Dynamic Difficulty Scheme in Bitcoin System: A Distributed and Balanced Consnsus Network

Chi-Ning Chou[1], I-Ping Tu[2], Shih-Wei Liao[3], and Che-Jui, Chang[4]

[1]CSIE, National Taiwan University
[2]Institute of Statistical Science, Academia Sinica
[3]CSIE, National Taiwan University
[4]Department of Blockchain Research, DiQi Inc.

April 11, 2016

Abstract

Bitcoin is the first secure decentralized electronic currency system. However, it is known to be inefficient and has the hazard of double spending. In this paper, we aimed to improve the intrinsic overhead of Bitcoin construction and decrease the probability of double spending. To fulfill our goal, we first formalized a Bitcoin-based decentralized secure network model in order to present a quantitative analysis. Next, to resolve the risk of double spending, we proposed a mechanism to dynamically modify the difficulty of each participants in the sense that those who wins more recently has less probability to win next time. To analyze the performance of the dynamic difficulty mechanism, we observed that the dynamics of the system enjoys a high-order Markov property, which can be modeled with a high-dimensional transition matrix. Finally, using the high-order Markov chain model, we showed that the dynamic difficulty mechanism effectively decreases the probability of double spending and results in a more efficient Bitcoin-based system without sacrificing the security.

Keywords: Bitcoin, Dynamic difficulty, High-order Markov chain, State reduction

Contents

1 Introduction

In 2008, Satoshi [Nak08] published an outbreaking decentralized cryptosystem: Bitcoin, which opened the door for fully distributed secure network. Seven years have passed, Bitcoin is indeed very successful in electronic currency, however, there are very few works adopt the brilliant idea and extend it into other related application. We believed that there are two main reasons for this phenomenon: the intrinsic overhead of the Bitcoin construction and the hazard of double spending.

The intrinsic overhead of Bitcoin lies in two parts of the system: the *information propagation* and the *proof of work mechanism*. The former is strongly related to the synchronization and consensus issue. Karame et al. in [KAC12] initiated the study of fast payment in order to accelerate the transaction confirmation in Bitcoin and found out that the double spending probability is non-negligible. Bamert et al. in [BDE+13] proposed *securing fast payments*, which improved the previous studies. They showed that under their construction the double spending probability diminishes to less than 0.088%. Moreover, they also presented a real application in vending machine. Stathakopoulou et al. in [Sta15] proposed a faster Bitcoin network based on pipeline. They showed that increasing the locality of connectivity among each node can accelerate the information propagation. By implementing a *Content Distribution Network (CDN)*, they achieved 60.6% average speed up. From these line of work, we can see that resolving the intrinsic overhead by accelerating the information propagation has limited performance.

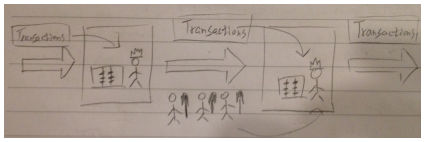
As a result, in this work, we tried to fix the intrinsic overhead of Bitcoin network by proposing a new proof of work mechanism. The central idea is to strengthen the power of proof-of-work so that it can guarantee the stronger security while in the meantime being more efficient.

In this section, we will introduce some background knowledge about Bitcoin and our mathematical model high-order Markov chain. The formal model will be constructed in Section 2. And we will analyze the rate of consecutive winning of the system we proposed in Section 3. In Section 4 we will present a reduction scheme to simplify the analysis. Finally, we will discuss the performance and compared with the current Bitcoin system in Section 5.

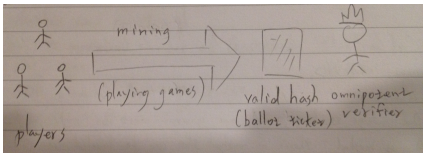
1.1 Bitcoin

To get more basic concepts about Bitcoin, we recommend the following works. [Nak08] is the original Bitcoin paper published by Satoshi and might be more technical. [Ros11] and [KDF13] presented Bitcoin in a economist point of view. On the other hands, [KAC12] and [Ros14] had a complete discussion about the double spending issue in Bitcoin. Here, we will simplify Bitcoin into a mathematical model for the convenience of analysis.

Bitcoin is a decentralized blockchain-based voting system. The goal of the system is to guarantee the validity of transactions among the participants without a centralized authority. There's a timestamp server forming a chain of blocks that contain the transactions in a certain period. Each block is verified by one of the participant in the Bitcoin system. Every one **trust** this verifier so that the economics in the system can sustain. See Figure 1a.



(a) Time-stamp system.



(b) Selecting omnipotent verifier.

Figure 1: Bitcoin Mechanism.

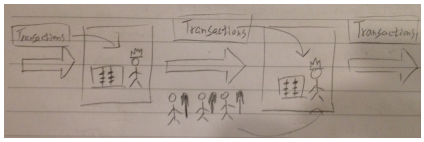
However, why should we trust this verifier and how the verifier being selected? Especially this is under a decentralized fashion!

In Bitcoin system, every participants have the right to be the verifier, but with different probability. The probability of each participant to be selected as a verifier is depended on how much has he/she devoted to the system. That is, the more one contribute to the system, the higher probability he/she has to be selected as a verifier. The mechanism to evaluate the amount of devotion is the so called *proof-of-work*. Simply speaking, each participant keeps playing a game (computing hash value). Who wins the game (find a small hash value) first will be regarded as the verifier for the current block. Intuitively, the participant that wins the game is in some sense being **voted** to be the verifier in that period. The central idea here is that every participant believes in the voting mechanism and thus also trust the verifier. As a result, the omnipotent verifier plays the rule of manager in that period and no one can run against he/she. See Figure 1b.

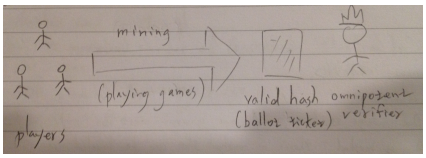
However, there's a potential hazard that the verifier is a bad guy. He/She might modifies the transaction record and benefits himself/herself. In [Nak08], Satoshi referred this kind of situation as **double spending** in which the verifier spends the same money twice, which is definitely not allowed to happen in a trusted system. This kind of double spending circumstances can be viewed as a failure of a network. Fig 2 elaborate more on double spending. The original double spending scenario in [Nak08] consider the possibility that some attacker in Bitcoin system building his/her own blockchain instead of mining on the main chain. See Fig 2a, the attacker builds a private blockchain that is longer than the public chain. Thus, for other participants in the system they will adopt the longer one, which in this case is the attacker's private blockchain. Once the attackers making some fake transactions in its private blockchain, *e.g.,* a double spending, the security of the system will break down. We call this kind of double spending scenario as *fork-style double spending*. And in [Nak08], Satoshi provided an upper bound on the probability of having fork-style double spending.

Another double spending scenario which is less studied and has'nt been formally proposed is called *direct double spending*. The idea is actually quite simple, see Fig 1b, now we consider the case when the attacker has consecutively won the mining and thus can create some fake transactions during his verification. Clearly, the probability of having direct double spending is less than that of fork-style double spending. Namely, the analysis of direct double spending provides a lower bound of the probability of double spending.

The main difference of these two kinds of double spending lies in the allowance of forking blockchain.



(a) Fork-style double spending.



(b) Direct double spending.

Figure 2: Bitcoin Mechanism.

If the system allows the participants to fork the blockchain in order to achieve fully decentralization, then we have to seriously analyze the probability of fork-style double spending. And now, the probability of direct double spending can only be a lower bound. However, once we take the so-called *address identifiability* assumption, the forking issues can be resolved, and thus it is sufficient to consider the probability of direct double spending because the fork-style double spending could not ever happen.

In the following discussion, we will first assume the address identifiability of the system and analyze the probability of direct double spending. The feasibility of address identifiability will be discussed in Section 5.

1.2 High-order Markov Chain

The idea of high-order Markov chain is simple: the transition probability of a stochastic process is only conditioned on the previous k events. Formally, suppose (X_n) is a k-th order Markov chain over state space $\mathcal{X}$, then for $n \geq k$ and $\forall x_i \in \mathcal{X}, 0 \leq i \leq n-1$

$$P[X_n = x_0 | X_{n-1} = x_1, ...X_1 = x_{n-1}] = P[X_n = x_0 | X_{n-1} = x_1, ..., X_{n-k} = x_k]$$

Raftery [Raf85] proposed the Mixture Transition Distribution model (MTD) in 1985 which models the high-order Markov chain with lag coefficient. Later, Berchtold and Raftery [BR02] generalized the idea into multi-matrix MTD, infinite-lag MTD, spatial MTD, etc., which can be used in various applications. In 2005, Ching et.al. [CNZ05] [NC06] relaxed the constraints in MTD model and yielded a more general results. Recently, Li and Ng [LN14] used probability tensor to model the high-order Markov chain, and found some sufficient conditions for the existence and uniqueness of stationary distribution. You can find more historical details and comparisons in Appendix A.

However, these previous works are not quite the same as what we concern here. They generalize the Markov property into high-order Markov chain, which conditions on more than one past states, and focused on the stationary distribution or asymptotic behaviors of a **single state**. Here, we not only utilize the high-order Markov property but also want to know the stationary distribution or asymptotic behaviors of a **sequence of states**. In other words, the event we care about is a period of time/state, not a single snapshot. In dynamic difficulty Bitcoin system, what we concern is the vent: **double spending**, which is a referred to a certain player sequentially winning more than one time.

In the later section, we will show that, under the dynamic difficulty mechanism plus the assumption of address identifiability, the probability of double spending will drastically decrease and in the meantime the winning distribution will still be balance and thus guarantee the security. We verified the security by using Markov chain to model the proposed mechanism. Although it becomes arduous for us to approximate the stationary distribution due to the non-time-invariant property of the system. We design a reduction scheme to lower the dimension of Markov chain from $O(n^k)$ to $O(k^k)$, where n is the number of participants in the system and k is the order of the dynamic difficulty mechanism.

2 Model

In this section, we use a general stochastic process to model the Bitcoin system and construct the dynamic difficulty Bitcoin system step by step. Start from the traditional Bitcoin system in Section 2.1, then introduce the concept of dynamic difficulty in Section 2.2 and give an example of difficulty function in Section 2.3.

2.1 Traditional Bitcoin system

Let's start with considering the traditional Bitcoin system. First, note that the system is basically a **discrete** system. Namely, the system is composed of a sequence of blocks and it's sufficient for us to use the index of each box to order the configurations in the system. In the following construction, we will use the small letter b to denote the the number of block we are at.

Next, there are n participants in the system competing to win the games. Each of them intrinsically has his/her own computing power, denoted as $C_i(b)$, where i refers to the index of the participant and b is the block index mentioned in the previous paragraph. Besides the computing power, there is a time-varying parameter recording the difficulty of the system. We denote the difficulty at block b as $D(b)$. With computing power $\{C_i(b)\}$ and difficulty $\{D(b)\}$ it is sufficient to model the traditional Bitcoin system with a two-tuple stochastic process as follow.

Definition 1 (traditional Bitcoin system)**.** *A traditional Bitcoin system is a 2-tuple*

$$\mathcal{B}_{traditional} = (\{C_i(b) : i = 1, ..., n;\ b \in \mathbb{N}\}, \{D(b) :\ b \in \mathbb{N}\})$$

, where $C_i(b)$ is the computing power of player i and $D(b)$ is the difficulty of the system at the bth block respectively.

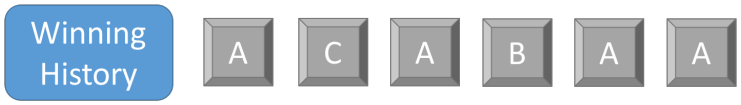
At block b, player i has computing power $C_i(b)$ and is assigned with difficulty $D(b)$. He/She then keep computing hash function until one of the participants find a valid hash value. And with some analysis, we can see that the probability of player i to win at block b is proportional to $\frac{C_i(b)}{D(b)} \sim C_i(b)$. This observation is formally stated in Theorem 3 and is proved in Appendix B. For now, the message is that the winning probability can be derived from the system parameter $\{C_i(b)\}$ and $\{D(b)\}$.

2.2 Dynamic Difficulty Bitcoin System

After modeling the traditional Bitcoin system with a stochastic process, let's go into dynamic difficulty setting. A dynamic difficulty Bitcoin system modifies the mechanism of itself according to the **recent** outcomes in the system. For example, see Figure 3, in the dynamic difficulty Bitcoin system, the difficulty index of each player will be configured according to the past winners in recent blocks. Suppose player A wins 4 blocks in the past 6 blocks, then his difficulty will be higher than player B who only wins 1 blocks in that period. Intuitively, we can think of the scenario as a dynamic stochastic process in which the transition mechanism will depend on a bunch of past results, but not all.

With the above intuition, now let's start formalizing the dynamic difficulty Bitcoin system. First, we can see that we need to add some additional features into the model so that we can achieve the dynamic difficulty scheme. Concretely, we increase the difficulty of the player who wins more times recently and decrease the difficulty of those who wins less recently. As a result, there are three added elements we need to include into the system: the winning history, the number of how many past blocks we concerned, and the difficulty function.

The winning history simply records who wins the block from the beginning till current block b. Formally, we can denote the winning history as $\{W(b)\}$, where $W(b)$ denotes the winner at the bth block. Next, we denote the number of past history we concern as k. That is to say, if we are at block b, then we are going to consider the winner at blocks $b-1, b-2, ..., b-k$. Finally, the difficulty function is specified by Φ_k from the past winning history to a difficulty assignment for every player. With the following notion, we can formally define the dynamic difficulty Bitcoin system as follow:



Player	A	B	C
Difficulty	16D	D	D

Figure 3: Example of dynamic difficulty.

Definition 2 (dynamic difficulty Bitcoin system)**.**
A dynamic difficulty Bitcoin system with n participants (players) is a 5-tuple

$$\mathcal{B} = (\{C_i(b) : i = 1, ..., n;\ b \in \mathbb{N}\}, \{D_i(b) : i = 1, ..., n;\ b \in \mathbb{N}\}, \{W(b) : b \in \mathbb{N}\}, k, \Phi_k)$$

, where

- $C_i(b) \in \mathbb{C}$ *denotes the computing power of player i at block b. For simplicity, if not specified, we assume the computing power of each player is a constant and being all the same. Intuitively, we can think of computing power as* **internal power***.*
- $D_i(b) \in \mathbb{D}$ *denotes the difficulty of player i at block b. Intuitively, we can think of difficulty as* **external power***.*
- $W(b) \in \{0, 1, ..., n-1\}$ *denotes who wins at block b.*
- k *is the number of blocks we look into the past.*
- Ψ_k *denotes the difficulty function that looks into k past blocks.*

We can immediately observe that the process is actually overdetermined. The difficulty $\{D_i(b)\}$ are recursively defined by the winning history $\{W(b)\}$, the number of blocks we look in the past k and the difficulty function Ψ_k. Similarly, we denote the probability of the ith player to win at block b as $P_i(b)$. And for convenience, we denote the recent m histories $\{W^{(b)}, W^{(b-1)}, ..., W^{(b-m+1)}\}$ as m-history.

The most importance thing we care here is the probability of winning. With some probability argument, we can derive a simple relation between winning probability, difficulty, and computing power stated in the following theorem.

Theorem 3. *The winning probability of player i at block b is proportional to the winning probability divided by the difficulty. That is,* $P_i^{(b)} \sim \frac{C_i(b)}{D_i(b)}$.

Proof. The proof is left in Appendix B. ■

Intuitively, Theorem 3 tells us that the winning probability of each player is proportional to the ratio of computing power and the assigned difficulty. As a result, once this ratio is approximately the same for all participants, then the winning probability will be closely to uniform.

With Theorem 3, we can simply calculate the winning probability for player i at block b by divide its ratio $\frac{C_i(b)}{D_i(b)}$ with the summation of all ratio $\sum_{j=0}^{n-1} \frac{C_j(b)}{D_j(b)}$. Remember that what we care about here is the winning probability of each player so that we can construct a transition function and drawing inferences. And Theorem 3 tells us the winning probability is proportional to the ratio of computing power and difficulty. Thus, it's convenient for us to analyze the whole system.

2.3 Difficulty function

Now, let's elaborate more on the difficulty function. We start from a formal definition for difficulty function.

Definition 4 (difficulty function)**.**

A difficulty function that considers k past blocks $\Psi_k : \{0, 1, ..., n-1\}^k \to [0,1]^n$ is a mapping from k past history $W(b-1), W(b-2), ..., W(b-k)$ to a difficulty vector for n players: $D_1(b), D_2(b), ..., D_n(b)$.

Also, with Theorem 3, we can assume $\sum_{j=0}^{n-1} D_j(b) = 1$ for simplicity.

Example 5 (α-exponential non-ordered difficulty function)**.**

Denote the winning times of player i in the past k blocks from current block b as $w_i(b) = \sum_{j=1}^{k} \mathbf{1}_{\{W(b-j)=i\}}$. An α-exponential non-ordered difficulty function Ψ_k^α maps

$$\begin{aligned} D_i(b) &= \big\langle \Psi_k(W(b-1), W(b-2), ..., W(b-k)) \big\rangle_i \\ &= \frac{\alpha^{w_i(b)}}{\sum_{j=0}^{n-1} \alpha^{w_j(b)}} \end{aligned}$$

Suppose we take $\alpha = 2, n = 3, k = 3$ and every player has the same computing power. If the 3-history right now is $(W(b) = 0, W(b-1) = 2, W(b-2) = 0)$, then the winning times of each player is:

$$\begin{aligned} w_0(b+1) &= 2 \\ w_1(b+1) &= 0 \\ w_2(b+1) &= 1 \end{aligned}$$

The difficulty for each player at block $b+1$ is:

$$\begin{aligned} D_0(b+1) &= \frac{2^{w_0(b+1)}}{\sum_{j=0}^{2} 2^{w_j(b+1)}} = \frac{2^2}{2^2 + 2^0 + 2^1} = \frac{4}{7} \\ D_1(b+1) &= \frac{2^{w_1(b+1)}}{\sum_{j=0}^{2} 2^{w_j(b+1)}} = \frac{2^0}{2^2 + 2^0 + 2^1} = \frac{1}{7} \\ D_2(b+1) &= \frac{2^{w_2(b+1)}}{\sum_{j=0}^{2} 2^{w_j(b+1)}} = \frac{2^1}{2^2 + 2^0 + 2^1} = \frac{2}{7} \end{aligned}$$

Finally, the winning probability for each player at block $b+1$ is:

$$\begin{aligned} Pr[W(b+1) = 0 | W(b) = 0, W(b-1) = 2, W(b-2) = 0] &= \frac{1/D_0(b+1)}{\sum_{j=0}^{2} 1/D_j(b+1)} = \frac{1}{7} \\ Pr[W(b+1) = 1 | W(b) = 0, W(b-1) = 2, W(b-2) = 0] &= \frac{1/D_1(b+1)}{\sum_{j=0}^{2} 1/D_j(b+1)} = \frac{4}{7} \\ Pr[W(b+1) = 2 | W(b) = 0, W(b-1) = 2, W(b-2) = 0] &= \frac{1/D_2(b+1)}{\sum_{j=0}^{2} 1/D_j(b+1)} = \frac{2}{7} \end{aligned}$$

, which shows the intuition of $\alpha-$exponential non-ordered difficulty function: the winning probability at block $b+1$ is proportional to $\alpha^{w_i(b+1)}$.

3 Rate of consecutive winning

After defining a mathematical model for dynamic Bitcoin system and the formal notion of difficulty function, now we want to examine how well the dynamic difficulty mechanism preventing us from consecutive winning.

First, let's define the scenario we care about.

- **Setting**: A dynamic difficulty Bitcoin system with n players: $\mathcal{B} = (\{C_i(b) : i = 1, ..., n;\ b \in \mathbb{N}\}, \{D_i(b) : i = 1, ..., n;\ b \in \mathbb{N}\}, \{W(b) : b \in \mathbb{N}\}, k, \Phi_k)$. Assume the computing power is constant for each player, *i.e.,* $C_i^{(b)} = C_j^{(b')}, \forall i, j \in \{0, ..., n-1\},\ \forall b, b'$.
- **Goal**: We want to know the probability of player 1 consecutively winning for k time, *i.e.,* $\lim_{n\to\infty} Pr[W(n) = 1, W(n-1) = 1, ..., W(n-k+1) = 1]$.

Before consider general difficulty function, let's consider the case with no difficulty function (or choosing the $1-$exponential non-ordered difficulty function).

3.1 No difficulty function

As the winning probability for each player at a single block is all the same: $\frac{1}{n}$, the probability of player 1 to consecutively win k blocks is $(\frac{1}{n})^k$. In another point of view, we can think of this no difficulty case as the winning probability of each block is i.i.d while in other cases, the winning probability of each block is correlated.

3.2 Arbitrary difficulty function

Now, we consider choosing an arbitrary difficulty function Ψ_k. After thinking for a while, we can immediately find out that this is not so trivial as the no difficulty case since the winning probability of each block is correlated. That is, we can not simply utilize the i.i.d. property to calculate the goal: $\lim_{n\to\infty} Pr[W(n) = 1, W(n-1) = 1, ..., W(n-k+1) = 1]$. Instead, joint probability of consecutive blocks should be considered.

It sounds complicated, but as we write down the probability of winning distribution at block $b+1$, interesting thing shows up

$$\begin{aligned} &Pr[W(b+1) = i | W(b) = w_1, W(b-1) = w_2, ..., W(1) = w_b] = \\ &Pr[W(b+1) = i | W(b) = w_1, W(b-1) = w_2, ..., W(b-k+1) = w_k] \end{aligned}$$

The system enjoys **kth order Markov property**!

With this observation, we can directly encode the winning probability into a transition matrix as we regard each possible k history as a state. Formally speaking, define the state space of the Markov chain as $\mathcal{M}(\mathcal{B}) = (S, P)$, where S is the state space and P is the transition probability function.

- $S = \{1, ..., n\}^k$: the k history.
- $P : S \times S \to [0, 1]$: suppose $s = (s_1, ..., s_k), s' = (s'_1, ..., s'_k)$ then

$$P(s, s') = \begin{cases} \dfrac{\frac{C_{s'_1}(b)}{\left\langle \Psi_k(s) \right\rangle_{s'_1}}}{\sum_{i=0}^{n-1} \frac{C_i(b)}{\left\langle \Psi_k(s) \right\rangle_i}} & \text{, if } s_i = s'_{i+1} \forall i = 1, ..., k-1, \\ 0 & \text{, otherwise} \end{cases}$$

, where $\left\langle \Psi_k(s) \right\rangle_i$ is the difficulty for player i conditioned on history s.

Let $\mathbb{W}_k(b)$ be the probability vector over S denoting the probability of k past history started from block b. Then, the stationary distribution of k past history is $\hat{\mathbb{W}}_k$ that satisfies $\hat{\mathbb{W}}_k = \lim_{b\to\infty} \mathbb{W}_k(b)$. Or, equivalently, $\hat{\mathbb{W}}_k = P\hat{\mathbb{W}}_k$.

3.3 $\alpha-$exponential non-ordered difficulty function

Now, suppose we choose the $\alpha-$exponential non-ordered difficulty function in a n participants, k past history dynamic Bitcoin system $\mathcal{B}$. Then, we get $\mathcal{M}(\mathcal{B}) = (S, P)$, where $S = \{1, ..., n\}^k$ and

$$P(s, s') = \begin{cases} \frac{1/\frac{\alpha^{w_{s'_1}}}{\sum_{j=0}^{n}\alpha^{w_j}}}{\sum_{i=0}^{n-1} 1/\frac{\alpha^{w_i}}{\sum_{j=0}^{n}\alpha^{w_j}}} = \frac{1/\alpha^{w_{s'_1}}}{\sum_{i=0}^{n-1} 1/\alpha^{w_i}} & \text{, if } s_i = s'_{i+1} \forall i = 1, ..., k-1, \\ 0 & \text{, otherwise} \end{cases}$$

Take a simpler example: $\alpha = 2, n = 3, k = 2$. P can be represented as a $3^2 \times 3^2$ matrix:

	(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
(0,0)	1/9	1/4	1/4	0	0	0	0	0	0
(0,1)	0	0	0	1/4	4/9	1/2	0	0	0
(0,2)	0	0	0	0	0	0	1/4	1/2	4/9
(1,0)	4/9	1/4	1/2	0	0	0	0	0	0
(1,1)	0	0	0	1/4	1/9	1/4	0	0	0
(1,2)	0	0	0	0	0	0	1/2	1/4	4/9
(2,0)	4/9	1/2	1/4	0	0	0	0	0	0
(2,1)	0	0	0	1/2	4/9	1/4	0	0	0
(2,2)	0	0	0	0	0	0	1/4		1/9

Table 1: The transition matrix of case: $\alpha = 2, n = 3, k = 2$.

And the stationary vector

$$\hat{\mathbb{W}}_k = (0.073, 0.130, 0.130, 0.130, 0.073, 0.130, 0.130, 0.130, 0.073)^T$$

That is, the probability of consecutive winning in the long run is

$$\lim_{b\to\infty} Pr[W^{(b)} = 0, W^{(b-1)} = 0] = 0.073$$

While for the no difficulty function system, the probability of consecutive winning is $(\frac{1}{3})^2 = 0.111$

Here we calculate the probability of consecutive winning with different number of past history and different difficulty functions. The number of participants is 5.

	2	3	4	5
No difficulty	4e-2	8e-3	1.6e-3	3.2e-4
2-exponential non-ordered	2.34e-2	1.59e-3	6.14e-5	1.35e-6
5-exponential non-ordered	1.12e-2	1.64e-4	6.38e-7	6.74e-10

Table 2: Probability of consecutive winning. $n = 5$.

3.4 Discussion and calculation issue

We have discussed the probability of consecutive winning with different difficulty functions. And in Table 2 we can see that the probability of consecutive winning is clearly drop down as we increase the number of past history and the difficulty ratio α.

However, once we want to increase the number of participants, the size of transition matrix will drastically increase. For example, if we consider 4 past history, the number of states in the transition matrix of 5

participants is $5^4 = 625$. As we consider 20 participants, the number will become $20^4 = 160000$, which is hardly compute by normal computer!As a result, once we create a new difficulty function and want to see it's performance of preventing from consecutive winning, we cannot efficiently compute the results of large amount of participants with the above calculation model.

But if we look deeper into the transition matrix, we can find out that there are so many 0. Namely, the matrix is extremely sparse. Moreover, it has meaningful structure. In the next section, we will demonstrate a cool technique which can efficiently reduce the number of states.

4 State reduction

In the previous section, we mentioned that we want to find out the stationary probability of consecutive winning. In some sense, what we want to know is the stationary probability of **a sequence of past events**. However, traditional techniques only provide us the way to compute stationary probability for **single time stamp**. As we want to consider more that one time stamps together, we must get our hands dirty into the complex correlation of past events since here we don't have the i.i.d. condition. Thus, the size of state space is exponentially growing due to the multiplicative property of the number of outcomes and it's computationally inefficient to compute directly!

While the number of states grows up, we can see that the outcomes are full of **super-symmetry**. That is, there are some sequence of events that behave exactly the same as other events. As a result, we can intuitively put them all together and reduce the number of outcomes, which will drastically reduce the computation consumption!

In this section, we first formalize our idea of state reduction, then provide a framework to construct the whole reduction process step by step. We hope that this framework provide those who want to perform state reduction on their own a rule of thumb to follow.

4.1 Intuition

When utilizing the idea in section 3.2 to encode the transition of $\mathcal{B}$, there are lots of redundant states. moreover, the transition matrix is extremely sparse. Since what we care about is the limiting probability of the event: $W(b) = 0, W(b-1), ..., W(b-k+1)$, we can see that there are a great number of states behave the same. For instance, $(0, 1, 2)$ and $(0, 3, 4)$. Thus, we can create a new state $(0, A, B)$ to represent these states: $(0, 1, 2)$, $(0, 2, 1)$, $(0, 3, 4)$, $(0, 1, 4)$...

Formally, we can construct a new state space S' as:

$$S' = \{(s_1, ..., s_k) \in \{0, A_1, ..., A_k\}^k | \forall i < j, \sum_{l=1}^{k} \mathbf{1}_{\{s_l = A_i\}} \geq \sum_{l=1}^{k} \mathbf{1}_{\{s_l = A_j\}}\}$$

Intuitively, here we use $\{A_i\}$ to represent the winner that is not player 0. We don't care who actually win the block. We only care what's the winning probability of player 0 under different difficulty function and this is only affected by how many other distinct players win in the past, not who. With this idea, we can construct the new state space S' with some combinatorial techniques. And the number of states has been drastically reduced. For example, if we take the number of participants $n = 10$ and $k = 3$, the new state space will be:

$$\begin{aligned} S' = \{&(0,0,0), (0,0,A_1), (0,A_1,0), (A_1,0,0), (0,A_1,A_1), \\ &(A_1,0,A_1), (A_1,A_1,0), (0,A_1,A_2), (A_1,0,A_2), (A_1,A_2,0) \\ &(A_1,A_1,A_1), (A_1,A_1,A_2), (A_1,A_2,A_1), (A_2,A_1,A_1), (A_1,A_2,A_3)\} \end{aligned}$$

The number of states has been reduced from $10^3 = 1000$ to 15.

Using a simple program to calculate the number of states, we can see that the number of states is notably reduced:

n\k	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	8	16	32	64
3	3	9	27	81	243	729
4	4	16	64	256	1024	4096
5	5	25	125	625	3125	15625
6	6	36	216	1296	7776	46656
7	7	49	343	2401	16807	117649
Others	n	n^2	n^3	n^4	n^5	n^6

(a) Number of non-reduced states.

n\k	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	8	16	32	64
3	2	5	14	41	122	365
4	2	5	15	51	187	715
5	2	5	15	52	202	855
6	2	5	15	52	203	876
7	2	5	15	52	203	877
Others	2	5	15	52	203	877

(b) Number of reduced states.

Table 3: Number of (reduced) states.

4.2 Abstract model

As we observe deep into the underlying mechanism of dynamic difficulty Bitcoin system and other similar dynamic system, we can find out that the basic dynamics are actually the same. What makes the system become complicated is that some parameters in the system will change over time according to the outcome. With this intuition, we can somehow divide the system into two parts: The base rules and the parameters. Moreover, the parameters can also be categorized into fixed parameters and dynamic parameters. See Figure 4.

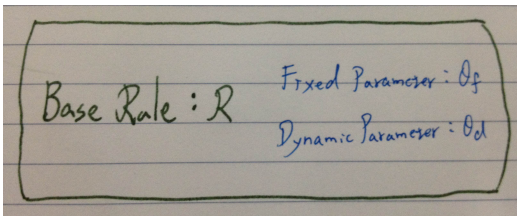


Figure 4: Abstract model for dynamic difficulty Bitcoin system.

Consider dynamic difficulty Bitcoin system, the base rules is analogous to the basic mechanisms such as mining policies, proof of work, timestamps etc. And the parameters are number of players, computing power, difficulty function, difficulty etc. The following is a table

Base rules		Mining policies, proof of work, timestamps.
Parameters	Fixed	Difficulty function, number of players*, computing power*.
	Dynamic	Difficulty.

Table 4: Analogy in dynamic difficulty Bitcoin system.

To be more precise, we formalize the abstract model of dynamic difficulty Bitcoin system as follow:

Parameters We define the fixed and dynamic parameters respectively as follow:

Definition 6 (fixed parameters)**.**
The fixed parameters of dynamic difficulty Bitcoin system, denoted as θ_f, is a 4-tuple

$$\theta_f := \{n, \{C_i\}, k, \Phi_k\}$$

, where

- n: *number of players.*
- $\{C_i\}$: *computing power of each player.*
- k: *number of history considered in the difficulty function.*
- Φ_k: *the dynamic difficulty function.*

Definition 7 (dynamic parameters)**.**
The dynamic parameters of dynamic difficulty Bitcoin system on block b, denoted as $\theta_d^{(b)}$*, is a 2-tuple*

$$\theta_d^{(b)} := \{\{D_i^{(b)}\}, \{W^{(b)}\}\}$$

, where

- $\{D_i^{(b)}\}$: *the difficulty of each player of block b.*
- $\{W^{(b)}\}$: *the winning history in the system. Note that, for our analysis, here the winner recorded in the history a the* **distribution***, not a certain player. That is, we record the probability for each player to win on block b.*

Base rules We define the base rules as follow:

Definition 8 (base rules)**.** *The base rules, denoted as R, is a function from* $(\theta_f, \theta_d^{(b)})$ *to a winning distribution* $W^{(b+1)}$*. That is,* $R : \theta_f \times \theta_d^{(b)} \to W^{(b+1)}$*,* $\forall b \geq 0$*.*

With the abstract model bear in mind, we can start the reduction process.

4.3 Framework

There are three steps in the reduction process:

1. Reduce states.
2. Transition matrix.
3. Stationary distribution.

In the first step, **Reduce states**, we follow the intuition above to scan through all possible past configurations and generate reduced states. Next, **Transition matrix**, we construct the corresponding transition matrix according to the reduced state, the basic parameters of the system, and the decay parameter of the exponential non-ordered model. In the end, **Stationary distribution**, we use iterative method to find the stationary distribution of reduced transition matrix and get the stationary probability of consecutive winning.

4.3.1 Reduce states

In this step, we are going to construct a mapping from the standard state space to the reduced state space based on the intuition in Section 4.1. Formally, we define the standard state space and reduced state space as follow:

Definition 9 (standard state space)**.** *The standard state space of the dynamic difficulty Bitcoin system, denoted as* S_s*, is the m-Cartesian product over the player space.*

$$S_s := \{0, 1, ..., n-1\}^m$$

, which is the state space for m-history.

Definition 10 (reduced state space)**.** *The reduced state space of the dynamic difficulty Bitcoin system, denoted as S_r, is a subset of S_s defined as follow:*

$$S_r := \{(s_1, ..., s_m) \in S_s : \begin{cases} \forall 0 < i < j < m, & \sum_{k=1}^{m} \mathbf{1}_{\{s_k=i\}} \geq \sum_{k=1}^{m} \mathbf{1}_{\{s_k=j\}} \\ \forall 0 < i < j < m, & \textit{if } \sum_{k=1}^{m} \mathbf{1}_{\{s_k=i\}} = \sum_{k=1}^{m} \mathbf{1}_{\{s_k=j\}}, \\ & \textit{then } \arg\min_{1\leq k\leq m} \mathbf{1}_{\{s_k=i\}} < \arg\min_{1\leq k\leq m} \mathbf{1}_{\{s_k=j\}} \end{cases} \}$$

, which is the reduced state space for m-history.

Note that the first constrains regulates the number of smaller index should not be less than the number of larger index. And the second constrain regulates the if two indexed appear the same number of times, then the smaller index should appear first than the larger one.

After defining the standard state space and reduced state space, now we are going to construct a mapping between them. And this is trivial since we can directly get the mapping by the definition of reduced state space.

Definition 11 (Reduced mapping)**.** *A reduced mapping from standard state space S_s to reduced state space S_r denoted as $R : S_s \to S_r$. $\forall s = (s_1, ..., s_m) \in S_s$, $R(s)$ is defined as follow*

$$R(s) = (s'_1, ..., s'_m) \in S_r \text{ s.t. } \forall i \neq j, s'_i = s'_j \Leftrightarrow s_i = s_j, \text{ and } \forall i \text{ s.t. } s_i = 0 \Leftrightarrow s'_i = 0$$

Note that the above construction is not unique. The definition of reduced state space and the corresponding reduced mapping depends on the usage. It will vary as the target event is different. As a result, for general application, one should observe the structure in their system and find a good way to reduce the number of states. Here, we just give an example in dynamic difficulty Bitcoin system.

4.3.2 Transition matrix

In this step, we are going to formalize the transition function over the reduced state space. Actually, it's quite straightforward, just simply apply the reduced mapping on the standard transition function. Suppose the standard transition function defined on standard probability space is $P_s : S_s \to S_s$. Then we defined the reduced transition function as follow:

Definition 12 (reduced transition function)**.** *Suppose $P_s : S_s \to S_s$ is the standard transition function, then the reduced transition function, denoted as $P_r : S_r \to S_r$, is defined as*

$$P_r(s_r, s'_r) := \sum_{\{s:R(s_r)=s\}} \sum_{\{s':R(s'_r)=s'\}} P_s(s, s')$$

That is, we sum up all the probability of the preimage of pair (s_r, s'_r).

4.3.3 Stationary distribution

To define the notion of stationary distribution in reduced state space, we first need to specify the notion of probability distribution over reduced stated space. We denote the space of probability distribution of reduced state space as

$$\mathcal{P}_r := \{p_r = (p_1, ..., p_{|S_r|}) : p_i \geq 0 \ \forall 1 \leq i \leq |S_r|, \sum_{i=1}^{|S_r|} p_i = 1\}$$

Note that we can view the stochastic process $W^{(1)}, W^{(2)}, ...$ as another stochastic process of m-**history**: $\{W^{(b)}, W^{(b-1)}, ..., W^{(b-m+1)}\}$,$\{W^{(b+1)}, W^{(b)}, ..., W^{(b-m+2)}\}$,... Moreover, we use a describe such stochastic process with random variables $W_m(b), W_m^{(b+1)}, ...$ where the support of $W_m^{(b)}$ is S_r. As a result, the distribution of $W_m^{(b)}$ can be represented by probability distribution in $\mathcal{P}_r$. That is, $W_m^{(b)} \sim p^{(b)} \in \mathcal{P}_r$.

Finally, we can define the stationary distribution of the stochastic process $W^{(b)}, W^{(b+1)}$, ... in the sense of m-history as

Definition 13 (reduced stationary distribution of m-history)**.** *Suppose $\bar{p} \in \mathcal{P}_r$, we say $\bar{p}$ is a reduced stationary distribution of reduced transition function P_r if*

$$\bar{p} = P_r(\bar{p}, \cdot)$$

4.4 Analysis

The following is a table of probability of consecutive winning with different system settings:

n\k	1	2	3	4	5	6
1	1	1	1	1	1	1
2	0.5	0.19	5.56e-2	1.18e-2	1.80e-3	1.97e-4
3	0.34	7.30e-2	1.04e-2	9.41e-4	5.39e-5	1.94e-6
4	0.25	3.83e-2	3.52e-3	1.93e-4	6.25e-6	1.20e-7
5	0.20	2.35e-2	1.59e-3	6.14e-5	1.35e-6	1.70e-8
6	0.17	1.59e-2	8.46e-4	2.52e-5	4.17e-7	3.84e-9
7	0.14	1.45e-2	5.03e-4	1.21e-5	1.60e-7	1.14e-9

Table 5: Probability of consecutive winning.

5 Discussion

Up to now, we have shown that the probability of double spending is drastically decreasing after using dynamic difficulty proof of work mechanism. In this section, we will first summary the results and compare with other's works. Then, discuss the major assumption, address identifiability, in this work. At the end, we will elaborate on some future works and open questions.

5.1 Summary and comparison

In Table 6, we summarize the double spending probability from different mechanism. The first row is the double spending probability computed from the program in [Nak08] and the following two rows are the results from the dynamic difficulty proof of work mechanism we proposed.

Mechanism\ k	1	2	3	4	5	6
Bitcoin	0.2046	0.0510	1.312e-2	3.455e-3	9.137e-4	2.428e-4
DD: 2-exponential	0.1011	0.0054	1.560e-4	1.214e-5	1.953e-8	8.447e-11
DD: 5-exponential	0.1030	0.0024	1.218e-5	1.412e-8	3.660e-12	2.135e-16

Table 6: Summary: Attacker has 10% computing power.

Clearly, we can see that the double spending probability of dynamic difficulty mechanism decrease much more faster than that of traditional Bitcoin setting. However, note that the double spending criteria in two mechanisms are actually not exactly the same. The traditional Bitcoin mechanism allows forking in their blockchain, so the double spending probability in their estimation will be a little higher than that under address-identifiability assumption. More details will be discussed in the Section 5.2. For now, Table 6 is a temporary results showing you the good performance of dynamic difficulty proof of work mechanism.

5.2 Address identifiability

In the Satoshi's paper [Nak08], he abandoned address identifiability to achieve fully anonymity. And as a consequences, the traditional Bitcoin system allow forking to happen in the blockchain and thus requiring

different analyzing model then our high-order Markov chain. In the analysis of our dynamic difficulty setting, we adopt the address identifiability assumption so that the double spending probability will be smaller than that in traditional fork-style Bitcoin network. However, here we try to argue the possibility of address identifiability both in a practical construction based perspective and in a performance-based evaluation point of view.

We present some possible construction to argue that instead of losing anonymity, address identifiability will increase the reliability of the whole network. The basic idea is to regulate the qualification of the participant in order to prevent malicious miners. However, problems come up immediately, how can we regulate the qualification of participants without losing the trust and justice? Here, we present two possible solutions. The first solution is to fix the address. Swanson in [Swa] proposed the idea of distributed ledger system, which is a permissioned identity system. User's identity in the system is recorded in a whitelist /blacklist through some public key system. The second solution is using the amount of Bitcoin to keep track of the identity of the participant. In [DH14], Evan and Kyle proposed the so called Darkcoin which uses master node to record the amount of coin among a group of miner. The master node should have enough amount of coin to have the qualification to vote. In this construction, we don't need to directly identify the address while in the meantime still preserve the security.

On the other hand, the performance of the network under address identifiability assumption is both computationally efficient and is less likely to occur double spending. Because if address identifiability, participants in the network no longer need to worrying about deciding which blockchain to take as there will only exist one unique blockchain. As to the probability of double spending to occur, since *direct double spending* is a special case of *fork-based double spending*, it's obvious that the former will is less likely to happen. To sum up, we can see that it is possible to have a secure address identifiable Bitcoin network with a better performance.

6 Conclusion

In this paper, we try to solve the intrinsic overhead problem of Bitcoin by proposing a new proof of work mechanism. We first formalize Bitcoin as a voting system and further generalize to a dynamic difficulty Bitcoin system. The dynamic difficulty mechanism balance the winning distribution in the network in the sense that those who win a lot recently will be less likely to become the verifier next time. We then use a high-order Markov chain to quantitatively model the dynamic difficulty system. Finally, we showed that the double spending rate drastically decreases from 0.02% to 0.00000008% after adopting the exponential non-ordered difficulty function. As a result, we can see that dynamic difficulty mechanism successfully improve the performance of Bitcoin system by proposing a modified proof-of-work protocol instead of the traditional approaches via information propagation.

6.1 Future work

In this work we have analyzed the double spending probability of dynamic difficulty mechanism. However, the performance in accelerating the transaction confirmation is not fully examined, which is a potential future work. On the other hand, ee would like to point out the possible breakdown of a balanced motivation system. Since we modify the difficulty of those who win recently, they might decide to have a break after the winning and thus breaking the dynamics of the system. Related problems was studied by Rosenfeld in [Ros14] and Kroll et al. in [KDF13] under the traditional setting. Another interesting study can be the analysis of winning distribution under dynamic difficulty mechanism. There might be the case that once we adopt the dynamic difficulty mechanism, the winning distribution no longer satisfy the participants so that they will have no motivation to join the proof of work process.

To sum up, the dynamic difficulty mechanism we proposed can effectively solve the two deadly drawbacks of Bitcoin: intrinsic overhead and the hazard of double spending at the same time. In this work we presented

the guarantee in security and some basic performance results. We hope that in the future this work can be extended and implemented in real word so that the brilliant idea of Bitcoin can be applied in more fields.

References

[BDE+13] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. Have a snack, pay with bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–5. IEEE, 2013.

[BR02] André Berchtold and Adrian E Raftery. The mixture transition distribution model for high-order markov chains and non-gaussian time series. *Statistical Science*, pages 328–356, 2002.

[CNZ05] Waiki Ching, Michael K Ng, and Shuqin Zhang. On computation with higher-order markov chains. In *Current Trends in High Performance Computing and Its Applications*, pages 15–24. Springer, 2005.

[DH14] Evan Duffield and Kyle Hagan. Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system. 2014.

[KAC12] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012:248, 2012.

[KDF13] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, 2013.

[LN14] Wen Li and Michael K Ng. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra*, 62(3):362–385, 2014.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.

[NC06] Michael K Ng and WK Ching. *Markov Chains: Models, Algorithms and Applications*. Springer, 2006.

[Raf85] Adrian E Raftery. A model for high-order markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 528–539, 1985.

[Ros11] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[Ros14] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[Sta15] Chrysoula Stathakopoulou. A faster bitcoin network. 2015.

[Swa] Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems.

A High-order Markov Chain Models

B Proof of Theorem 3