Matcoin: A high-efficiency Peer-to-Peer digital currency system based on blockchain matrix

Zhichao Duan

Abstract

In blockchain systems, the number of transactions per second is limited by the fact that there exists only one chain. Therefore the transaction efficiency is diminished to a relatively low level. In this article, we propose blockchain matrix, a new data model based on blockchain, which elevates transaction efficiency by utilizing parallel multi-chains.

The new method of parallel multi-chains requires more than the current protocols for blockchains. Hence we create a new protocol, which enables us to construct a high-efficiency Peer-to-Peer digital currency system, that, in theory, is capable of supporting unlimited amount of transactions per second.

Our approach to the new protocol is to first define equiprobable surjections between mining addresses and blockchains in a blockchain matrix by setting up constituencies of mining addresses. We then view each blockchain in a blockchain matrix as a transaction channel, and establish equiprobable surjections between transactions in the system and transaction channels. We then redefine the criteria for a valid transaction and those for effective selections of blockchain matrices. Finally, we end up with the consensus that constitutes our protocol.

The protocol defined in this article can be used by most of the commonly used consensus algorithms nowadays, such as POW, DPOS etc, after update. Moreover, the protocol proposed in this article is robust against majority attack. A majority attack only affects the system when it controls most of the nodes for a relatively long period of time, which means almost all nodes in the system have already accepted the new rule.

The structure of the article is as follows. In Section 1 to 5, we define the protocol for a blockchain matrix system. In Section 6, we give an example of a blockchain matrix system under the blockchain matrix system protocol.

1 Blockchain matrix

Let $A = \{a_{i,j}\}$ be an $m \times n$ matrix, where $m, n \in \mathbb{Z}_+, 1 \leq i \leq m, 1 \leq j \leq n$, and $a_{i,j}$ is a blockchain for any i, j. We call such a matrix A a **blockchain matrix**.

For any $a_{i,j} \in A$, let $L(a_{i,j})$ denote the **height** of the blockchain $a_{i,j}$. It is clear that $L(a_{i,j}) \in \mathbb{Z}$, and $L(a_{i,j})$ increases over time.

For any $a_{i,j} \in A$ and any integer $0 \leq k \leq L(a_{i,j})$, let $B(i,j,k)$ denote the block in $a_{i,j}$ of height k.

Via complex trans-chain transactions, we ensure that the digital currencies generated by each blockchain are identical.

2 Constituency

2.1 Mining address

In a blockchain system, members of the network are called nodes. Every node is competing for the right of logging, which we call the **competition**. Winning the competition means getting transaction fee and coinbase reward for each of the discovered block. The rewards are sent to an address given by miners, which we call **mining address**.

2.2 Constituency

In blockchain systems represented by Bitcoin, mining addresses are independent of the mining process, in the sense that miners can associate an account to the blocks he or she discovered at any time. On the contrary, in blockchain matrix systems, mining addresses are linked to the mining process. In other words, if a miner cannot provide valid mining addresses after the discovery of blocks, other nodes are still able to deny the validity of the mining process. Miners can avoid such inconvenience by preparing valid addresses in advance, as arbitrary number of addresses can be attached to a node.

A blockchain matrix A has $m \times n$ blockchains, and each one of them has their online nodes competing for the right of logging, which we can view as the competition of the right of logging of all online mining addresses. To maintain the order of this competition, we have to distribute the mining addresses to the blockchains represented by entries of A equiprobably. In other words, we provide the mining addresses with a piece of information called constituency, each one of which corresponds to a unique blockchain in A.

The definition of constituency is as follows. Let M be the set of all mining addresses in a blockchain matrix system, and let A' be the set of all elements in A. Clearly A' has $m \times n$ elements.

Consider a map $f : M \to A'$. For any $a_{i,j} \in A'$, we define the set of **pre-image** of $a_{i,j}$ as $f^{-1}(a_{i,j}) := \{addr \in M \mid f(addr) = a_{i,j}\}$. We will call $f : M \to A'$ an **equiprobable surjection** if f is a surjection and the probability of mapping any address $addr \in M$ to any $a \in A'$ are about the same. If f is an equiprobable surjection, we will call $f^{-1}(a_{i,j})$, the set of pre-image of $a_{i,j} \in A'$, the **constituency** of $a_{i,j}$ under f.

By equiprobable surjection, we actually mean equiprobable to a certain extent. Although the more equiprobable the function is, the better it fits our system, we do not require the function to be strictly equiprobable. As long as the probability of the function mapping the input to one value is not wildly

different from the probability of f mapping it to another value, the function is acceptable.

Different systems may have different definitions on addresses, which implies that different equiprobable surjections are needed in order to generate constituencies. If, for a given system, a scheme provides an equiprobable surjection so that the constituencies are reasonably generated, then we call such a scheme in compliance with the blockchain matrix protocol. We will give an example of such an f in Section 6.

2.3 Summary

In blockchain matrix systems, the set of intersection between any two constituencies is empty, hence the competitions can be viewed as independent processes.

3 Transaction

3.1 Basis for transaction

A transaction in a blockchain matrix system, denoted by Tx, is a transaction in the Bitcoin system together with the information presented in Sections 3.2 to 3.4. Here we start by familiarizing the readers with the definition of a transaction in the Bitcoin system.

In a Bitcoin system, a transaction consists of input and output, together with the data encoding the information of the transfer of Bitcoin value from the initial point (input) to the target (output). The basic unit of a transaction is an unspent transaction output, denoted by $UTXO$. Every transaction has a unique hash value associated to it, denoted by $txid$. In principle, an $UTXO$ corresponds to a unique Tx, but a Tx can be associated to many $UTXO$, while a Tx can only be attached to a unique $txid$.

3.2 Transaction channel

Let A be a blockchain matrix. We call the blockchain represented by $a_{i,j}$ in A the $(i,j)-$th **transaction channel**. Clearly there exist $m \times n$ transaction channels in a blockchain matrix.

Similar to the case of constituencies, we are looking for an equiprobable surjection g from the set of all transactions to the set of all transaction channels. In Section 6, we show the existence of such a g.

When a transaction Tx happens, its transaction channel is determined by g. As a consequence, all the nodes on the network know the transaction channel of Tx. The constituency corresponding to the transaction channel will accept the transaction, and then pack it in the blocks of the transaction channel.

When there are a large number of transactions taking place simultaneously in the system, g can distribute these transactions relatively uniformly onto transaction channels. As a result, the blockchain matrix model is able to increase

the efficiency of transactions by approximately $m \times n$ times, since the process of block generation on each transaction channel occur independently.

3.3 100% verified transaction

In a Bitcoin system, it is generally accepted that a transaction is verified if we add 5 blocks to the tail of the block on which the transaction is recorded. We call the above procedure a 6 block verification. The commonly used Bitcoin wallet is using 3 block verifications, while the DPOS algorithm is using 15 block verification.

However, in a blockchain transaction system, there is no 100% verified transaction. All we can do is to increase the number of verifications so that the verification rate gets closer and closer to 100%. In other words, there are still chances, despite of being extremely small, that a transaction completely fails due to attacks, even though all the users participating the system view the transaction as successful. The existence of such possibilities is not tolerated by industrial applications.

We now propose a protocol under which transactions are 100% verified. We consider a blockchain matrix A with blockchains $a_{i,j}$ and a given consensus algorithm P (in this article, all the consensus algorithms are referred to those who can run stably in blockchain systems, for example, the POW for Bitcoin and the DPOS for BTS). For any i, j, there exists a smallest positive integer c such that all the transactions recorded in all the blocks of $a_{i,j}$ of height less than or equal to $L(a_{i,j}) - c$ are commonly verified transactions. We will call these transactions as 100% **verified transactions** (in the rest of the article, we will not distinguish the concept of verified and 100% verified transactions).

We call the set $S(a_{i,j}) := \{B(i,j,k) | 0 \leq k \leq L(a_{i,j}) - c\}$ a **verified subchain**, and denote it pictorially as in Figure 1.

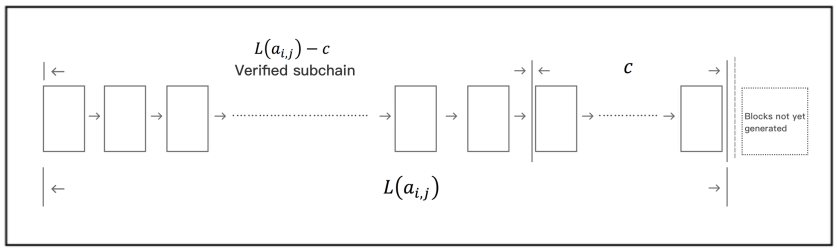

Figure 1: A verified subchain

Clearly, in a blockchain matrix, all $a_{i,j}$ has the same value of c, as for given A, the algorithm P is fixed.

3.4 Types of transactions

Based on our setup, transactions are related to each other. The input of every transaction is a set of $UTXO$, which enables us to trace a transaction to its

origin. Let Tx be a transaction. Denote the set of all $UTXO$ in the input of Tx by UT. Replacing every $UTXO' \in UT$ by its corresponding transaction Tx', we get a set of transactions T. By definition, the number of elements of the set T is less than or equal to that in UT. We call T the set of **relevant transactions** of Tx. For any $Tx' \in T$, we say Tx is relevant to Tx', and we denote the relation by $Tx \gg Tx'$.

3.4.1 Con-chain transaction

Consider a transaction Tx and its relevant transaction set T. If for any $Tx' \in T$, the transaction channel of Tx' is the same as that of Tx, we call Tx a **con-chain transaction**. If Tx is a con-chain transaction, we call Tx a valid con-chain transaction.

3.4.2 Trans-chain transaction

Let Tx and T be as in the last subsection. If there exists some $Tx' \in T$ such that the transaction channel of Tx' is different from that of Tx, we call Tx a **trans-chain transaction**.

For a trans-chain transaction Tx, if there exists some $Tx' \in T$ such that Tx' is not a verified transaction, we call it an **invalid transaction**, otherwise, we call it a valid trans-chain transaction, as shown in Figure 2.

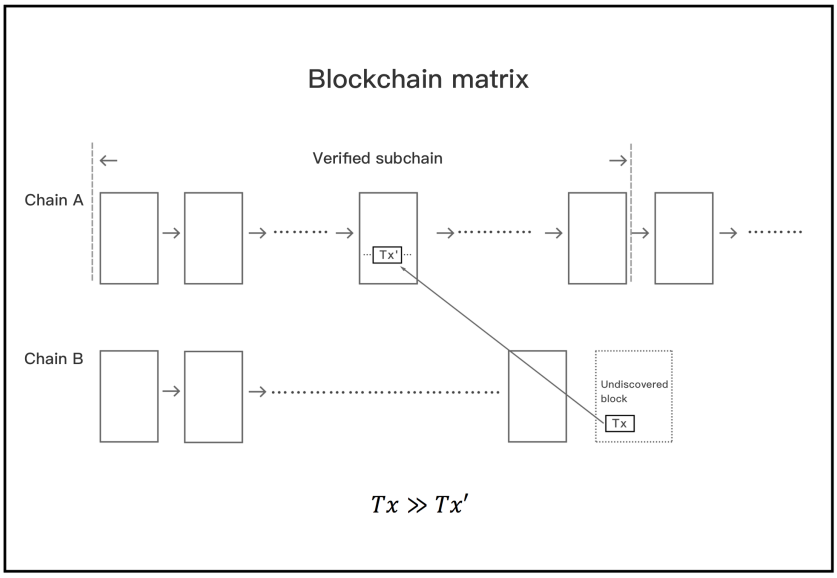

Figure 2: A valid trans-chain transaction

Valid con- and trans-chain transactions are called **valid transactions**.

4 Block identification and submatrix

4.1 Block identification

In blockchain systems, the header of a block only has the hash value of the previous block, which means that the block does not know its height. In order to identify the position of a single block, we need the information of the blockchain containing the target block and the height of the block in the blockchain. We will insert the above information into the header of a block.

4.2 Submatrix and verified submatrix

Let A be a block chain matrix. For a blockchain $a_{i,j}$, a subchain of height between l_1 and l_2 is denoted by $subchain(l_1, l_2)_{i,j}$. where $l_1, l_2 \in \mathbb{Z}_+$. If $l_1 = 0$, we call the subchain a **original subchain**.

Suppose A is of size $m \times n$. If there exists an $m \times n$ blockchain matrix $G = \{b_{i,j}\}$ such that for any $1 \leq i \leq m$, and any $1 \leq j \leq n$, $b_{i,j}$ is a original subchain of $a_{i,j}$, then we call G a **submatrix** of A. Moreover, if every transaction in G is 100% verified transaction in A, then we call G a **verified submatrix** of A.

If G_1, G_2 are two submatrices of A, and if for any $b_{i,j} \in G_1$, and any $b'_{i,j} \in G_2$, we have $L(b_{i,j}) \leq L(b'_{i,j})$, then we call G_2 is higher than G_1. If G is a submatrix of A, and there is no other submatrix of A higher than G, then G is called the highest submatrix of A. Clearly, A is the highest submatrix of itself.

If G is a verified submatrix of A, and there is no other verified submatrix of A that is higher than G, we call G the highest verified submatrix.

5 Decentralized consensus of blockchain matrix

Based on Sections 1 to 4 in this article, we give a protocol for blockchain matrix. Blockchain matrix's decentralized consensus emerges from the interplay of four processes that occur independently on nodes across the network:

(1) Independent verification of each transaction, by every full node.

(2) Independent generation of new blocks by mining nodes.

(3) Independent verification of the new blocks by every node and assembly into a chain.

(4) Independent selection of the most valid blockchain matrix, by every node.

5.1 Independent verification of transactions

Each transaction occurring at a node is later sent to nodes nearby so that the transaction propagate throughout the whole blockchain matrix network.

However, every blockchain matrix node receiving a transactions verifies the transaction before it is sent to nearby nodes, which ensures that only valid transactions are propagating in the network, while invalid transactions are discarded at the first node.

A consensus algorithm P in a blockchain matrix is clearly capable of performing independent verification on transactions. Based on the verification by the algorithm P, the following rule is imposed on a blockchain matrix system: **every transaction has to be a valid transaction**. In other words, a transaction is either a con-chain transaction or a valid trans-chain transaction.

Once a node setup its mining address, it collects, in certain reasonable order, the transactions in the transaction channel corresponding to the constituency, with which the mining address is provided. Transactions such collected are placed into pools, and each transaction channel corresponds to an individual pool.

5.2 Construction of new blocks

A consensus algorithm P used by a blockchain matrix must be capable of generating new blocks independently on each blockchain in the blockchain matrix. For each blockchain $a_{i,j}$ in the blockchain matrix, only the live mining nodes which have mining addresses belong to the constituency, which corresponds to the blockchain $a_{i,j}$,can compete the right of generating new block on this blockchain according the algorithm P. Upon the generation of a block, the winning miner will pack transactions belong to the transaction channel and the mining addresses into the block.

After the construction of a block, the node will send the block to all the neighboring nodes, which will continue spreading the block after verifying the block.

The construction of blocks in every transaction channel, if we don't take the transaction inside the block into consideration, is completely independent. It only depends on the algorithm P.

5.3 Verification of new blocks

When a node receives a new block, it will, according to the identification of the block, trace back to the blockchain $a_{i,j}$ which the block corresponds to. If the block passes the verification, it will be appended to $a_{i,j}$, otherwise, it will be rejected.

The verification of a new block contains the following parts:

- Verify if the block conforms to the standards provided by the consensus algorithm P;
- verify if the mining address of the block belongs to the correct constituency, if not, the block will be discarded;
- if the blockchain matrix before accepting the new block is A, verify that all transactions in the new block are valid transactions in A, in the sense of the validity provided in Section 3. In other words, the transactions have to be either con-chain transaction or valid trans-chain transaction;

- if the above criterion is not met, the block will be placed in the pool of isolated blocks until all the above criteria are met, or until it is swept from the pool of isolated blocks.

Due to the fact that there are parallel multi-chains in the receiving process, we need to prove the following statement: if the current blockchain matrix is A, there do not exist two new blocks $B(i,j,k)$ and $B(p,q,t)$ such that whether $B(i,j,k)$ can be inserted to A depends on whether $B(p,q,t)$ can be inserted to A first and vice versa. In other words, there does not exist deadlock.

Suppose deadlock described above does exist, then it is clear that the blockchain $a_{i,j}$ corresponding to $B(i,j,k)$ and the blockchain $a_{p,q}$ corresponding to $B(p,q,t)$ are different. Then there exists $t' < t$ and a block $B(p,q,t')$ in $a_{p,q}$ such that $B(p,q,t')$ is not a block in the verified subchain $S(a_{p,q})$ of $a_{p,q}$. After $B(p,q,t)$ is added to A, $B(p,q,t')$ becomes a block in $S(a_{p,q})$. In addition, there exists a transaction Tx'_1 recorded in $B(p,q,t')$ and a transaction Tx recorded in $B(i,j,k)$ such that $Tx \gg Tx'_1$.

If both blocks are given by honest nodes, then there exists a node M_1 on which $B(p,q,t)$ is generated before $B(i,j,k)$. However, by assumption, whether $B(p,q,t)$ can be added to A is dependent on whether $B(i,j,k)$ can be added to A first, contradiction.

From the argument above, we know that there does not exist deadlock. Similarly we can show that there does not exist cyclic deadlock (such as B_1 dependent on B_2, B_2 dependent on B_3, and B_3 dependent on B_1 etc).

5.4 The choice of blockchain matrix

Initially, a new node in the network has only $m \times n$ blocks, in other words there is only one block on each blockchain.

In a blockchain matrix, local node will acquire $m \times n$ blockchains in order. Acquisition of blockchains has to be in compliance with the following rules:

- Every local node has a highest verified submatrix F. For a new node, the highest maximal submatrix is a blockchain matrix such that each of its blockchain has height 0;
- every node (i,j) in a blockchain matrix $A = \{a_{i,j}\}$ will choose a longest blockchain according to the consensus algorithm P and satisfy the following criterion;
- if we replace the previous blockchain $a_{i,j}$ by the chosen longest blockchain, then the highest verified submatrix of the new blockchain matrix, denoted by F', has to be a submatrix of F (defined above), otherwise the blockchain will not be accepted.

5.5 Proof and analysis

In blockchain systems, the problem of blockchain fork is resolved by the consensus algorithm P. The precise statement and proof is given as follows.

Given a consensus algorithm P in a blockchain system, if P is valid, then after updating the system to a blockchain matrix system, there cannot be a fork of the highest verified submatrix. Here, we call P valid if a transaction is valid forever after it is recorded and it is viewed as verifiable under P.

In fact, if a consensus transaction under P can be successfully attacked then the decentralized digital currency system is unusable.

If we assume that there exists fork. Then there exist at least two different forks of a highest verified submatrix. In other words, there exists a blockchain in the blockchain matrix such that it has at least two blocks at the same height. Every transaction in the two blocks are verified and there is at least one transaction recorded in the two blocks that differently. This is the same as to say that there exists a transaction that is successfully attacked under P, contradiction.

In fact, it is easy to see that after the blockchain matrix system operate normally for a period of time, blockchains on an honest node will be immune to attacks since they will in some sense acquire the ability of identifying attacks. Attackers are not able to achieve their goals to make the honest nodes accept their blockchains in a short period of time. All they can do is to maintain a majority (51%) of the total net computing power for a long period of time. This is much more difficult than majority attack. From an alternative point of view, the success of such attack means that the majority of the nodes accepted the new blockchain matrix of the attacker.

The purpose of con-chain transactions in this paper is that it will satisfy users' demand of super fast transfer in small amount. We can think of the con-chain transaction as a way to transfer money without verification, while trans-chain transactions requires the verification of the others. This means it is possible for wallets to predict potential con-chain transactions, which makes super fast transfer in small amount possible.

6 Matcoin: an introduction

Here we give a simple example to show that the equiprobable surjections in Sections 2.2 and 3.2 exist. The example surjections may not be equiprobable in a strict sense, but it is enough for our system. We will provide more discussions on equiprobable surjections in subsequent articles as it is not the main topic of this paper.

In the following example, we will directly use the methods in Bitcoin systems to generate the *txid*s of transactions and to generate mining addresses.

Our choice of blockchain matrix has size $m = 1024$, $n = 1$.

6.1 A equiprobable surjection from mining addresses to constituencies

We define a map f from mining addresses to constituencies as follows. For any mining address *addr*, we first convert it to ASCII code, then we convert

the binary number into decimal number. Finally we take the congruence class modulo $m \times n$ of the decimal number.

We take, as a sample set, mining addresses emerged in the Bitcoin system. There are 56888 mining addresses in total, and they are taken from the 8 blocks in the Bitcoin system of height 508176 to 508183.

According to our definition of f, we get the distribution of the constituencies as shown in Figure 3.

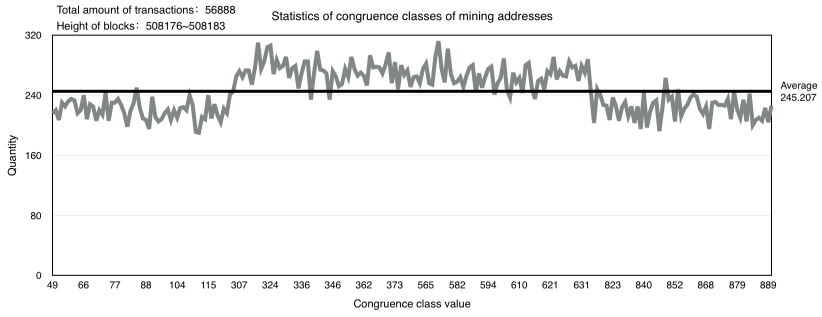

Figure 3: Modulus function statistics

6.2 A equiprobable surjection from transactions to transaction channels

We define a map g from the set of all transactions to the set of all transaction channels as follows. For any transaction Tx, convert its unique $txid$ to decimal number, and then take its congruence class modulo $m \times n$. It is clear that this map is a surjection. Note that we obtain $txid$ using the method in the Bitcoin system.

We take, as a sample set, 50178 occurred transactions from 50 blocks (of height 508176 to 508225) in the Bitcoin system, and we take the $txid$ of them.

According to the definition of g, we obtained the following distribution of results as shown in Figure 4.

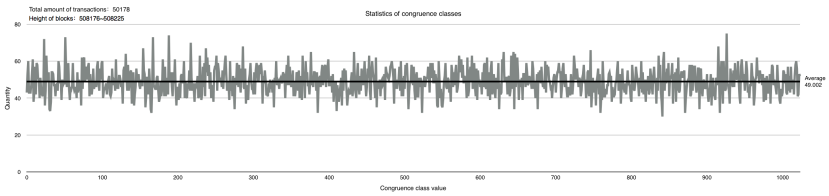

Figure 4: Transactions to transaction channels

6.3 Further discussion

The major feature of the current blockchain systems is that the total amount of digital currencies they generate are fixed. This is against the principles of economics: in real life, fixing the total amount of currencies will result in recessions. It is henceforth healthy from an economics point of view to ensure the existence of inflation rate. Mining rewards are also fundamental and inseparable for every transaction, otherwise the system will be short of users to participate in mining. When a block is discovered, the first reward given should be decided by a economics model based on the current total amount of currency.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electric Cash System*.
`https://bitcoin.org/bitcoin.pdf`, 2008.

[2] Andreas M Antonopoulos, *Mastering Bitcoin*.
`https://github.com/bitcoinbook/bitcoinbook/blob/develop/book.asciidoc`, 2017.

[3] Daniel Larimer, *DPOS Consensus Algorithm - The Missing White Paper*.
`https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper`, 2017.

[4] *Delegated Proof-of-Stake Consensus*.
`https://bitshares.org/technology/delegated-proof-of-stake-consensus`.

[5] Vitalik Buterin, *A Next-generation Smart Contract and Decentralized Application platform*.
`https://github.com/ethereum/wiki/wiki/White-Paper`, 2013.

[6] Vlad Zamfir, *Introducing Casper "the Friendly Ghost"*.
`https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`, 2015.