chip asset operations with 3rd games with the help of smart oracles - DPOS delegates play an important role in the integration as smart oracles.

The core layer performs the blockchain and ledger tasks. The rule layer is designed to allow others to develop play games on it and to keep the tokens economically self-sustained, safe and integrity at the same time.

Game assets must be able to safely follow their contract with PLAY, which is untrusted and may actually be malicious. This 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.



5.2 Built-in Chance Games Like Dice , Betting and Lottery As Part of The System

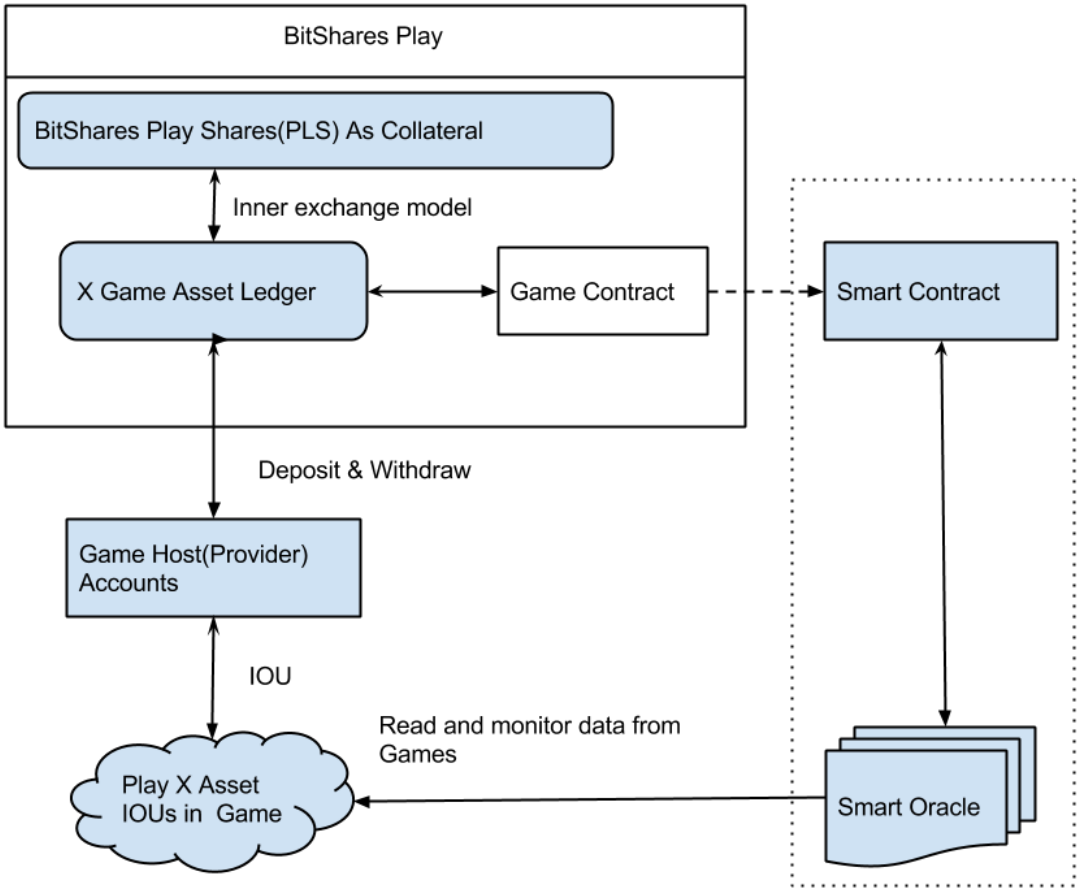
- RNG (Random Number Generation). A random number needs to be generated by the BitShares Play DAC, which is used to calculate the winning number.
- Play/Game rule definition. There are a lot of game rules, lottery, dice, etc, but their models are pretty similar. Actually they have a lot in common, so 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. For lotteries' combination input, we can use the Combinatorial Number System (CNS) to help us.
- Well designed rule models with a good economic balance are necessary, which could keep the DAC self-sustained and continuous. There should be no jackpots given out by failure because of reward flaws.

- Large scale of prize should not corrupt the market of BitShares Play, e.g. prize owners might dump their jackpot to the market.
    - To prevent winners of large jackpots from dumping their (possibly) huge shares on the market, the payout should be delayed and spread over many blocks. This mechanism should be part of transaction validation, using a feature like similar to "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 provide this feature. 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.

5.3 Integration With Third Party Games

Assets or points in traditional games can not withdraw to sell, some others which have external market are essentially IOUs users stored in game provider. These kind of game assets have no collaterals to back them, and normally have no built-in support for deposit and withdraw to other assets, so their market is not fluid.

Here we introduce a integration model with games which can take advantage of BitShares Play's Game Asset.



Game hosts here is entities which not only support PLAY games assets deposit and withdraw like digital exchanges, but also provide games softwares and services for that game assets. A game session is an

semi-permanent interactive information interchange or conversation between user and game. For example, during a session, a user buys game assets, plays games, and withdraws the assets after the games are over. For third party games, the assets might exist in the form of IOUs in those games sessions. The best practice for users is to withdraw the assets from games to PLAY after each game session, to protect them from malicious games hosts or providers. Game hosts can also choose to prove their game assets reserves using technology like merkle tree. But this can not prevent game hosts from creating non-exist assets in their games assets. Cheating by creating non-exist assets could be exposed when user can not withdraw from the game hosts. And It's better for game hosts to provide transparent API for smart oracles to monitor and audit.

Smart oracles are playing an important role as bridge between DAC and "real world", by providing consistent API between DAC and outside systems, including traditional centralized servers. Assuming that there are Y smart oracles in total, if X-of-Y of them are returning the same (or consistent) output for the same input, than we can conclude that this call of API is valid. In this way, we provide a robust decentralized solution for DACs, and can interactive with "real world" at the same time. The DACs need to trust Y smart oracles, but the risk is very low, as soon as X-of-Y oracles are not colluded, than the result is honest. We can get the optimization if the smart oracles have votes and are selected by all the shareholders of PLAY, just like the delegates in DPOS, further more, we can even use delegates as the smart oracles of the system directly. For more details about smart oracles, refer to Section 6.0 References [14].

Third party games can themselves being as DACs, in this case, the deposit/withdraw could slightly be different from centralized games. The assets in DAC games are no longer IOUs which have higher risk. The deposit/withdraw between other DACs and PLAY can be achieved by system escrow mechanism, a way to transfer assets cross-chain from one DAC to another DAC. For example, there is a pair between two DACs A and B, both supporting such a mechanism, there is a A-B escrow address, the tokens send to it will disappear in one DAC, and show up in another.

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 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.

In this way, assets in DACs are interoperable. This sounds like bitcoin side-chain proposal and two-way pegging[15], but the difference is that with bitcoin side-chain, there is only one token in bitcoin, a.k BTC, so any kind of escrow will have the risk to dilute that token or double spend that. The problem of bitcoin side-chain proposal is that it need merge mining between two chains, in the case of side-chains between POW based chains, the weak chain with less hash power can be easily 51% double spend attacked by stronger chain which has much more hash power. If the assets between two chains are pegged, then they should be protected by the same hash power, thus requiring merge mining. But this problem does not apply to system escrow mechanisms because there are special user issued asset for each chain, representing exactly the same token mirror of it own chain. Besides, shares in DPOS chains like BitShares X or BitShares Play are protected by the shareholders of each DAC, trying such 51% attack means they want to change the system escrow consensus part.

6.0 References

[1] https://bitcoin.org/bitcoin.pdf
[2] http://en.wikipedia.org/wiki/Combinatorial_number_system
[3] http://bitshares.org/security/delegated-proof-of-stake/
[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
[12] http://www.random.org/randomness/
[13] https://blog.ethereum.org/2014/07/22/ethereum-and-oracles/
[14] https://github.com/codius/codius/wiki/Smart-Oracles:-A-Simple,-Powerful-Approach-to-Smart-Contracts
[15] http://www.coindesk.com/bitcoin-core-developers-bitcoin-side-chains/