PoS Cryptocurrency wtih No Blockchain

by qianxiaochao

fxxfzx@gmail.com

btc: 18Bf718HZNboFvEqQuhJCTdKJtM8YxHPZW

Jun 2014, rev7

Abstract

There are some alternative Cryptocurrency systems which claim that they are based on PoS are actually based on PoSTW which denotes the Proof of Stake(coin), Time(day) and Work(hashing), while the other pure PoS Cryptocurrency systems are actually centralized. In this paper we propose a new framework of Cryptocurrency system. The major parts what we have changed include, a fast transparent distribution solution which can avoid deceptions between the sponsor and the audience, removing the bloated history transactions from data synchronization, no mining, no blockchain, it's environmentally friendly, no checkpoint, no exchange hub needed, it's truly decentralized and purely based on proof of stake. The logic is very simple and intuitive, 51% of stakes talk. The highlight of this paper is a proposal of a new concise data synchronization mechanism named "Converged Consensus" which ensures the system reaches a consistent distributed consensus. We think the famous blockchain mechanism based on PoW is no longer an essential element of a Cryptocurrency system. In aspect of security, we propose TILP & SSS strategies to secure our system. At the end, we try to give an explicit definition of decentralization.

1 Transparent distribution

As the new system is purely based on proof of stake, naturally, we must have an initial statistical distribution of stakes among the users. In this new framework, the initial distribution of stakes and the implement of the system is completely separated and independent. The initial distribution of stakes is just treated as a parameter to be input into the system, a different initial distribution of stakes means a different Cryptocurrency system. Whatever way the initial distribution is acquired, it's qualified if it's widely acknowledged and trusted by the users. We can even directly fork a distribution of stakes in the blockchain of some existing Cryptocurrency system, such as Bitcoin system, to be input into a new system, then the new system which inherited the distribution of stakes of the Bitcoin system will start to run under the new framework [APPENDIX A].

Here, we'd like to introduce a new method of fast distribution of stakes within several months. It's a completely transparent distribution, no one can cheat including the sponsor himself. Suppose an internet company X-Company plans to issue 10 billion of Cryptocurrency named "X-Coin".

Who are the audience of the distribution?
All existing accounts from some existing Cryptocurrency systems, say Bitcoin Litecoin or Dogecoin(sorry, there is no absolutely fair distribution), which had activities before the day Jan 1st, 2014 and the balance is above zero.

What's the manner?
A qualified participator signs a message with his old private key of the original system and sends it to the X-Company to register an account. The message consists of the old public key of the original system and the new public key of the X-Coin system. If the message is verified by the X-Company, he will be given some free(or for sale, it depends) stakes. An example division of stakes is, the X-

Company reserves 30% of all stakes, the other 70% of stakes are given to every registered account equally, the number of registered accounts is constrained under 50,000. The distribution will be expired within 6 months.

What are the advantages of this distribution policy?

1. The target audience are the most active and the most experienced players in the Cryptocurrency field.
2. The X-Company can't cheat people to reserve more than 30% of stakes. Because the ledgers of Bitoin Litecoin or Dogecoin are public and unchangeable.
3. Players can't register tons of accounts to cheat the X-Company.
4. It's simple, fast and consequently highly feasible.

OK, the distribution of X-Coin is now done. Then the X-Company publishes all related data during the distribution and hardcodes the initial distribution of stakes into the source of the X-Coin system, and will gradually open the source. Now people start trading X-Coins. The distribution result and the initial price of X-Coin both depend on the operation of the X-Company. At this point, the state of the X-Coin system is similar to the state of the Bitcoin system when all Bitcoins are mined off by the year 2140. There is no mining process in the X-Coin system.

2 Data synchronization

2.1 Throw bloated history transactions away

Why should we strip off history transactions from data synchronization?

1. Over time the history transactions become more and more bloated and harder and harder to deal with.
2. Only the data that is necessary for everyone has to be synchronized by the whole network. In real life, will you keep other people's transactions that have nothing to do with you? No, everyone only needs to keep his own history transactions.
3. The most important, during thousands of years of history of gold trading, there was no common history transactions at all. Since we build the X-Coin system to mimic gold on the internet, it's reasonable not to synchronize the whole network's history transactions. Whereas, everyone can choose to keep history transactions, it's optional.

2.2 Balance view

Since we don't need mining anymore, we replace the concept of block and blockchain with normal data list. Then how could we synchronize data? Our solution is, all nodes in the X-Coin system share one record list of everyone's balance. Balance view is a data list composed of everyone's balance record. It's a snapshot of all accounts' current balances. The initial balance view is provided by the X-Company after the currency distribution. We will elaborate on how a node initiates or updates balance view later in the security section.

user's balance record:
[pubkey, balance, time last activity]

We set a special balance record for recycled money in the X-Coin system:
[special key, balance, time last modified]

whole system's balance view:

```
[
sn,               //sequence number of a balance view increasing continuously from 0.
base view sn,     // it equals to sn – 1.
base view hash,   // hash of baseview based on which this balance view is calculated
hash of tx_package_51,    // hash of tx_package_51 which is used to calculate this balance view
[record1, record2, record3... recycled money record],    //every user's balance record
hash    //hash of this balance view
]
```

Baseview := A balance view based on which some important actions are taken. Baseview is a core concept in data synchronization. Almost all interactions between nodes are based on certain common baseview.

2.3 Transaction processing

2.3.1 Transaction agent

For processing transactions more efficiently, every transaction is executed by a transaction agent. A transaction agent is a node that collects transactions that match his criteria setting from network into a transactions package(tx_package). Then broadcast the tx_package to the network to be verified and to be vested with other nodes' balance numbers. Tx_package has a upper limit of size.

```
transaction:
[
base view sn,        //based on which this transaction was generated.
base view hash,
sender pubkey,
receiver pubkey,
volume,
tx_fee,              //max tx_fee the sender will pay for this transaction
timestamp,
signature of sender
]
```

2.3.2 Applying for account

An existing account sends some money to a fresh pubkey. After the transaction is confirmed, a new account is created.

2.3.3 Generating transaction

The sender should set a max tx_fee he will pay, for every new transaction. Every new transaction will be collected by some agents who share the same baseview with the sender and the sender's transaction criteria setting matches a agent's criteria setting. It's possible that one transaction is collected by multiple agents. The agent who wins the vesting competition actually gets tx_fee. Every node has the right to set a minimum of balance of qualified agent to make sure the agents are all serious and constrain the number of qualified agents and consequently the number of tx_packages generated by the agents. It's a bi-directional selection between sender and agent. We try to let the nodes independently adjust the parameters as many as possible to make the nodes compete freely and eventually make the system self-equilibrated and self-organized.

```
transactions list
[
base view sn,       //common base view shared by all txs in this txs list
base view hash,
[tx1, tx2, tx3... ]
]
```

```
vester item     //vester means someone who vests in a tx_package with his balance number
```

```
[
vester's pubkey,
vester's signature      // every vester signs its predecessor's signature in a tx_package.
]

vester items list
[
[item1, item2, item3... ],
timestamp of last item
]

transactions package
[
base view sn,        //common base view in this tx_package
base view hash,
agent pubkey,
transactions list,
timestamp,
signature of agent,
flag_51,                 //indicates this tx_package has gotten 5.1 billion vesting
vester items list
]
```

2.3.4 Vesting

After receiving and verifying a tx_package, every vester node will vest in this tx_package with his balance number and then broadcast the tx_package. A vester has the right to set criteria such as minimum of total fx_fees in a tx_package and a minimum of balance of an agent when vesting. The agent also can set a minimum of balance of a vester. It's a bi-directional selection between vester and agent. There must be a vester node finding that the sum volume of the vester items list in a tx_package is bigger than 5.1 billion(51% of total volume, "tx_package_51" for short). That indicates transactions in this tx_package_51 are all accepted by the whole network. Then this vester node will iterate every transaction in this tx_package_51 to calculate a new balance view by simulating the execution of all transactions in this tx_package_51 and then divide tx_fees to the agent and all vesters' accounts according to a defined formula and update his balance view and set the flag_51 of this tx_package to true, finally, broadcast this tx_package_51 to announce that a new balance view is accepted by the whole network. Every node will calculate a new balance view after receiving this tx_package_51.

Rules for data synchronization :

1. An agent only collects transactions which share the same baseview with him and match his criteria setting. (Same baseview refers to same baseview_sn and same baseview hash.)
2. An agent makes and broadcasts at most one tx_package per baseview_sn exclusively. (This rule prevents the agent from making multiple tx_packages based on one baseview to increase the probability of success, that is meaningless and will increase the load of communication.)
3. A vester vests in one tx_package only once per baseview_sn exclusively. (Multiple tx_package_51s is possible. This rule ensures tx_package_51s are all from one agent.)
4. A tx_package gets more than 5.1 billion of vesting indicates it had been accepted by the whole network.
5. Every node only accepts one tx_package_51 which has the same baseview with him per baseview_sn exclusively.

Agents and vesters have incentive to break the rule 2, 3. So we set a rule to punish the users who break the rule 2, 3. Every node randomly buffers a small ratio(say 0.1%) of history tx_packages with current balance_view_sn when it went through and checks if there is someone signed more than one tx_package or signed some tx_package more than once. In the X-Coin system, everyone is monitoring everyone, by checking these buffered history tx_packages, so it's impossible for a node who breaks the rule to escape being punished.

violation report transaction:
[
violation type,
reporter pubkey,
accused pubkey,
proofs,
timestamp,
signature of reporter
]

Every node will broadcast a violation report transaction when he detects a violation event. Every agent will collect this special transaction if verified. The last vester will verify proofs and will recycle 50% of balance of the accused account if verified. The violation report transaction has a higher priority than common transaction. We generalize the concept of transaction. All import actions can be executed by such a generalized transaction.

2.3.5 Insufficient vesting

If the vesting is insufficient to generate a tx_package_51 then we just wait and retry. Any agent or vester can manually rebroadcast the last tx_package if he can't wait.

2.4 Converged Consensus

Here we come to the heart part of this paper. The phrase "consensus" is a common saying of data synchronization. If the balance view data of the whole network is consistent then there is a consensus, otherwise, the consensus is split. According to my understanding, a Cryptocurrency system is basically a distributed database system and the data synchronization problem is the core problem of a distributed database system. So the most crucial and the most difficult point of a decentralized Cryptocurrency system is to ensure the whole network reaches a distributed consensus. In the past,the famous blockchain mechanism based on PoW is the only feasible approach for a decentralized system to reach a distributed consensus. But now, we propose a new mechanism named "Converged Consensus" purely based on PoS which can also achieve the same goal and will be more efficient and cleaner.

2.4.1 Rules for Converged Consensus

First, let's think about two math problems:
There are 100 red balls and 100 black balls in a bag, someone randomly selects 5 balls, if red balls are more than black balls then one black ball will turn to red ball, vice versa. He does this time and time again, eventually colors of all 200 balls will converge to one single color, all black or all red. This can be mathematically proven.
Another one is, there are 200 balls in a bag, if someone randomly selects 5 balls are all red balls and this happens 20 times continuously, then the probability that most of 200 balls are red balls is extremely high.

We use a similar concept to resolve the split consensus of balance view of the whole network.
It's possible that more than one tx_package_51 is found by different vesters. Then the consensus of

the new balance view is split. Suppose the consensus is split to partA, partB, and partC.
Referring to previous math problems, we add 3 more rules:

6. Every node keeps updating (corresponding to change the color of a ball) balance view periodically and all the time to make partA, partB, and partC converge to one stable and consistent balance view.
7. Every node takes action (send transaction, make tx_package or vest in tx_package) only after continuously N (N controls the degree of convergence of consensus) times of updating balance view with no change. (It's supposed that the current balance view is stable and consistent if there are N times updating balance view with no change and it becomes the new current baseview).
8. When updating balance view, the winner balance_view_sn has to be bigger than a node's current baseview_sn.

We set a rule for punishing someone who breaks rule 7 to win the race against time. If someone is found that he had taken action on a non-main-consensus part, i.e. taken action on a different consensus part with you, then it is supposed that he had not done enough updating. He will be locally blacklisted. Although this is not 100% accurate, we can adjust some parameters to get the best equilibrium. Every node has the responsibility to make sure that he will be in the main-consensus-part, otherwise, he violates the common baseview principle, and will be omitted by the mainstream. This circumstance is analogous to miners trying to mine on the main blockchain otherwise his work won't be acknowledged. Now we have two forms of punishment, global punishment in the common balance view data and local punishment in a node's client.

These rules ensure that the penultimate baseview is always consistent and irreversible. Whenever a node gets a new baseview by N times continuously updating balance view with no change, it's the time for a transaction sender to completely confirm his transaction. Meanwhile, buffered history data relate to this baseview can now be discarded. Note that, the current baseview is always stable and the current balance view is volatile.

2.4.2 Proof of Converged Consensus

Model description of Converged Consensus

Suppose there are K nodes online, the distribution of weight(balance) is given, and the current distribution of balance view data s_0 is given.

Updating algorithm in this model:

```
i = 0
while( i++ <  n )
{
    randomly select M nodes        //M is a given natural number
    read the hashes of their current balance view data
    make weighted statistics
    get a winner hash of balance view data  //if there is a tie, any one is OK
    randomly select one node
    update its data with the winner data
}
```

Define S := { s | s is any possible distribution of balance view data starts from s_0 }

Obviously S is a finite set.

Define P : = (0,1], E := <S,S,P>

We make a Markov chain based on S, C := <S,E>.

Define S1 := { s | s $\in$ S and s has a outgoing edge with the value 1 in the Markov chain C }

Define S0 := S - S1

If s $\in$S1 then s has a recurrent edge with the value 1. So if we reach the set S1 then there will be a irreversible consistent consensus. The set S1 is our target.

The initial state $s_0 \in S$ is given. Suppose after n steps of transitions the state turns to $s_n \in S$.
Claim: $\lim_{n\to\infty} p\{s_n \in S1\} = 1$.

Proof :
Define Ω := <E,E,E...E>. It's all possible n steps paths.
Define Γ := { γ | γ ∈Ω and γ is any possible path of n steps transitions start from s_0 }
Define Γ1 := { γ | γ ∈Γ and the last state $s_n \in S1$}.
Define Γ0 := Γ – Γ1
Define k := The cardinal number of Γ0
Define v(e) := the probability value of the edge e ∈E
Define m := The max probability value of all edges in the Markov chain C and m != 1

The probability of path γ is $p(\gamma) = \prod_{i=1}^{i=n} v(e_i), \quad e_i \in \gamma, \gamma \in \Gamma$

$$
\begin{aligned}
\because\ & p\{s_n \in S1\} \\
= \ & 1 - p\{s_n \in S0\} \\
= \ & 1 - \sum_{i=1}^{i=k} p(\gamma_i), \quad \gamma_i \in \Gamma 0 \\
= \ & 1 - \sum_{i=1}^{i=k} \prod_{j=1}^{j=n} v(e_{ij}), \quad e_{ij} \in \gamma_i \\
>= \ & 1 - \sum_{i=1}^{i=k} \prod_{j=1}^{j=n} m \\
= \ & 1 - k * m^n \\
\therefore\ & 1 \geq p\{s_n \in S1\} \geq 1 - k * m^n \\
\because\ & \lim_{n\to\infty} (1 - k * m^n) = 1 - 0 = 1 \\
\therefore\ & \lim_{n\to\infty} p\{s_n \in S1\} = 1
\end{aligned}
$$

Proven.

We can imagine that at first the convergence rate is relatively slow, it will become faster and faster as the consensus will become more and more consistent. There is a powerful positive feedback effect like snowballing phenomenon in the converging process. This implies that the average convergence rate should be fast, so this is a very good merit for practical application.

2.4.3 Example formulas for N and M

Define A := The number of all serious accounts.
We start from an initial distribution, so suppose A > 10000.
Define B := [lg(A)] + 1
Define M := The number of sample nodes in once sampling,
M := B*B.
Define N := If there are continuously N times updating balance view with no change then the balance view data of the whole network is supposed to be stable and consistent.
N := B*B.

Table 1. Examples of N and M

A	B	**M**	**N**
100-999	3	**9**	**9**
1000-9999	4	**16**	**16**
10000-99999	5	**25**	**25**

100000-999999	6	**36**	**36**
1000000-9999999	7	**49**	**49**
10000000-99999999	8	**64**	**64**
100000000-999999999	9	**81**	**81**
1000000000-9999999999	10	**100**	**100**

The problems about statistical inference seem very complicated. Some professional statisticians are needed to study the details. It seems PoW is more about game theory and PoS is more about statistics. Anyway, it's all about mathematics including cryptography.

2.4.4 Assertion

We assert a truly decentralized PoS Cryptocurrency system can't bypass the “Converged Consensus” mechanism.
If you want a top stands stably, make it keep spinning.
If you want the system reaches a consensus, make all nodes keep updating.
Don't you think the “Converged Consensus” mechanism is extraordinarily simple?
“Simplicity is the ultimate sophistication”
“大道至简”

Figure from Mader tops

3 Miscellaneous

3.1 History transactions

Since we don't synchronize the whole system's history transactions, how can we ensure the receiver will not deny reception? The answer is, once a node receives a tx_package_51 he may choose to save the tx_package_51 to a disk as memo or as proof. If you trust your receiver you just simply send money to him, and then keep tx_package_51s as memo. If you don't trust your receiver, you synchronize baseview with the receiver by getting a common baseview agreement with the receiver's signature before you send money to him and then keep two history tx_package_51s(one is the tx_package_51 which is based on the agreement baseview and actually took effect, the other is any tx_pakcage_51 which based on the new baseview) after you send money. You also can acquire tx_package_51s from other nodes, especially the agent of this transaction, he has the obligation to keep tx_package_51s for you. With common baseview agreement and two history tx_package_51s, the receiver has no way to deny reception.

3.2 Client load

The data load of a client will decrease dramatically, because we eliminate history transactions from data synchronization. The Bitcoin system has now about 1 million active accounts(data from bitcoinrichlist.com, at block 295,000). Suppose the X-Coin system has the same number of

accounts as the Bitcoin system. A balance record needs about 32+8+8=48 bytes. Then the data size of balance view is about 48m bytes. In aspect of computing load, the main computing tasks are produced by the verification of tx_packages, so the computing load depends on the scale and structure of the X-Coin system.
Basing on the transparent distribution, the length of the vester items list is about :

1 + (5.1b – 3.0b) / (7.0b / 50,000) = 15,001.

The size of a vester item is :

32 + 64 = 96 bytes.

The generation rate of transaction of the Bitcoin system is about 0.8 tx/s (data from blockchain.info,May 2014), a Bitcoin block carrys about:

0.8* 10 * 60 = 480 transactions.

The size of a transaction is :

8 + 20 + 32 + 32 + 8 + 8 + 8 + 20 = 136 bytes.

The size of a tx_package is about :

15,001 * 96 + 480 * 136 = 1505376 bytes = 1.44 MB.

The computing load is about :

(15,001 + 480) / (10 * 60) = 26 verification/s.

If we make one tx_package carry 10,000 transactions and suppose the transaction generation rate is 10,000 tx/ 600s = 17 tx/s. Then the computing load is merely :

(15,001 + 10,000) / 600 = 42 verification/s.

The size of a tx_package is about :

15,001 * 96 + 10,000 * 136 = 2800096 bytes = 2.67 MB.

For a ordinary home PC produced nowadays, such scale of client load is trivial, it's easy for any user to synchronize data or verify tx_packages or vest in tx_packages. Under such condition, everyone can open up a bank at his home and trade coins by the client without any medium of exchange hub. Our slogan is "Everyone has a bank at home". Generally speaking, the agents tend to contact vester nodes with large balance first and the vesters tend to merge. Like mining pools in the Bitcoin system, stake pools may emerge. So the length of the vester items list will be shorter than 15,001. Don't worry about the stake centralization problem. Because if any single entity controls too many stakes, say 50% of all stakes, that must compromise the X-Coin system's robustness and credit and consequently directly compromise that entity's interest, there is a negative feedback effect to resist the stake centralization.

3.3 Deflation

A well known problem in Cryptocurrency system is, over time, more and more private keys will be lost forever. To make sure the money supply is relatively stable, we set a recycled account to recycle the money of those dead accounts. An example solution is, if an account has no activity for 10 years, it will be taxed. A small part of money in that account will be recycled and the ratio will exponentially increases by the number of decades (say -1%,-2%,-4%..) and finally will be divided to every account proportionally.

3.4 Application extension

If it's necessary to add a new application to the X-Coin system, we just simply create a new data section aside the balance view data section. All important actions of the new application are vested by the majority of stakes. All modifications on the new data section are synchronized by the Converged Consensus mechanism independently.

4 Security

4.1 Sybil attack

An attack form named "sybil attack" is a common threat to p2p systems. It's one of the main barriers of realizing a truly decentralized Cryptocurrency system. Simply put, a sybil attack is an attack that an attacker tries to isolate a local node by faking tons of sybil nodes and then feeding the local node with fake or malformed data.

4.2 TILP

Now let’s talk about how a new coming node initiates its first balance view safely under the threat of a sybil attack. Suppose people trust and accept the X-Coin system, then the IP addresses of nodes of the X-Coin network will scatter over the whole world. So we propose a strategy of setting "trusted IP location pattern (TILP)" to anti sybil attack. This idea originated from [5].
Every honest client sets a "trusted IP location pattern (TILP)".
TILP examples:
ClientA set "I only trust 10 random IPs located in 10 random different countries (cities)"
ClientB set "I only trust 10 random IPs located in specific 10 countries (cities)"
and so on... TILPs are secret and vary from client to client.

A new coming node tries to contact other nodes with different IPs as many as possible and read hashes of their current balance view and then make statistics of hashes weighting IPs matching TILP with 1 weighting other IPs with 0. You can imagine the probability that more than 50% of IPs matching TILP all happen to be sybil nodes is extremely low. The winner hash should be clean. We acquire balance view with this winner hash from other nodes as our initial balance view. There is a problem that it's possible that communications between local node and the clean world are totally cut off by sybil nodes. If this happens, as a result, you will always fail to reach IPs matching your TILP. That is a strong sign that you are in the trap of a sybil attack. Then you are alerted. You will try to figure things out manually. Obviously the sybil attack must fail.

Another more automatic approach is to randomly set a small trusted IP subset(say 0.01%) of all allocated IPs in the world.

Someone may say TILP is not safe enough, what if an attacker controls a botnet that has a similar IP distribution as X-Coin's? To make sure the initial balance view we got is clean, we may compare it with balance views provided by our friends or by some famous sites. If all different sources of balance views are all consistent, that's a mutual-confirmation, we should trust it. In fact, every time we use X-Coin to buy something, it's a confirmation of balance views between you and the seller. We need to do such mutual-confirmation at most once, only once in our whole lifetime. Cause we deploy another strategy SSS to guarantee we will be always in the clean world of the X-Coin system after the initialization of balance view.

4.3 Secret Security Seeds (SSS)

For a short term updating strategy, since we have gotten a clean initial balance view, we can get a list of accounts with some amount of money randomly selected from initial balance view as our trustable source of hashes of balance views. We update balance view by making weighted statistics of hashes from these accounts. Their balance numbers are the weight of their hashes. The winner hash must be clean. We acquire balance view with this hash from other nodes.

For a long term solution, every time we connect to a node, we try to left a secret security seed on his computer. The seed is the number of his balance encrypted with our public key and then with his public key. This number is the weight of the hash next time we get from this node. Suppose we leave the X-Coin network one year and come back. Our balance view is out of date or is empty. We try to contact as many nodes as possible, and ask them "Had I left a secret security seed on your computer? If yes, please send it and the hash of your current balance view to me". After getting enough SSSes, we make weighted statistics, the winner hash is a clean hash of the current balance view. We acquire balance view with this winner hash from other nodes as our initial balance view. In addition, we can leave a new SSS or update existing SSS on another node's computer. One point should be noticed, for making statistics, the exact number of balance of an account is statistically not that important. Some accounts maybe decrease and some accounts maybe increase. Because our purpose is just to keep sybil nodes out. An attacker can simulate tons of sybil nodes, can control IPs scattered over the whole world, can he deposit those sybil nodes with real money to get sufficient weight to win the hash voting? Definitely not! So accounts that are ever deposited with some amount of money are unlikely to become sybil accounts in future. Other security problems and monopoly problems of PoS are discussed in great detail in [4].

5 Conclusion

We agree that maintaining a trustable financial system is very expensive. But people have already invested many resources including human resources, communication instrument, electricity, normal computing power. Do we really need to waste that much electricity and computing resources to do meaningless hashing? No, we don’t think so. It is feasible to build an environmentally friendly, truely decentralized, trustable, efficient and handy Cryptocurrency system purely based on proof of stake. To achieve this goal we redesign three key elements of a Cryptocurrency system, including fast currency distribution with no mining process, data synchronization realized by the Converged Consensus mechanism and security strategies focusing on defeating a sybil attack. We have a strong feeling that the Cryptocurrency world is advancing to a No Blockchain Pure PoS era!

References

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008
[2] Douceur, John R., The Sybil Attack, *International workshop on Peer-To-Peer Systems*, 2002.
[3] Daniel Larimer, Transactions as Proof-of-Stake, November, 2013
[4] https://en.bitcoin.it/wiki/Proof_of_Stake
[5] https://groups.google.com/forum/#!topic/twister-dev/tH3HlVQ_wmo[26-50-false]
[6] https://ripple.com/wiki/Consensus
[7] http://en.wikipedia.org/wiki/Markov_chain

APPENDIX A

Port the Bitcoin system to a X-Coin system.

Bitcoin is a great invention. I like Bitcoin because it is considered to be open free and decentralized. However, when the first time I learnt about the Bitcoin system has a checkpoint mechanism, I was a little bit surprised and disappointed. Intuition tells me checkpoint mechanism is obviously a violation of the spirit of decentralization. Other defects of the Bitcoin system what I can't accept, such as inefficient mining, bloated history transactions and the requirement of an exchange hub which leads to additional safety dependence which severely compromises the safety of Bitoins. You know the Mt.Gox event. My slogan is "Everyone has a bank at home". So I am trying to eliminate these annoying defects to build a more efficient cleaner and safer Cryptocurrency system.

The client of the X-Coin system and the distribution of currency can be separately implemented. The distribution is just for acquiring a widely acknowledged and trusted initial balance view to be input into the X-Coin system which is a general Cryptocurrency system.

To port the Bitcoin system to a X-Coin system, we plan to publicly make a snapshot of the distribution of all accounts' balances at a specific height of the blockchain of the Bitcoin system in future. Then make a balance view by dividing 1.0 billion of X-Coins to all pubkeys according to the snapshot proportionally. We input this balance view into the X-Coin system as the initial balance view and then start running this new system. Now our coins are doubled in these two systems. The new X-Coin system seems like a fork of the Bitcoin system and only the balance view is adopted. We let these two systems run simultaneously and compete freely, only the winner will survive. You may not agree on the porting program, I understand that, change will always be painful. If by any chance the X-Coin system defeat the original Bitcoin system, the whole mining sector may collapse. Workers of mining or exchange hub must be unhappy. Fortunately, as I emphasized, the Cryptocurrency world is open and free. We publicly fork the balance view of the Bitcoin system by all means. If you don't agree on the porting program, that means you abandon your coins in the new X-Coin system. Of course, you can get them back whenever you change your mind, the premise is that you keep the private key well. In fact, if we manage to implement a reliable client of the X-Coin system and fork the balance view of the Bitcoin system, all users of the original Bitcoin system will accept this new X-Coin system. Because in any circumstance, acceptance gains more than rejection. It's a Nash equilibrium.

Sidechain? Sidedata!

If it's necessary to add a new application to the X-Coin system, we just simply create a new data section aside the balance view data section. All important actions of the new application are vested by the majority of stakes. All modifications on the new data section are synchronized by the Converged Consensus mechanism independently.

Explicit definition of decentralization

When we talk about decentralization, the exact meaning of decentralization is always unclear.
I would try to give an explicit definition of decentralized Cryptocurrency system from the perspective of safety dependence.
Define: A decentralized Cryptocurrency system is a Cryptocurrency system which has only one safety dependence that is the private key.
钥在钱在, 钥失钱亡!
This is an abstract concept and doesn't exist in practice. It's the ultimate goal of designing a Cryptocurrency system. Now we can estimate how decentralized a Cryptocurrency system is by comparing the set of safety dependances with the set {private-key}. In my opinion, every Cryptocurrency system has the obligation to reveal its complete safety dependance set before it

claims it is decentralized. Please check the table below, I believe that a Cryptocurrency system's safety dependence set can't be smaller then the X-coin system's safety dependance set in practice. So the X-coin system is truly decentralized in the sense of practice.

Table 2. Safty dependence comparison

Safety Dependence	Privilege Nodes	Exchange Hub	Checkpoint	51% Attack	Private Key
Bitcoin	No	Yes	Yes	Open to the outer	Yes
X-coin	No	No	No	Closed and suicidal	Yes
Idealized	No	No	No	No	Yes
...	?	?	?	?	Yes

APPENDIX B

Table 3. Contributions of this paper

New Nomenclatures/Concepts/Ideas	Counterpart	Comment
PoSTW / PoSW	PoW, PoS	PoSTW / PoSW is not PoS. Lets make the concept of PoS clear.
Transparent distribution of stakes	Mining, IPO	The sponsor and the audience can't cheat each other. All information are transparent.
Balance view / Basevew	Blockchain	2D balance view instead of 3D blockchain, the inner structure and the consensus mechanism are totally different, no mining, no history transactions.
Balance view sequence number , baseview_sn	Height	Any single balance view carrys complete balances information, while any single block depends on all blocks below it.
Common baseview principle	Main blockchan	Almost all interactions between nodes are based on certain common baseview.
Agent, Vester	Miner	New responsibilities new roles, no mining no miner.
Generalized transaction		Important actions are executed by a generalized transaction, like the violation report transaction.
Transaction package, tx_package, tx_package_51	Block	Tx_packages are independent, any block is linked with its predecessor. The vester items list in a tx_package is elongating, the block is static.
Common baseview agreement based deny reception prevention	Blokchain based	Baseview agreement plus two tx_package_51s.
The plurality stakes mechanism	The longest blockchain	Some rules especially the common baseview principle, expressed by the vester items list

Converged Consensus mechanism and its mathematical proof	The longest blockchain	Ensures the system reaches a consistent distributed consensus.
TILP / TIP strategy		An auxiliary strategy to anti sybil attack
SSS strategy	The longest blockchain	This is the essential strategy to anti sysbil attack
Explicit definition of decentralization	Implicit saying	An abstract concept, a contrast for measuring the degree of decentralization.
Port the Bitcoin system to a X-Coin system	Hard fork	
...	...	...

The Author's Claim:

Every new nomenclature in this paper has its own unique meaning. If you refer to my ideas in this paper and respect my work, please don't change the nomenclatures without a good reason.