Build a Compact Cryptocurrency System Purely Based on PoS

by qianxiaochao

fxxfzx@gmail.com

btc: 18Bf718HZNboFvEqQuhJCTdKJtM8YxHPZW

Abstract

In this paper we propose a new framework of cryptocurrency system. The major parts what we have changed include removing the bloated history transactions from data synchronization, no mining, no blockchain, it's environmentally friendly, no checkpoint, no exchange hub needed, it's purely decentralized and purely based on proof of stake. The logic is very simple and intuitive, 51% stakes talk. A new data synchronization mechanism named "Converged Consensus" is proposed to ensure 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.

Initial distribution of stakes

As the new system is purely based on proof of stake, naturally, we must have a initial statistical distribution of stakes among the users. In this new framework, the initial distribution of stakes and the implement of 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 as money 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 Bitcoin system will start to run under the new framework [APPENDIX I].

Here, we'd like to introduce a new method of fast distribution of stakes within 1 year, by utilizing a high credit individual as initial source of credit, taking Google Company as an example.

Suppose Google plans to issue 10 billion of cryptocurrency named "Compactcoin" (Chinese name 简币). Why Google Company? Actually it could be any celebrity or corporation or even a country with a good reputation and high credit and there are many people who trust them around the whole world. We need an initial source of credit to generate a qualified initial distribution of stakes, just for the initial stage, it will eventually become purely decentralized and independent. Then what should he do?

Google announces the distribution policy:

1. The price decreases exponentially from a very high value to zero (say \$10 - \$0.25, Fig.1) but never reaches zero every day.
2. Anyone who buys Compactcoin during distribution time has the right to sell them back to Google at the same price he bought them after 1 year later since the distribution is expired. However after the distribution, people buy Compactcoin at their own risk.
3. 3. The distribution will expire within 1 year any time at which Google thinks buyers are enough and the price is proper. Then Google buys all the left volume (say 20%, 10%, anyway, much fairer than some creators possess 90% of stakes, right?) at the lowest price. Google keeps the process of distribution secret.

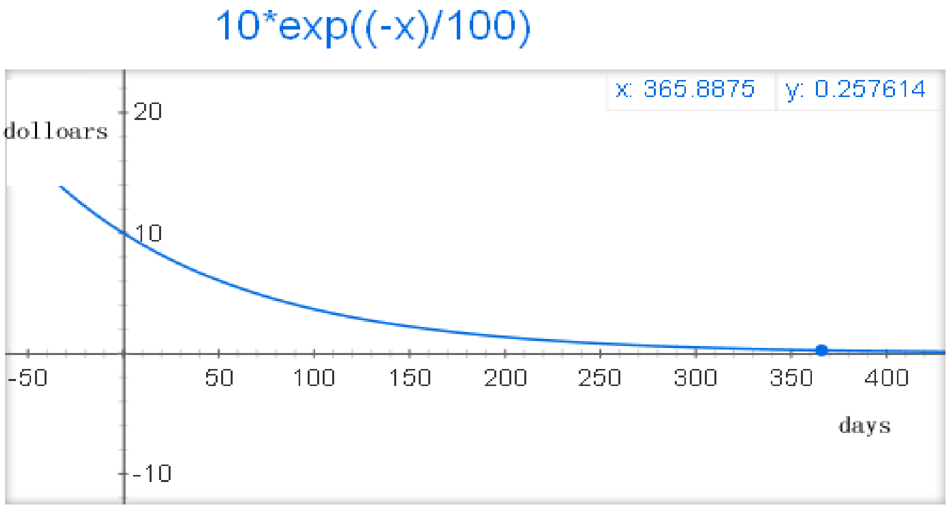

Fig.1. An example pricing strategy F(x) = 10*exp((-x)/100).

Let me explain you what these rules mean.

1. Why not set a fixed price?
   First, it's difficult to determine what is a proper price. Second, it can prevent someone with big money from buying most of the total volume. If so, the currency is dead due to lack of liquidity. Our target is to distribute currency to the potential audience as broad as possible and as many as possible. This rule can distinguish buyers to different layers by their different characters and personalities (e.g. wealth, risk appetite, trust degree, knowledge about cryptocurrency, intelligence, education level...). Buyers with different characters will get into the distribution process at different price intervals respectively. For example, buyers who are wealthy and risk hungry and know little about cryptocurrency may enter at relatively high price. Buyers who are poor and risk averse and know a lot about cryptocurrency may enter at a relatively low price. Third, for discovering a proper price to start trading at after the distribution. Suppose the statistical distribution of price-number_of_buyers is normal distribution, e is the expectation price. Since people buy Compactcoin after the currency distribution is expired at their own risk, we suggest e/2 to be the initial price. Then what if people won't sell any Compactcoin because they expect the price will go up? Obviously, if so, the currency is dead due to no liquidity. All participators get negative profit, and therefore buyers who got coins at a relatively low price, especially Google who bought large volume of coins at the lowest price will sell a part of coins to activate the market.
2. The second rule is to reduce the risk for a buyer to get into the distribution to stimulate more people to participate in the currency distribution process.
3. Will Google force the distribution to expire early so that he can possess most of the currency?No, he won't. If he does this, he can't discover a proper price to start trading at, and it's harmful to the Compactcoin system because of lack of liquidity and the Compactcoin system will be vulnerable to single point (Google) failure. Then people won't trust the Compactcoin system anymore, the coins will become worthless. Don't forget we made an assumption that Google is a famous company with high credit. Apparently, the later people get into the distribution the lower price they get. However, only Google knows the exact time when the distribution will be expired. So if too late, they may miss the opportunity of buying Compactcoin with nearly zero risk. This is a psychology game. People speculate when is probably the expired time and have to judge the proper time when to get in, according to their different characters.

OK, the distribution of Compactcoin is now done. Then Google publishes all the transactions during the distribution and hardcodes the initial distribution of stakes into the source of Compactcoin system, and of course, open source by convention. Now people start trading Compactcoin at the price e/2. At this point, the state of the Compactcoin system is similar to the state of Bitcoin system when all Bitcoins are mined off by the year 2140. There is no mining process in the Compactcoin system.

Data synchronization

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 more and more hard 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 Compactcoin 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.

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 Compactcoin 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 Google 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:
[address, balance, time last modified]

We set a special balance record for recycled money in Compactcoin system:
[special address, balance, time last modified]

whole system's balance view:

```
[
sn,    //sequence number of a balance view increasing continuously from 0.
[record1, record2, record3... recycled money record]    //every user's balance record
hash    //hash of this balance view
]
```

Baseview := A balance view based on which a transaction is generated or current balance view of a node. Base view is a core concept in data synchronization. Almost all interactions between nodes are based on certain common base view.

Transaction processing
Transaction agent
For processing transactions more efficiently, every transaction is executed by a transaction agent. A transaction agent is a node that collect transactions that match his criteria setting from network into a transactions package(tx-package). Then broadcast the tx-package to network to be verified and to be vested with other nodes' balance numbers. Tx-package has a request of minimum total volume

and a upper limit of size.

```
transaction:
[
base view sn,         //based on which this transaction was generated.
base view hash,
sender address,
receiver address,
volume,
tx-fee,           //max tx-fee the sender will pay for this transaction
timestamp
]
```

Applying for account

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

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 sets a minimum of balance of qualified agent to mak sure the agents are all serious and constrain the number of qualified agents and the number of tx-packages generated by the agents.

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

```
vester item         //vester means someone who vests in tx-package
[
base view sn,        //vester's base view sequence number
base view hash,
vester address,
balance of vester,
timestamp
]
```

```
vester items list
[
base view sn,        //common base view shared by all items in this items list
base view hash,
[item1, item2, item3... ]
]
```

```
transactions package
[
base view sn,        //common base view in this tx-package
```

base view hash,
agent address,
transactions list,
vester items list,
51-flag //indicate I have gotten 5.1 billion vesting
timestamp
]

Vesting

After receiving and verifying a tx-package, every vester node will vest in this tx-package with his balance number. 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 vester items list in a tx-package is bigger than 5.1 billion(51% of total volume, "51-tx-package" for short). That indicates transactions in this 51-tx-package are all accepted by the whole network. Then this vester node will iterate every transaction in this 51-tx-package to calculate a new balance view by simulating the execution of all transactions in this 51-tx-package and then divide tx-fees to the agent and all vesters' accounts according to a defined formula and update his baseview and set the 51-flag of this tx-package to true. Finally, broadcast this 51-tx-package to announce that a new balance view is accepted by the whole network. Every node will calculate the new baseview after receiving this 51-tx-package.

Rules for data synchronization :

1. An agent only collects transactions which share the same baseview with him and match his criteria setting. (Same base view 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 prevent 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 tx-package gets more than 5.1 billion of vesting indicates it's accepted by the whole network.
4. A vester vests in at most one tx-package per baseview_sn exclusively. (Multiple 51-tx-packages is possible. This rule ensure 51-tx-packages are all from one agent per baseview_sn.)
5. Every node only accepts one 51-tx-pacakge which has the same baseview with him exclusively.

Agents and vesters have incentive to break the rule 2, 4. So we set a rule to punish the users who break the rule 2, 4. Every node randomly buffers a small ratio(say 1%) of history tx-packages with current baseview_sn when it went through and checks if there is someone signed more than one tx-package. In the Compactcoin 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 address,
accused address,
proofs,
time to be released,
timestamp
]

Every node will broadcast a violation report transaction when he finds a violation event. Every agent will collect this special transaction unconditionally. The last vester will verify proofs and set the “time last modified” field of accused address to the “time to be released” field of this violation report transaction. The accused address is locked until the time it is released.

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 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.

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 51-tx-package is found by different vesters. Then the consensus of 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) baseview 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 baseview with no change. (It's supposed that the current balance view is stable and consistent if there are N times updating baseview with no change)
8. When updating balance view, the baseview_sn of the winner baseview has to be bigger or equal to 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 penultimate baseview, by receiving a 51-tx-package or by updating baseview, 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.

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 their balance view data
    make weighted statistics
    get a winner 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
The updating operation will be performed node by node randomly. So if s $\in$S1 then ether s has a recurrent edge with the value 1, or s has a outgoing edge with the value 1 and its direct successor 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 $\in$Ω and γ is any possible path of n steps transitions start from s_0 }
Define Γ1 := { γ | γ $\in$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 $\in$E
Define m := The max probability value of all edges in the markov chain C and m != 1
The probability of path γ is $p(\gamma) = \prod_{i=1}^{i=n} v(e_i), \quad e_i \in \gamma, \gamma \in \Gamma$

$$
\begin{aligned}
\because\ & p\{s_n \in S1\} \\
= \ & 1 - p\{s_n \in S0\} \\
= \ & 1 - \sum_{i=1}^{i=k} p(\gamma_i), \qquad \gamma_i \in \Gamma 0 \\
= \ & 1 - \sum_{i=1}^{i=k} \prod_{j=1}^{j=n} v(e_{ij}), \qquad e_{ij} \in \gamma_i
\end{aligned}
$$

>= $1-\sum_{i=1}^{i=k}\prod_{j=1}^{j=n} m$

= $1-k*m^n$

∴ $1 \ge p\{s_n \in S1\} \ge 1-k*m^n$

∵ $\lim_{n\to\infty}(1-k*m^n)$ = 1 - 0 = 1

∴ $\lim_{n\to\infty} p\{s_n \in S1\}=1$

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.

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 updating baseview,

M := B*B.

Define N := If there are N times continuously updating baseview with no change then the consensus of 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
1-9	1	**1**	**1**
10-99	2	**4**	**4**
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.

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 51-tx-package he may choose to save tx-package to disk as memo or as proof. If you trust your receiver you just simply send money to him, and then keep tx-packages as memo. If you don't trust your receiver, you synchronize baseview with receiver by getting a common baseview agreement with the receiver's signature before you send money to him. You keep every history tx-package after you send money until your transaction is confirmed. You also can acquire tx-packages from other nodes, especially the agent of this transaction. He has the obligation to keep tx-packages for you. With common baseview agreement and history tx-packages until your transaction is confirmed, the receiver has no way to deny reception.

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 Compactcoin system has the same number of accounts as the Bitcoin system. A balance record needs about 20+8+8=36 bytes. Then the data size of balance view is about 36m 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 Compactcoin system. Generally speaking, the agents tend to contact vester nodes with large balance first and the vesters tend to merge. Like mining pools in Bitcoin system, stake pools may emerge. So the length of vester items list should not be too long. The average computing load of verifying one transaction is at most 10 times of hashing. The generation rate of transaction of Bitcoin system is about 0.8 tx/s (data from blockchain.info,May 2014), so the computing load is about 10 * 0.8 = 8 hashes/s . The space complexity is O(c), c = 36m. For a ordinary home PC produced nowadays, such scale of client load is almost negligible, 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".

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 finally will be divided to every account proportionally.

Security

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 purely decentralized cryptocurrency system. Simply put, a sybil attack is an attack form 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.

TILP & SSS

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 Compactcoin system, then the IP addresses of nodes of the Compactcoin 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 connect to other nodes with different IPs as many as possible and read hashes of their baseview and then make statistics of hashes weighting IPs matching TILP with 1 weighting other IPs with 0. You can imagine the probability of 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.

Someone that has extremely high security level requirement may say TILP is not safe enough, what if an attacker controls a botnet that has a similar IP distribution as Compactcoin'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. 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 Compactcoin system after the initialization of balance view.

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 Compactcoin network one year and come back. Our balance view is out of date or is empty. We try to connect to 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 baseview to me". After getting enough SSSs, 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. 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].

Conclusion

We agree that maintaining a reliable 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, purely 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.

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] http://en.wikipedia.org/wiki/Markov_chain

APPENDIX I

Port the Bitcoin system to a Compactcoin system.

Bitcoin is a great invention. I like Bitcoin because it is considered to be open free and decentralized. When the first time I learnt about the Bitcoin system has a checkpoint mechanism, I was a little bit surprised and disappointed. To my intuition, checkpoint mechanism is obviously a violation of the spirit of decentralization. Other defects of 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 Compactcoin 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 Compactcoin system which is a general cryptocurrency system.

To port the Bitcoin system to a Compactcoin system, we plan to publicly make a snapshot of the distribution of all accounts' balances at a specific height of blockchain of Bitcoin system in future. Then make a balance view by dividing 1.0 billion of Compactcoins to all addresses according to the snapshot proportionally. We input this balance view into the Compactcoin system as the initial balance view and then start running this new system. Now our coins are doubled in these two systems. The new Compactcoin 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 Compactcoin 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 Bitcoin system by all means. If you don't agree on the porting program, that means you abandon your coins in the new Compactcoin 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 Compactcoin system and fork the balance view of Bitcoin system, all users of original Bitcoin system will accept this new Compactcoin 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 Compactcoin 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.